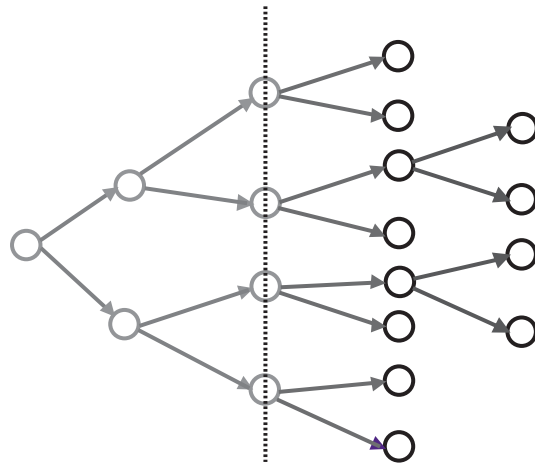

PROCEEDINGS

OF THE FIFTH

SIAM INTERNATIONAL

CONFERENCE

ON DATA MINING



SIAM PROCEEDINGS SERIES LIST

- Glowinski, R., Golub, G. H., Meurant, G. A., and Periaux, J., *First International Conference on Domain Decomposition Methods for Partial Differential Equations* (1988)
- Salam, Fathi M. A. and Levi, Mark L., *Dynamical Systems Approaches to Nonlinear Problems in Systems and Circuits* (1988)
- Datta, B., Johnson, C., Kaashoek, M., Plemmons, R., and Sontag, E., *Linear Algebra in Signals, Systems and Control* (1988)
- Ringeisen, Richard D. and Roberts, Fred S., *Applications of Discrete Mathematics* (1988)
- McKenna, James and Temam, Roger, *ICIAM '87: Proceedings of the First International Conference on Industrial and Applied Mathematics* (1988)
- Rodrigue, Garry, *Parallel Processing for Scientific Computing* (1989)
- Caflish, Russel E., *Mathematical Aspects of Vortex Dynamics* (1989)
- Wouk, Arthur, *Parallel Processing and Medium-Scale Multiprocessors* (1989)
- Flaherty, Joseph E., Paslow, Pamela J., Shephard, Mark S., and Vasilakis, John D., *Adaptive Methods for Partial Differential Equations* (1989)
- Kohn, Robert V. and Milton, Graeme W., *Random Media and Composites* (1989)
- Mandel, Jan, McCormick, S. F., Dendy, J. E., Jr., Farhat, Charbel, Lonsdale, Guy, Parter, Seymour V., Ruge, John W., and Stüben, Klaus, *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods* (1989)
- Colton, David, Ewing, Richard, and Rundell, William, *Inverse Problems in Partial Differential Equations* (1990)
- Chan, Tony F., Glowinski, Roland, Periaux, Jacques, and Widlund, Olof B., *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations* (1990)
- Dongarra, Jack, Messina, Paul, Sorensen, Danny C., and Voigt, Robert G., *Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing* (1990)
- Glowinski, Roland and Lichniewsky, Alain, *Computing Methods in Applied Sciences and Engineering* (1990)
- Coleman, Thomas F. and Li, Yuying, *Large-Scale Numerical Optimization* (1990)
- Aggarwal, Alok, Borodin, Allan, Gabow, Harold, N., Galil, Zvi, Karp, Richard M., Kleitman, Daniel J., Odlyzko, Andrew M., Pulleyblank, William R., Tardos, Éva, and Vishkin, Uzi, *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms* (1990)
- Cohen, Gary, Halpern, Laurence, and Joly, Patrick, *Mathematical and Numerical Aspects of Wave Propagation Phenomena* (1991)
- Gómez, S., Hennart, J. P., and Tapia, R. A., *Advances in Numerical Partial Differential Equations and Optimization: Proceedings of the Fifth Mexico-United States Workshop* (1991)
- Glowinski, Roland, Kuznetsov, Yuri A., Meurant, Gérard, Périaux, Jacques, and Widlund, Olof B., *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations* (1991)
- Alavi, Y., Chung, F. R. K., Graham, R. L., and Hsu, D. F., *Graph Theory, Combinatorics, Algorithms, and Applications* (1991)
- Wu, Julian J., Ting, T. C. T., and Barnett, David M., *Modern Theory of Anisotropic Elasticity and Applications* (1991)
- Shearer, Michael, *Viscous Profiles and Numerical Methods for Shock Waves* (1991)
- Griewank, Andreas and Corliss, George F., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application* (1991)
- Frederickson, Greg, Graham, Ron, Hochbaum, Dorit S., Johnson, Ellis, Kosaraju, S. Rao, Luby, Michael, Megiddo, Nimrod, Schieber, Baruch, Vaidya, Pravin, and Yao, Frances, *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms* (1992)
- Field, David A. and Komkov, Vadim, *Theoretical Aspects of Industrial Design* (1992)
- Field, David A. and Komkov, Vadim, *Geometric Aspects of Industrial Design* (1992)
- Bednar, J. Bee, Lines, L. R., Stolt, R. H., and Weglein, A. B., *Geophysical Inversion* (1992)
- O'Malley, Robert E. Jr., *ICIAM 91: Proceedings of the Second International Conference on Industrial and Applied Mathematics* (1992)
- Keyes, David E., Chan, Tony F., Meurant, Gérard, Scroggs, Jeffrey S., and Voigt, Robert G., *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations* (1992)
- Dongarra, Jack, Messina, Paul, Kennedy, Ken, Sorensen, Danny C., and Voigt, Robert G., *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing* (1992)

Corones, James P., Kristensson, Gerhard, Nelson, Paul, and Seth, Daniel L., *Invariant Imbedding and Inverse Problems* (1992)

Ramachandran, Vijaya, Bentley, Jon, Cole, Richard, Cunningham, William H., Guibas, Leo, King, Valerie, Lawler, Eugene, Lenstra, Arjen, Mulmuley, Ketan, Sleator, Daniel D., and Yannakakis, Mihalis, *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (1993)

Kleinman, Ralph, Angell, Thomas, Colton, David, Santosa, Fadil, and Stakgold, Ivar, *Second International Conference on Mathematical and Numerical Aspects of Wave Propagation* (1993)

Banks, H. T., Fabiano, R. H., and Ito, K., *Identification and Control in Systems Governed by Partial Differential Equations* (1993)

Sleator, Daniel D., Bern, Marshall W., Clarkson, Kenneth L., Cook, William J., Karlin, Anna, Klein, Philip N., Lagarias, Jeffrey C., Lawler, Eugene L., Maggs, Bruce, Milenkovic, Victor J., and Winkler, Peter, *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (1994)

Lewis, John G., *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra* (1994)

Brown, J. David, Chu, Moody T., Ellison, Donald C., and Plemmons, Robert J., *Proceedings of the Cornelius Lanczos International Centenary Conference* (1994)

Dongarra, Jack J. and Tourancheau, B., *Proceedings of the Second Workshop on Environments and Tools for Parallel Scientific Computing* (1994)

Bailey, David H., Bjørstad, Petter E., Gilbert, John R., Mascagni, Michael V., Schreiber, Robert S., Simon, Horst D., Torczon, Virginia J., and Watson, Layne T., *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing* (1995)

Clarkson, Kenneth, Agarwal, Pankaj K., Atallah, Mikhail, Frieze, Alan, Goldberg, Andrew, Karloff, Howard, Manber, Udi, Munro, Ian, Raghavan, Prabhakar, Schmidt, Jeanette, and Young, Moti, *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms* (1995)

Becache, Elaine, Cohen, Gary, Joly, Patrick, and Roberts, Jean E., *Third International Conference on Mathematical and Numerical Aspects of Wave Propagation* (1995)

Engl, Heinz W., and Rundell, W., *GAMM-SIAM Proceedings on Inverse Problems in Diffusion Processes* (1995)

Angell, T. S., Cook, Pamela L., Kleinman, R. E., and Olmstead, W. E., *Nonlinear Problems in Applied Mathematics* (1995)

Tardos, Éva, Applegate, David, Canny, John, Eppstein, David, Galil, Zvi, Karger, David R., Karlin, Anna R., Linial, Nati, Rao, Satish B., Vitter, Jeffrey S., and Winkler, Peter M., *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms* (1996)

Cook, Pamela L., Roytburd, Victor, and Tulin, Marshal, *Mathematics Is for Solving Problems* (1996)

Adams, Loyce and Nazareth, J. L., *Linear and Nonlinear Conjugate Gradient-Related Methods* (1996)

Renardy, Yuriko Y., Coward, Adrian V., Papageorgiou, Demetrios T., and Sun, Shu-Ming, *Advances in Multi-Fluid Flows* (1996)

Berz, Martin, Bischof, Christian, Corliss, George, and Griewank, Andreas, *Computational Differentiation: Techniques, Applications, and Tools* (1996)

Delic, George and Wheeler, Mary F., *Next Generation Environmental Models and Computational Methods* (1997)

Engl, Heinz W., Louis, Alfred, and Rundell, William, *Inverse Problems in Geophysical Applications* (1997)

Saks, Michael, Anderson, Richard, Bach, Eric, Berger, Bonnie, Blum, Avrim, Chazelle, Bernard, Edelsbrunner, Herbert, Henzinger, Monika, Johnson, David, Kannan, Sampath, Khuller, Samir, Maggs, Bruce, Muthukrishnan, S., Ruskey, Frank, Seymour, Paul, Spencer, Joel, Williamson, David P., and Williamson, Gill, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (1997)

Alexandrov, Natalia M. and Hussaini, M. Y., *Multidisciplinary Design Optimization: State of the Art* (1997)

Van Huffel, Sabine, *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling* (1997)

Ferris, Michael C. and Pang, Jong-Shi, *Complementarity and Variational Problems: State of the Art* (1997)

Bern, Marshall, Fiat, Amos, Goldberg, Andrew, Kannan, Sampath, Karloff, Howard, Kenyon, Claire, Kierstead, Hal, Kosaraju, Rao, Linial, Nati, Rabani, Yuval, Rödl, Vojta, Sharir, Micha, Shmoys, David, Spielman, Dan, Spinrad, Jerry, Srinivasan, Aravind, and Sudan, Madhu, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (1998)

DeSanto, John A., *Mathematical and Numerical Aspects of Wave Propagation* (1998)

Tarjan, Robert E., Warnow, Tandy, Amenta, Nina, Benham, Craig, Corneil, Derek G., Edelsbrunner, Herbert, Feigenbaum, Joan, Gusfield, Dan, Habib, Michel, Hall, Leslie, Karp, Richard, King, Valerie, Koller, Daphne, McKay, Brendan, Moret, Bernard, Muthukrishnan, S., Phillips, Cindy, Raghavan, Prabhakar, Randall, Dana, and Scheinerman, Edward, *Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms* (1999)

Hendrickson, Bruce, Yelick, Katherine A., Bischof, Christian H., Duff, Iain S., Edelman, Alan S., Geist, George A., Heath, Michael T., Heroux, Michael H., Koelbel, Chuck, Schrieber, Robert S., Sincovec, Richard F., and Wheeler, Mary F., *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing* (1999)

Henderson, Michael E., Anderson, Christopher R., and Lyons, Stephen L., *Object Oriented Methods for Interoperable Scientific and Engineering Computing* (1999)

Shmoys, David, Brightwell, Graham, Cohen, Edith, Cook, Bill, Eppstein, David, Gerards, Bert, Irani, Sandy, Kenyon, Claire, Ostrovsky, Rafail, Peleg, David, Pevzner, Pavel, Reed, Bruce, Stein, Cliff, Tetali, Prasad, and Welsh, Dominic, *Proceedings of the Eleventh ACM-SIAM Symposium on Discrete Algorithms* (2000)

Bermúdez, Alfredo, Gómez, Dolores, Hazard, Christophe, Joly, Patrick, and Roberts, Jean E., *Fifth International Conference on Mathematical and Numerical Aspects of Wave Propagation* (2000)

Kosaraju, S. Rao, Bellare, Mihir, Buchsbaum, Adam, Chazelle, Bernard, Graham, Fan Chung, Karp, Richard, Lovász, László, Motwani, Rajeev, Myrvold, Wendy, Pruhs, Kirk, Sinclair, Alistair, Spencer, Joel, Stein, Cliff, Tardos, Eva, Vempala, Santosh, *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms* (2001)

Koelbel, Charles and Meza, Juan, *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing* (2001)

Grossman, Robert, Kumar, Vipin, and Han, Jiawei, *Proceedings of the First SIAM International Conference on Data Mining* (2001)

Berry, Michael, *Computational Information Retrieval* (2001)

Eppstein, David, Demaine, Erik, Doerr, Benjamin, Fleischer, Lisa, Goel, Ashish, Goodrich, Mike, Khanna, Sanjeev, King, Valerie, Munro, Ian, Randall, Dana, Shepherd, Bruce, Spielman, Dan, Sudakov, Benjamin, Suri, Subhash, and Warnow, Tandy, *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2002)

Grossman, Robert, Han, Jiawei, Kumar, Vipin, Mannila, Heikki, and Motwani, Rajeev, *Proceedings of the Second SIAM International Conference on Data Mining* (2002)

Estep, Donald and Tavener, Simon, *Collected Lectures on the Preservation of Stability under Discretization* (2002)

Ladner, Richard E., *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments* (2003)

Barbará, Daniel and Kamath, Chandrika, *Proceedings of the Third SIAM International Conference on Data Mining* (2003)

Olshevsky, Vadim, *Fast Algorithms for Structured Matrices: Theory and Applications* (2003)

Munro, Ian, Albers, Susanne, Arge, Lars, Brodal, Gerth, Buchsbaum, Adam, Cowen, Lenore, Farach-Colton, Martin, Frieze, Alan, Goldberg, Andrew, Hershberger, John, Jerrum, Mark, Johnson, David, Kosaraju, Rao, López-Ortiz, Alejandro, Mosca, Michele, Muthukrishnan, S., Rote, Günter, Ruskey, Frank, Spinrad, Jeremy, Stein, Cliff, and Suri, Subhash, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2004)

Arge, Lars and Italiano, Giuseppe F., *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithms and Combinatorics* (2004)

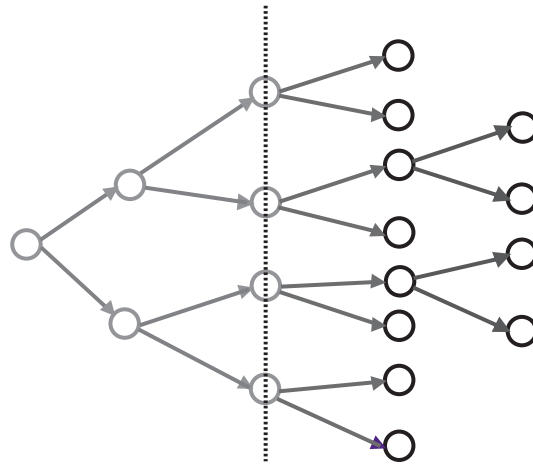
Hill, James M. and Moore, Ross, *Applied Mathematics Entering the 21st Century: Invited Talks from the ICIAM 2003 Congress* (2004)

Berry, Michael W., Dayal, Umeshwar, Kamath, Chandrika and Skillicorn, David, *Proceedings of the Fourth SIAM International Conference on Data Mining* (2004)

Azar, Yossi, Buchsbaum, Adam, Chazelle, Bernard, Cole, Richard, Fleischer, Lisa, Golin, Mordecai, Goodrich, Michael, Grossi, Roberto, Guha, Sudipto, Halldorsson, Magnus M., Indyk, Piotr, Italiano, Giuseppe F., Kaplan, Haim, Myrvold, Wendy, Pruhs, Kirk, Randall, Dana, Rao, Satish, Shepherd, Bruce, Torng, Eric, Vempala, Santosh, Venkatasubramanian, Suresh, Vu, Van, and Wormald, Nick, *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2005)

Kargupta, Hilol, Srivastava, Jaideep, Kamath, Chandrika, and Goodman, Arnold, *Proceedings of the Fifth SIAM International Conference on Data Mining* (2005)

PROCEEDINGS OF THE FIFTH SIAM INTERNATIONAL CONFERENCE ON DATA MINING



Edited by
Hillol Kargupta
University of Maryland Baltimore County
Baltimore, Maryland

Jaideep Srivastava
University of Minnesota
Minneapolis, Minnesota

Chandrika Kamath
Lawrence Livermore National
Laboratory
Livermore, California

Arnold Goodman
University of California, Irvine
Irvine, California

siam

Society for Industrial and Applied Mathematics
Philadelphia

PROCEEDINGS OF THE FIFTH SIAM INTERNATIONAL CONFERENCE ON DATA MINING

Proceedings of the Fifth SIAM International Conference on Data Mining, Newport Beach, CA, April 21–23, 2005

Copyright © 2005 by the Society for Industrial and Applied Mathematics.

10 9 8 7 6 5 4 3 2 1

All rights reserved. Printed in the United States of America. No part of this book may be reproduced, stored, or transmitted in any manner without the written permission of the publisher. For information, write to the Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688.

Library of Congress Catalog Card Number: 2005923615

ISBN 0-89871-593-8

siam is a registered trademark.

xi	Message from the Conference Co-Chairs
xiii	Preface
1	Computational Developments of ψ -learning <i>Sijin Liu, Xiaotong Shen, and Wing Hung Wong</i>
12	A Random Walks Perspective on Maximizing Satisfaction and Profit <i>Matthew Brand</i>
20	Surveying Data for Patchy Structure <i>Ronald K. Pearson</i>
32	2-Dimensional Singular Value Decomposition for 2D Maps and Images <i>Chris Ding and Jieping Ye</i>
44	Summarizing and Mining Skewed Data Streams <i>Graham Cormode and S. Muthukrishnan</i>
56	Online Analysis of Community Evolution in Data Streams <i>Charu C. Aggarwal and Philip S. Yu</i>
68	Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window <i>Chih-Hsiang Lin, Ding-Ying Chiu, Yi-Hung Wu, and Arbee L. P. Chen</i>
80	On Abnormality Detection in Spuriously Populated Data Streams <i>Charu C. Aggarwal</i>
92	Privacy-Preserving Classification of Customer Data without Loss of Accuracy <i>Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright</i>
103	Privacy-Aware Market Basket Data Set Generation: A Feasible Approach for Inverse Frequent Set Mining <i>Xintao Wu, Ying Wu, Yongge Wang, and Yingjiu Li</i>
115	On Variable Constraints in Privacy Preserving Data Mining <i>Charu C. Aggarwal and Philip S. Yu</i>
126	Clustering with Model-Level Constraints <i>David Gondek, Shivakumar Vaithyanathan, and Ashutosh Garg</i>
138	Clustering with Constraints: Feasibility Issues and the k -Means Algorithm <i>Ian Davidson and S. S. Ravi</i>
150	A Cutting Algorithm for the Minimum Sum-of-Squared Error Clustering <i>Jiming Peng and Yu Xia</i>
161	Dynamic Classification of Defect Structures in Molecular Dynamics Simulation Data <i>Sameep Mehta, Steve Barr, Tat-Sang Choy, Hui Yang, Srinivasan Parthasarathy, Raghu Machiraju, and John Wilkins</i>
173	Striking Two Birds with One Stone: Simultaneous Mining of Positive and Negative Spatial Patterns <i>Bavani Arunasalam, Sanjay Chawla, and Pei Sun</i>
183	Finding Young Stellar Populations in Elliptical Galaxies from Independent Components of Optical Spectra <i>Ata Kabán, Louisa A. Nolan, and Somak Raychaudhury</i>
195	Hybrid Attribute Reduction for Classification Based on a Fuzzy Rough Set Technique <i>Qinghua Hu, Daren Yu, and Zongxia Xie</i>

205	HARMONY: Efficiently Mining the Best Rules for Classification <i>Jianyong Wang and George Karypis</i>
217	On Error Correlation and Accuracy of Nearest Neighbor Ensemble Classifiers <i>Carlotta Domeniconi and Bojun Yan</i>
227	Lazy Learning for Classification Based on Query Projections <i>Yiqiu Han and Wai Lam</i>
239	Mining Non-derivable Association Rules <i>Bart Goethals, Juho Muhonen, and Hannu Toivonen</i>
250	Depth-First Non-derivable Itemset Mining <i>Toon Calders and Bart Goethals</i>
262	Exploiting Relationships for Domain-Independent Data Cleaning <i>Dmitri V. Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen</i>
274	A Spectral Clustering Approach to Finding Communities in Graphs <i>Scott White and Padhraic Smyth</i>
286	Mining Behavior Graphs for "Backtrace" of Noncrashing Bugs <i>Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S. Yu</i>
298	Learning to Refine Ontology for a New Web Site Using a Bayesian Approach <i>Tak-Lam Wong and Wai Lam</i>
310	Exploiting Parameter Related Domain Knowledge for Learning in Graphical Models <i>Radu S. Niculescu, Tom M. Mitchell, and R. Bharat Rao</i>
322	Exploiting Geometry for Support Vector Machine Indexing <i>Navneet Panda and Edward Y. Chang</i>
334	Parallel Computation of RBF Kernels for Support Vector Classifiers <i>Shibin Qiu and Terran Lane</i>
346	Loadstar: A Load Shedding Scheme for Classifying Data Streams <i>Yun Chi, Philip S. Yu, Haixun Wang, and Richard R. Muntz</i>
358	Topic-Driven Clustering for Document Datasets <i>Ying Zhao and George Karypis</i>
370	Variational Learning for Noisy-OR Component Analysis <i>Tomas Singliar and Milos Hauskrecht</i>
380	Summarizing Sequential Data with Closed Partial Orders <i>Gemma Casas-Garriga</i>
392	SUMSRM: A New Statistic for the Structural Break Detection in Time Series <i>Kwok Pan Pang and Kai Ming Ting</i>
404	Markov Models for Identification of Significant Episodes <i>Robert Gwadera, Mikhail Atallah, and Wojciech Szpankowski</i>
415	Efficient Mining of Maximal Sequential Patterns Using Multiple Samples <i>Congnan Luo and Soon M. Chung</i>
427	Gaussian Processes for Active Data Mining of Spatial Aggregates <i>Naren Ramakrishnan, Chris Bailey-Kellogg, Satish Tadepalli, and Varun N. Pandey</i>
439	Correlation Clustering for Learning Mixtures of Canonical Correlation Models <i>Xiaoli Z. Fern, Carla E. Brodley, and Mark A. Friedl</i>
449	On Periodicity Detection and Structural Periodic Similarity <i>Michail Vlachos, Philip Yu, and Vittorio Castelli</i>
461	Cross Table Cubing: Mining Iceberg Cubes from Data Warehouses <i>Moonjung Cho, Jian Pei, and David W. Cheung</i>

466	Decision Tree Induction in High Dimensional, Hierarchically Distributed Databases <i>Amir Bar-Or, Assaf Schuster, Ran Wolff, and Daniel Keren</i>
471	Slope One Predictors for Online Rating-Based Collaborative Filtering <i>Daniel Lemire and Anna Maclachlan</i>
476	Sparse Fisher Discriminant Analysis for Computer Aided Detection <i>M. Murat Dundar, Glenn Fung, Jinbo Bi, Sandilya Sathyakama, and Bharat Rao</i>
481	Expanding the Training Data Space Using Emerging Patterns and Genetic Methods <i>Hamad Alhammady and Kotagiri Ramamohanarao</i>
486	Making Data Mining Models Useful to Model Non-paying Customers of Exchange Carriers <i>Wei Fan, Janak Mathuria, and Chang-tien Lu</i>
491	Matrix Condition Number Prediction with SVM Regression and Feature Selection <i>Shuting Xu and Jun Zhang</i>
496	Cluster Validity Analysis of Alternative Results from Multi-objective Optimization <i>Yimin Liu, Tansel Özyer, Reda Alhajj, and Ken Barker</i>
501	ClosedPROWL: Efficient Mining of Closed Frequent Continuities by Projected Window List Technology <i>Kuo-Yu Huang, Chia-Hui Chang, and Kuo-Zui Lin</i>
506	Three Myths about Dynamic Time Warping Data Mining <i>Chotirat Ann Ratanamahatana and Eamonn Keogh</i>
511	PCA without Eigenvalue Calculations: A Case Study on Face Recognition <i>E. Kokiopoulou and Y. Saad</i>
516	Mining Top-K Itemsets over a Sliding Window Based on Zipfian Distribution <i>Raymond Chi-Wing Wong and Ada Wai-Chee Fu</i>
521	Hierarchical Document Classification Using Automatically Generated Hierarchy <i>Tao Li and Shenghuo Zhu</i>
526	On Clustering Binary Data <i>Tao Li and Shenghuo Zhu</i>
531	Time-Series Bitmaps: A Practical Visualization Tool for Working with Large Time Series Databases <i>Nitin Kumar, Venkata Nishanth Lolla, Eamonn Keogh, Stefano Lonardi, Chotirat Ann Ratanamahatana, and Li Wei</i>
536	Pushing Feature Selection Ahead of Join <i>Rong She, Ke Wang, Yabo Xu, and Phillip S. Yu</i>
541	Discarding Insignificant Rules During Impact Rule Discovery in Large, Dense Databases <i>Shiying Huang and Geoffrey I. Webb</i>
546	SPID4.7: Discretization Using Successive Pseudo Deletion at Maximum Information Gain Boundary Points <i>Somnath Pal and Himika Biswas</i>
551	Iterative Mining for Rules with Constrained Antecedents <i>Zheng Sun, Phillip S. Yu, and Xiang-Yang Li</i>
556	Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach <i>Al Mamunur Rashid, George Karypis, and John Riedl</i>
561	Mining Unconnected Patterns in Workflows <i>Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, and Domenico Saccà</i>
566	The Best Nurturers in Computer Science Research <i>Bharath Kumar M. and Y. N. Srikant</i>

571	Knowledge Discovery from Heterogeneous Dynamic Systems Using Change-Point Correlations <i>Tsuyoshi Idé and Keisuke Inoue</i>
576	Building Decision Trees on Records Linked through Key References <i>Ke Wang, Yabo Xu, Philip S. Yu, and Rong She</i>
581	Efficient Allocation of Marketing Resources Using Dynamic Programming <i>Giuliano Tirenni, Abderrahim Labbi, André Elisseeff, and Cèsar Berrospi</i>
586	Near-Neighbor Search in Pattern Distance Spaces <i>Haixun Wang, Chang-Shing Perng, and Philip S. Yu</i>
591	An Algorithm for Lattice-Structured Subspace Clusters <i>Haiyun Bian and Raj Bhatnagar</i>
596	CBS: A New Classification Method by Using Sequential Patterns <i>Vincent S. M. Tseng and Chao-Hui Lee</i>
601	SeqIndex: Indexing Sequences by Sequential Pattern Analysis <i>Hong Cheng, Xifeng Yan, and Jiawei Han</i>
606	On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering <i>Chris Ding, Xiaofeng He, and Horst D. Simon</i>
611	Kronecker Factorization for Speeding up Kernel Machines <i>Gang Wu, Zhihua Zhang, and Edward Chang</i>
616	Symmetric Statistical Translation Models for Automatic Image Annotation <i>Feng Kang and Rong Jin</i>
621	Correcting Sampling Bias in Structural Genomics through Iterative Selection of Underrepresented Targets <i>Kang Peng, Slobodan Vucetic, and Zoran Obradovic</i>
626	Statistical Models for Unequally Spaced Time Series <i>Emre Erdogan, Sheng Ma, Alina Beygelzimer, and Irina Rish</i>
631	CLSI: A Flexible Approximation Scheme from Clustered Term-Document Matrices <i>Dimitrios Zeimpekis and Efstratios Gallopoulos</i>
636	WFIM: Weighted Frequent Itemset Mining with a Weight Range and a Minimum Weight <i>Unil Yun and John J. Leggett</i>
641	Model-Based Clustering with Probabilistic Constraints <i>Martin H. C. Law, Alexander Topchy, and Anil K. Jain</i>
647	Author Index

MESSAGE FROM THE CONFERENCE CO-CHAIRS

The Fifth SIAM International Conference on Data Mining continues the tradition of providing an open forum for the presentation and discussion of innovative algorithms as well as novel applications of data mining. This is reflected in the talks by the four keynote speakers, who will discuss a diverse set of topics from models for customer-based analysis (Peter S. Fader), through embedded sensor networks (Mark Hansen) and the practice of cluster analysis (Jon R. Kettenring), to visual data mining (Edward J. Wegman).

In addition to the keynote talks, the conference also features two tutorials and six workshops on a range of subjects. The tutorials will provide the participants an in-depth exposition on segmentation algorithms for time series data and pattern discovery in biosequences. The workshops are a forum for discussing new ideas, brainstorming on work in progress, and identifying new algorithms and application areas for data mining. The workshops this year include several traditional favorites as well as a couple of new ones. The topics of the workshops are data mining in sensor networks; feature selection for data mining; clustering high-dimensional data and its applications; link analysis, counter-terrorism, and security; high-performance and distributed mining; and mining scientific and engineering datasets. These workshops and tutorials, in addition to the papers and the poster session, provide an exciting environment in which the participants can interact with each other.

This year, we also have two special sessions. The first, on industrial and government applications, is being organized by Mehran Sahami and Ashok Srivastava; it builds on the success of a similar session last year. The second, organized by Amy Braverman and Michael Turmon, is on statistics and data mining; it underlines the connections between the two fields and provides a venue for the practitioners of each to exchange ideas.

We would like to thank the entire organizing committee for the terrific job they have done in putting together a strong technical program: Hillol Kargupta and Jaideep Srivastava for assembling a well-rounded program committee and for overseeing the paper selection process; Philip Yu and his team for selecting the best papers; Eamonn Keogh for soliciting and assembling a top-notch tutorial program; Bing Liu and Ke Wang for selecting workshops on a diverse range of subjects, all of current interest; Osmar Zaiane for identifying sponsors for the conference; and finally, the international publicity team of Philip Chan, Daniel Keim, and Kyuseok Shim for their tireless efforts in publicizing the conference.

We would like to acknowledge our sponsor, the Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory, for its generous support, in particular the funding of student travel grants. This conference is being co-sponsored by the American Statistical Association; we hope that this will lead to greater collaboration between the two communities.

Finally, we thank the authors and the participants who are the primary reason for the success of the conference. We hope you all enjoy the conference!

Chandrika Kamath and Arnold Goodman, Conference Co-Chairs

Vipin Kumar, Steering Committee Chair

We are pleased to present the proceedings of the 2005 SIAM International Conference on Data Mining. The pervasiveness of data mining in fundamental research and applications continues to grow, and we are pleased to witness the growing contribution of this conference in further development of this field. We are excited to have a record number of paper submissions (218) this year as well as a record number of program committee members (94). Both of these are a testament to the growing importance of the SIAM series of conferences as the preferred venue for publishing exciting new results in the area. In addition, this year we had nine vice chairs for facilitating the review process. We hope that the research and experiences captured in these proceedings are insightful to both expert and novice users and practitioners of data mining approaches.

We received 218 paper submissions from 19 countries. Each submitted paper was reviewed by at least three members of the program committee. The reviewing period was followed by a discussion phase. Finally 40 papers (18.3%) were selected to appear in the program as regular papers, and another 37 papers (17%) were accepted as poster presentations. Full papers received 12 pages, and poster papers received five pages in the proceedings.

We would like to thank our program committee, consisting of high visibility researchers, whose dedication and diligence made the selection of papers for these proceedings possible. Special thanks go to the Vice Chairs: Chid Apte, Joydeep Ghosh, Diane Lambert, Lim Ee Peng, Sanjay Ranka, Jude Shavlik, Domenico Talia, Ramaswamy Uthrusamy, and Rebecca Wright. They brought their expertise into handling the review and discussion of papers in their respective areas, and helped us decide the final program.

We are grateful to Microsoft Corporation for providing the Conference Management Tool (CMT) that facilitated the collection and management of paper submissions. Special thanks to Chani Johnson for his help with the CMT and for troubleshooting when needed. We also thank the staff at SIAM (particularly Darrell Ross, Laura Helfrich, and Simon Dickey) for their help in the production of these proceedings and in all the necessary arrangements for the conference.

Of course, this conference would not be possible without the excellent papers and presentations represented by these proceedings. We thank all the authors for their participation in SDM 2005!

Hillol Kargupta and Jaideep Srivastava, Program Co-Chairs

Computational developments of ψ -learning*

Sijin Liu
Department of Statistics
The Ohio State University
Columbus, OH 43210

Xiaotong Shen
School of Statistics
University of Minnesota
Minneapolis, MN 55455

Wing Hung Wong
Department of Statistics
Stanford University
Stanford, CA 94305

Summary

One central problem in science and engineering is predicting unseen outcome via relevant knowledge gained from data, where accuracy of generalization is the key. In the context of classification, we argue that higher generalization accuracy is achievable via ψ -learning, when a certain class of non-convex rather than convex cost functions are employed. To deliver attainable higher generalization accuracy, we propose two computational strategies via a global optimization technique—difference convex programming, which relies on a decomposition of the cost function into a difference of two convex functions. The first strategy solves sequential quadratic programs. The second strategy, combining this with the method of Branch-and-Bound, is more computationally intensive but is capable of producing global optima. Numerical experiments suggest that the algorithms realize the desired generalization ability of ψ -learning.

* Shen's research is supported in part by NSF Grant IIS-0328802. Wong's research is supported by a grant from NSF-DMS.

Key words: DC programming, global optimization, sequential quadratic programming, support vectors.

1 Introduction

How does one accurately predict unseen outcome using relevant information? The simplest problem of this kind is classification. In statistical and computer sciences, classification has been particularly vital. Margin-based classification techniques are at the core of progress. Essentially all these techniques construct a classifier by minimizing a convex cost function. Examples include support

vector machines (SVM, Boser, Guyon and Vapnik, 1992; Cortes and Vapnik, 1995), import vector machines (Zhu and Hastie, 2004), among others.

Key issues: Convex vs Non-convex. While the margin-based techniques have proven effective and have achieved state-of-the-art performance, the recent work of Shen, Tseng, Zhang, and Wong (2003) suggests that substantial higher generalization accuracy can be achieved if one steps out from the paradigm of convexity; see also Lin (2000, 2002). Specifically, they constructed a certain class of ψ cost functions and showed that ψ -learning realizes sharp generalization error rates in some examples. The rationale behind their methodology is that classification is non-convex in nature and ultimately should be treated via non-convex cost functions. Indeed, any convex cost function is generally bound to suffer a loss in generalization accuracy as the price for easing computation (Fung and Mangasarian, 2000). An important practical issue then is how to meet the computational challenge of non-convex minimization.

Global optimization. On the basis of recent advances in global optimization, we develop computational tools for ψ -learning. This allows us to realize the generalization ability of ψ -learning in practice. The key ingredient of our proposed methods is difference convex (DC) programming, which uses a decomposition of the cost function into a difference of two convex functions. In our decomposition, the leading convex function is an equivalent SVM cost function, while the trailing one can be thought of as a correction to the SVM cost function so that the result is closer to the true generalization error. With this decomposition, a sequence of monotone approximations to the SVM cost function are constructed, and a sequence of quadratic programs are solved to yield an approximate

solution. We develop an efficient algorithm and refer this as SQP. This algorithm can be further enhanced by combining with the branch-and-bound (BB) search to obtain a provable convergence to global optima, which we call SQP-BB.

Numerical experiments suggest SQP performs well for large problems, whose termination requires only a small number of iterations, for instance, 4-7 iterations would be common. We use SQP-BB to check globality of the solutions of SQP in some examples. The result indicates that SQP yields global optima with high likelihood of occurrence.

Six benchmark examples are examined using SQP-BB. Computational results demonstrate that the significant generalization advantage of ψ -learning is realized by the computational tools developed here. For every single example, ψ -learning yields higher generalization accuracy than SVM.

2 ψ -learning

From a statistical perspective, training data $(x_i, y_i)_{i=1}^n$ are sampled from a true yet unknown probability distribution, with $y_i = \pm 1$ in binary classification.

Linear classification uses hyperplanes $f(\tilde{x}) = \langle w, \tilde{x} \rangle$ as decision functions, with $\tilde{x} = (x, 1)$ and $w = (w^*, w_{d+1}) = (w_1, \dots, w_d, w_{d+1}) \in \mathcal{R}^{d+1}$. Here $\langle \cdot, \cdot \rangle$ represents the inner product in the corresponding Euclidean space. The basic form of linear SVM, originated from the optimal separating hyper-plane in the separable case, minimizes a convex cost function: $s(w) = \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi_{svm}(y_i f(\tilde{x}_i))$, where $\psi_{svm}(z) = (1 - z)_+$ is the hinge loss, and z_+ represents the positive part of z . Instead of using ψ_{svm} , linear ψ -learning seeks w to minimize

$$s(w) = \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi(y_i f(\tilde{x}_i)), \quad (1)$$

where $C > 0$ controls the balance between the margin and training, and $\frac{2}{\|w^*\|^2}$ is the geometric margin in the separable case. Here ψ is required to satisfy the property:

$$\begin{aligned} U \geq \psi(z) &> 0 && \text{if } z \in (0, \tau] \\ \psi(z) &= U(1 - \text{Sign}(z)) && \text{otherwise,} \end{aligned} \quad (2)$$

where $0 < \tau \leq 1$ and $U > 0$ are some constants.

In implementation, a specific choice of ψ should be chosen depending on one's optimization strategy. In what follows, we shall use a ψ function, defined as $\psi(z) = 0$ if $z \geq 1$, $\psi(z) = 2(1 - z)$ if $0 \leq z \leq 1$, and 2 otherwise. This ψ function, as displayed in Figure 2, has the desirable DC property in that it has a DC representation. This property is the key to develop efficient computational algorithms. Since no differentiability is used in our algorithms, there is no obvious advantage of applying a smooth version of ψ . See Shen et. al (2003) for some discussions with regard to the choice of ψ .

Nonlinear classification uses flexible representations, with $f(\tilde{x}) = \langle \tilde{x}, w \rangle$, $w = (w^*, w_{n+1}) = (w_1, \dots, w_{n+1}) \in \mathcal{R}^{n+1}$, and $\tilde{x} = (K(x, x_1), \dots, K(x, x_n), 1)$, defined by kernel $K(\cdot, \cdot)$ mapping from $S \times S$ to \mathcal{R} . The kernel is required to satisfy Mercer's condition, which assures that $\|w^*\| = (\sum_{i=1}^n \sum_{j=1}^n w_i w_j K(x_i, x_j))^{1/2}$ is a proper norm. The theory of reproducing kernel Hilbert space (RKHS), c.f., Wahba (1990, 1999), is useful to construct such a kernel. Then the cost function of nonlinear ψ -learning becomes

$$s(w) = \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi(y_i f(\tilde{x}_i)). \quad (3)$$

In the sequel, we shall adopt a generic form $f(\tilde{x}) = \langle \tilde{x}, w \rangle$ and the norm $\|w^*\| = \langle w^*, w^* \rangle$ to represent both the linear and nonlinear cases with w being respectively $d+1$ and $n+1$ dimensional. The estimated decision function of ψ -learning is then $\hat{f}(\tilde{x}) = \langle \hat{w}, \tilde{x} \rangle$, where \hat{w} is the minimizer of (1) or (3).

3 Non-convex minimization

For high-dimensional non-convex minimization, there is generally no efficient method to compute global optima. Figure 1 illustrates the level of difficulty of optimization in (1) and (3). Fortunately, by exploiting the DC property of the ψ -function, we are able to develop efficient algorithms.

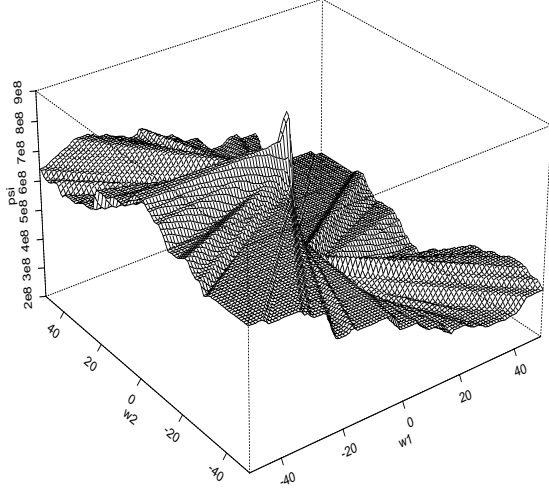


Figure 1: Perspective plot of s as a function (w_1, w_2) and $w_3 = 0.25$ for the example described in Section 4 with $n = 50$, $C = 10^7$ and 20% flipping.

3.1 DC decompositions

There have been major advances in computation of global optima when an objective function has a DC representation (An and Tao, 1997). Such a decomposition plays an **extremely critical** role in determining the speed of convergence, stability, robustness, and globality of sought solutions. For our problem, we utilize the problem structure and decompose our cost function s in (1) or (3) into:

$$s(w) = s_1(w) - s_2(w), \quad (4)$$

where $s_1 = \frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \psi_1(y_i f(\tilde{x}_i))$ and $s_2 = C \sum_{i=1}^n \psi_2(y_i f(\tilde{x}_i))$ are both convex in w . This decomposition is obtained from a DC decomposition of $\psi(z) = \psi_1(z) - \psi_2(z)$, where $\psi_1(z)$ is 0 if $z \geq 1$ and $-2(z - 1)$ otherwise; $\psi_2(z)$ is 0 if $z \geq 0$ and $-2z$ otherwise. The plot of this decomposition is given in Figure 2. Note that s_1 is equivalent to the SVM cost function induced by twice of the hinge loss. Further, since the true generalization error is defined by a bounded loss function $1 - \text{Sign}$, the unbounded nature of the SVM cost function introduces obvious bias when it is used to estimate

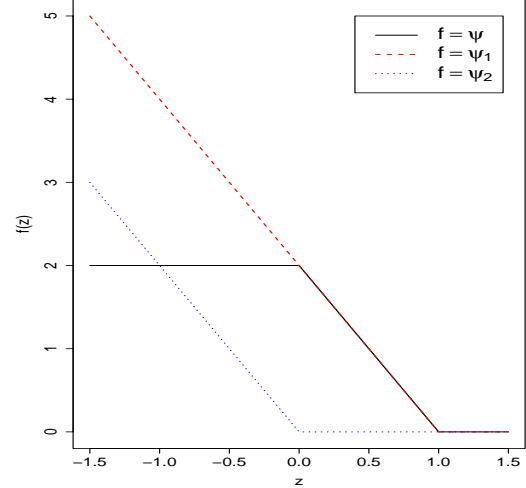


Figure 2: Plot of functions ψ_1 , ψ_2 and ψ , where $\psi = \psi_1 - \psi_2$ is a DC decomposition of ψ .

the generalization error. In light of this, we interpret s_2 in (4) as a bias correction to the SVM cost function.

3.2 Differenced Convex Algorithms

Differenced convex algorithm (DCA) is among the rare algorithms which allow to solve large-scale non-convex minimization problems. As shown in An and Tao (1997), when a DC decomposition is available, DCA constructs non-increasing upper envelopes of s , which yield sequential convex subproblems. This permits developing efficient algorithms for ψ -learning, especially so for large-scale problems.

There are basically two versions of DCA, regular and simplified, and we apply the simplified DCA. When (4) is given, the simplified DCA solves a sequence of primal and dual subproblems. It proceeds with construction of two sequences $(w^{(k)}, y^{(k)})$ iteratively. Given $(w^{(k)}, y^{(k)})$, the k th primal subproblem is $s_1(w) - s_2(w^{(k)}) - \langle w - w^{(k)}, y^{(k)} \rangle$, obtained by replacing s_2 by its affine minorization function $s_2(w^{(k)}) + \langle w - w^{(k)}, y^{(k)} \rangle$. Minimizing it with respect to w yields $w^{(k+1)}$. Similarly, $y^{(k+1)}$ is the minimizer of the k th dual subproblem after obtaining $w^{(k+1)}$, which amounts to selecting a suitable sub-

gradient of s_2 at $w^{(k)}$. By convexity, these subproblems provide a sequence of non-increasing upper approximations to the original problem, leading to convergence of $w^{(k)}$.

In our case, we derive a subgradient of s_2 at $w^{(k)}$ without solving the dual problem. Specifically, this subgradient $\nabla s_2(w^{(k)})$ is defined as $(V_1^{(k)}, V_2^{(k)})$, where $V_1^{(k)}$ is $C \sum_{i=1}^n \nabla \psi_2(y_i f^{(k)}(\tilde{x}_i)) y_i x_i$, $V_2^{(k)}$ is $C \sum_{i=1}^n \nabla \psi_2(y_i f^{(k)}(\tilde{x}_i)) y_i$, and $f^{(k)}(\tilde{x}_i)$ is $\langle w^{(k)}, \tilde{x}_i \rangle$. Here $\nabla \psi_2(z) = 0$ if $z > 0$ and $\nabla \psi_2(z) = -2$ otherwise.

Our algorithm solves a sequence of subproblems. At iteration k , only the primal subproblem is required to solve, which is equivalent to

$$\min_w (s_1(w) - \langle w, \nabla s_2(w^{(k)}) \rangle). \quad (5)$$

This problem can be solved via quadratic programming (QP). By Kuhn-Tucker (KKT)'s condition, it is equivalent to the dual QP in Theorems 1 and 2.

Theorem 1: (Linear) The k th dual subproblem of (5) with $\alpha = (\alpha_1, \dots, \alpha_n)$ is

$$\begin{aligned} \max_{\alpha} W(\alpha) = & \sum_{i=1}^n \alpha_i [1 - y_i \langle V_1^{(k)}, x_i \rangle] \\ & - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \end{aligned} \quad (6)$$

subject to $\sum_{i=1}^n \alpha_i y_i = -V_2^{(k)}$, $2C \geq \alpha_i \geq 0$; $i = 1, \dots, n$. Then the solution of (5) $(w_1^{(k+1)}, \dots, w_d^{(k+1)})$ is $V_1 + \sum_{i=1}^n \alpha_i^{(k)} y_i x_i$, $w_{d+1}^{(k+1)}$ satisfies KKT's condition: $y_i \langle w^{(k+1)}, \tilde{x}_i \rangle = 1$ for any i with $2C > \alpha_i^{(k)} > 0$. Here $\{\alpha_i^{(k)}\}_{i=1}^n$ is the solution of (6).

Theorem 2: (Nonlinear) The dual subproblem of (5) is (6) with $\langle x_i, x_j \rangle$ being replaced by $K(x_i, x_j)$ and $\langle V_1^{(k)}, x_i \rangle$ being replaced by $C \sum_{j=1}^n \psi_2'(y_j f^{(k)}(\tilde{x}_j)) y_j K(x_i, x_j)$. The solution $\{\alpha_i^{(k)}\}_{i=1}^n$ yields that of (5) $w_j^{(k+1)} = y_j \langle \alpha_j^{(k)} + C \nabla \psi_2(y_j f^{(k)}(\tilde{x}_j)) \rangle$; $j = 1, \dots, n$, and $w_{n+1}^{(k+1)}$ satisfies KKT's condition: $y_i \langle w^{(k+1)}, \tilde{x}_i \rangle = 1$ for any i with $2C > \alpha_i^{(k)} > 0$.

Algorithm 1: (SQP, Linear and Nonlinear)

Step 1: (Initialization) Specify initial value $w^{(0)}$ and tolerance error $\varepsilon > 0$.

Step 2: (Iteration) At iteration k , compute $w^{(k+1)}$ by solving (6).

Step 3: (Stopping rule) Stop if $|s(w^{(k+1)}) - s(w^{(k)})| \leq \varepsilon$. Then the final solution w^s is $w^{(k+1)}$, which yields $\hat{f}(\tilde{x}) = \langle w^s, \tilde{x} \rangle$ for (1) or (3).

Our numerical experience suggests that a good initial value enhances the chance of DCA to locate global optima. For both Algorithms 1 and 2, we recommend to use a SVM solution or any point with a smaller cost function value.

Two important features are built into SQP to guard against potential numerical problems and enhance its stability. First, linear programming (LP) is employed for w_{d+1} or w_{n+1} when there are no instances $2C > \hat{\alpha}_i > 0$ such that they can be determined by KKT's condition. Specifically, minimize (5) with respect to w_{d+1} or w_{n+1} via LP after substituting the values of $\{\alpha_i^{(k)}\}_{i=1}^n$ in $f^{(k+1)}$. Second, a regularization technique is applied to replace the leading matrix K in QP by $K + \rho I$ for small $\rho > 0$ when it becomes ill-posed, although K is supposed to be positive definite. This regularization technique is equivalent to a different DC decomposition: $s = (s_1 + \rho \|\cdot\|^2) - (s_2 + \rho \|\cdot\|^2)$, for some $\rho > 0$ for improving the strength of convexity of the decomposition.

Theorem 3: (Convergence) The sequence $s(w^{(k)})$ is non-increasing, $\lim_{k \rightarrow \infty} s(w^{(k)}) \geq \min_w s(w)$, and $\lim_{k \rightarrow \infty} \|w^{(k+1)} - w^{(\infty)}\| = 0$ for some $w^{(\infty)}$. Moreover, convergence of SQP is superlinear in that $\lim_{k \rightarrow \infty} \|w^{(k+1)} - w^{(\infty)}\| / \|w^{(k)} - w^{(\infty)}\| = 0$ provided that there does not exist an instance x^* on the decision boundary such that $f^{(\infty)}(x^*) = \langle w^{(\infty)}, \tilde{x}^* \rangle = 0$.

As shown in Theorem 3, SQP converges superlinear in that the number of iterations required for it to achieve precision ε is $o(\log(1/\varepsilon))$. Based on our numerical experience, it normally terminates in 4-10 steps. The computational complexity of $o(\log(1/\varepsilon))$ multiplied by that of QP, which is usually $O(n^3)$.

An improvement over SVM in generalization usually occurs even when global optima have not been reached by SQP; see Tables 1 and 2. This is mainly because s_2 corrects the bias due to imposed convexity to s_1 in (4). This aspect has been confirmed by our numerical experi-

ence.

3.3 DCA and Branch-and-Bound

The method of BB can be used to globally solve minimization in (1) and (3). When it is suitably combined with SQP, it leads to a promising global minimization routine, which can substantially improve efficiency of BB and enhance globality of DCA. In what follows, we shall derive such an algorithm.

BB is composed of two critical operations: bounding and subdivision. The bounding operation constructs both upper and lower bounds of s , while the subdivision operation divides regions. A combination of both exclude infeasible regions and determine optimality of a solution. Because SQP tends to yield a sharp upper bound when it does not give global optima, convergence of BB expedites.

For the bounding operation, we obtain a good upper bound via the solution w^s of SQP in that $s(w^s) \geq \min_w s(w)$. To construct a good lower bound, we first construct a tight convex envelop of the concave function $-s_2$ in (4) over a simplex S using a result of Falk and Hooeman (1976). Specifically in the linear case, let $v_j \in V(S)$; $j = 1, \dots, d+2$, be a collection of vertices of S . The convex envelop $l_s(w)$ of $-s_2(w)$ is $\langle a, w \rangle + a_{d+2}$, obtained by solving a linear system of $(d+2)$ equations with respect to $(a = (a_1, \dots, a_{d+1}, a_{d+2})$: $l_s(v_j) = -s_2(v_j)$; $j = 1, \dots, d+2$. By concavity, $-s_2(w) \geq l_s(w)$ for $w \in S$. Second, solve a quadratic problem:

$$L(S) = \min_{w \in S} (s_1(w) + l_s(w)), \quad (7)$$

yielding a lower bound $L(S)$, which is equivalent to solving the following problem.

Theorem 4: (Lower bound, Linear) The dual problem of (7) is

$$\min_z W(z) = \frac{1}{2} z^T H z - g^T z \quad (8)$$

subject to $2C \geq \alpha_i \geq 0$; $i = 1, 2, \dots, n$, $\beta \geq 0$, $\theta \geq 0$, $t \geq 0$, and $\sum_{i=1}^n \alpha_i y_i + \beta^T A_2 + \theta A_4 - t a_6 = a_2$. The solution $(\tilde{w}_1, \dots, \tilde{w}_d)$ of (7) is $\sum_{i=1}^n \hat{\alpha}_i y_i x_i + A_1^T \hat{\beta} + \hat{\theta} A_3^T - a_1 - \hat{t} a_5$, and \tilde{w}_{d+1} is chosen so that $y_i \langle \tilde{w}, \tilde{x}_i \rangle = 1$ for any i with $2C > \hat{\alpha}_i > 0$. Here $(\hat{\alpha}_1, \dots, \hat{\alpha}_n, \hat{\beta}, \hat{\theta}, \hat{t})$

is the solution of (8), and H, g, A_i ; $i = 1, \dots, 4$, a_i ; $i = 1, \dots, 6$, are defined in the Appendix.

For the subdivision operation, we use a simplicial subdivision, and combine bisection via the longest edge with the radial partition, c.f., Horst and Tuy (1989). An adaptive partition is possible but will not be studied in here. A simplex subdivision divides S into $d+2$ subsimplices $\{S_i\}_{i=1}^{d+2}$, and replaces each vertex v_i of S by a subdivision point v , which is the average of the vertices of S for the radial partition and is the middle point of the longest edge for the longest edge partition. Note that the latter partition only generates two non-degenerate subsimplices.

Our subdivision rule first uses the radial partition, then switches to bisection via the longest edge once the lower bounds become sufficiently good. This allows us to combine the advantages of the both partitions to enhance computational efficiency. Note that the first yields good lower bounds due to division of smaller subsimplices but does not assure convergence, while the second one assures convergence but is inefficient with computation of lower bounds for a large simplex. As a result, computational efficiency is enhanced. In the partition process, switching takes place for S if one of three conditions is met: 1) Five consecutive partitions have been completed, 2) $L(S)$ becomes positive, and 3) the shortest edge of S is less than 10. Here 3) is for numerical stability.

To define an initial simplex S_0 centered at $w^{(0)} = (w_1^{(0)}, \dots, w_{d+1}^{(0)})$, we first construct a cube centered at $w^{(0)}$ with half width h . Let $\alpha_j = w_j^{(0)} - h$; $j = 1, \dots, d+1$ and $\alpha = \sum_{j=1}^{d+1} w_j^{(0)} + (d+1)h$. This defines the smallest simplex containing the cube: $S_0 = \{w \in R^{d+1} : \alpha_i - w_j \leq 0, \sum_{j=1}^{d+1} w_j - \alpha \leq 0\}$ with vertices $v_0 = (\alpha_1, \dots, \alpha_{d+1})$ and $v_j = (\alpha_1, \dots, \alpha_{j-1}, \alpha - \sum_{i \neq j} \alpha_i, \alpha_{j+1}, \dots, \alpha_{d+1})$; $j = 1, \dots, d+1$.

Let R_k , $\epsilon > 0$, w^s , and N be a collection of feasible regions at iteration k , prespecified tolerance error, the final solution, and the upper bound of the numbers of iterations, respectively. Here N and h control run time with suitable choice of N and h yielding global optima while expedite convergence.

Algorithm 2:(SQP-BB)

Step 1: (Initialization) Specify $w^{(0)}$, h , N , and ϵ . With $w^{(0)}$ the current best feasible point, compute upper bound $U(S_0)$ obtained via the solution of SQP with initial value as the solution from lower bound $L(S_0)$ via the solution of

(7) via (8) with $S = S_0$. If $U(S_0) - L(S_0) \leq \varepsilon(L(S_0) + 1)$, then $R_0 = \emptyset$ and terminate; otherwise $R_0 = S_0$.

Iteration $k = 0, \dots$

Step 2: (Check optimality) If $R_k = \emptyset$, then terminate with $w^s = w^{(k)}$.

Step 3: (Selection) If $R_k \neq \emptyset$, then select $S_k \in R_k$ such that $U(S_k) = \min\{U(S) : S \in R_k\}$.

Step 4: (Division) For each $S_{kj}; j = 1, \dots, j_k$, apply the subdivision rule to divide S_k into subsimplices S_{kj} and compute $L(S_{kj})$ via (8) with $S = S_{kj}$.

Step 5: (Updating and elimination) Apply SQP to S_{kj} using the lower bound solution as an initial value and compute $U(S_{kj}); j = 1, \dots, j_k$. Update the current best feasible point $w^{(k+1)}$ and the best upper bound $\alpha_{k+1} = \min\{U(S_{kj}) : j = 1, \dots, j_k\}$ at iteration k . Let $R_{k+1} = \{S \in \Delta_{k+1} : \alpha_{k+1} - L(S) > \varepsilon(L(S) + 1)\}$, where $\Delta_{k+1} = (R_k \setminus S_k) \cup \{S_{kj} : j = 1, \dots, j_k\}$.

Theorem 5: (Convergence) The sequence $w^{(k)}$ converges to the global minimizer, i.e., $\lim_{k \rightarrow \infty} s(w^{(k+1)}) = \min_w s(w)$. Moreover, SQP-BB terminates finitely with precision $\varepsilon > 0$ in that $|s(w^s) - \min_w s(w)| \leq \varepsilon$.

Theorem 5 says that w^s is an ε -global minimizer. Our numerical experience suggests that SQP-BB usually converges reasonably fast but more slowly than SQP. Computational complexity of SQP-BB is roughly $o(\log(1/\varepsilon)Nn^3)$, with an upper bound of N being of order of $1/\varepsilon$. Note that SQP is a special case of SQP-BB when $h = \infty$ and $N = 0$.

As shown in Table 1, even if N is set to be small, it usually yields better solution than SQP. The computational cost of SQP-BB for linear problems is acceptable, while SQP with a good initial value is recommended for large problems. Interestingly, SQP-BB is parallelizable, permitting fast computation.

4 Numerical Analysis

The following numerical example examines the effectiveness of SQP and SQP-BB in terms of speed of convergence and globality of sought solutions. Here the QP and LP involved in SQP and SQP-BB are implemented via the IMSL QP and LP routines.

Consider a two-dimensional linear example of Shen et al. (2003), in which $f(x) = \sum_{i=1}^2 w_i x_i + w_3$. A random

Table 1: Globality of SQP and SQP-BB(N) in percent over 100 simulation replications as well as the average number of iterations for SQP.

C=1				
Flip	SQP	Ave # iter	SQP-BB (N=100)	SQP-BB (N=200)
0%	95%	2.43	100%	100%
10%	92%	2.82	100%	100%
20%	83%	2.85	99%	100%
C=10 ³				
Flip	SQP	Ave # iter	SQP-BB (N=100)	SQP-BB (N=200)
0%	99%	1.20	99%	99%
10%	46%	2.07	95%	97%
20%	29%	2.10	85%	93%

training sample $\{X_{i1}, X_{i2}, Y_i\}_{i=1}^n$ is generated as follows. First, $(X_{i1}, X_{i2})_{i=1}^n$ are sampled from the uniform distribution over the unit disk $\{(x_1, x_2) : x_1^2 + x_2^2 \leq 1\}$, and Y_i is assigned to 1 if $X_{i1} \geq 0$ and -1 otherwise. Then randomly selected labels $\{Y_i\}_{i=1}^n$ are flipped, which generates a random sample for non-separable cases.

Three levels of contamination are considered: 0-flip, 10%-flip and 20%-flip, each with two different values $C = 1, 10^3$. In each case, the percents of time for SQP to yield global optima based on 100 simulation replications are reported in Table 1. The globality of solutions of SQP is determined by its agreement with the solutions of SQP-BB(N= ∞), ignoring numerical rounding error that is less than $\varepsilon > 0$ used by SQP-BB. In the simulations, ε is set to be $10^{-3}C$.

Numerical analyses for this linear problem indicate that SQP yields global optima with high likelihood, and termination occurs in 2-3 steps on average. It appears that whether it does so is random. This conclusion seems concordant with that of An and Tao (1997) for different problems, where numerical experiments for up to 30-dimensional problems were conducted. Furthermore, BB with N=100 seems to suffice in the case.

5 Performance Comparison

In this section we investigate the effectiveness of ψ -learning via SQP and SQP-BB(N), and compare it to SVM, in both simulated and benchmark examples. A testing error $T(\cdot)$ is used for any given method, which is averaged over 100 independent testing samples.

For simulation comparisons, we define the amount of improvement of ψ -learning over SVM as the percent of improvement of in terms of corresponding Bayesian regrets, that is,

$$\frac{(T(SVM) - T(Bayes)) - ((T(\psi) - T(Bayes)))}{T(SVM) - T(Bayes)}, \quad (9)$$

where $T(SVM)$, $T(\psi)$, and $T(Bayes)$ are the testing errors for SVM, ψ -learning, and the Bayes error, respectively, with $T(SVM) - T(Bayes)$ and $T(\psi) - T(Bayes)$ the corresponding Bayesian regrets. This measure seems to be more sensible because a comparison is performed against the baseline error—the Bayes error $T(Bayes)$, which is the testing error over a testing sample of large size, say 10^5 .

For benchmark comparisons, because $T(Bayes)$ is unknown, we then define the amount of improvement as

$$\frac{T(SVM) - T(\psi)}{T(SVM)}, \quad (10)$$

which may underestimate the improvement from the baseline error.

5.1 Simulation

For ψ -learning, SQP is applied to Examples 1 and 2 with the corresponding SVM solution as an initial value. To eliminate dependence of the performances of SVM and ψ -learning on tuning parameters, we perform a grid search to maximize the performances with respect to the tuning parameters.

Example 1 (Linear): A random training sample of size $n = 150$ is generated as follows. First, generate (t_1, t_2) from the standard bivariate t -distribution with degree 1. Second, randomly assign ± 1 to each (t_1, t_2) . Third, generate (x_1, x_2) as: $x_j = t_j + a_j$; $j = 1, 2$, with $(a_1, a_2) = ((\sqrt{3.5}, 0.5), (-\sqrt{3.5}, -0.5))$ for positive and negative classes, respectively. In this example,

Table 2: Bayesian regrets defined in (9), for SVM and ψ -learning as well as the standard errors (in parentheses) in Example 1, minimized over tuning parameter C . The Bayes error is 15.18%.

		Testing	# SV
n=150	SVM	19.04(0.826)%	109.88
	ψ	16.10(0.469)%	54.05
	% Improv	76.2%	

we maximize the performance with respect to C over an interval $(0, 10^4]$, with 9, 9, 9, 9, 99, and 10 uniformly grid points over $[10^{-3}, 10^{-2})$, $[10^{-2}, 10^{-1})$, $[10^{-1}, 1)$, $[1, 10)$, $[10, 10^3)$ and $[10^3, 10^4]$, for evaluation, that is, $10^{-3}i$; $i = 1, \dots, 9$, $10^{-2}i$; $i = 1, \dots, 9$, $10^{-1}i$; $i = 1, \dots, 9$, i ; $i = 1, \dots, 9, 10$; $i = 1, \dots, 99$, 10^3i ; $i = 1, \dots, 10$. The smallest average testing errors as well as the average number of support vectors of SVM and ψ -learning are summarized in Table 2.

Example 1 shows that ψ -learning is more robust to outliers than SVM.

Example 2 (Nonlinear): A random sample of size $n = 150, 300$ is generated as follows. First, randomly sample positive and negative class labels. For the positive class, generate (x_1, x_2) from the standard bivariate t -distribution with degree 1. For the negative class, randomly generate (x_1, x_2) from the mixture of the standard bivariate t -distribution with degree 1 and the standard bivariate normal distribution. Gaussian kernel $K(x, y) = \exp(-\frac{1}{\sigma^2} \|x - y\|^2)$ is applied to SVM and ψ -learning. Here the optimal C is chosen via grid search over interval $(0, 10^4]$, to maximize the performance of each method. The grid points are chosen in the same manner as in Example 1. For σ , it is set to be the median distance between the positive and negative classes. This is because C and σ^2 play the similar role, and it is easier to optimize over C if σ^2 is estimated. The numerical results are summarized in Table 3.

As expected, ψ -learning outperforms SVM as in Example 1. The amount of improvement, however, depends on the sample size. In this nonlinear case, the choice of tuning parameters C and σ^2 appears to be more critical.

In summary, ψ -learning outperforms SVM in both the linear and nonlinear cases with improvement ranging from 20.5% to 76.2%. In addition, the average number

Table 3: Bayesian regrets defined in (9), for SVM and ψ -learning as well as the standard errors (in parentheses) in Example 2, minimized over tuning parameters C and σ^2 . The Bayes error is 24.90%.

		Testing	# SV
n=150	SVM	29.16(0.467)%	132.77
	ψ	28.29(0.432)%	87.9
	% Improv	20.5%	
n=300	SVM	27.47(0.361)%	23.71
	ψ	26.40(0.319)%	15.72
	% Improv	41.6%	

of support vectors of ψ -learning is smaller than that of SVM in all the cases. This suggests that ψ -learning yields more sparse solutions than SVM. Moreover, ψ -learning is insensitive to outliers while SVM seems quite sensitive.

5.2 Benchmark

We now examine ψ -learning using SQP-BB($N=100$) and compare it to SVM on 6 different benchmark examples: Wisconsin Breast Cancer (WBC, Wolberg and Mangasarian, 1990), Liver-disorders and Page-Block (the UCI Machine Learning Repository, Murphy and Aha, 1992), and Heart, Breast Cancer (Breast C) and Thyroid (Rätsch, Onoda and Müller, 2001). The examples used here are those reasonable for linear or kernel-based learning. For WBC, Liver and Page-Block examples, we randomly divide each data set into two halves, for testing and training. In the Page-Block example, in particular, we choose the horizontal line and picture classes with 329 and 115 cases respectively to be the binary classes. In the case where the sample size is odd, the size of training is one larger than that of testing. For Heart, Breast C, and Thyroid examples, they are originally not binary classification, hence that a random partition into two classes is applied and are available on <http://mlg.anu.edu.au/~raetsch/data/>, c.f., Rätsch et al. (2001).

For the same randomly selected training and testing sets, SVM and ψ -learning are compared, where SQP-BB($N=100$) is used for ψ -learning, with $N = 100$ and $h = 25$ for linear training. Their performances averaged over these 100 randomly selected pairs are compared, which are minimized over C in interval $(0, 10^4]$.

Table 4: Averages of testing errors of linear SVM and ψ -learning as well as the standard errors (in parentheses), minimized over tuning parameter C in the five benchmark examples.

Data		Testing	# SV
Obs \times Dim			
WBC 682 \times 9	SVM	3.48(0.05)%	30.71
	ψ	3.05(0.04)%	16.26
	% Improv	12.4%	
Liver 345 \times 6	SVM	32.00(0.29)%	123.46
	ψ	30.38(0.28)%	51.69
	% Improv	5.1%	
Heart 270 \times 13	SVM	16.99(0.30)%	60.35
	ψ	16.58(0.33)%	32.80
	% Improv	2.4%	
Breast C 277 \times 9	SVM	28.87(0.43)%	138.17
	ψ	23.56(0.37)%	53.70
	% Improv	18.4%	
Thyroid 215 \times 5	SVM	9.43(0.25)%	35.26
	ψ	7.39(0.27)%	16.02
	% Improv	21.6%	

Table 5: Averages of testing errors of SVM and ψ -learning as well as the standard errors (in parentheses), minimized over C in polynomial learning in the two benchmark examples.

Data		Testing	# SV
Obs \times Dim			
Liver 345 \times 6	SVM	27.46(0.225)%	101.29
	ψ	26.63(0.210)%	53.13
	% Improv	3.0%	
Breast C 277 \times 9	SVM	29.09(0.475)%	92.82
	ψ	27.26(0.462)%	67.51
	% Improv	6.3%	

Table 6: Averages of testing errors of SVM and ψ -learning as well as the standard errors (in parentheses), minimized over tuning parameters C and σ^2 in learning with Gaussian kernels for the benchmark example.

Data		Testing	# SV
Obs \times Dim			
Page Block	SVM	5.06(0.202)%	50.46
444 \times 10	ψ	4.96(0.222)%	43.50
	% Improv	1.87%	

Particularly, 5, 45, 10 uniformly grid points respectively over $(0, 10]$, $(10, 10^3]$, and $(10^3, 10^4]$ are used, which are $2i; i = 1, \dots, 5$, $20i; i = 1, \dots, 50$, $2000i; i = 1, \dots, 5$. For the Gaussian kernel, σ^2 is set to the same as in Example 2. The average smallest testing errors as well as the average number of support vectors for SVM and ψ -learning are summarized in Table 4 for linear learning with SQP-BB(N=100), in Table 5 for polynomial kernels, and in Table 6 for Gaussian kernels with SQP. Here $K(x, y) = \langle x, y \rangle$ in the linear case, $K(x, y) = (1 + \langle x, y \rangle)^2$ in the polynomial kernel case, and $K(x, y) = \exp(-\frac{1}{\sigma^2} \|x - y\|^2)$ in the Gaussian kernel case.

We make the following observations regarding SVM and ψ -learning based on Tables 4-6.

- (1) Testing correctness of ψ -learning is higher than SVM on all benchmark datasets, for both linear and nonlinear learning.
- (2) On average, ψ -learning reduces the number of support vectors of SVM. The percent of reduction, however, varies.
- (3) Computing times for ψ -learning were about 7 times higher than those of SVM on average. The additional times are used for iteration, correcting the bias introduced by imposed convexity for SVM, while storage space for ψ -learning is only slightly higher.

In Tables 5 and 6, the percentage improvement is very modest. This is likely due to the fact that with kernel learning, the excess error rate (9) of the SVM over the Bayes error rate is probably of less than 10% already; so even if ψ -learning can reduce the excess error rate (9) by 50%, it will only show up as an improvement of just a

few percent according the computable index (10), which is only a lower bound of (9).

The numerical results here are consistent with the theoretical findings in Shen et al. (2003).

6 Appendix

Proof of Theorem 1: The k th subproblem (5) can be written as $\frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi_1(y_i f(\tilde{x}_i)) - \langle w, V^{(k)} \rangle$, which, after introducing n slack variables $\xi_i; i = 1, \dots, n$, is equivalent to $\min_w (\frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \xi_i - \langle w, V^{(k)} \rangle)$ with constraints: $\xi_i \geq 2(1 - y_i f(\tilde{x}_i)); \xi_i \geq 0$, or $y_i \langle w, \tilde{x}_i \rangle \geq 1 - \frac{1}{2} \xi_i; i = 1, \dots, n$.

To solve this problem, we introduce Lagrangian multipliers to yield

$$L(w, \xi, \alpha, \gamma) = \frac{1}{2} \langle w^*, w^* \rangle + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i (\langle w, \tilde{x}_i \rangle) - 1 + \frac{1}{2} \xi_i] - \sum_{i=1}^n \gamma_i \xi_i - \langle w, \nabla s_2(w^{(k)}) \rangle, \quad (11)$$

where $\alpha_i \geq 0$ and $\gamma_i \geq 0; i = 1, \dots, n$. After differentiating L with respect to (w, ξ, α, γ) and letting the derivatives be zero, we obtain that $w^* = V_1^{(k)} - \sum_{i=1}^n \alpha_i y_i x_i = 0$, $C - \frac{1}{2} \alpha_i - \gamma_i = 0$, and $-V_2^{(k)} = \sum_{i=1}^n \alpha_i y_i$. Substituting these identities in (11), $L(w, \xi, \alpha, \gamma) = \sum_{i=1}^n \alpha_i [1 - y_i \langle V_1^{(k)}, x_i \rangle] - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \frac{1}{2} \langle V_1^{(k)}, V_1^{(k)} \rangle$. This yields (6) after ignoring constant terms. To derive the corresponding constraints, we note that $C - \frac{1}{2} \alpha_i - \gamma_i = 0$, together with $\gamma_i \geq 0$, implies $\alpha_i \leq 2C$. Furthermore, $\xi_i \neq 0$ implies that $\gamma_i = 0$ and $\alpha_i = 2C$. Hence, KKT's condition becomes $\alpha_i [y_i \langle w, \tilde{x}_i \rangle - 1 + \frac{1}{2} \xi_i] = 0$ and $\xi_i [\alpha_i - 2C] = 0; i = 1, \dots, n$, implying that non-zero ξ_i 's can only occur when $\alpha_i = 2C$.

Proof of Theorem 2: The proof is essentially the same as that in Theorem 1 with slight modifications, and thus is omitted.

Proof of Theorem 3: We will only prove the linear case as the nonlinear case can be treated similarly. It follows from Theorem 6 of An and Tao (1997) that $s(w^{(k)})$ is

non-increasing with respect to k and $\lim_{k \rightarrow \infty} \|w^{(k+1)} - w^{(\infty)}\| = 0$. From (6), we know that $w^{(k+1)} = w^{(k)} + F(w^{(k)})$, where F is a continuous mapping from R^{d+1} to R^{d+1} . By the assumption, there does not exist x^* such that $\langle w^{(\infty)}, x^* \rangle = 0$. By continuity, this property holds in a small w -neighborhood of $w^{(\infty)}$. Because $\psi(z)$ is smooth when z stays away from the origin, the derivative F' exists and satisfies the Lipschitz condition $\|F'(w) - F'(w^{(\infty)})\| \leq c\|w - w^{(\infty)}\|$ for a constant $c > 0$ for any w in a small neighborhood of $w^{(\infty)}$. The convergence result then follows from Theorem 2.2 of Dennis and Moré (1974).

Proof of Theorem 4: We will solve the problem via QP. Let $v = (v_1, \dots, v_{d+2})$ be the vertices of a simplex, with $v_i = (v_{i1}, \dots, v_{i(d+1)})$. Invoking the simplex representation, any point $w = (w^*, w_{d+1})$ within the simplex can be written as $w = \sum_{i=1}^{d+2} \lambda_i v_i$, $\lambda_i \geq 0$; $i = 1, \dots, d+2$, $\mathbf{1}^T \lambda = 1$, where $\mathbf{1} = (1, 1, \dots, 1)^T$ and $\lambda = (\lambda_1, \dots, \lambda_{d+2})$. Since $\lambda_{d+2} = 1 - \sum_{i=1}^{d+1} \lambda_i$, w can be written as $A\lambda + v_{d+2}$ with $A = (v_1 - v_{d+2}, \dots, v_{d+1} - v_{d+2})$. This yields that $\lambda = A^{-1}(w - v_{d+2})$, subject to $\lambda \geq 0$ and $\mathbf{1}^T \lambda \leq 1$. Write the convex envelop $l_s(w)$ as $\langle a_1, w^* \rangle + a_2 w_{d+1} + a_{d+2}$, where a_1 and a_2 are $(d+1)$ -dimensional and one-dimensional, respectively. Then (7) becomes $\min_w (\frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \psi_1(y_i f(\tilde{x}_i)) + a_1^T w^* + a_2 w_{d+1})$ subject to $A^{-1}(w - v_{d+2}) \geq 0$, $\mathbf{1}^T A^{-1}(w - v_{d+2}) \leq 1$. After introducing n slack variables ξ_i ; $i = 1, \dots, n$, it reduces to

$$\min_w (\frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \xi_i + a_1^T w^* + a_2 w_{d+1})$$

subject to $A_1 w^* + A_2 w_{d+1} \geq a_3$, $A_3 w^* + A_4 w_{d+1} \geq a_4$, $a_5 w^* + a_6 w_{d+1} \leq a_7$, $y_i \langle w, \tilde{x}_i \rangle \geq 1 - \frac{1}{2} \xi_i$, $\xi_i \leq 0$; $i = 1, \dots, n$. Let A^{-1} be $\begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$, where A_i ; $i = 1, 2, 4$ are $d \times d$, $d \times 1$ and 1×1 , $v_{d+2} = (v_{d+2}^*, v_{d+2})$, $a_3 = A_1 v_{d+2}^* + A_2 v_{d+2}$, $a_4 = A_3 v_{d+2}^* + A_4 v_{d+2}$, $a_5 = \mathbf{1}^T A_1 + A_3$, $a_6 = \mathbf{1}^T A_2 + A_4$, and $a_7 = 1 + (\mathbf{1}^T A_1 + A_3) v_{d+2}^* + (\mathbf{1}^T A_2 + A_4) v_{d+2} = 1 + \mathbf{1}^T a_3 + a_4$. Now introduce the Lagrangian multipli-

ers:

$$\begin{aligned} L = & \frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \xi_i + a_1^T w^* + a_2 w_{d+1} \\ & - \sum_{i=1}^n \alpha_i (y_i \langle w, \tilde{x}_i \rangle - 1 + \frac{1}{2} \xi_i) - \sum_{i=1}^n r_i \xi_i \\ & - \beta^T (A_1 w^* + A_2 w_{d+1} - a_3) \\ & - \theta (A_3 w^* + A_4 w_{d+1} - a_4) \\ & + t (a_5 w^* + a_6 w_{d+1} - a_7), \end{aligned}$$

where $\alpha \geq 0$, $r \geq 0$, $\beta \geq 0$, $\theta \geq 0$, and $t \geq 0$. After differentiating L with respect to (w, ξ) and letting the derivatives be zero, we obtain that $C = \frac{1}{2} \alpha_i + r_i$, $w^* = \sum_{i=1}^n \alpha_i y_i x_i + A_1^T \beta + \theta A_3^T - a_1 - t a_5^T$, $\sum_{i=1}^n \alpha_i y_i = a_2 + t a_6 - \beta^T A_2 - \theta A_4$. Therefore, $L = -\frac{1}{2} z^T H z + g^T z - \frac{1}{2} a_1^T a_1^T$, where the matrix $H = (h_{ij})$, $h_{11} = y_x y_x^T$, $h_{12} = h_{21}^T = y_x A_1^T$, $h_{13} = h_{31}^T = y_x A_3^T$, $h_{14} = h_{41}^T = -y_x a_5^T$, $h_{22} = A_1 A_1^T$, $h_{23} = h_{32}^T = A_1 A_3^T$, $h_{24} = h_{42}^T = -A_1 a_5^T$, $h_{33} = A_3 A_3^T$, $h_{34} = h_{43}^T = -A_3 a_5^T$, $h_{44} = a_5 a_5^T$, $g = (a_1^T y_x^T + \mathbf{1}^T, a_1^T A_1^T + a_3^T, a_1^T A_3^T + a_4, -a_1^T a_5^T - a_7)$, and $y_x^T = (y_1 x_1, \dots, y_n x_n)$. This yields the desired result. Similarly as in the proof of Theorem 1, the corresponding constraints can be derived.

Proof of Theorem 5: Note that $U(S) - L(S) \leq |\langle w, \nabla s_2(w^s) \rangle - \langle w, \alpha \rangle|$, where $\alpha \in \mathcal{R}^{d+1}$ is the coefficient vector for l_s . Furthermore, when the longest edge partition is employed, the volume of S shrinks to zero, thus $\|\alpha - \nabla s_2(w^s)\| \rightarrow 0$. Consequently, $|U(S) - L(S)|$ shrinks to zero as the number of iterations increases. The algorithm stops finitely when $\varepsilon > 0$.

References

- [1] An, L. T. H., and Tao, P.D. (1997). Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *J. global opt.*, **11**, 253-285.
- [2] Boser, B., Guyon, I., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Fifth Annual Conference on Computational Learning Theory*, Pittsburgh ACM, 142-152.
- [3] Cortes, C., and Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, **20**, 273-297.

- [4] Dennis and Moré (1974). A characterization of super-linear convergence and its application to quasi-Newton methods. *Math. comput.*, **28**, 549-560.
- [5] Fung, G., and Mangasarian, O.L. (2000). Data selection for support vector machine classifiers. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 20-23, 2000, Boston, MA, R. Ramakrishnan & S. Stolfo, editors, ACM, NY 2000, 64-70.
- [6] Falk, J. E., and Hooeman, K. L. (1976). A successive underestimation method for concave minimization problems. *Mathematics of operations research*, **1**, 251-259.
- [7] Horst, R., and Tuy, H. (1989). *Global optimization-deterministic approaches*. Springer-Verlag.
- [8] Lin, Y. (2000) Some Asymptotic Properties of the Support Vector Machine. Technical report 1029r. Revised February 2002. Department of Statistics, University of Wisconsin.
- [9] Lin, Y. (2002) Support Vector Machines and the Bayes Rule in Classification. *Data Mining and Knowledge Discovery.*, **6**, 259-275.
- [10] Murphy, P.M., and Aha, D.W. (1992). UCI repository of machine learning databases. (www.ics.uci.edu/~mlearn/MLRepository.html).
- [11] Rätsch, G., Onoda, T., and Müller, K.R (2001). Soft Margins for AdaBoost. *Machine Learning*, **42**, 287-320.
- [12] Shen, X., Tseng, G., Zhang, X., and Wong, W. H. (2003). On ψ -learning. *J. Ameri. Statist. Assoc.*, **98**, 724-734.
- [13] Wahba, G. (1990). *Spline models for observational data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, xii + 169 pp, Vol. 59.
- [14] Wahba, G. (1999). Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. *Advances in Kernel Methods Support Vector Learning*, MIT Press, 69-88.
- [15] Wolberg, W.H., and Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci.*, **87**, 9193-9196.
- [16] Zhu, J., and Hastie, T. (2004). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, to appear.

A random walks perspective on maximizing satisfaction and profit

Matthew Brand*

Abstract

We model consumer behavior such as web browsing, shopping, and entertainment choices as random walks on a weighted association graph. The graph is derived from a relational database that links products, consumers, and attributes such as product categories, consumer demographics, market segments, etc. The Markov chain that describes this walk amalgamates consumer behavior over the whole population; individuals are distinguished by their current state in the chain. We develop a geometrization of the chain that furnishes a key similarity measure for information retrieval—the cosine (correlation) angle between two states. Empirically, this proves to be highly predictive of future choices made by individuals, and is useful for recommending and semi-supervised classification. This statistic is obtained through a sparse matrix inversion, and we develop approximation strategies that make this practical for very large Markov chains. These methods also make it practical to compute recommendations to maximize long-term profit.

Keywords: collaborative filtering; random walks; Markov chain; cosine correlations; semi-supervised classification.

1 Introduction

Collaborative filtering seeks to make recommendations to individuals based on the choices made by a population. An extensive literature treats this as a missing value problem, wherein an individual’s history of choices is a fragment of a hypothetical vector containing that individual’s ratings or rankings of all possible consumables. The goal is to fill in that vector (imputation) or identify the relative ranking of the of the unknown elements. A rich literature has grown up around this problem, with successful demonstrations of Bayesian, nonparametric, and even linear methods; see [1] for a broad survey. All methods essentially match the individual to others who have made similar choices, and use some combination of their experiences to predict future choices.

In this paper we explore the idea of making recommendations on the basis of associations in a relational database. The database may connect categories to products to pur-

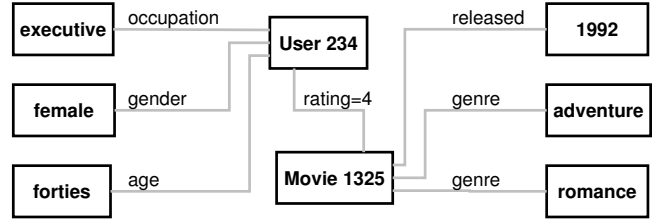


Figure 1: A fragment of an association graph representing a relational database. Affinities between pairs or groups of vertices can be computed from statistics of a random walk on the entire graph.

chasers to demographics, etc., and we may be interested in finding out what products a customer is likely to buy next, what product categories are preferred by specific demographic groups, or even how to sequence sales pitches to maximize likely profits. We answer these questions by looking at the expected behavior of a random walk on the database’s association graph (e.g., see figure 1). The expected travel time between states gives us a distance metric that has a natural transformation into a similarity measure. The random walks view has been highly successful in social networks analysis (e.g., [10, 13]) and web search (e.g., [4, 11, 3]) and in many respects is formally identical to the analysis of electrical resistive networks [5]. We develop a novel measure of similarity based on random walk behavior—the *cosine correlation between states*—and show that it is much more predictive of individual’s future choices than classic graph-based dissimilarity measures. A particularly nice feature of the random walks view is that it can naturally incorporate large amounts of contextual information beyond the usual who-liked-what of collaborative filtering, including categorical information. All this comes at a heavy computational price, but we outline approximation strategies that make these computations practical for very large graphs. These also make it practical to compute a classically useful statistic—the expected discounted profit of states, and make recommendations that optimize vendor profit.

2 Markov chain statistics

Let $\mathbf{W} \in \mathbb{R}^{N \times N}$ be a sparse nonnegative matrix that specifies the edges of a graph. \mathbf{W} may count events, e.g., W_{ij} is the number of times event j followed event i , or more generally,

*Mitsubishi Electric Research Labs, Cambridge MA 02139 USA

we may view \mathbf{W} as an arbitrarily weighted association matrix with $W_{ij} > 0$ iff person i has viewed movie j , or if web page i contains keyword j , etc. We are interested in a random walk on the directed graph specified by \mathbf{W} . (If \mathbf{W} is symmetric the graph is undirected.) The row-normalized stochastic matrix $\mathbf{T} = \text{diag}(\mathbf{W}\mathbf{1})^{-1}\mathbf{W}$ contains the transition probabilities of the associated Markov chain ($\mathbf{1}$ is a vector of 1's). We assume that the chain is irreducible and has no unreachable or absorbing states; it may be asymmetric and self-transitions are allowed to model repeat purchases. If the statistics in \mathbf{W} are a fair sample of the collective behavior of a population, then a random walk on this Markov chain will mimic, over the short term, the behavior of individuals randomly drawn from this population.

Various statistics of this walk are useful for prediction tasks. The *stationary distribution* $\mathbf{s} \in \mathbb{R}^N$ describes the relative frequencies of visiting each state in an infinitely long random walk, and can be used to flag the most popular products. Formally, $\mathbf{s}^\top \doteq \mathbf{1}^\top \mathbf{T}^\infty$ satisfies $\mathbf{s}^\top = \mathbf{s}^\top \mathbf{T}$ and $\mathbf{s}^\top \mathbf{1} = 1$. If \mathbf{W} is symmetric then $\mathbf{s} = \frac{\mathbf{1}^\top \mathbf{W}}{\mathbf{1}^\top \mathbf{W} \mathbf{1}}$; otherwise it may be computed from the recurrence $\mathbf{s}_{i+1}^\top \leftarrow \mathbf{s}_i^\top \mathbf{T}$, $\mathbf{s}_0 = \mathbf{1}/N$. The *recurrence times* $\mathbf{r} \in \mathbb{R}^N : r_i = s_i^{-1}$ describes the expected time between two visits to the same state (and should not be confused with the self-commute time $C_{ii} = 0$ described below). The *expected hitting time* H_{ij} for a random walk starting at state i to hit state j can be computed from

$$(2.1) \quad \mathbf{A} \doteq (\mathbf{I} - \mathbf{T} - \mathbf{1}\mathbf{f}^\top)^{-1}$$

for any vector $\mathbf{f} > \mathbf{0}$ satisfying $\mathbf{f}^\top \mathbf{s} \neq 0$ as

$$(2.2) \quad H_{ij} = (A_{jj} - A_{ij})/s_j$$

and the expected round-trip *commute time* is

$$(2.3) \quad C_{ij} = C_{ji} = H_{ij} + H_{ji}.$$

For the special case of $\mathbf{f} = \mathbf{s}$, \mathbf{A} is the inverse of the *fundamental matrix* and we recover the classic formula for hitting times [2]. The two dissimilarity measures C_{ij} and H_{ij} have been proposed as a basis for making recommendations [6] but they can be dominated by the stationary distribution, often causing the same popular items to be recommended to every consumer, regardless of individual consumer tastes. Ad-hoc normalizations have been proposed, but none are clearly advantageous. In this regard, it will prove useful to develop an understanding of how the chain embeds in normed spaces.

2.1 Random walk correlations Here we establish a connection to one of most useful statistics of information retrieval: the *cosine correlation*. In information retrieval, items are often represented by vectors that count various attributes. For example, if we view a document as a sample from a process that generates a particular distribution of words, its attribute vector counts (or log-counts) how many times each

(stemmed) word appears. Similar documents employ similar vocabulary, thus the inner product of their attribute vectors is large. However, longer documents sample this distribution more, resulting in more words and larger inner products. In order to be invariant to this sampling artifact, one normalizes the vectors, so that the inner product measures the empirical correlation between any two word distributions. This measure is called the cosine correlation because the normalized inner product is the cosine of the angle between two vectors.

To extend this idea to random walks, we will take two states to be similar if their relations to all other states are similar, just as similar documents have similar relationships to words.

The key idea for formalizing this intuition is a geometrization of the chain's long-term behavior: The square-root commute times are metric, satisfying the triangle inequality $\sqrt{C_{ij}} + \sqrt{C_{jk}} \geq \sqrt{C_{ik}}$, symmetry $\sqrt{C_{ij}} = \sqrt{C_{ji}}$, and identity $\sqrt{C_{ii}} = 0$ [7]. Identifying commute times with squared distances $C_{ij} \sim \|\mathbf{x}_i - \mathbf{x}_j\|^2$ sets the stage for a geometric embedding of a Markov chain in a Euclidean space¹, with each state assigned to a point $\mathbf{x}_i \in \mathbb{R}^N$, and similar states located near to each other. Because raw commute times reflect the stationary distribution, popular states will crowd near the origin regardless of dissimilarity, so raw Euclidean distance is unsuitable for most applications. However, the angle $\theta_{ij} \doteq \angle(\mathbf{x}_i, \mathbf{x}_j)$ between the embedding vectors $\mathbf{x}_i, \mathbf{x}_j$ of states i and j factors out the centrality of popular states. More importantly, its cosine measures the correlation between these two state's travel times to the rest of the graph—how similar their roles are in a random walk. E.g., if two states are perfectly correlated ($\cos \theta_{ij} = 1$), then jumping instantaneously from one to the other would not change the statistics of the random walk over the remaining states.

We need not actually compute the embedding to obtain the cosines. We can convert the matrix of squared distances \mathbf{C} to a matrix of inner products \mathbf{P} by observing that

$$(2.4) \quad C_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

$$(2.5) \quad = \mathbf{x}_i^\top \mathbf{x}_i - \mathbf{x}_i^\top \mathbf{x}_j - \mathbf{x}_j^\top \mathbf{x}_i + \mathbf{x}_j^\top \mathbf{x}_j$$

¹Both $\sqrt{C_{ij}}$ and C_{ij} are metrics; why prefer $\sqrt{C_{ij}}$? Consider square lattice graphs of varying dimension and uniform transition probabilities. The identification $C_{ij} \sim \|\mathbf{x}_i - \mathbf{x}_j\|^2$ leads to embeddings of 1D lattice graphs with uniform distances between adjoining states, but higher dimensional lattices are embedded with a pin-cushion radial distortion (corners are pulled away from the origin). Concentrating the graph in the corner spikes makes near-corner vertices have larger distances but smaller angles to central vertices than other non-corner vertices—undesirable because they are not similar to central vertices. The identification $C_{ij} \sim \|\mathbf{x}_i - \mathbf{x}_j\|^2$ leads to embeddings with a lattice-axis-parallel barrelling distortion (straight lines in the lattice are preserved, but the spacing of lattice lines is compressed according to the sigmoid $x \rightarrow \sin x$ on $(-\pi, \pi)$; angles properly increase with distance in the graph. Proof: Embedding uses the (nonconstant) eigenvectors of the graph Laplacian which comprise the lowest frequencies of a Fourier basis on the domain of grid spacings.

$$(2.6) \quad = P_{ii} - P_{ij} - P_{ji} + P_{jj}.$$

Thus, removing the row- and column-averages $P_{ii} = \mathbf{x}_i^\top \mathbf{x}_i$ and $P_{jj} = \mathbf{x}_j^\top \mathbf{x}_j$ from \mathbf{C} by a double-centering

$$(2.7) \quad -2 \cdot \mathbf{P} = (\mathbf{I} - \frac{1}{N} \mathbf{1}\mathbf{1}^\top) \mathbf{C} (\mathbf{I} - \frac{1}{N} \mathbf{1}\mathbf{1}^\top)$$

yields $P_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$ [12]. The cosine correlation is then

$$(2.8) \quad \cos \theta_{ij} = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|} = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\sqrt{\mathbf{x}_i^\top \mathbf{x}_i} \cdot \sqrt{\mathbf{x}_j^\top \mathbf{x}_j}} = \frac{P_{ij}}{\sqrt{P_{ii}P_{jj}}}.$$

In appendix A we will show efficient ways to compute \mathbf{P} directly from sparse \mathbf{T} or \mathbf{W} without computing dense \mathbf{C} . One result established there is that for the special case of symmetric, zero-diagonal \mathbf{W} , \mathbf{P} simplifies to the pseudo-inverse of the graph Laplacian $\text{diag}(\mathbf{W}\mathbf{1}) - \mathbf{W}$.

For an alternate geometric interpretation of the cosine correlations, consider projecting all embedded points onto a unit hypersphere (thereby removing the effect of generic popularity) and denoting the resulting pairwise Euclidean distances as $\overset{\circ}{d}_{ij}$. Then

$$(2.9) \quad \cos \theta_{ij} = 1 - (\overset{\circ}{d}_{ij})^2/2.$$

In this embedding the correlation between two states is negatively proportional to their squared Euclidean distance. Thus summing and averaging correlations is a geometrically meaningful way to measure similarity between two *groups* of states.

In large chains the norm $\|\mathbf{x}_i\| = \sqrt{P_{ii}}$ is usually a close approximation (up to a constant factor) of the recurrence time $r_i = s_i^{-1}$, roughly the inverse “popularity” of a state, so the cosine correlations may be interpreted as a measure of similarity that factors out artifacts of uneven sampling. For example, if two web pages are very popular the expected time to hit either from any page will be low, thus they will have a small mutual commute time. But if they are usually accessed by different groups of people or are connected to different sets of attributes, the angle between them may be large, implying decorrelation or anticorrelation. Similarly, with a movie database described below we find that the horror thriller “Silence of the Lambs” to the children’s film “Free Willy” have a smaller than average mutual commute time because both were box-office successes, yet the angle between them is larger than average because there was little overlap in their audiences.

As presented, these calculations require the construction and inversion of a dense $N \times N$ matrix, an $O(N^3)$ proposition that is clearly impractical for large chains. It is also wasteful because most queries will involve submatrices of \mathbf{P} and the cosine matrix. Section A will show how to efficiently estimate the submatrices directly from the sparse Markov chain parameters.

3 Recommending as semi-supervised classification

To make recommendations, we select one or more query states and then rank other states by their summed (or averaged) correlation to the query states. The query states may represent customers, recent purchases, demographic categories, etc.

Recommending in this model is strongly related to the semi-supervised classification problem: The states are embedded in a space as points, one or more points are given class labels, and we seek to compute an affinity (similarity measure) between each unlabelled point and each class. Unlike fully supervised classification, the affinity between a point and the labelled examples is mediated by the distribution of other unlabelled points in the space, because they influence the (locally varying) distance metric over the entire space. Similarly, in a random walk on a graph, the similarity between two states depends on the distribution of all possible paths in the graph.

To make this visually intuitive, we revisit a classification problem recently proposed by Zhou & Schölkopf [14] in the machine learning literature (see figure 2): 80 points are arranged in two normally distributed clusters in the 2D plane, surrounded by an arc of 20 points. An undirected graph is made by connecting every point to its k nearest neighbors (figure 2 left), giving a sparse graph, or to all neighbors within some distance ϵ (figure 2 right), giving a denser graph. Edge weights are chosen to be a fast-decaying function of Euclidean distance, e.g., $W_{ij} \propto \exp(-d_{ij}^2/2)$. Although connectivity and edge weights are loosely related to Euclidean distance, similarity is mediated entirely by the graph, not its layout on the page. Given three labelled points (one on the arc and one on each cluster) representing two classes, Zhou & Schölkopf ask how the rest should be classified, and propose the similarity measure $((1 - \alpha)\mathbf{I} + \alpha\mathbf{N})^{-1}$, with $\mathbf{N} = \mathbf{I} - \text{diag}(\mathbf{W}\mathbf{1})^{-1/2} \mathbf{W} \text{diag}(\mathbf{W}\mathbf{1})^{-1/2}$ the normalized combinatorial Laplacian, and $0 < \alpha < 1$ a user-specified regularization parameter. This is similar to our framework in the special case of an undirected graph with no self-arcs, but whereas we normalize the pseudo-inverted Laplacian to obtain cosines, they normalize the Laplacian, then regularize to make ordinary inversion feasible².

The similarity measure should be relatively insensitive to perturbations of the graph, especially those inflicted by a user varying the graph parameter k or ϵ . Since these mainly affect the density of edges and thus the stationary distribution, we may expect some classification robustness from cosine correlations. Figure 2 shows two such labellings. Clas-

²Zhou & Schölkopf suggest their measure is the cosine associated with commute time norms on a “lazy” random walk, but equations 3.4, 3.8 and 3.9 in their analysis only hold for $\alpha = 1$ (where their inverse is undefined), and neither the inverse nor the pseudo-inverse will yield true cosines unless $\alpha = 0$ (i.e., the graph is ignored). A secondary motivation from calculus on graphs is much more satisfying.

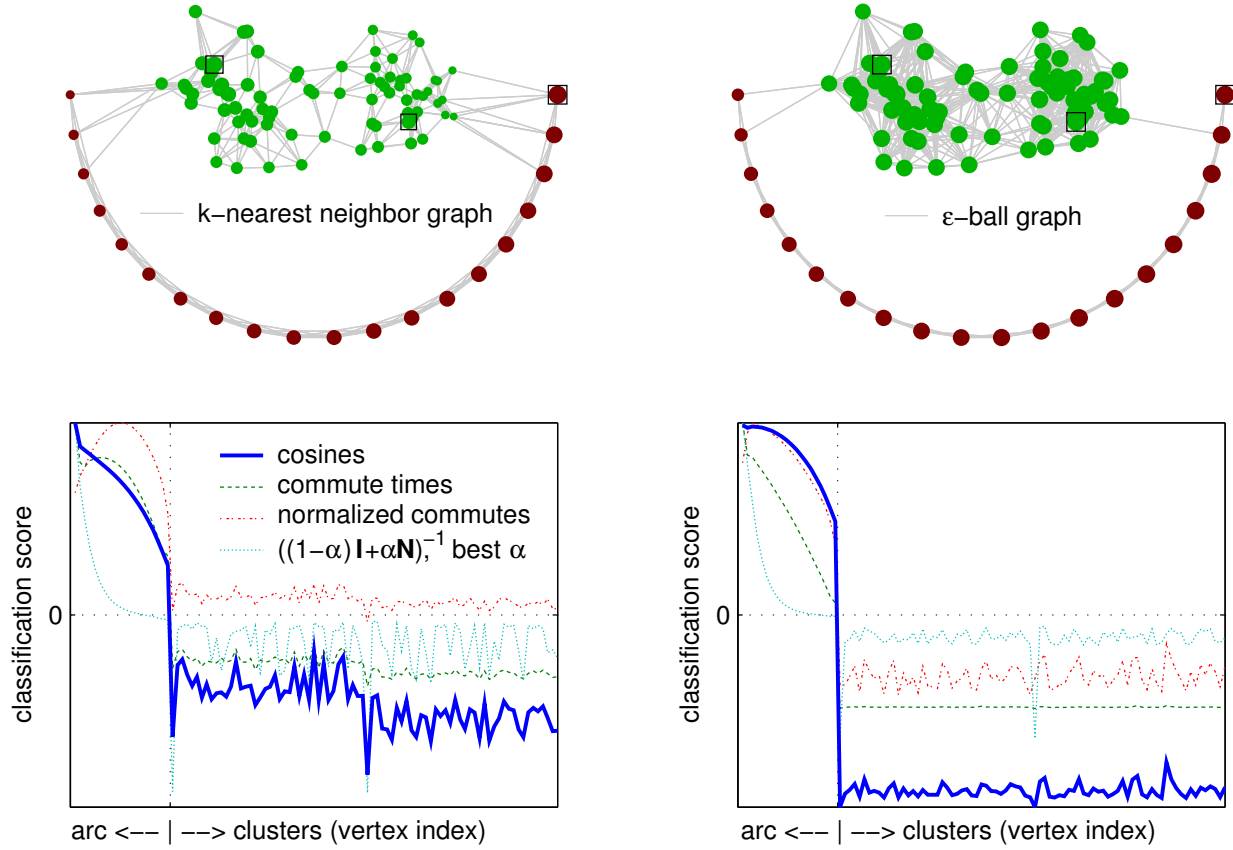


Figure 2: TOP: Classification of graph vertices according to random walk cosine correlations with labelled (boxed) vertices. Size and color of each vertex dot indicates the magnitude and sign of its classification score. BOTTOM: Point-by-point classification scores using various graph-based similarity and dissimilarity matrices. Each horizontal ordinate represents a point; the vertical ordinate is its classification score. Points scoring > 0 are classified as belonging to the arc. Classification scores are connected into lines only for the purpose of visual grouping. The cosine matrix offers the widest classification margin and most stability to small changes in the graph.

sification scores, depicted by the size and color of the graph vertices, are simply the difference between the recommendation score for two classes. The left and right panels illustrate how the classification varies when the criteria for adding edges to the graph changes. We experimented with different values for k and ϵ , and found that cosine correlations and commute times both perform well, in the sense of giving an intuitively correct classification that is relatively stable as the density of edges in the graph is varied. However, cosines offer a considerably wider classification margin and thus more robustness to graph perturbations. Normalized commute times, the Zhou & Schölkopf measure, hitting times, reverse hitting times, and their normalized variants (not shown) classify poorly to well on denser graphs but quite poorly on sparser graphs. The Zhou & Schölkopf measure in particular has a small margin because it is designed

to vary smoothly over the graph. From this small informal experiment we may expect cosine correlations to give consistent recommendations under small variations in the association graph; this is borne out below in large cross-validation experiments.

4 Expected profit

While the consumer is interested in finding the next most interesting product, the vendor wants to recommend products that are also profitable. Assuming that most customers will make more than one purchase in the future and that customers' purchase decisions are independent of vendor profit margins, decision theory tells us that the optimal strategy is to recommend the product (state) with the greatest expected profit, discounted over time. That is, the vendor wants to nudge a consumer into a state from which a random walk

will pass through highly profitable states (hence retail strategies such as “loss leaders”). Moreover, these states should be traversed early in the walk, because money is worth more now than it is in the indefinite future.

Let $\mathbf{p} \in \mathbb{R}^N$ be a vector of profit (or loss) for each state, and $e^{-\beta}, \beta > 0$ be a discount factor that determines the time value of future profits. The expected discounted profit v_i of the i^{th} state is the averaged profit of every state reachable from it, discounted for the time of arrival. In vector form:

$$(4.10) \quad \mathbf{v} = \mathbf{p} + e^{-\beta} \mathbf{T} \mathbf{p} + e^{-2\beta} \mathbf{T}^2 \mathbf{p} + \dots$$

Using the identity $\sum_{i=0}^{\infty} \mathbf{X}^i = (\mathbf{I} - \mathbf{X})^{-1}$ for matrices of less than unit spectral radius ($\lambda_{\max}(\mathbf{X}) < 1$), we rearrange the series into a sparse linear system:

$$(4.11) \quad \mathbf{v} = \left(\sum_{t=0}^{\infty} e^{-\beta t} \mathbf{T}^t \right) \mathbf{p} = (\mathbf{I} - e^{-\beta} \mathbf{T})^{-1} \mathbf{p}.$$

The most profitable recommendation for a consumer in state i is thus the state j in the neighborhood of i that has the largest expected discounted profit: $j = \arg \max_{j \in \mathcal{N}(i)} T_{ij} v_j$. If the chain is structured so that states representing saleable products are k steps away from the current state, then the appropriate term is $\arg \max_{j \in \mathcal{N}(i)} T_{ij}^k v_j$.

5 Experiments

The MovieLens database [8] contains ratings on a scale of 1-5 for 1682 movies by 943 individuals. The data is a snapshot of what movies the university community considered worth seeing in 1997. Viewers rated 20-737 movies (average=106); movies received 1-583 ratings (average=60). The ratings table is 93.7% empty, which we interpret to mean that most viewers have not seen most movies. Movies are also tagged with nonexclusive memberships in 19 genres; individuals have 2 possible genders, 21 possible vocations, and 8 overlapping age groups. We constructed an $N = 2675$ state Markov chain with $W_{ij} = 1$ for each of these connections, except for movie ratings, which were copied directly into \mathbf{W} on the principle that more highly rated movies are more likely choices. \mathbf{W} is very sparse with less than 3% nonzero values. To evaluate the many measures of similarity and dissimilarity described above, we compared their performance in the following tasks.

5.1 Recommendation to maximize satisfaction We performed extensive cross-validation experiments to determine which statistic can best predict one part of the data from the rest. In each trial we randomly partitioned the data into a test set containing 10 ratings from each viewer, and a training set containing the remainder of the data. The goal is to “predict” each viewer’s held-out movies. A Markov chain was constructed from the training set and a variety of similarity (e.g., cosine correlation) and dissimilarity (e.g., commute times)

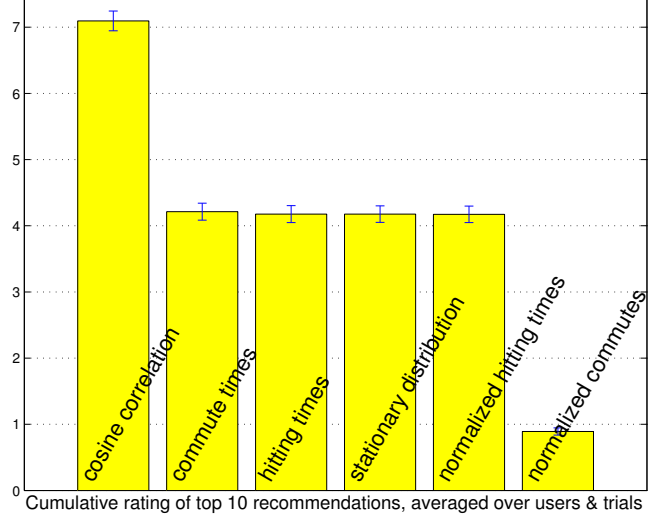


Figure 3: Cosine correlation is almost twice as effective as all other measures for predicting what movies a viewer will see and like.

matrices were computed. Sorting the rows of these matrices gives a predicted per-viewer ranking of all movies. To score each measure’s prediction quality, we took as recommendations the 10 top-ranked movies that were not in the training set, and summed the viewer’s held-out ratings of each recommended movie. A cumulative score of zero means that the viewer did not elect to rate (or presumably, see) any of the recommendations. A cumulative score of 50 would mean that the viewer did indeed see all 10 recommendations and gave all the highest possible rating. When the data’s average rating and sparsity is considered, an omniscient oracle could score no better than 35.3 on average; random guessing will score 2.2 on average. We performed 2500 trials with different random partitions and averaged scores over all viewers and trials. Figure 3 shows that cosine correlation is almost twice as successful as any other measure, with an average score slightly over 7. We also looked at how the predictors ranked the held-out movies: If a viewer had three held-out movies that the predictor ranked 5th, 17th, and 205th in her personalized recommendation list, then that predictor would be assessed a penalty of $5 + 17 + 205 = 227$. Cosine correlation had the smallest average penalty, roughly 1/4 the average penalty of commute times, the next best predictor.

Both sets of experiments were repeated with all ratings flattened to 1 ($W_{ij} \in \{0, 1\}$), yielding almost identical comparative results. When ratings are not flattened, all methods show a bias for highly rated movies.

Consistent with results reported in [6], we found that commute times are slightly more informative than hitting times. That paper advocated commute times and demon-

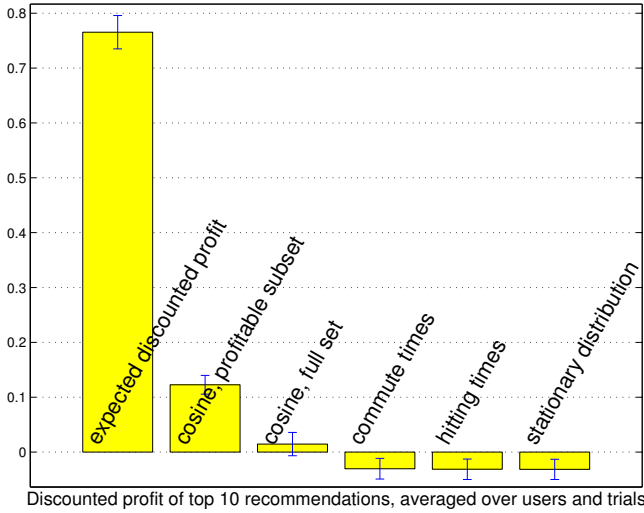


Figure 4: Making recommendations that maximize long-term profit is a much more successful strategy than recommending strictly profitable films, and profit-blind recommendations make no profit at all.

strated that they outperform k -nearest neighbors, Katz’ influence measure [10], Dijkstra shortest-path distances, and cosine coefficient computed from attribute vectors (*not* to be confused with cosine correlations of the random walk). However, we found that recommending from commute times is only slightly better than simply recommending the most popular movies identified by the stationary distribution, most likely because the commute and hitting times are both dominated by the stationary distribution. We tried several ways of pre-normalizing the data matrix \mathbf{W} or post-normalizing the hitting/commute times to ameliorate this problem but failed to improve their predictions substantially and usually worsened performance, presumably because most normalizations are not consistent with the geometry of these statistics. For example, because commute times are small where stationary probabilities are large, [14, eq. (3.6-7)] proposed post-normalizing the commute times by the recurrence time (i.e., $C_{ij}/\sqrt{r_i r_j} = C_{ij} \cdot \sqrt{s_i s_j}$); we found this promoted unpopular movies so strongly that recommendation scores averaged worse than chance. The most successful normalization, after cosine correlations, was obtained by projecting the transition matrix to the closest doubly stochastic matrix prior to computing commute times, which makes the stationary distribution uniform (when such a projection exists).

5.2 Recommendations to maximize profit We repeated the experiments above with a slightly different scoring protocol: Before trials, each movie was randomly assigned a unique profit (or loss) p_j from a unit normal distribution.

stationary distribution	correlated to ‘male’	correlated to ‘female’
Star Wars	Star Wars	The English Patient
Fargo	Contact	Contact
Return of the Jedi	Fargo	Titanic
Contact	Return of the Jedi	Jerry Maguire
Raiders of the Lost Ark	Air Force One	Conspiracy Theory
The Godfather	Scream	Sense and Sensibility
Toy Story	Toy Story	The Full Monty
Silence of the Lambs	Liar Liar	L.A. Confidential
Scream	The Godfather	Good Will Hunting

Table 1: Top recommendations made from the stationary distribution and by correlation to ‘male’ and ‘female’ states.

During trials, ten recommendations are given to the viewer in sequence; if the i^{th} recommendation is in the viewer’s held-out list, the viewer accepts the movie and we receive a time-discounted profit of $e^{-\beta} p_j$, with $e^{-\beta} = 0.9$. The goal is to maximize the profit over the entire sequence. In addition to the profit-blind predictors evaluated above, we considered a short-term profit maximizer—a cosine correlation predictor that only recommends movies with positive profits—and the long-term profit maximizer of section 4. This works by first suggesting the movie in a local graph neighborhood of the viewer’s state that has maximum expected discounted profit. If the user declines, it suggests the next most profitable movie in the same neighborhood. If the user accepts, the state shifts to that of the accepted movie and the next suggestion comes from the graph neighborhood of that state. (This is one of many ways in which the state could be updated.) Figure 4 shows that the expected discounted profit maximizer strongly outperforms the greedy short-term maximizer, and that profit-blind recommenders effectively make no profit at all. (They show slight losses only because the random pricing happened to make some of the more popular movies unprofitable.)

5.3 Market analysis Recommendations can be made from any state in the chain, making it possible to identify products that are particularly successful with a consumer demographic, or customers that are particularly loyal to specific product categories. For example, the MovieLens data has *male* and *female* attributes that are indirectly linked to all movies through the viewers, and thus we may ask which movies are preferentially watched by men or women. Ranking movies by their commute times or expected hitting times from these states turns out to be uninformative, as the ranking is almost identical to the stationary distribution ranking. (This is understandable for men because the database is mostly male.) However, ranking by cosine correlation produces two very different lists, with males preferring ac-

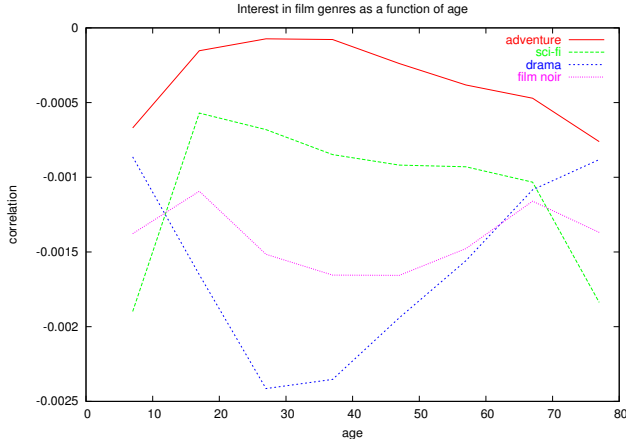


Figure 5: Correlation of age to genre preferences is weak but clearly shows that interest in sci-fi peaks in the teens and twenties. Soon after, interest in adventure peaks and interest in drama begins to climb.

tion and sci-fi movies and females preferring romances and dramas. Table 1 lists the top ten recommendations for each gender.

By the same method, we can ask which genres are preferentially watched by people of particular occupations and/or age groups. Figure 5 shows that age is indeed weakly predictive of genre preferences.

6 Conclusion

The random walks view of association graphs is a very natural way to study affinity relations in a relational database, providing a way to make use of extensive contextual information such as demographics and product categories in collaborative filtering tasks. We derived a novel measure of similarity—the cosine correlation of two states in a random walk—and showed that it is highly predictive for recommendation and semi-supervised classification tasks. Cross-validation experiments indicate that correlation-based rankings are more predictive and robust to perturbations of the graph’s edge set than rankings based on commute times, hitting times, normalized Laplacians, and related graph-based dissimilarity measures. This is very encouraging because recommendations ought to be stable with respect to random omissions in the database, a challenge presented by most data-collection scenarios. We also sketched some efficient approximation methods for very large graphs; a forthcoming paper will detail very fast exact methods based on a modified sparse L-U decomposition.

Acknowledgments Thanks to anonymous readers and reviewers for helpful comments and pointers to [6, 14].

References

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Recommendation technologies: Survey of current methods and possible extensions. MISRC working paper 03-29, <http://misrc.umn.edu/workingpapers/abstracts/0329.aspx>, May 2003.
- [2] D. Aldous and J. Fill. Reversible Markov chains and random walks on graphs. Manuscript, <http://www.stat.berkeley.edu/users/aldous/RWG/book.html>, In prep.
- [3] P. Baldi, P. Frasconi, and P. Smyth. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. John Wiley and Sons, 2003.
- [4] S. Brin and L. Page. The anatomy of a large-scale hyper-textual web search engine. In *Proc. 7th International World Wide Web Conference*, 1998.
- [5] P.G. Doyle and J.L. Snell. *Random Walks and Electric Networks*. Mathematical Association of America, 1984.
- [6] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering. In *Proc., ECML workshop on Statistical Approaches for Web Mining*, 2004.
- [7] F. Gobel and A. Jagers. Random walks on graphs. *Stochastic Processes and their Applications*, 2:311–336, 1974.
- [8] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. 1999 Conference on Research and Development in Information Retrieval*, 1999.
- [9] Ngoc-Diep Ho and Paul Van Dooren. On the pseudo-inverse of the Laplacian of a bipartite graph. In *Proc. AML’04*, 2004.
- [10] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.
- [11] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [12] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, Beverly Hills, CA, 1978.
- [13] J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, London, 2000.
- [14] D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data using random walks. 2004.

A Computational strategies

For chains with $N \gg 10^3$ states, it is currently impractical to compute the full matrix of commute times or even a large matrix inversion of the form $(\mathbf{I} - \mathbf{X})^{-1} \in \mathbb{R}^{N \times N}$. To get around $O(N^2)$ memory and $O(N^3)$ time costs, we exploit the fact that most computations have the form $(\mathbf{I} - \mathbf{X})^{-1} \mathbf{G}$ where \mathbf{X} is sparse and \mathbf{G} has only a few columns. For many queries, only a subset of states are being compared (\mathbf{G} is sparse as well), making only a subset of columns of the inverse necessary. These can be computed via the series expansions

$$(1.12) \quad (\mathbf{I} - \mathbf{X})^{-1} = \sum_{i=0}^{\infty} \mathbf{X}^i = \prod_{i=0}^{\infty} (\mathbf{I} + \mathbf{X}^{2^i}),$$

which can be truncated to yield good approximations for fast-mixing sparse chains. In particular, an n -term sum of the additive series (middle) can be evaluated via $2\log_2 n$ sparse matrix multiplies via the multiplicative expansion (right). For any one column of the inverse this reduces to sparse matrix-vector products.

One problem is that these series only converge for matrices of less than unit spectral radius ($\lambda_{\max}(\mathbf{X}) < 1$). For inverses that do not conform, the associated series expansions will have a divergent component that can be incrementally removed to obtain the numerically correct result. For example, in the case of hitting times, we have $\mathbf{X} = \mathbf{T} + \mathbf{1}\mathbf{s}^\top$ which has spectral radius 2. By expanding the additive series one can see that unwanted multiples of $\mathbf{1}\mathbf{s}^\top$ accumulate very quickly in the sum. Instead, we construct an iteration that removes them as they arise:

$$(1.13) \quad \mathbf{A}_0 \leftarrow \mathbf{I} - \mathbf{1}\mathbf{s}^\top$$

$$(1.14) \quad \mathbf{B}_0 \leftarrow \mathbf{T}$$

$$(1.15) \quad \mathbf{A}_{i+1} \leftarrow \mathbf{A}_i + \mathbf{B}_i - \mathbf{1}\mathbf{s}^\top$$

$$(1.16) \quad \mathbf{B}_{i+1} \leftarrow \mathbf{T}\mathbf{B}_i,$$

which converges to

$$(1.17) \quad \mathbf{A}_{i \rightarrow \infty} \rightarrow (\mathbf{I} - \mathbf{T} - \mathbf{1}\mathbf{s}^\top)^{-1} + \mathbf{1}\mathbf{s}^\top.$$

Note that this is easily adapted to compute an arbitrary subset of the columns of \mathbf{A}_i and \mathbf{B}_i , making it economical to compute submatrices of \mathbf{H} . Because sparse chains tend to mix quickly, \mathbf{B}_i rapidly converges to the stationary distribution $\mathbf{1}\mathbf{s}^\top$, and we often find that \mathbf{A}_i is a good approximation even for $i < N$. We can construct a much faster converging recursion for the multiplicative series:

$$(1.18) \quad \mathbf{A}_0 \leftarrow \mathbf{I} - \mathbf{1}\mathbf{s}^\top$$

$$(1.19) \quad \mathbf{B}_0 \leftarrow \mathbf{T}$$

$$(1.20) \quad \mathbf{A}_{i+1} \leftarrow \mathbf{A}_i + \mathbf{A}_i\mathbf{B}_i$$

$$(1.21) \quad \mathbf{B}_{i+1} \leftarrow \mathbf{B}_i^2.$$

This converges exponentially faster but requires computation of the entire \mathbf{B}_i . In both iterations, one can substitute $\mathbf{1}/N$ for \mathbf{s} ; this merely shifts the column averages, which are removed in the final calculation

$$(1.22) \quad \mathbf{H} \leftarrow (\mathbf{1}\text{diag}(\mathbf{A}_i)^\top - \mathbf{A}_i)\text{diag}(\mathbf{r}).$$

The recurrence times $r_i = s_i^{-1}$ can be obtained from the converged $\mathbf{B}_i = \mathbf{1}\mathbf{s}^\top$.

It is possible to compute the inner product matrix \mathbf{P} directly from the Markov chain parameters. The identity

$$(1.23) \quad \mathbf{P} = (\mathbf{Q} + \mathbf{Q}^\top)/2$$

with

$$(1.24) \quad \begin{aligned} \mathbf{Q} - \frac{1}{iN}\mathbf{1}\mathbf{1}^\top &= (\mathbf{I} - \mathbf{T} - \frac{i}{N}\mathbf{r}\mathbf{1}^\top)^{-1}\text{diag}(\mathbf{r}) \\ &= (\text{diag}(\mathbf{s}) - \text{diag}(\mathbf{s})\mathbf{T} - \frac{i}{N}\mathbf{1}\mathbf{1}^\top)^{-1}, 0 < i \leq N \end{aligned}$$

can be verified by expansion and substitution. For a submatrix of \mathbf{P} , one need only compute the corresponding columns of \mathbf{Q} using appropriate variants of the iterations above.

Once again, if \mathbf{s} (and thus \mathbf{r}) are unknown prior to the iterations, one can make the substitution $\mathbf{s} \rightarrow \mathbf{1}/N$; at convergence the resulting $\mathbf{A}' = \mathbf{A}_i - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$, $\mathbf{s} = \mathbf{1}^\top\mathbf{B}_i/\text{cols}(\mathbf{B}_i)$, $r_i = s_i^{-1}$ satisfy

$$(1.25) \quad \mathbf{A}' - \frac{1}{N}(\mathbf{A}'\mathbf{r} - \mathbf{1})\mathbf{s}^\top = (\mathbf{I} - \mathbf{T} - \frac{1}{N}\mathbf{r}\mathbf{1}^\top)^{-1}$$

and

$$(1.26) \quad \mathbf{Q} = \mathbf{A}'\text{diag}(\mathbf{r})(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top).$$

However, one pays a price for not pre-computing the stationary distribution \mathbf{s} : The last two equalities require full rows of \mathbf{A}_i , which defeats our goal of economically computing submatrices \mathbf{P} .

Such partial computations are quite feasible for undirected graphs with no self-loops: When \mathbf{W} is symmetric and zero-diagonal, \mathbf{Q} (equation 1.24) simplifies to the Laplacian kernel

$$(1.27) \quad \mathbf{Q} = \mathbf{P} = (\mathbf{1}^\top\mathbf{W}\mathbf{1}) \cdot (\text{diag}(\mathbf{W}\mathbf{1}) - \mathbf{W})^+,$$

a pseudo-inverse because the Laplacian $\text{diag}(\mathbf{W}\mathbf{1}) - \mathbf{W}$ has a null eigenvalue. In our setting, the Laplacian has a sparse block structure that allows the pseudo-inverse to be computed via smaller singular value decompositions of the blocks [9], but even this can be prohibitive. We avoid expensive pseudo-inversion entirely by shifting the null eigenvalue to 1, inverting via series expansion, and then shifting the eigenvalue back to zero. These operations are collected together in the equality

$$(1.28) \quad \begin{aligned} \frac{1}{\mathbf{1}^\top\mathbf{W}\mathbf{1}}\mathbf{P} &= \mathbf{D}((\mathbf{I} - \{\mathbf{D}(\mathbf{W} - \frac{i}{N}\mathbf{1}\mathbf{1}^\top)\mathbf{D}\})^{-1}\mathbf{D} \\ &\quad - \frac{1}{iN}\mathbf{1}\mathbf{1}^\top), \end{aligned}$$

where $\mathbf{D} \doteq \text{diag}(\mathbf{W}\mathbf{1})^{-1/2}$ and $i > 0$. By construction, the term in braces $\{\cdot\}$ has spectral radius < 1 for $i \leq 1$, thus any subset of columns of the inverse (and of \mathbf{P}) can be computed via straightforward additive iteration.

One advantage of couching these calculations in terms of sparse matrix inversion is that new data, such as a series of purchases by a customer, can be incorporated into the model via lightweight computations using the Sherman-Woodbury-Morrison formula for low-rank updates of the inverse.

Surveying Data for Patchy Structure

Ronald K. Pearson*

Abstract

The term “data surveying” refers to the preliminary examination of a dataset to assess its overall character, and this process typically involves simple descriptive statistics to characterize the available variables, along with detection of data anomalies (e.g., outliers or incomplete records) and possibly other “interesting” or “unusual” features that may be worthy of careful scrutiny. In the survey sampling literature, an important distinction is made between responses that are *missing at random*, the simplest form of *ignorable missing data*, and *non-random* alternatives that can lead to *non-ignorable missing data*. The distinction is practically important because non-ignorable missing data can cause severe biases in analytical results, while ignorable missing data typically causes an undesirable but less serious increase in the variability of these results. Analogous distinctions can also be usefully made for other types of data anomalies (e.g., i.i.d. vs. correlated outliers) or other unusual data subsets of potential interest. In particular, the observation of systematic behavior with respect to time, position, or other ordered index sequences (e.g., primary key in a database) can often give insight into the nature or generation mechanism of these data subsets. Motivated by these observations, this paper considers the problem of detecting structure in distinguished subsets of data records, including missing data, outliers and other “interesting data records.” Depending on the nature of the dependences considered, this problem is closely related to a number of others, including the detection of “streaks” in athletic performance records, the quantification of association between variables, or binary classification.

1 Introduction

Pyle [18] describes data mining in terms of three components: data preparation, data surveying, and data modeling. The second of these steps—data surveying—is concerned with identifying and characterizing what is present in the dataset. Useful data surveying tools include simple descriptive statistics (e.g., how many variables constitute each data record? what kind are they (nominal, ordinal, or real)? what are the ranges and typical values for each?), along with somewhat more complex characterizations like entropy measures [18, Ch. 11]. In addition, it is also useful to characterize

both data anomalies (e.g., outliers and missing data) and other “interesting” or “unusual” data subsets that may be worthy of separate analysis. The problem of interest in this paper is the detection of significant *structure* in these subsets, which may lead to useful insights concerning their nature and origin.

As a particularly important example, a useful distinction is made in the survey sampling literature between data values that are *missing at random (MAR)* and those that are *systematically missing* [20]. The MAR model generally represents *ignorable missing data*, which may be regarded as a nuisance that causes the uncertainty of our analytical results to increase, effectively reducing our sample size. Conversely, *nonignorable* missing data patterns in which the probability of being included in the dataset depends on the missing data values themselves are generally more serious as they can cause large biases in our results. Further, the identification of nonignorable missing data can be the first step in discovering *why* these data values are missing, which can have important practical implications.

Although it is not as widely discussed, a somewhat analogous distinction is that between outliers that are randomly distributed throughout the dataset and *dependent outliers*, sometimes known as “patchy outliers.” In particular, Davies and Gather [6] express concern that, “in almost all simulations in the literature the outliers are taken to be iid random variables.” To illustrate that such working assumptions are not always appropriate, they discuss a highly contaminated weather balloon dataset in which the outliers do not conform to this assumption. This distinction is important because outlier sequences of the same magnitude and concentration but with different dependence structures can have very different influences on dynamic characterizations like spectrum analysis or linear system identification [16]. Again, detection of systematic patterns in outliers or other data anomalies can be useful in determining the mechanisms and sources responsible for these anomalies.

The analogy between dependent outliers and systematic missing data becomes clear if we adopt the *replacement model* for outliers [14]. There, the sequence $\{y_k\}$ of available data samples is modelled as:

$$(1.1) \quad y_k = (1 - z_k)x_k + z_k o_k,$$

*ProSanos Corporation, Harrisburg, PA.

where $\{x_k\}$ is the nominal (i.e., uncontaminated) data sequence of interest, $\{o_k\}$ is a sequence of outlying values, and $\{z_k\}$ is a binary selection sequence, assuming the value $z_k = 0$ whenever the nominal data value is observed, and $z_k = 1$ whenever the outlying data value is observed. The outlier analog of the missing at random data model then corresponds to assuming that $\{z_k\}$ is an iid binary sequence (i.e., a sequence of Bernoulli trials). Similarly, the outlier analog of systematic missing data corresponds to the case where the binary sequence $\{z_k\}$ either exhibits a significant dependence structure (e.g., patchy outliers) or depends on other contaminated variables. In particular, *common mode effects* (e.g., partial system failures) can be responsible for the presence of outliers in several different variables simultaneously, again in violation of the random occurrence model. The influence of these strongly correlated outlier sequences in different variables can profoundly influence the results of otherwise reasonable joint characterizations like cross-correlation analysis [15, Sec. 8.1.2].

Useful distinctions can also be made between random and systematic occurrence of other types of anomalous, interesting, or unusual data records \mathcal{R}_k within a dataset \mathcal{D} . Specific examples include *inliers*, defined as observations that lie within the distribution of nominal (i.e., non-anomalous) data values, but which are in error [21], *near-duplicate records* (e.g., web documents) [3], or subsets of data records that have been deemed “interesting” by some quantitative *interestingness measure* [12]. DesJardins [7] notes that isolated inliers may not be a problem and may be almost indistinguishable from nominal data values, but that moderate-sized sets of inliers can have more serious analytical consequences. (It is important to note that the term “inlier” is sometimes used as a synonym for “nominal data” [13], quite distinct from the meaning assumed here.) Finally, another case where data records of particular interest are not randomly distributed throughout a dataset is the case of alarms in telecommunication network data [22]. There, different alarm sequences are known to be correlated and to occur in intermittent bursts; one of the key practical challenges is in extracting cause information from the complicated patterns generated by these related alarm sequences.

The general problem considered in this paper is the following one. We are given a class \mathcal{K} of data records $\{\mathcal{R}_k\}$ that are of particular interest. This class could consist of missing or incomplete records, outliers, inliers, near-duplicate records, or records identified on the basis of some interestingness measure, either objective or subjective [12]. The essential requirement here is that we have available a classification scheme that partitions data records into those belonging to class \mathcal{K} and those

not belonging to this class. Given this partitioning, define the *status sequence* $\{z_k\}$ as:

$$(1.2) \quad z_k = \begin{cases} 1 & \text{if } \mathcal{R}_k \in \mathcal{K} \\ 0 & \text{if } \mathcal{R}_k \notin \mathcal{K}, \end{cases}$$

generalizing the binary sequence $\{z_k\}$ on which the replacement outlier model (1.1) is based. An obvious extension of this idea would be to consider multiple classes of interesting data records, but this paper considers only a single class \mathcal{K} . The key problem of interest here thus reduces to the characterization and interpretation of the binary sequence $\{z_k\}$. As one reviewer noted, it is important to emphasize that the utility of the results obtained from analyses like those described here depends strongly on the accuracy of the classification procedure that generates the status sequence $\{z_k\}$. To keep the length of this paper manageable, the problem of missclassification is minimized here by considering cases like missing data where accurate construction of the sequence $\{z_k\}$ is straightforward.

2 The problem of assessing patchiness

The primary question considered in this paper is whether the records belonging to class \mathcal{K} occur randomly or systematically through the dataset \mathcal{D} , based on their record index k . If the dataset consists of a sequence of real values indexed by time and if the class \mathcal{K} corresponds to local outliers in this sequence, the question considered here reduces to one of determining whether these outliers are isolated or patchy. More generally, even if the records are much more complex and the record index has no obvious interesting interpretation, the detection of patchiness in the status sequence $\{z_k\}$ can lead us to discover unexpected structure in these records, a point illustrated in Sec. 6.

Despite the simplicity of the concept—records from class \mathcal{K} are either grouped together into patches or they are not—the practical assessment of patchiness in binary data sequences is harder than it sounds. This point is illustrated in Fig. 1, which shows a portion of the status sequence $\{z_k\}$ discussed in Sec. 6.2. Briefly, this sequence identifies a subset of adverse event incident reports in the U.S. Food and Drug Administration’s Adverse Event Reporting System (AERS) database in which an outcome of “death” is listed. The visual appearance of this binary sequence is strongly suggestive of patchiness, but it is desirable to have a quantitative characterization that permits us to objectively assess patchiness and quantify its extent.

The assessment of patchiness in status sequences is essentially the same as that of detecting “streakiness” in sports performance statistics. As a specific example, Albert and Williamson consider the question, “Was

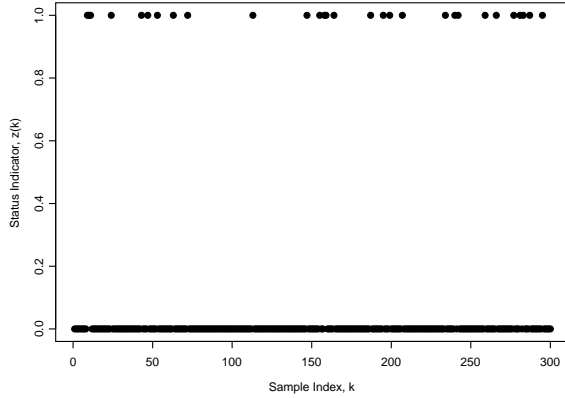


Figure 1: Binary status sequences derived from the AERS database example discussed in Sec. 6.2.

Javy Lopez a ‘streaky hitter’ when he played for the Atlanta Braves in 1998?” [1]. These authors discuss several different approaches to this problem, and the one they adopt is a simulation-based Bayesian strategy that incorporates any one of several different parametric streakiness models and draws inferences about the model parameters. In addition, this approach also requires a “streakiness statistic” and the authors consider six of these: two are based on moving averages, two are based on characterization of runs of successive 0’s or 1’s in individual at-bat results, one is based on a logistic model relating the probability of hitting in a given game to batting averages in previous games, and one is based on the standard deviation of batting averages across subgroups of games.

The approach to assessing patchiness adopted in this paper is based on the empirical patchiness measures described in Sec. 4, together with a random permutation strategy that provides a reference standard for tests of the patchiness hypothesis. To assess the performance of these measures, they are first applied to simulated sequences based on the patchy sequence model described in Sec. 3. One of these measures is then applied in Sec. 6 to the AERS database mentioned above.

3 A patchy sequence model

To assess the performance of patchiness characterizations like those described here, it is useful to have a simulation procedure for generating patchy sequences with well-defined, controllable characteristics. Here, the following patchy sequence model is used as the basis for such a procedure. The idea is to specify a distribution $\{p_w\}$ of patch widths w and use this distribution

to generate a binary sequence $\{z_k\}$ of length N having patches of successive 1’s that are drawn from this distribution. Specifically, given $\{p_w\}$, the sequence $\{z_k\}$ is generated as follows. First, a sequence $\{w_k\}$ of N possible patch widths is generated having probabilities p_w for $w = 0, 1, \dots, w^*$ where $w^* \leq N$ is width of the widest patch considered. The binary sequence $\{z_k\}$ is then constructed according to the following procedure:

0. Initialize: set $k = 1$
1. Do while $k \leq N$:
 - $w_k = 0 \Rightarrow z_k = 0$ and $k \rightarrow k + 1$,
 - $w_k = 1 \Rightarrow z_k = 1$ and $k \rightarrow k + 1$,
 - $w_k = w > 1 \Rightarrow z_k = z_{k+1} = \dots = z_{k+w} = 1$ and $k \rightarrow k + w$.
2. Return $\{z_k\}$ for $k = 1, 2, \dots, N$.

Note that taking $p_1 = q$ and $p_0 = 1 - q$ yields a Bernoulli sequence with probability q that $z_k = 1$; an example is shown in the upper plot in Fig. 2, which was obtained by taking $p_0 = 0.95$ and $p_1 = 0.05$. Conversely, taking $p_w > 0$ for $w > 1$ yields binary sequences with patches of width w . As a specific example, the sequence shown in the lower plot in Fig. 2 was obtained by setting $p_0 = 0.95$ and $p_3 = 0.05$, giving a sequence that always exhibits patches of length 3, an outcome that occurs 5% of the time. Note, however, that since each 1 generated by this model occurs three times in succession, the probability that $z_k = 1$ is 15% rather than 5% as in the Bernoulli example. More generally, note that the expected number of 1’s in the binary sequence $\{z_k\}$ generated according to the procedure just described is

$$(3.3) \quad N_a = N \sum_{w=0}^N w p_w.$$

Hence, if we wish to generate a patchy sequence of fixed length N having a specified value for N_a (e.g., a patchy outlier sequence with “10% contamination”), it is necessary to reduce the probabilities p_w accordingly for $w > 0$ to account for the patch effects. This modification is equivalent to increasing the probability p_0 , an idea closely related to the use of zero-inflated Poisson models [4, 11], zero-inflated binomial models [11], or zero-inflated negative binomial models [4] in analyzing count data.

4 Empirical measures of patchiness

To assess the patchiness of a status sequence $\{z_k\}$ of length N , this paper considers three closely related empirical measures. The first is the *empirical concentration*

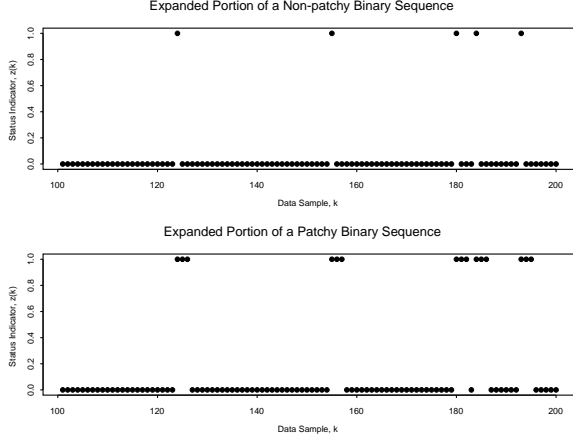


Figure 2: Two simulated binary sequences: a non-patchy Bernoulli sequence, assuming the value $z_k = 1$ with probability 5% (top), and a sequence that exhibits only patches of width 3 (bottom).

measure:

$$(4.4) \quad \hat{\phi}_w = \frac{N_w(w+1)}{N-1},$$

which represents the fraction of the maximum possible number of patches of width w , based on the following definitions. A *patch of width w* in the binary sequence $\{z_k\}$ is defined by the following two conditions, which must hold for some index k satisfying $k \geq 2$ and $k \leq N - w$:

$$(4.5) \quad \begin{aligned} z_k = z_{k+1} = \dots = z_{k+w-1} &= 1 \\ \text{and } z_{k-1} = z_{k+w} &= 0. \end{aligned}$$

It follows from this definition that the maximum possible number of patches satisfies:

$$(4.6) \quad N_w w \leq (N-2) - (N_w - 1).$$

The term on the left-hand side of this inequality counts the number of points included in the patches of width w , while the first term on the right-hand side is the maximum number of points in the sequence $\{z_k\}$ that can assume the value $z_k = 1$ (i.e., z_1 and z_N must both be zero), and the second term on the right-hand side is the minimum number of zero values required to separate successive patches of width w . It follows from Eq. (4.6) that

$$(4.7) \quad N_w \leq \frac{N-1}{w+1},$$

meaning that the empirical concentration measure $\hat{\phi}_w$ satisfies $0 \leq \hat{\phi}_w \leq 1$ for all patch widths w .

The numerical results presented in the following examples were computed in the *S-plus* software package;

while it is possible to compute $\hat{\phi}_w$ by brute force, this involves a very slow nested loop construction, and it is *much* faster to use an algorithm based on the following observations. First, define the complementary sequence $\{z_k^c\}$ to the status sequence $\{z_k\}$ by:

$$(4.8) \quad z_k^c = 1 - z_k.$$

Next, note that the two defining conditions given in Eq. (4.5) for a patch of width w may be expressed as:

$$(4.9) \quad \begin{aligned} z_{k-1}^c = z_k = z_{k+1} = \dots = z_{k+w-1} = z_{k+w}^c &= 1 \\ \Leftrightarrow z_{k-1}^c \cdot z_k \cdot z_{k+1} \cdot \dots \cdot z_{k+w-1} \cdot z_{k+w}^c &= 1. \end{aligned}$$

The advantage of this observation is that it immediately yields the following expression for the number N_w of patches of width w in the sequence $\{z_k\}$, i.e.

$$(4.10) \quad N_w = \sum_{k=2}^{N-w} z_{k-1}^c \cdot z_k \cdot z_{k+1} \cdot \dots \cdot z_{k+w-1} \cdot z_{k+w}^c.$$

To compute $\hat{\phi}_w$ over a range of values from 1 to some maximum patch width w^* , simply construct a matrix with N rows and w^* columns where each column contains the vector appearing on the right-hand side of Eq. (4.10). The number N_w can then be efficiently computed as the vector of row sums of this matrix and $\hat{\phi}_w$ can be computed directly from this result.

By itself, a sequence of values $\{\hat{\phi}_w\}$ computed from a given binary status sequence $\{z_k\}$ is not easy to interpret: does $\hat{\phi}_w = 0.3$ give strong or weak evidence in support of the hypothesis that $\{z_k\}$ exhibits an unusually large number of patches of width w ? Conversely, how small must $\hat{\phi}_w$ be to suggest *fewer* patches of width w than we would expect under the homogeneous Bernoulli alternative? In subsequent discussions, this latter phenomenon will be called *sparseness*. To address these questions, this paper adopts a permutation strategy [10], analogous to that used previously in assessing the significance of clustering results [17]. Specifically, given a sequence $\{z_k\}$, applying a random permutation to the index sequence should destroy any patchiness that may be present, effectively reducing the randomized sequence $\{\tilde{z}_k\}$ to a Bernoulli sequence. Hence, if the number of patches of width w is unusually large in the original sequence, relative to a Bernoulli alternative, this randomization should cause $\hat{\phi}_w$ to decrease significantly. Similarly, if the sequence $\{z_k\}$ contains significantly fewer patches of width w than expected under the Bernoulli alternative, randomization should cause $\hat{\phi}_w$ to increase. Repeating this process for M statistically independent random permutations gives a sequence $\{\tilde{\phi}_w^j\}$ of patchiness measures that can be used

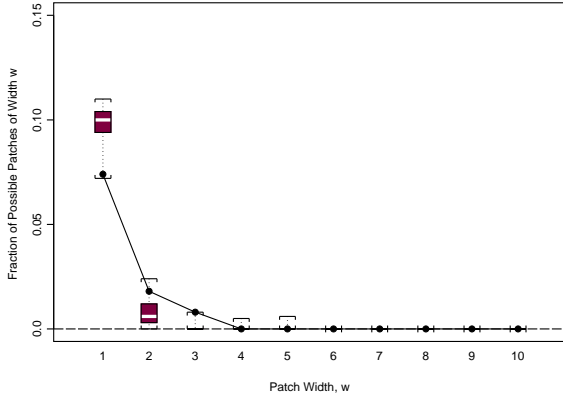


Figure 3: Empirical patchiness characterization for the Bernoulli sequence shown in the upper plot in Fig. 2. The line through the solid circles corresponds to the computed values of $\hat{\phi}_w$. The boxplots each summarize the $\hat{\phi}_w^j$ values obtained from 200 randomizations of the original sequence.

to assess the significance of the original patchiness measure $\hat{\phi}_w$. In particular, $\hat{\phi}_w$ gives evidence of patchiness if it lies above the range of the randomization results, and it gives evidence of sparseness if it lies below this range. Since these comparisons define a two-sided hypothesis test, $\hat{\phi}_w$ values falling outside this range are significant at the level $2/M$.

Fig. 3 summarizes the results obtained for the Bernoulli sequence shown in the upper plot in Fig. 2 using the empirical patchiness characterization proposed here, for patch widths between $w = 1$ and $w = 10$. The values of $\hat{\phi}_w$ computed from the original sequence $\{z_k\}$ are shown in Fig. 3 as solid circles, connected by a line, and the results $\{\hat{\phi}_w^j\}$ obtained for 200 independent randomizations are summarized with boxplots. Overall, the results are exactly what we expect for a Bernoulli sequence, which may be taken as a reference standard of “non-patchiness.” Specifically, these results show no evidence of patchiness since none of the $\hat{\phi}_w$ values fall outside the range of values generated by the 200 randomizations. Also, note that the quantity $\hat{\phi}_w$ shown here is not the same as the *patchy contamination at width w* $\hat{\gamma}_w$, defined as the number of contaminants contributed by patches of width w and given by

$$(4.11) \quad \hat{\gamma}_w = \frac{N_w w}{N} = \left(\frac{w}{w+1} \right) \left(\frac{N-1}{N} \right) \hat{\phi}_w.$$

In particular, note that $\hat{\gamma}_1 \simeq 0.05$, corresponding to the total contamination level of the Bernoulli sequence considered here, while $\hat{\phi}_1 \simeq 0.10$, twice this level.

For comparison, Fig. 4 shows the corresponding results obtained from the patchy sequence shown in the bottom plot in Fig. 2, which consists entirely of patches of width 3. As before, the original ϕ_w values are represented by the solid circles connected with the smooth curve, and the 200 randomization results $\{\hat{\phi}_w^j\}$ are summarized with boxplots. Because of the special character of the status sequence $\{z_k\}$ considered here, these results illustrate both significant patchiness and significant sparseness, relative to the Bernoulli alternative. In particular, since no patches of widths 1 or 2 appear in this sequence, $\hat{\phi}_1 = \hat{\phi}_2 = 0$ here and these results fall well below the range of the corresponding randomization values. In other words, these results correctly reflect the extreme sparseness of the sequence $\{z_k\}$ with respect to patches of widths $w = 1$ and $w = 2$. Conversely, the results for $\hat{\phi}_3$ lie well above the range of the randomization results, giving strong evidence in support of the patchiness hypothesis for $w = 3$. None of the other results are significant, as they all fall within the range of the corresponding randomizations. Note, however, that although the value of $\hat{\phi}_6$ is not significant relative to the randomizations, it is the only nonzero result other than $\hat{\phi}_3$, reflecting the fact that two successive patches of width 3 were generated here by the random sequence generator described in Sec. 3 with no intervening zero value, converting them into a single patch of width 6.

The visual similarity of this result to a second harmonic in a power spectrum suggests the following alternative graphical representation of the results presented here. Define $\hat{\psi}_w$ as the *patch spectrum of width w*, given by

$$(4.12) \quad \begin{aligned} \hat{\psi}_w &= \frac{N_w w}{\sum_{k=1}^N z_k} \\ &= \left(\frac{(N-1)w}{(w+1) \sum_{k=1}^N z_k} \right) \hat{\phi}_w. \end{aligned}$$

Note that this quantity represents the fractional contribution of patches of width w to the total number of 1’s in the $\{z_k\}$ sequence. Because $\hat{\psi}_w$ is linearly related to $\hat{\phi}_w$ by a constant that is invariant under random permutations, it follows that the randomization results $\hat{\psi}_w^j$ may be computed directly from the corresponding randomization results $\hat{\phi}_w^j$, i.e.,

$$(4.13) \quad \hat{\psi}_w^j = \left(\frac{(N-1)w}{(w+1) \sum_{k=1}^N z_k} \right) \hat{\phi}_w^j.$$

An advantage of $\hat{\psi}_w$ over $\hat{\phi}_w$ is that, while both patchiness measures are normalized to lie in the unit interval $[0, 1]$, this upper limit has a more useful interpretation

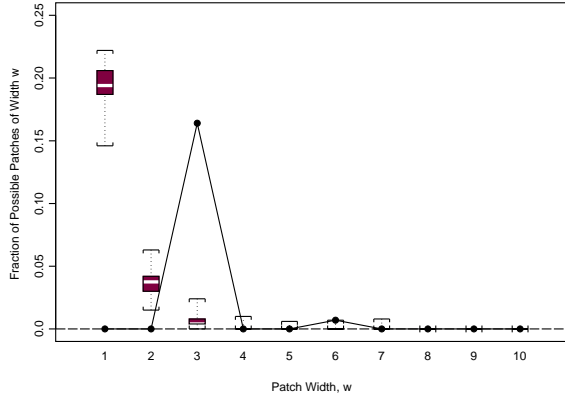


Figure 4: Empirical patchiness characterization $\hat{\phi}_w$ for the patchy binary sequence shown in the lower plot in Fig. 2, in the same format as Fig. 3.

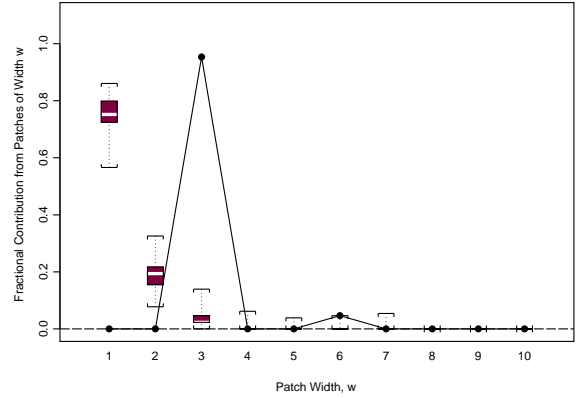


Figure 5: Computed patch spectrum $\hat{\psi}_w$ for the patchy binary sequence shown in the lower plot in Fig. 2, in the same format as Fig. 3.

for the patch spectrum $\hat{\psi}_w$ than it does for the empirical patchiness measure $\hat{\phi}_w$. For example, note that if $\hat{\phi}_1 = 1$, the sequence $\{z_k\}$ is completely specified: it is a periodic sequence of odd length N that takes the value $z_k = 0$ whenever k is odd and $z_k = 1$ whenever k is even. Although this sequence certainly can arise, it is not a typical status sequence we might expect to see in practice. Conversely, the result $\hat{\psi}_1 = 1$ is easily seen to arise if and only if every data anomaly is isolated, a much more likely situation in practice, and one that we might well be interested in detecting.

More generally, the scaling of the patch spectrum $\hat{\psi}_w$ appears to be much more informative than that of the empirical patchiness measure $\hat{\phi}_w$, as may be seen by comparing Figs. 4 and 5. In particular, the fact that $\hat{\psi}_3 \simeq 1$ demonstrates clearly that almost all of the 1's in the sequence $\{z_k\}$ appear in patches of width $w = 3$, a point that is not obvious from the numerical values of $\hat{\phi}_w$ plotted in Fig. 4. Similar conclusions apply for the Bernoulli sequence: the patch spectrum results in Fig. 6 show that $\sim 65\%$ of the 1's in this sequence occur in patches of width $w = 1$, $\sim 20\%$ in patches of width $w = 2$, and the remaining $\sim 15\%$ in patches of width $w = 3$. Again, this quantitative interpretation is not obvious from the numerical values for $\hat{\phi}_w$ shown in Fig. 3. Overall, because the results for $\hat{\psi}_w$ are easier to interpret than those for $\hat{\phi}_w$, the remainder of this paper focuses entirely on the patch spectrum $\hat{\psi}_w$.

5 Two numerical summary statistics

Although plots like Figs. 3 through 6 give useful qualitative characterizations of patchiness, it is desirable to also

have simple numerical summary statistics. The following discussion presents two such summaries: the z-score z_w of the patch spectrum $\hat{\psi}_w$ relative to the randomized results $\{\tilde{\psi}_w^j\}$, and a quantity α_w that gives a normalized measure of the distance $\hat{\psi}_w$ lies from the most extreme $\tilde{\psi}_w^j$ value, relative to the range of possible $\hat{\psi}_w$ values.

More specifically, let $\tilde{\mu}_w$ denote the mean of the M randomized values $\{\tilde{\psi}_w^j\}$ and let $\tilde{\sigma}_w$ denote the standard deviation of these values. The z-score for the $\hat{\psi}_w$ value computed from the original data sequence is then defined as

$$(5.14) \quad z_w = \frac{\hat{\psi}_w - \tilde{\mu}_w}{\tilde{\sigma}_w}.$$

If, as in the case of $\hat{\psi}_w$ for $w > p$, all of the randomized values $\tilde{\psi}_w^j$ are equal (e.g., zero in this case), it follows that $\tilde{\sigma}_w = 0$. When $\hat{\psi}_w$ also exhibits this common value, again as in the case of $\hat{\psi}_w$ for $w > p$, the value of z_w will be defined as zero; otherwise, z_w will be defined as $\pm\infty$, depending on the sign of $\hat{\psi}_w - \tilde{\mu}_w$.

If the randomized values $\{\tilde{\psi}_w^j\}$ exhibit an approximately normal distribution, we would expect to see no z-scores larger in magnitude than $|z_w| \sim 3$ for non-patchy status sequences $\{z_k\}$ (specifically, the probability of observing a normal random variable with a z-score of magnitude larger than 3 is approximately 0.3%). However, the shape of some of the boxplot summaries is strongly suggestive of significant asymmetry, bringing the appropriateness of approximate normality assumptions seriously into question. Still, it follows from Chebyshev's inequality [2, p. 75] that:

$$(5.15) \quad \mathcal{P} \left\{ \left| \frac{x - \mu}{\sigma} \right| > \beta \right\} = \mathcal{P}\{|z| > \beta\} \leq \frac{1}{\beta^2},$$

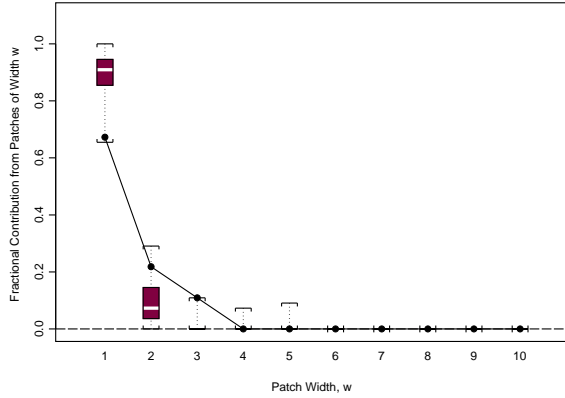


Figure 6: Computed patch spectrum $\hat{\psi}_w$ for the Bernoulli sequence shown in the upper plot in Fig. 2, in the same format as Fig. 3.

for any finite variance distribution. Hence, even under this very weak distributional assumption, it follows that z-scores of large magnitude are unlikely. In particular, note that $|z_w| > 10$ has probability less than 1% even under this extremely conservative working assumption.

The second summary statistic considered here is α_w , defined as follows. First, define the minimum and maximum random permutation values as:

$$(5.16) \quad \begin{aligned} \tilde{\psi}_w^- &= \min_j \{\tilde{\psi}_w^j\}, \\ \tilde{\psi}_w^+ &= \max_j \{\tilde{\psi}_w^j\}. \end{aligned}$$

The α_w value is then defined as:

$$(5.17) \quad \alpha_w = \begin{cases} \frac{\hat{\psi}_w - \tilde{\psi}_w^+}{1 - \tilde{\psi}_w^+} & \hat{\psi}_w > \tilde{\psi}_w^+ \\ 0 & \tilde{\psi}_w^- \leq \hat{\psi}_w \leq \tilde{\psi}_w^+ \\ -\left(\frac{\tilde{\psi}_w^- - \hat{\psi}_w}{\tilde{\psi}_w^-}\right) & \hat{\psi}_w < \tilde{\psi}_w^- \end{cases}$$

Note that α_w is nonzero if and only if $\hat{\psi}_w$ is significant with respect to the random permutation values $\{\tilde{\psi}_w^j\}$. For $\hat{\psi}_w$ values lying above the range of these permutation values, α_w is positive, bounded above by its maximum achievable value of 1. Since this behavior is precisely what we expect for a patchy status sequence, positive α_w values may be interpreted as a measure of the strength of evidence in support of the hypothesis that $\{z_k\}$ exhibits an unusually large number of patches of width w , relative to the Bernoulli model. In particular, note that $\alpha_w \simeq 1$ implies that essentially all of the anomalies identified by the status sequence $\{z_k\}$ occur in a patch of width w that has low probability under the Bernoulli model. Conversely, a sparse sequence will

exhibit *fewer* patches of width w than expected under the Bernoulli model, and this will give rise to negative α_w values, with the same general interpretation. Specifically, negative α_w values lie between -1 and 0 , and they may be viewed as a measure of the strength of evidence in support of the *sparseness hypothesis* that $\{z_k\}$ exhibits significantly fewer patches of width w than would be expected under the Bernoulli model. More specifically, $\alpha_w \simeq -1$ implies that patches of width w that are expected to be present under the Bernoulli model are largely absent in the observed status sequence $\{z_k\}$.

For the patchy status sequence $\{z_k\}$ shown in the bottom plot in Fig. 2, nonzero α_w values are observed only for $w = 1, 2$, and 3 , and these values are $\alpha_1 = \alpha_2 = -1$ and $\alpha_3 = 0.95$. These results reflect first, the complete absence of patches of width $w = 1$ and $w = 2$, both expected under the Bernoulli alternative, and an overwhelming predominance of patches of width $w = 3$, which are relatively rare under the Bernoulli alternative. For comparison, none of the α_w values computed from the Bernoulli sequence shown in the upper plot in Fig. 2 are nonzero, in perfect agreement with our expectations.

It is particularly instructive to consider the z-scores for this example. For the patchy sequence shown in the bottom plot in Fig. 2, nonzero z-scores are obtained for all patch widths w between 1 and 7, although four of these values are quite small in magnitude (specifically, $z_4 = -0.4$, $z_5 = -0.1$, and $z_7 = -0.1$). The results for patch widths of 1 and 2 exhibit negative z-scores, consistent with the absence of expected patches of these widths in this example: $z_1 = -15.1$ and $z_2 = -4.3$. Not surprisingly, the results for patches of width 3 exhibit the largest magnitude z-score seen: $z_3 = 32.6$. What is somewhat surprising is that $z_6 = 9.9$, an extremely large z-score, especially for a result that is not significant with respect to the random permutations. Less extreme but somewhat similar results are obtained for the Bernoulli sequence shown in the top plot in Fig. 2: $z_1 = -3.6$, $z_2 = 2.2$, and $z_3 = 5.0$, with much smaller values for $w = 4$ and $w = 5$ ($z_4 = z_5 = -0.1$). The large magnitudes of some of these non-significant z-scores further emphasizes the point noted above that approximate normality should not be assumed for the permutation values $\{\tilde{\phi}_w^j\}$ here, since many of these z-scores would be extremely significant under the Gaussian model. As a practical matter, the best strategy is probably to consider only the z-scores for those values having nonzero α_w values.

6 Applications to the AERS database

The following examples serve both to illustrate the application of the patch spectrum $\hat{\psi}_w$ to real data, and to demonstrate that the characterization of patchiness

can lead to the identification of unusual structure even when applied to record indices (i.e., access keys) k with little or no inherent real-world significance. Both examples are based on the U.S. Food and Drug Administration’s Adverse Event Reporting System (AERS) database, which summarizes medical adverse events reported in conjunction with the use of specific drugs. A detailed description is available through the website <http://www.fda.gov/cder/aers/>. In general terms, exploratory analysis of this database is of interest because it can provide evidence of significant associations between specific drugs and adverse reactions, or between pairs of drugs. As a specific example, DuMouchel presents a Bayesian characterization of interesting drug-event combinations [9]. This database is examined here for two reasons: first, that it represents a real database of moderately large size as a practical testbed for the analysis methods described here and second, because unrecognized structure in a dataset can have a deleterious influence on analyses based on working assumptions that are inconsistent with this structure. As a specific example, Dodge [8] discusses the analysis of a dataset that has been adopted as a standard benchmark in the applied statistics literature (the Brownlee stack-loss dataset), noting that many authors have analyzed this dataset based on the assumption that the measurements represented a uniformly sampled time-series. He argues convincingly that this is not the case, bringing a number of these earlier conclusions (e.g., identification of specific observations as outliers) into question.

The AERS database is organized by quarter and year, and the specific data values used in both examples considered here were obtained from the the following five datasets from first quarter 2001 portion of this database: DEMO01Q1 gives demographic information, REAC01Q1 lists specific adverse reactions, DRUG01Q1 lists the drugs involved, OUTC01Q1 gives outcome information (e.g., “death,” “hospitalization,” or “other”), and RPSR01Q1 gives information pertaining to the source of each adverse event report. These files are linked via an integer primary key designated the ISR (Individual Safety Report) number for each record, which corresponds to a report logged by the FDA. For example, each record in the DEMO01Q1 dataset consists of the ISR, together with 13 other values, including the date the manufacturer first received information reported to the FDA, the name of the manufacturer sending the report, and the age and gender of the patient associated with the report. Similarly, each record in the REAC01Q1 data file consists of the ISR and a single character string describing a reported reaction. Since each report typically lists more than one adverse reaction, however, ISR’s generally appear more than

once in the REAC01Q1 dataset, unlike the DEMO01Q1 dataset, where each ISR appears only once. Analogous observations apply to the DRUG01Q1 and OUTC01Q1 data files: each adverse event report may have more than one entry.

6.1 Application 1: missing data As is often the case, the fraction of missing data in the datasets that make up the AERS database varies strongly between fields. For example, the file DEMO01Q1 contains 51,012 records, each corresponding to a unique ISR number. Since this field is used as the primary access key for matching records across the first quarter 2001 datasets, is extremely well-maintained, containing neither duplicate nor missing entries. Similarly, the FDA report date, the field indicating the date that the FDA was notified of the reported adverse event, is also free of missing data. In contrast, some of the other twelve fields exhibit significant fractions of missing data:

- patient age: $\sim 27.5\%$ missing,
- patient gender: $\sim 6.8\%$ missing,
- manufacturer reporting date: $\sim 8.4\%$ missing,
- date of adverse event: $\sim 28.7\%$ missing.

Further, certain subsets of the data may exhibit much higher missing data percentages. As a specific example, consider the set of records for which both the manufacturer reporting date and the date of adverse event are missing. While this situation only arises in $\sim 0.9\%$ of the total data records, within this subset of records, gender is also missing about 38% of the time, and age is also missing about 59% of the time.

To explore this case further, consider the following question: do these missing records occur at random throughout the DEMO01Q1 data file, or do they exhibit evidence of significant patchiness? One reason this question is interesting is that, if these particular incomplete records group together by ISR number, they may also exhibit other common characteristics that are of significantly greater interest. To consider this question, define the binary status sequence $\{z_k\}$ as:

$$z_k = \begin{cases} 1 & \text{if both event date and manufacturer date} \\ & \text{are missing,} \\ 0 & \text{otherwise.} \end{cases} \quad (6.18)$$

Values of $\hat{\psi}_w$ were computed for $w = 1$ to $w = 20$ and these results satisfied the normalization condition, indicating that all patches had been found. Comparing these results with the corresponding values $\tilde{\psi}_w^j$ for $M = 200$ random permutations show that too few patches of

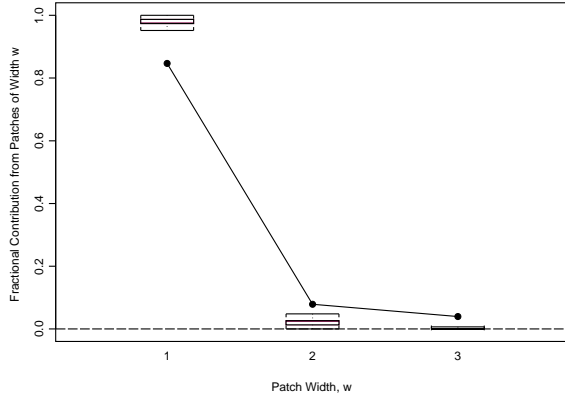


Figure 7: Computed patch spectrum $\hat{\psi}_w$ and $M = 200$ corresponding random permutation results for the AERS missing data status sequence $\{z_k\}$ for patch widths $w = 1, 2$, and 3 .

width $w = 1$ were seen in the sequence $\{z_k\}$ relative to the Bernoulli alternative, and too many patches of widths $2, 3, 4, 5$ and 7 were observed.

Fig. 7 shows these results for patch widths $w = 1, 2$ and 3 . Specifically, the solid circles in this plot represent the patch spectrum values $\hat{\psi}_w$ computed from the original status sequence for these values of w , while the boxplots summarize the range of $\tilde{\psi}_w^j$ values obtained for the 200 random permutations considered here. It is clear from this plot that the value $\hat{\psi}_1$ lies well below the range of the randomizations, indicating as noted above that there are too few patches of width $w = 1$, relative to the Bernoulli alternative. Similarly, the values of $\hat{\psi}_2$ and $\hat{\psi}_3$ both lie above the range of the randomization values.

The patch spectrum results $\hat{\psi}_w$ and their associated randomizations $\tilde{\psi}_w^j$ are summarized in Fig. 8 for patch widths $w = 3$ through $w = 10$. This plot was separated from Fig. 7 to emphasize small but significant details; in particular, note that the scale of Fig. 7 spans the range from 0 to 1 on the vertical axis, while Fig. 8 spans the narrower range from 0 to 0.05 . This plot emphasizes that, while patches of width $w = 3$ do occur in the randomizations, they are much less frequent than in the non-randomized results. In addition, it is clear that patches of width $w = 4, 5$, and 7 appear in the original sequence but *not* in the 200 randomizations.

It is instructive to examine the results for $w = 7$ in more detail, which corresponds to a single patch. Examination of these seven adverse event reports reveals that:

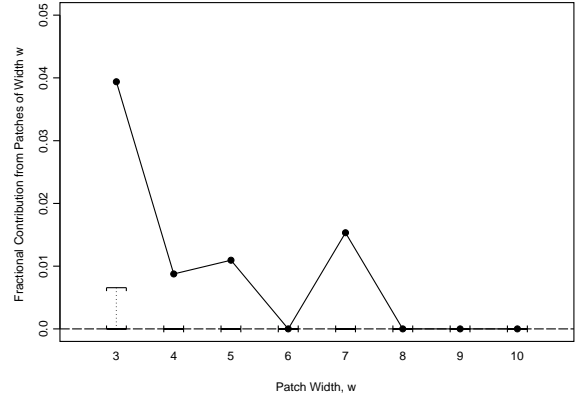


Figure 8: Computed patch spectrum $\hat{\psi}_w$ and $M = 200$ corresponding random permutation results for the AERS missing data status sequence $\{z_k\}$ for patch widths $w = 3$ through $w = 10$.

- the same FDA report date is listed for all ISR's,
- both age and gender are always missing,
- none of the fields in dataset DEMO01Q1 that identify manufacturer contain entries,
- none of these ISR's have a corresponding entry in the RPSR01Q1 report source information file,
- the ISR's all implicate different drugs,
- *all ISR's list the same, single reaction: "drug maladministration."*

The key points here are first, that this collection of seven successive records share many unusual characteristics in common. Hence, even though ISR number is a completely un-interesting data field by itself, patches of successive ISR's sharing a few anomalous characteristics (here, missing manufacturer report date and event date) actually share a much wider range of unusual characteristics. The second key point is that the patches detected by the method described here represent *extremely* small subsets of the data: in this example, a sequence of 7 anomalous records is detected, out of a total of $51,012$ (approximately 0.01%).

6.2 Application 2: death outcomes Of the $51,012$ adverse event reports with records in the DEMO01Q1 dataset, $5,048$ have "death" listed as an outcome in the OUTC01Q1 dataset. Here, we adopt this outcome as a subjective interestingness measure

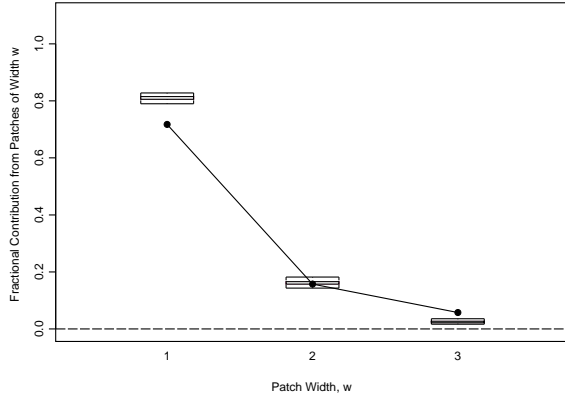


Figure 9: ISR patch spectrum $\hat{\psi}_w$ for patches of width $w = 1, 2$ or 3 for the 1Q 2001 AERS data listing “death” as the outcome.

[12] and consider the following status sequence:

$$(6.19) z_k = \begin{cases} 1 & \text{if ISR } k \text{ has outcome “death,”} \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 9 shows the patch spectrum $\hat{\psi}_w$ values for $w = 1, 2$, and 3 , denoted by solid circles connected by a line, with the results from 200 random permutations shown as boxplots. It is clear from this plot that isolated ISR’s (i.e., patches of width $w = 1$) occur less frequently than would be expected under the Bernoulli alternative, that patches of width $w = 2$ are consistent with what we would expect from a Bernoulli sequence, and patches of width $w = 3$ occur more frequently than we would expect for a Bernoulli sequence. Fig. 10 shows the corresponding results for w between 3 and 20 , again plotted on a different scale to show the details more clearly. It may be seen from this figure that the status sequence $\{z_k\}$ defined in Eq. (6.19) exhibits an unusually large number of patches of widths 3 through 10 . Even more unusual is the result for patches of width $w = 19$, which have essentially zero probability under the Bernoulli model.

A more complete quantitative summary of these results is given in Table 1, which lists the values computed for $\hat{\psi}_w$, the associated z-score z_w , the corresponding α_w value, and the number of patches N_w for all results giving nonzero $\hat{\psi}_w$ values between $w = 1$ and $w = 50$. Since these $\hat{\psi}_w$ values sum to 1 , they account for all of the ISR’s listing “death” as their associated outcome. In fact, it follows from the results shown in Table 1 that approximately 12.5% of the death ISR’s occur in successive groupings of width 3 or more, with the most extreme one having a width of $w = 35$. Also, note that

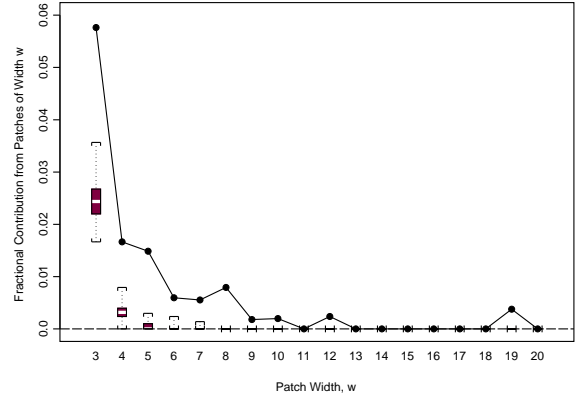


Figure 10: ISR patch spectrum $\hat{\psi}_w$ for patches of width $w = 3$ through $w = 20$ for the 1Q 2001 AERS data listing “death” as the outcome.

this example illustrates the separate utility of the z_w and α_w values. In particular, the fact that the α_w values are small but nonzero means that, while no single patch width $w > 2$ contributes a majority of interesting ISR’s, these patches are all significant relative to the Bernoulli alternative. In particular, it is not difficult to show that $\alpha_w \leq \hat{\psi}_w$ whenever $\alpha_w > 0$ and that this bound holds with equality if and only if $\hat{\psi}_w^+ = 0$, further implying that $\hat{\psi}_w^j = 0$ for all randomizations j . In this case it also follows that $\tilde{\sigma} = 0$, which is responsible for the infinite z-scores shown in Table 1 for $w = 8, 9, 10, 12, 19$, and 35 since this condition holds for these cases.

It is particularly instructive to look at the most extreme case, $w = 35$. As indicated in Table 1, only a single patch of this width occurs, and examining the corresponding records from the DEMO01Q1 file reveal that 31 of these 35 successive ISR’s share a common reporting manufacturer (Company A). To examine this result further, we can adopt this reporting manufacturer as a measure of interestingness and repeat the previous analysis. Specifically, define the binary status sequence:

$$(6.20) z_k = \begin{cases} 1 & \text{if reported by Company A,} \\ 0 & \text{otherwise.} \end{cases}$$

A plot of the patch spectrum computed from this sequence is shown in Fig. 11 for $w = 1$ through $w = 10$. Although this plot does not show the peak at width $w = 31$ that led to the construction and examination of this status sequence, it is clear from Fig. 11 that the number of isolated ISR’s is vastly smaller than would be expected under the Bernoulli alternative, and that the number of patches of widths 2 through 8 is much larger than would be expected. Even more significantly,

w	$\hat{\psi}_w$	z_w	α_w	N_w
1	0.7173	-13.0	-0.0918	3621
2	0.1573	-0.7	0.0000	397
3	0.0576	9.5	0.0228	97
4	0.0166	9.0	0.0088	21
5	0.0149	24.0	0.0119	15
6	0.0059	22.5	0.0036	5
7	0.0055	56.5	0.0042	4
8	0.0079	$+\infty$	0.0079	5
9	0.0018	$+\infty$	0.0018	1
10	0.0020	$+\infty$	0.0020	1
<hr/>				
12	0.0024	$+\infty$	0.0024	1
19	0.0038	$+\infty$	0.0038	1
35	0.0069	$+\infty$	0.0069	1

Table 1: Patch spectrum $\hat{\psi}_w$, z-scores z_w , α_w values and number of patches of width w present in the status sequence for the AERS ISR’s with outcome “death.”

the single patch of width 31 in the Company A status sequence—the only patch of width wider than 10 in this sequence—accounts for approximately 26% of the total number of ISR’s associated with this manufacturer.

Further examination of the Company A results reveals the following details. Altogether, this company appears as reporting manufacturer in 118 ISR’s. Of these, 111 list “death” as an outcome, with the following additional characteristics:

- patient gender is missing in all ISR’s,
- one or both of the following reactions is listed for every ISR: “Non-Accidental Overdose,” or “Overdose Nos (Not Otherwise Specified),”
- the same manufacturer date is listed, corresponding to the date the manufacturer initially recieved notification of the adverse event.

In view of these results, it seems likely that the patchiness seen in these ISR sequence is due to the manner in which these adverse events were reported and processed by the FDA. Despite the fact that this patch generation mechanism is not especially interesting, it has led us to focus on a very interesting group of ISR’s. In particular, the results presented here demonstrate that the detection and interpretation of patchiness in sequences of data records can ultimately lead us to groups

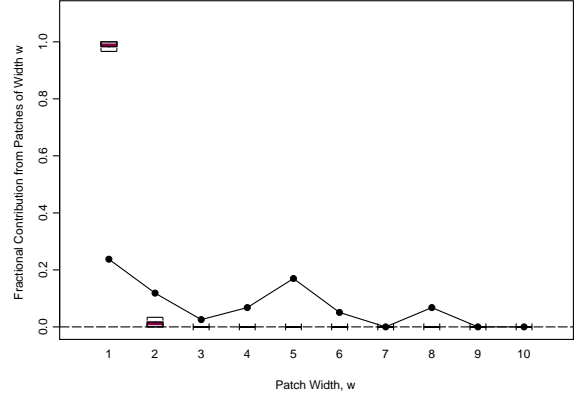


Figure 11: Patch spectrum for the Company A binary sequence defined in Eq. (6.20).

of records that are very strongly associated by characteristics that may be of significant interest (e.g., death by drug overdose).

7 Summary

This paper has considered the problem of detecting, characterizing, and interpreting non-random membership patterns of some class \mathcal{K} of data records in a larger dataset. Specific examples include missing or incomplete records, various types of data anomalies (e.g., outliers, inliers, or near-duplicate records), or record classes selected by either objective or subjective interestingness measures. The fundamental basis for these results is the binary status sequence $\{z_k\}$ defined in Eq. (1.2) indicating whether the record \mathcal{R}_k belongs to class \mathcal{K} or not. The main tools introduced here to characterize this sequence are the patch spectrum $\hat{\psi}_w$ introduced in Sec. 4 and the associated summary statistics z_w and α_w introduced in Sec. 5. The effectiveness of these tools was demonstrated first for a pair of simulation-based examples (a non-patchy Bernoulli sequence and a sequence exhibiting only random patches of width $w = 3$), and then with respect to status sequences constructed from the FDA’s AERS adverse event database. These examples illustrate that, even though the patches appearing in these record sequences are almost certainly data entry artifacts, they are interesting because the records involved were processed together for a reason related to some underlying common structure. For example, observation that 31 of 35 records in the longest observed patch of successive ISR’s with “death” listed as an outcome were associated with the same manufacturer led to an examination of all ISR’s associated with this man-

ufacturer. In turn, this led to the discovery that 111 of the 118 reports associated with this manufacturer were fatal drug overdoses, all with the same reporting date.

These examples also illustrated that, even in very long binary status sequences, the observation of wide patches is rare enough that they are easy to detect even if they represent an extremely small portion of the total sequence. For example, the AERS sequences considered here were of length 51,012, but the analysis methods presented here had no difficulty detecting single patches of width ~ 10 as unexpected features in the data, even though they correspond to $\sim 0.02\%$ of the data. As a corollary, this observation means that even if some unusual event or data recording anomaly places a single small patch of interesting records together, the fact that they are grouped together greatly enhances our ability to detect them. As a practical matter, even if this grouping is primarily due to the details of the data entry procedure, the fact that a group of records sharing the same characteristic of interest were entered together usually means that these records share several characteristics in common, as in the examples considered here. The results presented here suggest that patchiness analysis may be a very useful first step in uncovering these associations.

Finally, note that once we have constructed the binary status sequence $\{z_k\}$, we can apply a range of standard binary data analysis methods like logistic regression to explore possible relationships with other variables [5]. Alternatively, since $\{z_k\}$ defines a binary classification of records, we can also adopt the methodology of *case-control studies* [5, p. 217] or *case-referent studies* [19, p. 7]. There, the idea is to match each member of the “interesting class” (i.e., each record \mathcal{R}_k with $z_k = 1$) to one or more records from the “nominal class” (i.e., records \mathcal{R}_k with $z_k = 0$), usually subject to an approximate matching constraint on other record characteristics. The objective of these studies is to identify systematic differences in other characteristics that may be responsible for the difference in interestingness.

References

- [1] J. Albert and P. Williamson, “Using Model/Data Simulations to Detect Streakiness,” *Amer. Statistician*, vol. 55, 2001, pp. 41–50.
- [2] P. Billingsley, *Probability and Measure*, 2nd ed., Wiley, 1986.
- [3] A.Z. Broder, “Identifying and Filtering Near-Duplicate Documents,” in *Combinatorial Pattern Matching*, R. Giancarlo and D. Sankoff, eds., Springer-Verlag, 2000, pp. 1–10.
- [4] Y.B. Cheung, “Zero-inflated models for regression analysis of count data: a study of growth and development,” *Statist. Med.*, v. 21, 2002, pp. 1461–1469.
- [5] D. Collett, *Modelling Binary Data*, 2nd ed., Chapman and Hall, 2003.
- [6] L. Davies and U. Gather, “The identification of multiple outliers,” *J. Amer. Statist. Assoc.*, v. 88, 1993, pp. 782–801.
- [7] D. DesJardins, “Outliers, Inliers and Just Plain Liars—New Graphical EDA+ (EDA Plus) Techniques for Understanding Data,” *CEASAR Conf. Proc. Statistics Italy*, Rome, 2001, paper no. 169–26.
- [8] Y. Dodge, “The Guinea Pig of Multiple Regression,” in *Robust Statistics, Data Analysis, and Computer Intensive Methods*, H. Rieder, ed., Springer-Verlag, 1986, pp. 91–117.
- [9] W. DuMouchel, “Bayesian Data Mining in Large Frequency Tables, with an Application to the FDA Spontaneous Reporting System (with discussion),” *American Statistician*, v. 53, 1999, pp. 177–202.
- [10] P. Good, *Permutation Tests*, Springer-Verlag, 2000.
- [11] D.B. Hall and K.S. Berenhaut, “Score tests for heterogeneity and overdispersion in zero-inflated Poisson and binomial regression models,” *Canadian J. Statistics*, v. 30, 2002, pp. 1–16.
- [12] R.J. Hilderman and H.J. Hamilton, “Evaluation of Interestingness Measures for Ranking Discovered Knowledge,” *Proc. 5th Pacific-Asia Conf. Knowledge Discovery and Data Mining*, 2001, pp. 247–259.
- [13] K.-M. Lee, P. Meer, and R.-H. Park, “Robust Adaptive Segmentation of Range Images,” *IEEE Trans. Pattern Analysis Machine Intelligence*, v. 20, 1998, pp. 200–205.
- [14] R.D. Martin and V.J. Yohai, “Influence functionals for time-series,” *Ann. Statist.*, vol. 14, 1986, pp. 781–785.
- [15] R.K. Pearson, *Discrete-Time Dynamic Models*, Oxford, 1999.
- [16] R.K. Pearson, “Outliers in Process Modelling and Identification,” *IEEE Trans. Control Systems Technology*, v. 10, 2001, pp. 55–63.
- [17] R.K. Pearson, T. Zylkin, J.S. Schwaber, and G.E. Gonye, “Quantitative Evaluation of Clustering Results Using Computational Negative Controls,” *Proc. 2004 SIAM International Conference on Data Mining*, April, 2004, Lake Buena Vista, Florida, pp. 188–199.
- [18] D. Pyle, *Data Preparation for Data Mining*, Academic Press, 1999.
- [19] P.R. Rosenbaum, *Observational Studies*, 2nd ed., Springer-Verlag, 2002.
- [20] D.B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, Wiley, 1987.
- [21] W.E. Winkler, “Problems with Inliers,” paper presented at the European Conference of Statisticians, Prague, October, 1997.
- [22] Q. Zheng, K. Xu, W. Lv, and S. Ma, “Intelligent Search of Correlated Alarms from Database Containing Noise Data,” *Proc. 8th IEEE/IFIP Network and Operations Management Symposium (NOMS)*, Florence, Italy, 2002, pp. 405–419.

2-Dimensional Singular Value Decomposition for 2D Maps and Images

Chris Ding* and Jieping Ye†

LBNL-56481. October 3, 2004.

Abstract

For a set of 1D vectors, standard singular value decomposition (SVD) is frequently applied. For a set of 2D objects such as images or weather maps, we form 2dSVD, which computes principal eigenvectors of row-row and column-column covariance matrices, exactly as in the standard SVD. We study optimality properties of 2dSVD as low-rank approximation and show that it provides a framework unifying two recent approaches. Experiments on images and weather maps illustrate the usefulness of 2dSVD.

1 Introduction

Singular value decomposition (SVD)[5, 7] plays the central role in reducing high dimensional data into lower dimensional data which is also called principal component analysis (PCA)[8] in statistics. It often occurs that in the reduced space, coherent patterns can be detected more clearly. Such unsupervised dimension reduction is used in very broad areas such as meteorology[11], image processing[9, 13], and information retrieval[1].

The problem of low rank approximations of matrices has recently received broad attention in areas such as computer vision, information retrieval, and machine learning [1, 2, 3, 12]. It becomes an important tool for extracting correlations and removing noise from data. However, applications of this technique to high-dimensional data, such as images and videos, quickly run up against practical computational limits, mainly due to the high time and space complexities of the SVD computation for large matrices.

In recent years, increasingly more data items come naturally as 2D objects, such the 2D images, 2D weather maps. Currently widely used method for dimension reduction of these 2D data objects is based on SVD. First, 2D objects are converted into 1D vectors and are packed together as a large matrix. For example, each of the 2D maps of $A_i, A_i \in \mathbb{R}^{r \times c}, i = 1, \dots, n$ is converted to a vector \mathbf{a}_i of length rc . The standard

SVD is then applied to the matrix containing all the vectors: $A = (\mathbf{a}_1, \dots, \mathbf{a}_n)$. In image processing, this is called Eigenfaces[9]. In weather research, this is called Empirical Orthogonal Functions (EOF) [11]. Although the conventional approach is widely used, it does not preserve the 2D nature of these 2D data objects.

Two recent studies made first proposals to capture the 2D nature explicitly in low rank approximation. Yang *et al.* [13] propose to use the principal components of (column-column) covariance matrix for image representation. Ye *et al.* [14, 15] propose to use a $LM_i R^T$ type decomposition for low rank approximation.

In this paper, we propose to construct 2-dimensional singular value decomposition (2dSVD) based on the row-row and column-column covariance matrices. We study various optimality properties of 2dSVD as low-rank approximation. We show that the approach of Yang *et al.* [13] can be casted as a one-sided low-rank approximation with its optimal solution given by 2dSVD. 2dSVD also gives a near-optimal solution for the low rank approximation using $LM_i R^T$ decomposition by Ye [14]. Thus 2dSVD serves as a framework unifying the work of Yang *et al.* [13] and Ye [14].

Together, this new approach captures explicitly the 2D nature and has 3 advantages over conventional SVD-based approach: (1) It deals with much smaller matrices, typically $r \times c$ matrices, instead of $n \times (rc)$ matrix in conventional approach. (2) At the same or better accuracy of reconstruction, the new approach requires substantially smaller memory storage. (3) Some of the operations on these rectangular objects can be done much more efficiently, due to the preservation of the 2D structure.

We note there exists other type of decompositions of high order objects. The recently studied orthogonal tensor decomposition [16, 10], seeks an f -factor trilinear form for decomposition of X into A, B, C : $x_{ijk} = \sum_{\alpha=1}^f \sum_{\beta=1}^f \sum_{\gamma=1}^f a_{i\alpha} b_{j\beta} c_{k\gamma}$ where columns of A, B, C mutually orthogonal within each matrices.

Our approach differs in that we keep explicit the 2D nature of these 2D maps and images. For weather map, the i, j dimensions are longitude and latitude which are of same nature. For 2D images, the i, j dimensions are vertical and horizontal dimensions, which are of the same

*Lawrence Berkeley National Laboratory, University of California, Berkeley, CA 94720. Email: chqding@lbl.gov

†Department of Computer Science, University of Minnesota, Minneapolis, MN 55455. Email: jieping@cs.umn.edu

nature. The k dimension refers to different data objects. (In contrast, in the multi-factor trilinear orthogonal decomposition, the i, j, k dimensions are of different nature, say “temperature”, “intensity”, “thickness”.)

These inherently 2D datasets are very similar to 1D vector datasets, $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, for which the singular value decomposition (SVD) is often applied to obtain the optimal low-rank approximation:

$$(1.1) \quad X \approx \tilde{X}, \quad \tilde{X} = U_k \Sigma_k V_k^T, \quad \Sigma_k = U_k^T X V_k,$$

where U_k contains k principal eigenvectors of the covariance matrix¹ XX^T and V contains k principal eigenvectors of the inner-product matrix $X^T X$.

We define 2-dimensional SVD for a set of 2D maps in the same way as SVD is computed for a set of 1D vectors. Define the averaged row-row and column-column covariance matrices,

$$(1.2) \quad \begin{aligned} F &= \sum_{i=1}^n (A_i - \bar{A})(A_i - \bar{A})^T, \\ G &= \sum_{i=1}^n (A_i - \bar{A})^T (A_i - \bar{A}). \end{aligned}$$

where $\bar{A} = \sum_i A_i / n$.¹ F corresponds to XX^T and G corresponds to $X^T X$. Let U_k contains k principal eigenvectors of F and V_s contains s principal eigenvectors of G :

$$(1.3) \quad F = \sum_{\ell=1}^r \lambda_\ell \mathbf{u}_\ell \mathbf{u}_\ell^T, \quad U_k \equiv (\mathbf{u}_1, \dots, \mathbf{u}_k);$$

$$(1.4) \quad G = \sum_{\ell=1}^c \zeta_\ell \mathbf{v}_\ell \mathbf{v}_\ell^T, \quad V_s \equiv (\mathbf{v}_1, \dots, \mathbf{v}_s).$$

Following Eq.(1.1), we define

$$(1.5) \quad \tilde{A}_i = U_k M_i V_s^T, \quad M_i = U_k^T A_i V_s, \quad i = 1, \dots, n,$$

as the extension of SVD to 2D maps. We say $(U_k, V_s, \{M_i\}_{i=1}^n)$ form the 2dSVD of $\{A_i\}_{i=1}^n$. In standard SVD of Eq.(1.1), U_k provides the common subspace basis for 1D vectors to project to. In 2dSVD, U_k, V_s provide the two common subspace bases for 2D maps to (right and left) project to (this will become more clear in §3, §4 §5). Note that $M_i \in \mathbb{R}^{k \times s}$ is not required to be diagonal, whereas in standard SVD, Σ_k is diagonal.

For standard SVD, the eigenvalues of XX^T and $X^T X$ are identical, $\lambda_\ell = \zeta_\ell = \sigma_\ell^2$. The Eckart-Young Theorem[5] states that the residual error

$$(1.6) \quad \left\| X - \sum_{\ell=1}^k \mathbf{u}_\ell \sigma_\ell \mathbf{v}_\ell^T \right\|^2 = \sum_{\ell=k+1}^r \sigma_\ell^2.$$

¹In general, SVD is applied to any rectangular matrix. while PCA applying SVD on centered data: $X = (\mathbf{x}_1 - \bar{\mathbf{x}}, \dots, \mathbf{x}_n - \bar{\mathbf{x}})$, $\bar{\mathbf{x}} = \sum_i \mathbf{x}_i / n$. In the rest of this paper, we assume $\bar{A} = 0$ to simplify the equations. This can be recovered by $A_i \rightarrow A_i - \bar{A}$.

We will see that 2dSVD has very similar properties.

Obviously, 2dSVD provides a low rank approximation of the original 2D maps $\{A_i\}$. In the following we provide detailed analysis and show that 2dSVD provides (near) optimal solutions to a number of different types of approximations of $\{A_i\}$.

2 Optimality properties of 2dSVD

Definition. Given a 2D map set $\{A_i\}_{i=1}^n$, $A_i \in \mathbb{R}^{r \times c}$, we define the low rank approximation

$$(2.7) \quad \begin{aligned} A_i &\approx \tilde{A}_i, \quad \tilde{A}_i = L M_i R^T, \\ L &\in \mathbb{R}^{r \times k}, \quad R \in \mathbb{R}^{c \times s}, \quad M_i \in \mathbb{R}^{k \times s}. \end{aligned}$$

Here k, s are input parameters for specifying the rank of the approximation. We require L, R have orthonormal columns $L^T L = I_k$, $R^T R = I_s$. A less strict requirement is: columns of L be linearly independent and columns of R be linearly independent. However, given a fixed L, R with these constraints, we can do QR factorization to obtain $L = Q_L \tilde{L}$ and $R = Q_R \tilde{R}$ where Q_L, Q_R are orthogonal. We can write $L M_i R^T = Q_L \tilde{L} M_i \tilde{R} Q_R^T = Q_L \tilde{M}_i Q_R^T$. This is identical to the form of $L M_i R^T$.

The 2dSVD of Eq.(1.5) is clearly one such approximation:

$$(2.8) \quad L = U_k, \quad R = V_s, \quad M_i = U_k^T A_i V_s.$$

What's the significance of 2dSVD?

- The optimal solution for the low-rank approximation using the *1-sided* decomposition

$$(2.9) \quad \min_{M_i \in \mathbb{R}^{r \times k}, R \in \mathbb{R}^{c \times k}} J_1(\{M_i\}, R) = \sum_{i=1}^n \|A_i - M_i R^T\|^2$$

is given by the 2dSVD: $R = V_k$, $M_i = A_i V_k$.

- The optimal solution for the *1-sided* low-rank approximation

$$(2.10) \quad \min_{L \in \mathbb{R}^{r \times k}, M_i \in \mathbb{R}^{c \times k}} J_2(L, \{M_i\}) = \sum_{i=1}^n \|A_i - L M_i^T\|^2$$

is given by the 2dSVD: $L = U_k$, $M_i = A_i^T U_k$.

- The 2dSVD gives a near-optimal solution for the low-rank approximation using the *2-sided* decomposition [14]

$$(2.11) \quad \begin{aligned} \min_{L \in \mathbb{R}^{r \times k}, R \in \mathbb{R}^{c \times s}, M_i \in \mathbb{R}^{k \times s}} J_3(L, \{M_i\}, R) \\ = \sum_{i=1}^n \|A_i - L M_i R^T\|^2. \end{aligned}$$

When $k = r$, $\min J_3$ reduces to $\min J_1$. When $s = c$, $\min J_3$ reduces to $\min J_2$.

- When $A_i = A_i^T, \forall i$, the 2dSVD gives a near-optimal solution for the symmetric approximation (2.12)

$$\min_{L \in \mathbb{R}^{r \times k}, M_i \in \mathbb{R}^{k \times k}} J_4(L, \{M_i\}) = \sum_{i=1}^n \|A_i - LM_i L^T\|^2.$$

2dSVD provides a unified framework for rectangular data matrices. Our 2dSVD generalizes the work of Yang *et al.* [13] where they consider only the matrix G , but give no discussion on the optimality property and the objective of the approximation. On other hands, the 2dSVD provides a near-optimal solution of the 2D low rank approximation of Ye [14], the symmetric decomposition of J_3 which we believe is key to the low rank approximation of these rectangular data matrices.

We discuss the 3 decompositions in details in §3, §4, §5.

3 $A_i = M_i R^T$ Decomposition

Theorem 1. The *global* optimal solution for $A_i = M_i R^T$ approximation of J_1 in Eq.(2.9) is given by

$$(3.13) \quad R = V_k, M_i = A_i V_k,$$

$$J_1^{\text{opt}} = \sum_i \|A_i - A_i V_k V_k^T\|^2 = \sum_{j=k+1}^c \zeta_j.$$

Remark. Theorem 1 is very similar to Eckart-Young Theorem of Eq.(1.6) in that the solution is given by the principal eigenvectors of the covariance matrix and the residual is the sum of the eigenvalues of the retained subspace.

Note that this solution is unique² up to an arbitrary k -by- k orthogonal matrix Γ : for any given solution $(L, \{M_i\})$, $(L\Gamma, \{M_i\Gamma\})$ is also a solution with the same objective value. When $k = c$, R becomes a full rank orthogonal matrix, i.e., $RR^T = I_c$. In this case, we set $R = I_c$ and $M_i = A_i$.

Proof. Using $\|A\|^2 = \text{Tr}(A^T A)$, and $\text{Tr}(AB) = \text{Tr}(BA)$, we have

$$\begin{aligned} J_1 &= \sum_{i=1}^n \text{Tr}(A_i - M_i R^T)^T (A_i - M_i R^T) \\ &= \text{Tr} \sum_{i=1}^n [A_i^T A_i - 2A_i^T M_i R^T + M_i M_i^T] \end{aligned}$$

This is a quadratic function w.r.t. M_i . The minimum occur at where the gradient is zero: $0 = \partial J_1 / \partial M_i = -2A_i R + 2M_i$. Thus $M_i = A_i R$. With this, we have

$$J_1 = \sum_{i=1}^n \|A_i\|^2 - \text{Tr}[R^T (\sum_{i=1}^n A_i^T A_i) R]$$

²If eigenvalue $\zeta_j, j \leq k$ is degenerate, the corresponding columns of V_k could be any orthogonal basis of the subspace, therefore not unique.

Now $\min_R J_1$ becomes

$$\max_{R | R^T R = I_k} J_{1a} = \text{Tr}(R^T G R)$$

By a well-known result in algebra, the optimal solution for R is given by $R = (\mathbf{v}_1, \dots, \mathbf{v}_k)\Gamma$, Γ is an arbitrary k -by- k orthogonal matrix noted earlier. The optimal value is the sum of the large k eigenvalues of G : $J_{1a}^{\text{opt}} = \sum_{j=1}^k \zeta_j$. Note that

$$(3.14) \quad \sum_{j=1}^c \zeta_j = \text{Tr}(V_c^T G V_c) = \text{Tr}(G) = \sum_i \|A_i\|^2.$$

Here we have used the fact that $V_c V_c^T = I$ because V_c is a full rank orthonormal matrix. Thus $J_1^{\text{opt}} = \sum_{i=1}^n \|A_i\|^2 - \sum_{j=1}^k \zeta_j = \sum_{j=k+1}^c \zeta_j$.

To see why this is the global optimal solution, we first note that for any solution \tilde{M}_i, \tilde{R} , the zero gradient condition holds, i.e, $\tilde{M}_i = A_i^T \tilde{R}$. With this, we have $J_1 = \sum_{i=1}^n \|A_i\|^2 - \text{Tr} \tilde{R}^T G \tilde{R}$. Due to the positive definiteness of G , the solution for the quadratic function must be unique, up to an arbitrary rotation: $\tilde{R} = R\Gamma$. ■

4 $A_i = L M_i^T$ Decomposition

Theorem 2. The *global* optimal solution for $A_i = L M_i^T$ approximation of J_2 in Eq.(2.10) is given by

$$(4.15) \quad L = U_k, M_i = A_i^T U_k,$$

$$J_2^{\text{opt}} = \sum_i \|A_i - U_k U_k^T A_i\|^2 = \sum_{j=k+1}^r \lambda_j.$$

The proof is identical to Theorem 1, using the relation

$$(4.16) \quad \sum_{j=1}^r \lambda_j = \text{Tr}(U_r^T F U_r) = \text{Tr}(F) = \sum_i \|A_i\|^2.$$

For this decomposition, when $k = r$, we have $L = I_r$ and $M_i = A_i^T$.

5 $A_i = L M_i R^T$ Decomposition

Theorem 3 The optimal solution for $A_i = L M_i R^T$ approximation of J_3 in Eq.(2.11) is given by

$$(5.17) \quad \begin{aligned} L &= \tilde{U}_k = (\tilde{\mathbf{u}}_1, \dots, \tilde{\mathbf{u}}_k), \\ R &= \tilde{V}_s = (\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_s), M_i = \tilde{U}_k^T A_i \tilde{V}_s, \end{aligned}$$

where $\tilde{\mathbf{u}}_k, \tilde{\mathbf{v}}_k$ are simultaneous solutions of the eigenvector problems

$$(5.18) \quad \tilde{F} \tilde{\mathbf{u}}_k = \tilde{\lambda}_k \tilde{\mathbf{u}}_k, \tilde{G} \tilde{\mathbf{v}}_k = \tilde{\zeta}_k \tilde{\mathbf{v}}_k,$$

of the reweighted covariance matrices \tilde{F} and \tilde{G} (see Eq.(1.2)) :

$$\begin{aligned} \tilde{F} &= \sum_i A_i R R^T A_i^T = \sum_i A_i \tilde{V}_s \tilde{V}_s^T A_i^T, \\ (5.19) \quad \tilde{G} &= \sum_i A_i^T L L^T A_i = \sum_i A_i^T \tilde{U}_k \tilde{U}_k^T A_i. \end{aligned}$$

The optimal objective function value is given by

$$\begin{aligned} J_3^{\text{opt}}(k, s) &= \sum_i \|A_i - \tilde{U}_k \tilde{U}_k^T A_i \tilde{V}_s \tilde{V}_s^T\|^2 \\ (5.20) \quad &= \sum_i \|A_i\|^2 - \sum_{j=1}^k \tilde{\lambda}_j \\ (5.21) \quad &\geq \sum_{j=k+1}^r \tilde{\lambda}_j + \sum_{j=s+1}^c \zeta_j, \\ (5.22) \quad &J_3^{\text{opt}}(k, s) = \sum_i \|A_i\|^2 - \sum_{j=1}^s \tilde{\zeta}_j \geq \sum_{j=k+1}^r \lambda_j + \sum_{j=s+1}^c \tilde{\zeta}_j. \end{aligned}$$

In the following special cases, the problem of maximization of J_3 is greatly simplified:

(A) When $k = r$, L becomes a full rank orthogonal matrix. In this case, $LL^T = I_c$, and we can set $L = I_r$. \tilde{G} becomes identical to G . The problem of maximization of J_3 is reduced to the maximization of J_2 .

(B) When $s = c$, R becomes a full rank orthogonal matrix. and can be set as $R = I_c$. \tilde{F} becomes identical to F . Maximization of J_3 is reduced to the maximization of J_1 .

(C) When $k = r$ and $s = c$, the optimization problem becomes trivial one: $L = I_r, R = I_c, M_i = A_i$.

Proof. We write $J_3 = \sum_{i=1}^n \text{Tr}[A_i^T A_i - 2LM_i R^T A_i^T + M_i^T M_i]$. Taking $\partial J_3 / \partial M_i = 0$, we obtain $M_i = L^T A_i R$, and $J_3 = \sum_{i=1}^n \|A_i\|^2 - \sum_{i=1}^n \|L^T A_i R\|^2$. Thus $\min J_3$ becomes

$$(5.23) \quad \max_{L, R} J_{3a}(L, R) = \sum_{i=1}^n \|L^T A_i R\|^2.$$

The objective can be written as

$$\begin{aligned} J_{3a}(L, R) &= \text{Tr} L^T \left(\sum_{i=1}^n A_i R R^T A_i^T \right) L = \text{Tr} L^T \tilde{F} L \\ (5.24) \quad &= \text{Tr} R^T \left(\sum_{i=1}^n A_i^T L L^T A_i \right) R = \text{Tr} R^T \tilde{G} R \end{aligned}$$

As solutions for these traces of quadratic forms, L, R are given by the eigenvectors of \tilde{F}, \tilde{G} , and the optimal value are given by the equalities in Eqs.(5.21, 5.22).

To prove the inequality in Eq.(5.21), we note

$$(5.25) \quad \sum_{j=1}^r \tilde{\lambda}_j = \text{Tr} \tilde{U}_r^T \left(\sum_i A_i \tilde{V}_s \tilde{V}_s^T A_i^T \right) \tilde{U}_r$$

$$(5.26) \quad = \text{Tr} \sum_i A_i \tilde{V}_s \tilde{V}_s^T A_i^T$$

$$(5.27) \quad = \text{Tr} \tilde{V}_s^T \left(\sum_i A_i^T A_i \right) \tilde{V}_s$$

$$(5.28) \quad \leq \text{Tr} V_s^T \left(\sum_i A_i^T A_i \right) V_s = \sum_{j=1}^s \zeta_j$$

Re-writing the RHS of above inequality using Eq.(3.14) and splitting the LHS into two terms, we obtain

$$\sum_{j=1}^k \tilde{\lambda}_j + \sum_{j=k+1}^r \tilde{\lambda}_j \leq \sum_i \|A_i\|^2 - \sum_{j=s+1}^c \zeta_j.$$

This gives the inequality in Eq.(5.21). The inequality in Eq.(5.22) can be proved in the same fashion. ■

In practice, simultaneous solutions of the \tilde{U}, \tilde{V} eigenvectors are achieved via an iterative process:

Iterative Updating Algorithm. Given initial r -by- k matrix $L^{(0)}$, we form \tilde{G} and solve for the k largest eigenvectors ($\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_s$) which gives $R^{(0)}$. Based on $R^{(0)}$, we form \tilde{F} and solve for the k largest eigenvectors ($\tilde{\mathbf{u}}_1, \dots, \tilde{\mathbf{u}}_k$) which gives $L^{(1)}$. This way, we obtain $L^{(0)}, R^{(0)}, L^{(1)}, R^{(1)}, \dots$.

Proposition 6. $J_{3a}(L, R)$ is step-wise nondecreasing, i.e.,

$$J_{3a}(L^{(0)}, R^{(0)}) \leq J_{3a}(L^{(1)}, R^{(0)}) \leq J_{3a}(L^{(1)}, R^{(1)}) \leq \dots$$

Proof. Suppose we have currently $L^{(t)}, R^{(t)}$. Using $L^{(t)}$ we form \tilde{G} , solve for k largest eigenvectors and obtain a new $R^{(t+1)}$. By definition, $R^{(t+1)}$ is the one that maximizes

$$\text{Tr} R^T \left(\sum_i A_i^T L^{(t)} L^{(t)T} A_i \right) R = \sum_i \|L^{(t)T} A_i R\|^2.$$

Thus $\sum_i \|L^T A_i R\|^2$ must be non-decreasing. Similarly, using $R^{(t)}$ we can form \tilde{F} , solve for k largest eigenvectors and obtain a new $L^{(t+1)}$. $\sum_i \|L^T A_i R\|^2$ must be also non-decreasing. ■

Proposition 7. An upper-bound exists for $\max \sum_i \|L^T A_i R\|^2$:

$$\max_{L \in \mathbb{R}^{r \times k}, R \in \mathbb{R}^{c \times s}} \sum_i \|L^T A_i R\|^2 < \min \left(\sum_{j=1}^k \lambda_j, \sum_{j=1}^s \zeta_j \right).$$

Proof. Assume $k < r$. For any r -by- k matrix L , with orthonormal columns, we can always find additional $r-k$

orthonormal columns \tilde{L} such that (L, \tilde{L}) span the space. Thus $LL^T + \tilde{L}\tilde{L}^T = I_r$. Noting that $\sum_i A_i^T(\tilde{L}\tilde{L}^T)A_i$ is positive definite, we have

$$\begin{aligned} \max_R \quad & \text{Tr} R^T \left(\sum_i A_i^T LL^T A_i \right) R \\ < \quad & \max_R \text{Tr} R^T \left(\sum_i A_i^T (LL^T + \tilde{L}\tilde{L}^T) A_i \right) R \\ = \quad & \max_R \text{Tr} R^T \left(\sum_i A_i^T A_i \right) R. \end{aligned}$$

From Eqs.(1.2,1.4), the solution to the right-hand-side is given by the 2dSVD: $R = V_s$. We can similarly show the upper-bound involving U_k . The eigenvalues arise from Eqs.(1.2,1.4). ■

From Proposition 6, we obtain a simple lower bound,

$$(5.29) \quad J_3^{\text{opt}}(k, s) \leq \sum_i \|A_i\|^2 - \min \left(\sum_{j=1}^s \lambda_j, \sum_{j=1}^s \zeta_j \right)$$

With the nondecreasing property (Proposition 6) and the upper-bound (proposition 7), we conclude that the iterative update algorithm converges to a local maximum.

Is the local maximum also a global maximum? We have several arguments and some strong numerical evidence to support

Observation 8. When $A_i = LM_i R^T$ decomposition provides a good approximation to the 2D data, the iterative update algorithm (IUA) converges to the global maximum.

Discussion. (A) For $n = 1$, 2dSVD reduces to usual SVD and the global maximum is well-known. Fixing L , J_{3a} is a quadratic function of R and the only local maximum is the global one, achieved in IUA. Similarly, fixing R , IUA achieves the global maximum. (B) We may let $L^{(0)} = U_k$ as in 2dSVD, any random matrices, or a matrix of zeroes except one element being 1. For any of these starting point, IUA always converges to the same final solution (L^*, R^*) in 3 iterations. (C) We initialize L as $L^{(0)} \perp L^*$, i.e, as $L^{(0)}$ has zero overlap with the solution L^* . We run IUA again. Typically in 3 iterations, the IUA converges to the same (L^*, R^*) .³ These three experiments indicate it is unlikely IUA can be trapped in a local maximum, if it exists.

5.1 Comparison with $A_i = LM_i, A_i = M_i R^T$
We compare $A_i = LM_i R^T$ with $A_i = M_i R^T$ and

$A_i = LM_i^T$. The computer storage for the three approximations are

$$(5.30) \quad S_{\text{LMR}} = rk + nks + sc = 204,000,$$

$$(5.31) \quad S_{\text{MR}} = nrk + kc = 1,002,000,$$

$$(5.32) \quad S_{\text{LM}} = rk + nkc = 1,002,000,$$

where the last number assumes $r = c = 100, n = 500$ and $k = s = 20$. The reconstruction errors, i.e., the objective function values, have the relationship:

$$(5.33) \quad J_1^{\text{opt}}(s) < J_3^{\text{opt}}(k, s), k < r; J_1^{\text{opt}}(s) = J_3^{\text{opt}}(r, s).$$

$$(5.34) \quad J_2^{\text{opt}}(k) < J_3^{\text{opt}}(k, s), s < c; J_2^{\text{opt}}(k) = J_3^{\text{opt}}(k, c).$$

This comes from Proposition 7 and noting $J_1^{\text{opt}} = \sum_{j=s+1}^c \zeta_j$ and $J_2^{\text{opt}} = \sum_{j=k+1}^r \lambda_j$ from Theorems 1 and 2.

From the expressions for J_1^{opt} , J_2^{opt} , and J_3^{opt} in Eqs.(3.13, 4.15), and Theorem 5, we see that A_i is either left projected to the subspace $U_k U_k^T$, right projected to the subspace $V_k V_k^T$ or left and right projected simultaneously.

2dSVD as near-optimal solution for J_3

6 Bounding J_3 by 2dSVD

In this section, we give upper bounds on J_3 and show 2dSVD is the solution for minimizing these upper bounds.

Upper bound J_{3L}

We first let $R_i \in \mathbb{R}^{c \times k}$ be a temporary replacement of $M_i R^T$. Using triangle inequality of Frobenius norm, we can write J_3 as

$$\begin{aligned} J_3 &= \sum_{i=1}^n \|A_i - LR_i^T + LR_i^T - LM_i R^T\|^2 \\ &\leq \sum_{i=1}^n \|A_i - LR_i^T\|^2 + \sum_{i=1}^n \|LR_i^T - LM_i R^T\|^2 \equiv J_{3L}. \end{aligned}$$

Since L has orthonormal columns, the second term becomes $\|R_i^T - M_i R^T\|^2 = \|R_i - RM_i^T\|^2$. Thus the optimization of J_{3L} can be written as

$$\min_{L, R_i} \sum_{i=1}^n \|A_i - LR_i^T\|^2 + \min_{\substack{R, M_i \\ R_i \text{ fixed}}} \sum_i \|R_i - RM_i^T\|^2.$$

The first term is identical to $\min J_2$, and the optimal solution is given by Theorem 2,

$$(6.35) \quad L = U_k, R_i = A_i^T U_k, J_{3L}^{(1)} = \sum_{j=k+1}^r \lambda_j.$$

³Due to existence of Γ as discussed in Theorem 1, we measure the angle between the two subspaces. For 1-D subspaces, it is the angle between the two lines. This is generalized to multi-dimensional subspaces [7].

The second term of J_{3L} is equivalent to $\min J_2$, and by Theorem 2 again, optimal solution are given by (6.36)

$$R = \hat{V}_s \equiv (\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_s), \quad M_i = U_k^T A_i R, \quad J_{3L}^{(2)} = \sum_{j=k+1}^c \hat{\zeta}_j,$$

where $\hat{\mathbf{v}}_k, \hat{\zeta}_k$ are eigenvectors and eigenvalues of the weighted covariance matrix \hat{G} :

$$(6.37) \quad \hat{G} \hat{\mathbf{v}}_k = \hat{\zeta}_k \hat{\mathbf{v}}_k, \quad \hat{G} = \sum_i A_i^T U_k U_k^T A_i.$$

Combining these results, we have

Theorem 5. Minimizing the upper bound J_{3L} leads to the following near-optimal solution for J_3 :

$$(6.38) \quad L = U_k, \quad R = \hat{V}_s, \quad M_i = U_k^T A_i \hat{V}_s,$$

$$(6.39) \quad J_3^{\text{opt}} \leq \sum_{j=k+1}^r \lambda_j + \sum_{j=s+1}^c \hat{\zeta}_j.$$

To implement Theorem 5, we (1) compute U_k ; (2) construct the reweighted row-row covariance \hat{G} of Eq.(6.37) and compute its s eigenvectors which gives \hat{V}_s ; (3) compute $M_i = U_k^T A_i \hat{V}_s$. This $U \rightarrow V \rightarrow M_i$ procedure is a variant of 2dSVD, instead of computing U_k and V_s independent of each other (see Eqs.(1.3, 1.4)). The variant has the same computational cost. We call this LRLMi. Note that, in the iterative update algorithm of J_3 , if we set $L^{(0)} = U_k$, then $R^{(0)} = \hat{V}_k$. This 2dSVD variant can be considered as the initialization of the iterative update algorithm.

Upper bound J_{3R}

Alternatively, we may first let $L_i \in \mathbb{R}^{c \times k}$ be a temporary replacement of LM_i . Using triangle inequality, we obtain another upper bound of J_3 ,

$$\begin{aligned} J_3 &= \sum_{i=1}^n \|A_i - L_i R^T + L_i R^T - LM_i R^T\|^2 \\ &\leq \sum_{i=1}^n \|A_i - L_i R^T\|^2 + \sum_{i=1}^n \|L_i R^T - LM_i R^T\|^2 \equiv J_{3R}. \end{aligned}$$

R has orthonormal columns and drops out of the second term. The optimization of J_{3R} can be written as

$$\min_{L_i, R} \sum_{i=1}^n \|A_i - L_i R^T\|^2 + \min_{\substack{L, M_i \\ L_i \text{ fixed}}} \sum_i \|L_i - LM_i\|^2.$$

Following the same analysis leading to Theorem 5, we obtain

Theorem 6. Minimizing the upper bound J_{3R} leads to

the following near-optimal solution for J_3 :

$$(6.40) \quad L = \hat{U}_k \equiv (\hat{\mathbf{u}}_1, \dots, \hat{\mathbf{u}}_k), \quad R = V_s,$$

$$(6.41) \quad M_i = \hat{U}_k^T A_i V_s,$$

$$(6.42) \quad J_3^{\text{opt}} \leq \sum_{j=k+1}^r \hat{\lambda}_j + \sum_{j=s+1}^c \zeta_j,$$

where $\hat{\mathbf{p}}_k$ are eigenvectors of the weighted covariance matrix \hat{F}

$$(6.43) \quad \hat{F} \hat{\mathbf{u}}_k = \hat{\zeta}_k \hat{\mathbf{u}}_k, \quad \hat{F} = \sum_i A_i V_s V_s^T A_i^T.$$

The implementations are: (1) compute V_s ; (2) construct the reweighted row-row covariance \hat{F} of Eq.(6.43) and compute its k eigenvectors which gives \hat{U}_k ; (3) compute M_i . This is another variant of 2dSVD, which we call RLMI.

7 Error Analysis of J_3 and 2dSVD

For $A_i = LM_i R^T$ decomposition, from Theorems 5 and 6, and Eqs.(5.21, 5.22), we obtain the following lower and upper bounds for J_3 :

$$(7.44) \quad l_b(k, s) \leq J_3^{\text{opt}}(k, s) \leq u_b(k, s),$$

$$(7.45)$$

$$l_b(k, s) = \max\left(\sum_{j=k+1}^r \tilde{\lambda}_j + \sum_{j=s+1}^c \zeta_j, \sum_{j=k+1}^r \lambda_j + \sum_{j=s+1}^c \tilde{\zeta}_j\right),$$

$$(7.46)$$

$$u_b(k, s) = \min\left(\sum_{j=k+1}^r \hat{\lambda}_j + \sum_{j=s+1}^c \zeta_j, \sum_{j=k+1}^r \lambda_j + \sum_{j=s+1}^c \hat{\zeta}_j\right).$$

We have seen how 2dSVD arises in minimizing the upper bounds J_{3L} and J_{3R} . Now we analyze it in subspace approximation point of view. Let \tilde{U}_k be the subspace complement of \tilde{U}_k , i.e., $(\tilde{U}_k, \tilde{U}_k)$ spans the entire space. Thus $(\tilde{U}_k, \tilde{U}_k)(\tilde{U}_k, \tilde{U}_k)^T = I$. We say that the dominant structures of a 2D map dataset are well captured by the subspace $\tilde{U}_k \tilde{U}_k^T$ if

$$\sum_i A_i^T \tilde{U}_k \tilde{U}_k^T A_i \simeq \sum_i A_i^T (\tilde{U}_k \tilde{U}_k^T + \tilde{U}_k \tilde{U}_k^T) A_i = \sum_i A_i^T A_i.$$

which will happen if the largest k eigenvalues dominate the spectrum:

$$\sum_{j=1}^k \tilde{\lambda}_j / \sum_{j=1}^r \tilde{\lambda}_j \simeq 1, \quad \text{and} \quad \sum_{j=1}^s \tilde{\zeta}_j / \sum_{j=1}^r \tilde{\zeta}_j \simeq 1.$$

This is because the importance of these subspaces is approximately measured by their eigenvalues. This situation is similar to the standard SVD, where the first

k singular pairs provide a good approximation to the data when

$$\sum_{j=1}^k \sigma_j^2 / \sum_{j=1}^r \sigma_j^2 \simeq 1$$

This situation occurs when the eigenvalues λ_j approach zero rapidly with increasing j . The space is dominated by a few eigenstate.

In this case, the 2D maps can be well approximated by the 2dSVD, i.e., 2dSVD provides a near-optimal solution to $J_3(\cdot)$. In this case, the differences between $\hat{\lambda}_j, \tilde{\lambda}_j, \lambda_j$ tend to be small, and we set approximately

$$\sum_{j=k+1}^r \hat{\lambda}_j \simeq \sum_{j=k+1}^r \tilde{\lambda}_j \simeq \sum_{j=k+1}^r \lambda_j.$$

Similar results also hold for $\hat{\zeta}_j, \tilde{\zeta}_j, \zeta_j$. we obtain error estimation,

$$(7.47) \quad J_3^{\text{opt}}(k, s) \simeq \sum_{j=k+1}^r \lambda_j + \sum_{j=s+1}^c \zeta_j$$

$$(7.48) \quad \leq \sum_i \|A_i - U_k U_k^T A_i V_s V_s^T\|^2,$$

similar to the Eckart-Young Theorem. The two accumulative sums of eigenvalues correspond to the simultaneous left and right projections.

8 $A_i = LM_i L^T$ for symmetric A_i

Consider the case when A_i 's are symmetric: $A_i^T = A_i$, for all i . We seek the symmetric decomposition $A_i = LM_i L^T$ of J_4 in Eq.(2.12). Expand J_4 and take $\partial J_4 / \partial M_i = 0$, we obtain $M_i = L^T A_i L$, and $J_4 = \sum_{i=1}^n \|A_i\|^2 - \sum_{i=1}^n \|L^T A_i L\|^2$. Thus $\min J_4$ becomes

$$(8.49) \quad \max_L J_{4a}(L) = \sum_{i=1}^n \|L^T A_i L\|^2 = \text{Tr} L^T \left(\sum_{i=1}^n A_i L L^T A_i \right) L$$

Similar to the $A_i = LM_i R^T$ decomposition, 2dSVD gives a near-optimal solution

$$(8.50) \quad L = U_k, \quad M_i = U_k^T A_i U_k.$$

Starting with this, the exact optimal solution, can be computed according to the iterative update algorithm in §6. We write

$$(8.51) \quad \max_{L^{(t+1)}} J_{4a}(L^{(t+1)}) = \text{Tr} L^{(t+1)T} \left(\sum_{i=1}^n A_i L^{(t)} L^{(t)T} A_i \right) L^{(t+1)}.$$

From a current $L^{(t)}$, we form $\tilde{F} = \sum_i A_i L^{(t)} L^{(t)T} A_i$ and compute the first k -eigenvectors, which gives $L^{(t+1)}$.

From the same analysis of Propositions 6 and 7, we have

$$(8.52) \quad \begin{aligned} J_{4a}(L^{(0)}) &\leq J_{4a}(L^{(1)}) \leq J_{4a}(L^{(2)}) \leq \dots \\ &\leq \max_L \text{Tr} L^T \left(\sum_{i=1}^n A_i A_i \right) L = \sum_{i=1}^n \|U_k A_i\|^2. \end{aligned}$$

Thus the iterative algorithm converges to the optimal solution, $L^{(t)} \rightarrow \tilde{U} = (\tilde{\mathbf{u}}_1, \dots, \tilde{\mathbf{u}}_k)$, where

$$(8.53) \quad \tilde{F} \tilde{\mathbf{u}}_j = \tilde{\lambda}_j \tilde{\mathbf{u}}_j, \quad \tilde{F} = \sum_{i=1}^n A_i \tilde{U}_k \tilde{U}_k^T A_i.$$

The optimal objective value has the lower and upper bounds:

$$(8.54) \quad \sum_{j=k+1}^r (\lambda_j + \tilde{\lambda}_j) \leq J_4^{\text{opt}} \leq \sum_{j=k+1}^r (\lambda_j + \hat{\lambda}_j)$$

where $\hat{\lambda}_j$ are the eigenvalues of \hat{F} :

$$(8.55) \quad \hat{F} \hat{\mathbf{u}}_j = \hat{\lambda}_j \hat{\mathbf{u}}_j, \quad \hat{F} = \sum_{i=1}^n A_i U_k U_k^T A_i.$$

If eigenvalues $\tilde{\lambda}_j$ fall rapidly as j increases, the principal subspace \tilde{U}_k captures most of the structure, and 2dSVD provides a good approximation of the data. i.e., 2dSVD is the near-optimal solution in the sense of $J_4(\cdot)$. Thus we have

$$(8.56) \quad J_4^{\text{opt}} \simeq 2 \sum_{j=k+1}^r \lambda_j.$$

9 Application to images reconstruction and classification

Dataset A. ORL ⁴ is a well-known dataset for face recognition. It contains the face images of 40 persons, for a total of 400 images of sizes 92×112 . The major challenge on this dataset is the variation of the face pose.

Dataset B. AR ⁵ is a large face image dataset. The instance of each face may contain large areas of occlusion, due to sun glasses and scarves. The existence of occlusion dramatically increases the within-class variances of AR face image data. We use a subset of AR which contains 65 face images of 5 persons. The original image size is 768×576 . We crop face part of the image reducing size to 101×88 .

9.1 Image Reconstruction Figure 1 shows 8 reconstructed images from the ORL dataset, with a rather small $k = s = 5$. Images in the first row are reconstructed by the $A_i = LM_i$ decomposition using row-row

⁴<http://www.uk.research.att.com/facedatabase.html>

⁵http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html

Table 1: Test datasets and related storages for $k = s = 15$.

Dataset	n	Dimensions	# of classes	2dSVD Storage	SVD storage
ORL	400	92×112	40	93060	160560
AR	65	88×101	5	16920	143295



Figure 1: Reconstructed images by 2dLRi (first row), 2dLiR (second row), 2dSVD (third row), and LMR (fourth row) on ORL dataset at $k = s = 5$.

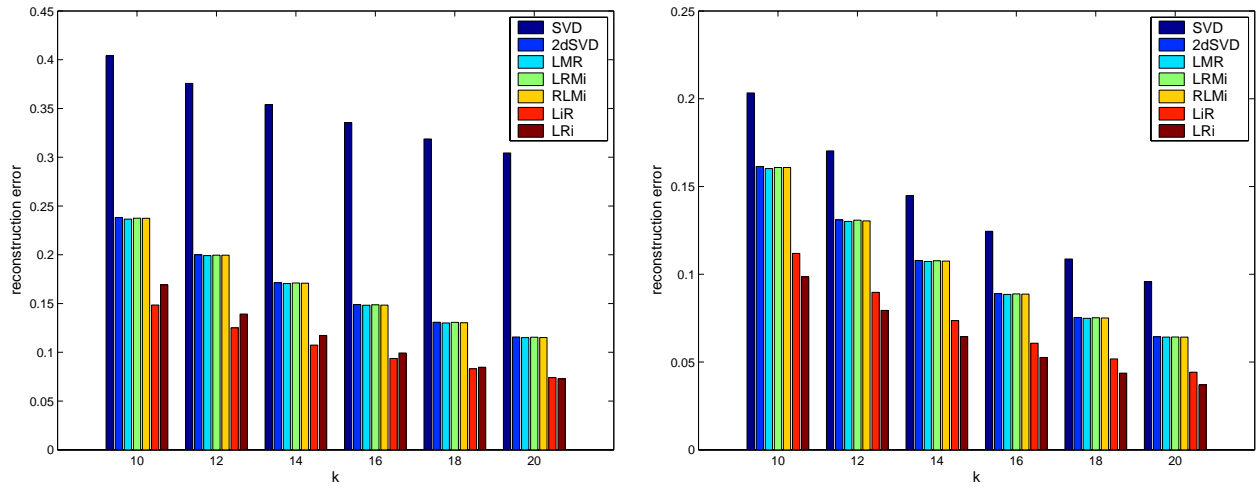


Figure 2: Reconstruction error for ORL dataset (left) and AR dataset (right). Compression methods, from left to right, are indicated in the insert panel.

Table 2: Convergence of LMR

t	2dSVD	Random	Rank-1	Orthogonal
0	0.15889029408304	0.58632462287908	0.99175445002735	0.96285808073065
1	0.15872269104567	0.15872372559621	0.15875108000211	0.15878339838255
2	0.15872268890982	0.15872268893458	0.15872268927498	0.15872268953111
3	0.15872268890976	0.15872268890977	0.15872268890981	0.15872268890981
4	0.15872268890976	0.15872268890976	0.15872268890976	0.15872268890976
angle	0	4.623e-10	3.644e-10	2.797e-10

correlation matrix F . We can see clearly the blurring along horizontal direction. Images in the second row are reconstructed by the $A_i = M_i R$ decomposition using column-column correlation matrix G . We can see clearly the blurring along vertical direction. Images in the 3rd row are reconstructed by the 2dSVD; Images in the 4th row are reconstructed by the $LM_i R^T$ decomposition; The symmetric decomposition of LMR and 2dSVD give better quality reconstruction. Figure 3 shows the same 8 reconstructed images from the ORL dataset, at $k = s = 15$ for 2dSVD and traditional SVD. One can see that 2dSVD gives better quality reconstruction.

Figure 2 shows the reconstruction errors for LMR of §6, 2dSVD, $M_i R^T$ decomposition of §3, LM_i^T decomposition of §4, LRMi of §6, and RLMi of §7. These experiments are done on AR and ORL datasets, with $k = s$ ranging between 10 and 20. We have the following observations: (a) LRi and LiR achieve the lowest residue errors; (b) LMR, 2dSVD, LRMi and RLMi lead to similar residue errors, with LMR the best; (c) SVD has the largest residue errors in all cases.

9.2 Convergence of $A_i = LM_i R^T$ decomposition

We examine the sensitivity of LMR on the initial choice. In Table 2, we show J_3 values for several initial choices of $L^{(0)}$ as explained in Discussion of Observation 8: 2dSVD, random matrices, Rank-1 start ($L^{(0)}$ is a matrix of zeros except $L_{1,1}^{(0)} = 1$), and orthogonal start ($L^{(0)}$ is orthogonal to the solution L^*).

We have the following observations. First, starting with all 4 initial $L^{(0)}$'s, the algorithm converges to the same final solution. In the last line, the angle between the different solutions and the one with 2dSVD start are given. They are all around 10^{-10} , practically zero within the accuracy of the computer precision. Considering the rank-1 start and the orthogonal start, this indicates the algorithm does not encounter other local minimums.

Second, 2dSVD is a good approximate solution. It achieves 3 effective decimal digit accuracy: $(J_3(2dSVD) - J_3^{\text{opt}})/J_3^{\text{opt}} = 0.1\%$. Starting from the 2dSVD, it converges to the final optimal solution in 3 iterations; it gets 6 digits accuracy in 1 iteration and gets 12 digit accuracy in 2 iterations.

Third, the convergence rate is quite good. In 1 iteration, the algorithm converges to 4 digits accuracy for all 4 initial starts. With 4 iterations, the algorithm converges to 14 digits, the computer precision with 64-bits, irrespective of any odd starting points.

To further understand the rapid convergence, we set $k = s = 1$ and run two experiments, one with $L^{(0)} = e_1$ and the other with $L^{(0)} = e_2$, where e_i is a vector of zeroes except that the i -th element is 1. The angle between the solutions at successive iterations, $L_1^{(t)}$ and $L_2^{(t)}$, are given in Table 3. One can see that even though the solution subspaces are orthogonal ($\pi/2$) at beginning, they run towards each other rapidly and become identical in 4 iterations. This indicates the solution subspace converges rapidly.

9.3 Bounds on J_3^{opt} In Figure 4, we show the bounds of J_3^{opt} provided by 2dSVD, Eq.(5.29) and Eq.(7.48). These values are trivially computed once 2dSVD are obtained. Also shown are the exact solutions at $k = s = 10, 15, 20$. We can see the 2dSVD provides a tight upper bound, because it provides a very close optimal solution. This bounds are useful in practice. Suppose one computes 2dSVD and wishes to decide the parameter k and s . Given a tolerance on reconstruction error, one can easily choose the parameters from these bound curves.

9.4 Classification One of the most commonly performed tasks in image processing is the image retrieval. Here we test the classification problem: given a query image, determine its class. We use the K-Nearest-Neighbors (KNN) method based on the Euclidean distance for classification [4, 6]. We have tested $k = 1, 2, 3$ in KNN. $k = 1$ always leads to the best classification results. Thus we fix $k = 1$. We use *10-fold cross-validation* for estimating the classification accuracy. In 10-fold cross-validation, the data are randomly divided into ten subsets of (approximately) equal size. We do the training and testing ten times, each time leaving out one of the subsets for training, and using only the omitted subset for testing. The classification accuracy reported is the average from the ten different random splits. The



Figure 3: Reconstructed images by 2dSVD (first row), and SVD (second row) on ORL dataset at $k = s = 15$.

Table 3: Convergence of LMR: $k = s = 1$ case

t	0	1	2	3	4
angle	$1.571=\pi/2$	1.486e-03	4.406e-05	1.325e-06	3.985e-08

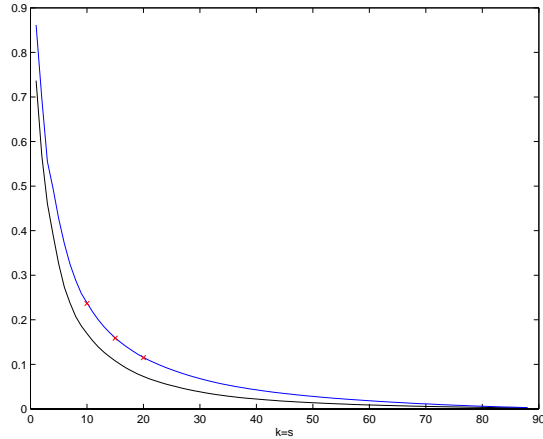


Figure 4: Lower and upper bounds of J_3^{opt} provided by 2dSVD. Also shown are the exact solutions at $k = s = 10, 15, 20$.

distance between two images $\|A_i - A_j\|$ are computed using the compressed data: $\|M_i - M_j\|$ for LMR, MR, and LM. For SVD, let $(a_i, \dots, a_n) = U\Sigma(\mathbf{v}_1, \dots, \mathbf{v}_n)$. The pairwise distance is $\|\Sigma(\mathbf{v}_i - \mathbf{v}_j)\|$. The results are shown in Fig.5. We see that LMR and 2dSVD consistently leads to small classification error rates, outperforming LiR, LRI and SVD, expect for AR dataset at large value of k (such as $k \geq 16$) where SVD is competitive.

9.5 Convergence for symmetric 2D dataset We tested the algorithm for the symmetric 2D dataset by generating the synthetic datasets $B_i = A_i^T A_i, i = 1, \dots, n$ for the ORL image dataset. Setting $k = 15$,

the reconstruction error J_4 is shown in Table 4. The iteration starts with 2dSVD solution, which is already accurate to 5 digits. After 1 iteration, the algorithm converges to the machine precision.

Table 4: Convergence for symmetric case

t	J_4
0	0.01245341543106
1	0.01245337811927
2	0.01245337811927

10 Surface temperature maps

The datasets are 12 maps, each of size 32 (latitude) x 64 (longitude). Each shows the distribution of average surface temperature of the month of January (100 years).

Table 5 shows the reconstruction of the temperature maps. One see that 2dSVD provides about the same or better reconstruction at much less storage. This shows 2dSVD provides a more effective function approximation of these 2D maps. The temperature maps are shown in Figure 6.

11 Summary

In this paper, we propose an extension of standard SVD for a set of vectors to 2dSVD for a set of 2D objects $\{A_i\}_{i=1}^n$. The resulting 2dSVD has a number of optimality properties which make it suitable for low-rank approximation. We systematically analyze the four decompositions, $A_i = M_i R^T$, $A_i = L M_i^T$, $A_i = L M_i R^T$, and $A_i = L M_i L^T$ for symmetric A_i . Their relationship with 2dSVD are shown. This provides a

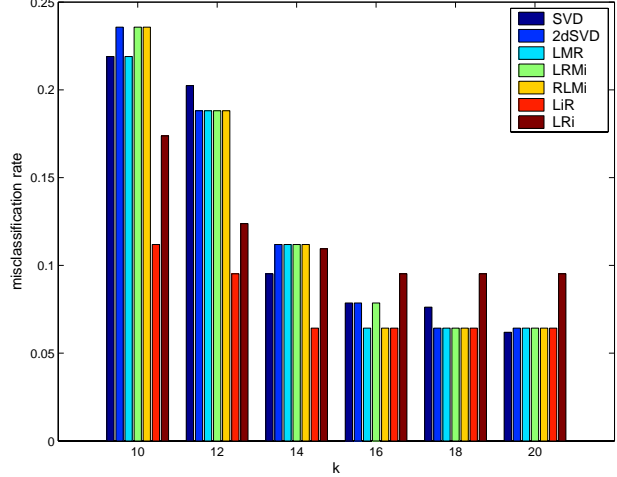
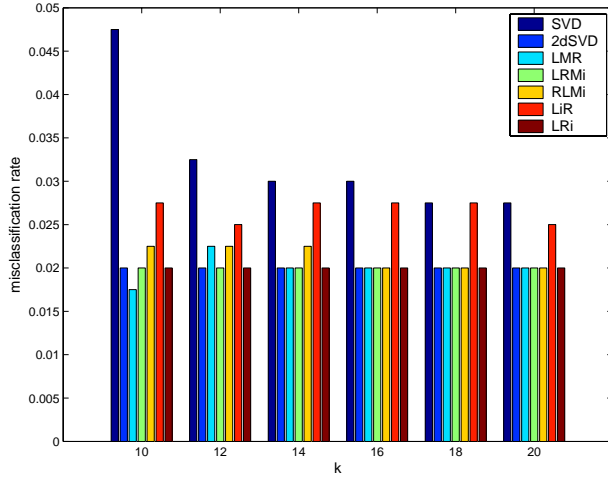


Figure 5: Classification (cross validation) error rate for ORL (left) and for AR (right)

Table 5: Reconstruction of the temperature maps

Method	k,s	storage	error
2dSVD	$k = 4, s = 8$	1024	0.0030
2dSVD	$k = 8, s = 16$	2816	0.0022
SVD	$k = 4$	8244	0.0040
SVD	$k = 8$	16488	0.0022

framework unifying two recent approaches by Yang *et al.*[13] and by Ye [14] for low-rank approximations which captures explicitly the 2D nature of the 2D objects, and further extend the analysis results. We carry out extensive experiment on 2 image datasets and compare to standard SVD. We also apply 2dSVD to weather maps. These experiments demonstrate the usefulness of 2dSVD.

Acknowledgements. We thank Dr. Haesun Park for discussions. This work is partially supported by Department of Energy under contract DE-AC03-76SF00098.

References

- [1] M.W. Berry, S.T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.
- [2] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Scienc*, 41, 391–407.
- [3] Dhillon, I., & Modha, D. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42, 143–175.
- [4] R.O. Duda, P.E. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

- [5] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:183–187, 1936.
- [6] K. Fukunaga. *Introduction to Statistical Pattern Classification*. Academic Press, San Diego, California, USA, 1990.
- [7] G. Golub and C. Van Loan. *Matrix Computations*, 3rd edition. Johns Hopkins, Baltimore, 1996.
- [8] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.
- [9] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. Pattern Analysis Machine Intelligence*, 12:103–108, 1990.
- [10] T.G. Kolda. Orthogonal tensor decompositions. *SIAM J. Matrix Analysis and App.*, 23:243–255, 2001.
- [11] R. W. Preisendorfer and C. D. Mobley. *Principal Component Analysis in Meteorology and Oceanography*. Elsevier Science Ltd, 1988.
- [12] N. Srebro & T. Jaakkola. Weighted low-rank approximations. *ICML Conference Proceedings* (pp. 720–727).
- [13] J. Yang, D. Zhang, A. Frangi, and J. Yang. Two-dimensional pca: A new approach to appearance-based face representation and recognition. *IEEE Trans. Pattern Analysis Machine Intelligence*, 26:131–137, 2004.
- [14] J. Ye. Generalized Low Rank Approximations of Matrices. *Proceedings of the Twenty-First International Conference on Machine Learning*. 887–894, 2004.
- [15] J. Ye, R. Janardan, and Q. Li. GPCA: An Efficient Dimension Reduction Scheme for Image Compression and Retrieval. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 354–363, 2004.
- [16] T. Zhang and G. H. Golub. Rank-one approximation to high order tensors. *SIAM Journal of Matrix Analysis and Applications*, 23:534–550, 2001.

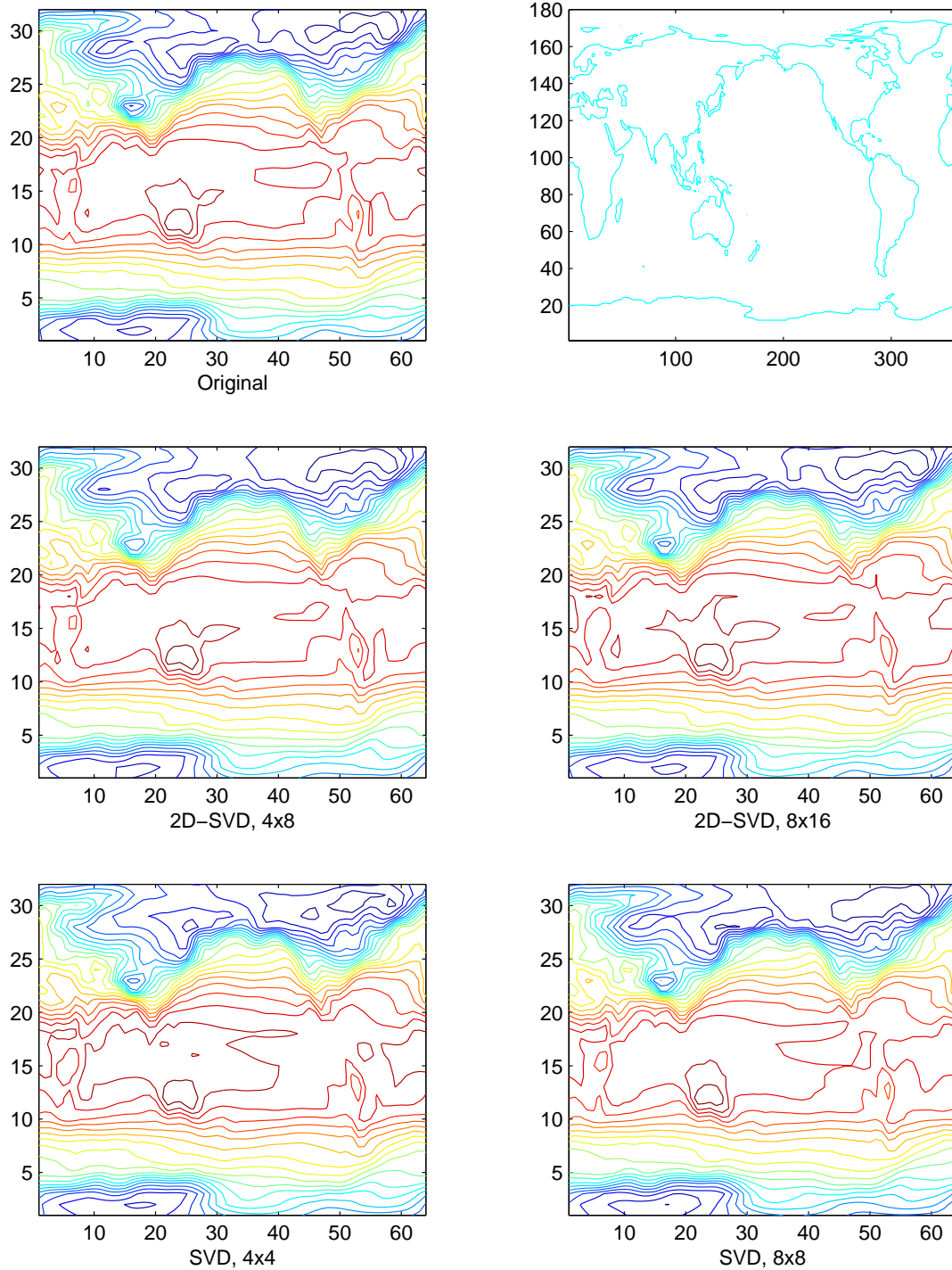


Figure 6: Global surface temperature. Top left: original data (January temperature for a randomly picked year. One can see that the central area of Australia is hottest spot on Earth). Top right: matching continental topography for location specification. Middle left: 2dSVD with $k = 4, s = 8$. The reduction ratio are kept same for both columns and rows: $8=32/4=64/8$. Middle right: 2D-SVD with $k = 8, s = 16$. Bottom left: conventional SVD with $k = 4$. Bottom right: conventional SVD with $k = 8$.

Summarizing and Mining Skewed Data Streams

Graham Cormode*

S. Muthukrishnan†

Abstract

Many applications generate massive data streams. Summarizing such massive data requires fast, small space algorithms to support post-hoc queries and mining. An important observation is that such streams are rarely uniform, and real data sources typically exhibit significant skewness. These are well modeled by Zipf distributions, which are characterized by a parameter, z , that captures the amount of skew.

We present a data stream summary that can answer point queries with ε accuracy and show that the space needed is only $O(\varepsilon^{-\min\{1, 1/z\}})$. This is the first $o(1/\varepsilon)$ space algorithm for this problem, and we show it is essentially tight for skewed distributions. We show that the same data structure can also estimate the L_2 norm of the stream in $o(1/\varepsilon^2)$ space for $z > \frac{1}{2}$, another improvement over the existing $\Omega(1/\varepsilon^2)$ methods.

We support our theoretical results with an experimental study over a large variety of real and synthetic data. We show that significant skew is present in both textual and telecommunication data. Our methods give strong accuracy, significantly better than other methods, and behave exactly in line with their analytic bounds.

Keywords: data stream analysis, data mining, Zipf distribution, power laws, heavy hitters, massive data.

1 Introduction

A number of applications—real-time IP traffic analysis, managing web clicks and crawls, sensor readings, email/SMS/blog and other text sources—are instances of massive data *streams*. Here new data arrives very rapidly and often we do not have the space to store all the data. Hence, managing such streams needs algorithmic methods that support *fast updates* and have a *small footprint* of space. See [17] for a detailed motivation for these constraints in the context of IP traffic analysis. Similar motivation can be found in high performance web data analysis [25], mining email streams [40], aggregating sensor data [39], analyzing

financial time series [52], processing high-volume scientific measurements [30], detecting communities in communication graphs [7] and others.

There are typically two aspects to analyzing data streams. The first is *summarizing* data streams for post-hoc queries. Data stream methods use a *synopsis* to summarize the data stream on the fly. “Synopsis” means a small space representation of the data stream that can be rapidly updated as the data stream unravels; typical synopses are samples and sketches. The second aspect of analyzing data streams is the type of queries that are supported, whether using synopses or without. Since many of these data stream applications tend to be about monitoring data sources say for adverse events such as intrusion detection, fraudulent communities, etc., an essential set of problems are of the data *mining* genre. For example, how to find heavy hitters or frequent items, large changes and evolving trends, or perform clustering and similarity searching, decision tree classification, regression and other statistical analyses. See the ensemble of websites for project descriptions [41, 26, 48, 1, 15, 28, 45], bibliographies [50], tutorials [20, 34], surveys [5, 24, 44] in addition to a number of special issues in journals, workshops, etc.

We study both summarization and mining problems with data streams. Our work here was initiated by our experience with IP traffic analysis using Gigascope [15], AT&T’s IP traffic analysis system, as well as our work on mining text streams [40]. Our observation was that data streams involved distributions that were not arbitrary, but rather typically quite *skewed*. For example, if one studied the distribution of the IP addresses that used a link of the backbone network or the distribution of flows or bytes sent by each IP address, the distribution was zipfian, fractal or multi-fractal. This has been studied in detail in [33]. Similarly, the word frequencies in natural text is well known to be zipfian; word frequencies in text streams such as email or blogs tend to be heavy-tailed [36]. In nature, zipfian distributions (also known as power laws) abound in citation distributions to web accesses for different sites, file sizes transferred over the Internet, etc [8].

Motivated by our experience and the widely-prevalent evidence of skew in data streams, we study summarization and mining problems with skewed data streams. We take a formal approach, focusing on zipfian (more generally, *Zipf-like* which we define later and which is quite general) skew on the domain $1 \dots U$. The Zipf distribution is also known

*cormode@bell-labs.com Work completed while author was at Center for Discrete Mathematics and Computer Science (DIMACS) Rutgers University, Piscataway NJ. Supported by NSF ITR 0220280 and NSF EIA 02-05116.

†muthu@cs.rutgers.edu Division of Computer and Information Systems, Rutgers University. Supported by NSF CCR 0087022, NSF ITR 0220280, NSF 0430852 and NSF EIA 02-05116.

as the Pareto distribution, and is essentially identical to so-called “power-laws” [2] with transformation of parameters. We work in well-established data stream algorithm model and study *probabilistic, approximate* algorithms, that is algorithms that provide ε approximation with probability of success at least $1 - \delta$. As is usual, we study the trade-off between space used by the synopsis and the time per new data update versus the quality of estimations given by ε and δ . Our *theoretical contributions* are as follows.

1. We present a synopsis that uses space $O(\varepsilon^{-\min\{1, 1/z\}} \ln 1/\delta)$ for (ε, δ) point queries on z -skewed data streams.

For $z > 1$, the space used is $o(1/\varepsilon)$; this is the first known $o(1/\varepsilon)$ space algorithm known for *any* synopsis—sample or sketch based—known for such problems. This is of interest since the $O(1/\varepsilon)$ space bound has long been taken as the gold-standard target for data stream algorithm design.

We can use this synopsis for a variety of mining tasks such as finding heavy-hitters, frequent items for association rule mining, finding significant changes from one time to another, significant differences between different streams, estimating wavelet decomposition of data stream, and so on. In all these cases, our methods improve the $O(1/\varepsilon)$ factor in previously known algorithms to $O(\varepsilon^{-\min\{1, 1/z\}})$ for space usage.

2. We prove a matching lower bound, that is, we show that any $(\varepsilon, O(1))$ algorithm for point queries needs at least $\Omega(\varepsilon^{-\min\{1, 1/z\}})$ space.
3. We extend our synopsis to estimate the second frequency moment, i.e., the sum of squares of the frequencies of items in the data stream. It is equivalently (the square of) the L_2 norm of the vector of frequencies of items in the data stream. The space used is $O(\varepsilon^{\frac{4}{1+2z}})$ for $1/2 < \varepsilon \leq 1$ and $O(\varepsilon^{\frac{-2}{1+z}})$ for $z > 1$.

For $z \geq 1/2$, the space used is $o(1/\varepsilon^2)$; all previously known algorithms in contrast use at least $\Omega(1/\varepsilon^2)$ space. Our bound above is additionally interesting because the synopsis methods for L_2 norm are implementations of the Johnson-Lindenstrauss lemma [29]. The lemma states that a set of vectors in Euclidean space can each be replaced by a vector in $O(\frac{1}{\varepsilon^2})$ -dimensional space, and inter-vector distances are preserved up to a $(1 \pm \varepsilon)$ factor with high probability. This dependency on ε is tight since a lower bound of $\Omega(\frac{1}{\varepsilon^2})$ has recently been shown [51] for general distributions. Our results show that for skewed data, this lower bound can be avoided.

We can use this result for a variety of mining tasks. For example, anomaly detection methods in IP traffic analysis use the second frequency moment [35]. Also, it

is used as a subroutine in counting subgraphs in massive web graphs in [6], and for quantiles, histograms and other statistical descriptors of the data stream [21].

Our synopsis above is a sketch, i.e., inner products of the input distribution with certain random vectors. There are many known sketches [4, 10, 23, 49, 13]. Here, we adopt the Count-Min sketch [13] which dominates all other sketches in terms of space and update time needed to guarantee $\varepsilon \|a\|$ accuracy. Our main theoretical contribution here is to *analyze* estimation methods based on this sketch and prove improved bounds on space usage without compromising any of the other parameters, i.e., update time and accuracy. Our algorithms are essentially *oblivious* of the skew value, z but our analysis is skew-aware. We can approach this in two ways: given a desired error bound ε and bound on the skew z , we can allocate space to the sketch as a function of these parameters; alternatively, we can allocate a fixed amount of space for the sketch, and based on the observed skew z , give tight bounds on the worst case error as a function of z . In the latter case irrespective of the data distribution, a simple analysis gives a universal bounds on the error, and using the skew of the distribution we can give tighter bounds.

Our results give additional evidence that CM sketch is versatile and suited for a variety of problems under a range of data distributions. This, coupled with their proven performance within the operational AT&T’s IP traffic analysis tool Gigascope [11] at the rate of OC48 links, makes our methods for skewed data stream summarization and mining suitably efficient for real-life data stream management systems in practice.

Our *experimental contributions* are as follows. We consider large streams of both real and synthetic data. We observe that all the real data we consider, from IP network and phone call data to “blogs” and Shakespeare’s plays, exhibit significant skew to varying degrees, and our methods capitalize on this. Not only do they outperform other methods, but they behave closely in accordance with our stated bounds. The correlation is sufficiently good that not only can we compare our method to that predicted by our theory, but also we can use our results to compute the skewness of the data with high accuracy. Our conclusion is that by understanding and building skew into our model of data streams, we can realize much stronger results, both in terms of theoretical analysis and practical performance.

1.1 Map.

We give preliminaries in Section 2, then define the CM Sketch in Section 3. We discuss skewed distributions and related work in Section 4. Section 5 gives our results for Point Queries, and Section 6 those for L_2 norm estimation. Our experimental study on a mixture of real and synthetic data is reported in Section 7.

2 Model and Queries

We consider a vector \mathbf{a} , which is presented in an implicit, incremental fashion. This vector has dimension n , and its current state at time t is $\mathbf{a}(t) = [a_1(t), \dots, a_i(t), \dots, a_n(t)]$. For convenience, we shall usually drop t and refer only to the current state of the vector. Initially, \mathbf{a} is the zero vector, $\mathbf{0}$, so $a_i(0)$ is 0 for all i . Updates to individual entries of the vector are presented as a stream of pairs. The t th update is (i_t, c_t) , meaning that

$$\begin{aligned} a_{i_t}(t) &= a_{i_t}(t-1) + c_t \\ a_{i'}(t) &= a_{i'}(t-1) \quad i' \neq i_t \end{aligned}$$

We assume throughout that although values of a_i increase and decrease with updates, each $a_i \geq 0$. Our results all generalize to the case where a_i s can be less than 0, with small factor increases in space, but we omit details of these extensions for simplicity of exposition.

It is easy to see how this model maps to the motivating data stream applications. For example, for the IP traffic case, each new IP packet with source IP address s and size of the packet p may be seen as updating $a[s] \leftarrow a[s] + p$ to count the total size of flows from source IP address s . Similarly, in the text streaming application, when new text input such as email arrives, we can parse it into words and track word usage frequency in order to track frequent and recently popular (“bursty”) words, to attribute authorship based on usage patterns, etc. Here, each new text input updates many new $a[w]$ ’s for different words or phrases w in the input. See [44] for more examples. We consider two particular types of queries for summarization and mining.

- **Point query.** A point query is, given i , to return an estimate of a_i . Our goal is to give (ε, δ) approximations: the answer should be correct to within additive error of $\varepsilon \|\mathbf{a}\|_1$ with probability at least $1 - \delta$. We will analyze the space required as a function of ε and δ required to achieve this.
- **Second Frequency Moment and L_2 Norm** The L_2 norm of a vector, $\|\mathbf{a}\|_2$, is defined as $(\sum_i a_i^2)^{\frac{1}{2}}$. The goal is to estimate this within additive error of $\varepsilon \|\mathbf{a}\|_2$ (equivalently, with relative error $1 + \varepsilon$) with probability at least $1 - \delta$. The second frequency moment in our model is the square of the L_2 norm, $\|\mathbf{a}\|_2^2$.

These two queries appear to be abstract, but they have many concrete applications in a number of mining problems on data streams. Point queries can be used for estimating frequent items for association rule mining [42], heavy hitters¹ [12], significant differences [10] and significant relative changes [14], etc. The L_2 norm estimation is use-

ful in anomaly detection [35], counting triangles in massive web graphs in [6], and for quantiles [23], wavelets [22], histograms [21] and other statistical descriptors of the data stream. They are also useful for partitioning data stream into multiple zones of interest [16]. There is a maturing theory of data stream algorithms with many such applications. Rather than list these applications and show the improvements obtained by using our methods we focus on these primary queries and demonstrate the nature of our improvements in depth.

3 The CM Sketch

Many sketches are known [4, 10, 49, 13]. Here, we briefly recap the data structure that is used throughout. The important property is that, given the parameters of the sketch structure, the update procedure is the same no matter what the ultimate query operations are.

The CM sketch is simply an array of counters of width w and depth d , $\text{count}[1, 1] \dots \text{count}[d, w]$. Each entry of the array is initially zero. Additionally, d hash functions

$$h_1 \dots h_d : \{1 \dots n\} \rightarrow \{1 \dots w\}$$

are chosen uniformly at random from a pairwise-independent family. Once w and d are chosen, the space required is fixed as the wd counters and the d hash functions (which can each be represented in $O(1)$ machine words [43]).

Update and Query Procedure. When an update (i_t, c_t) arrives, meaning that item a_{i_t} is updated by a quantity of c_t , then c_t is added to one count in each row; the counter is determined by h_j . Formally, we set

$$\forall 1 \leq j \leq d : \text{count}[j, h_j(i_t)] \leftarrow \text{count}[j, h_j(i_t)] + c_t$$

The query procedure is similar: given a query point i , return $\min_{1 \leq j \leq d} \text{count}[j, h_j(i)]$ as the estimate. In [13], it was shown that the error for point queries, irrespective of the distribution, is $\varepsilon \|\mathbf{a}\|_1 = e/w \|\mathbf{a}\|_1$ with probability $1 - \delta = 1 - e^{-d}$. Hence, in order to get ε approximation with probability $1 - \delta$ for point queries, we need $w = e/\varepsilon$ and $d = \log(1/\delta)$.

4 Skew in Data Stream Distributions

In almost every practical setting, the distribution of frequencies of different items displays some amount of skew. Throughout, we will use the popular Zipf distribution to model skewed distributions. The Zipf distribution accurately captures a large number of natural distributions. It was introduced in the context of linguistics, where it was observed that the frequency of the i th most commonly used word in a language was approximately proportional to $1/i$ [53]. Zipf distributions are equivalent to Pareto distributions and power-laws [2].

¹The heavy hitters problem is to find all i such that $a_i \geq \|\mathbf{a}\|_1/k$ for some constant k .

Formally, a Zipf distribution with parameter z has the property that f_i , the (relative) frequency of the i th most frequent item is given by $f_i = \frac{c_z}{i^z}$, where c_z is an appropriate scaling constant. We will consider distributions over the range $[1 \dots U]$, where U is the range, or universe size. For the skewed distributions we consider, we can often allow U to be ∞ . c_z is determined by z (and U) since for a probability distribution we must have $\sum_{i=1}^U f_i = 1$. Given a vector \mathbf{a} whose entries are distributed according to a Zipf distribution, the count of the i th most frequent item is simply $\|\mathbf{a}\|_1 f_i$.

Many skewed distributions are well captured by Zipf distributions with appropriate parameters. Natural phenomena, such as sizes of cities, distribution of income, social networks and word frequency can all be modeled with Zipf distributions. Even the number of citations of papers demonstrates a highly skewed Zipf distribution [47]. More relevant to our study of large data streams, web page accesses for different sites have been observed to obey a skewed Zipf distribution with parameter between 0.65 and 0.8. [9]. The “depth” to which surfers investigate websites is also captured by a Zipf distribution, with parameter 1.5 [27]. Files communicated over the Internet display Zipf distribution in a variety of ways: transmission times are Zipf with parameter approximately 1.2; the size of files requested, transmitted, and available for download are all Zipf with parameters respectively measured as 1.16, 1.06 and 1.06 [8]. FTP traffic sizes was estimated to have z in the range 0.9 to 1.1. More strongly, such skewed behavior of requests appears not only over individual addresses but also when grouped into subnets or larger networks [33], meaning that the skewed distribution is self-similar (multi-fractal).

Related work on Mining Skewed Streams. A distinguishing element of our work is to bring the skew of the data into the analysis of summarizing and mining data whereas much of the extant work deals with arbitrary distributions (with some exceptions). For the heavy hitters problem Manku and Motwani [42] presented the “lossy counting” algorithm that requires space $O(\frac{1}{\epsilon} \log \epsilon \|\mathbf{a}\|_1)$ to give the same accuracy bounds as our results in general; but under the assumption that each new item is drawn from a fixed probability distribution, then the space is (expected) $O(\frac{1}{\epsilon})$ and the error guaranteed. Our results are dual to this, given guaranteed space bounds and expected error bounds; however, with more information about the distribution, our bounds are dependent on skew z , being much better for moderate to large skew, but never worse. For the top k problem, [10] specifically studied Zipfian data and showed that for $z > \frac{1}{2}$, $O(\frac{k}{\epsilon^2})$ space suffices. For large skew, our methods improve this bound to $O(\frac{k}{\epsilon})$. Using data skew is not uncommon in database research, but only recently there are examples of data mining in presence of skew in massive data such as [18] of analyzing trading anomalies. Our work differs from previous works by being a systematic algorithmic study of summarization and

mining problems in data streams with skew to give much improved bounds and performance.

Zipf tail bounds. For our analysis, we will divide up the range of the parameter z into three regions. We refer to $\frac{1}{2} < z \leq 1$ as *moderate skew*, and $1 < z$ as *skewed*. Otherwise, when $z \leq \frac{1}{2}$, we will say that the distribution has *light skew*.

The following facts result from bounding the discrete distribution by its continuous counterpart.

FACT 4.1. For $z > 1$, $1 - \frac{1}{z} \leq c_z \leq z - 1$.

FACT 4.2. For $z > 1$,

$$\frac{c_z k^{1-z}}{z-1} \leq \sum_{i=k}^U f_i \leq \frac{c_z (k-1)^{1-z}}{z-1}$$

FACT 4.3. For $z > \frac{1}{2}$,

$$\frac{c_z^2 k^{1-2z}}{2z-1} \leq \sum_{i=k}^U f_i^2 \leq \frac{c_z^2 (k-1)^{1-2z}}{2z-1}$$

Our analyses generalize to when the data distribution is dominated by zipfian or more generally, what we call *Zipf-like* distributions: a distribution is *Zipf-like* with parameter $z > 1$ if the tail after k largest items has weight at most k^{1-z} of the total weight (one could also allow scaling by a constant, eg, the tail has weight at most ck^{1-z} ; such extensions follow easily). Although we state results in this paper for zipfian data, with a few more technical details, the results hold for Zipf-like distributions as well.

5 Point Queries

5.1 Upper Bounds

The crucial insight for giving better bounds for the Count-Min sketch in the presence of skewed distributions is the observation that items with large counts can cause our estimates to be poor if they collide with other items in the array of counters. In a skewed distribution a few items consume a large fraction of the total count. When estimating the count of an item, if none of these large items collide with it under the hash functions, then we can argue that the estimates will be better than the $w = 1/\epsilon$ bound given by the generic argument in [13].

THEOREM 5.1. For a Zipf distribution with parameter z , the space required to answer point queries with error $\epsilon \|\mathbf{a}\|_1$ with probability at least $1 - \delta$ is given by $O(\epsilon^{-\min\{1, 1/z\}} \ln 1/\delta)$.

Proof. For $z \leq 1$, the best results follow from analysis in [13]. For $z > 1$, we use the same estimation technique to return an estimate for a_i as $\hat{a}_i = \min_j \text{count}[h_j(i)]$, but give

a new analysis. The estimate returns a_i plus some additional “error” coming from the counts of items that collide with i under the hash functions. We split the error in our estimate into two parts: (i) collisions with some of the largest items and (ii) noise from the non-heavy items. If the sketch has width w , then let $k = w/3$. With constant probability ($\frac{2}{3}$) over the choice of hash functions, none of the k heaviest items collide with the point we are testing in any given row.

The expectation of the estimate for i is

$$a_i + \frac{1}{w} \sum_{x=k+1, x \neq i}^U a_x \leq a_i + \|\mathbf{a}\|_1 k^{1-z}/w. \quad (5.1)$$

This uses Fact 4.2 from Section 4 to bound the weight of the tail of the Zipf distribution after the k largest items are removed. Setting $k^{1-z}/w = \varepsilon/3$ and recalling that $w = 3k$ leads us to choose $w = 3k = 3(\frac{1}{\varepsilon})^{1/z}$. We can now apply the Markov inequality to show that the error is bounded by $\varepsilon \|\mathbf{a}\|_1$ with probability at least $1 - \frac{1}{3} - \frac{1}{3} = \frac{1}{3}$. This applies to each estimate; since we take the minimum of all the estimates, then this probability is amplified to $1 - \frac{2}{3}^d$ over the d separate estimations. \square

5.2 Lower Bounds

We now present lower bounds for the space required to answer point queries, which shows that our analysis above is asymptotically tight (since [13] shows the CM sketch data structure gives ε error over general distributions with $O(\frac{1}{\varepsilon})$ space).

THEOREM 5.2. *The space required to answer point queries correctly with any constant probability and error at most $\varepsilon \|\mathbf{a}\|_1$ is $\Omega(\varepsilon^{-1})$ over general distributions, and $\Omega(\varepsilon^{-1/z})$ for Zipf distributions with parameter z , assuming $n = \Omega(\varepsilon^{-\min\{1, 1/z\}})$.*

Proof. Our proof relies on a reduction to the Index problem in communication complexity. There are two players, A and B . A holds a bitstring of length n , and is allowed to send a message to B who wishes to compute the i th bit of the bitstring. Since B is not allowed to communicate with A , then any protocol to solve this problem, even probabilistically, requires $\Omega(n)$ bits of communication [37]. We will reduce to this problem by encoding a bitstring in such a way that if we could answer point queries with sufficient accuracy, we could recover bits from the bitstring. This is sufficient to show a lower bound on the size of the data structure required to answer such queries.

For general distributions, we take a bitstring $B[1 \dots \frac{1}{2\varepsilon}]$ of length $n = \frac{1}{2\varepsilon}$ bits, and create a set of counts. We set $a_i = 2$ if $B[i] = 1$. Otherwise, we set $a_i = 0$ otherwise, and add 2 to a_0 . Now, observe that whatever the value of B , $\|\mathbf{a}\|_1 = 1/\varepsilon$. If we can answer point queries with accuracy $\varepsilon \|\mathbf{a}\|_1 = 1$, then we can test any a_i and determine the value

of $B[i]$ by reporting 1 if the estimated value of a_i is above εN , and 0 otherwise. Therefore, the space used must be at least $\Omega(\frac{1}{\varepsilon})$ bits.

The same idea applies when we restrict ourselves to Zipf distributions. However, the counts must follow the Zipf pattern. We again encode a bitstring B , this time using the k largest counts from the Zipf distribution. Now we set $a_{2i} = f_i N$ (for some suitably large value of N) if $B[i] = 1$, else we set $a_{2i+1} = f_i N$ if $B[i] = 0$. This time, we can recover the first k bits of B provided that $f_k \geq 2\varepsilon$: if f_k is less than this, then the error in approximation does not allow us to distinguish this value from zero. Using the bounds on f_i for skewed Zipf distributions, we have $f_k = \frac{c_z}{k^z} \geq 2\varepsilon$. To get the best lower bound, we choose k as large as possible subject to these constraints. Solving for k , we find $k = \frac{c_z^{1/z}}{2\varepsilon^{1/z}}$. The term $\frac{c_z^{1/z}}{2\varepsilon^{1/z}}$ is bounded below by $(z-1)/2$ for $1 < z \leq 2$, and may be treated as a constant. Thus, k is fixed as $c \frac{1}{\varepsilon^{1/z}}$ bits of B for some constant c . This results in the stated space bounds by again appealing to the Index problem. \square

5.3 An example application: Top-k items

As mentioned earlier, supporting point queries post-hoc on data stream synopsis has many applications. Here, we focus on describing one of them.

A common query for a variety of management and analysis settings is to find the top- k : for example, find the top 100 users of bandwidth on a network, or find the top 10 new terms in a message stream. Such queries can be answered by point queries, by tracking the most frequent items that are seen as the stream unravels. We need to choose the parameter ε appropriately: too large, and we will not be able to answer the query with sufficient accuracy, and the results may be unreliable. When the distribution is skewed, we can apply our above results and give very tight bounds.

To give the correct answer, we need to bound the error by εa_k (where, here, we use a_k to denote the frequency of the k th most frequent item in \mathbf{a}) instead of $\varepsilon \|\mathbf{a}\|_1$. Using the above analysis for the expectation of the error in the estimation of any frequency from equation (5.1), we set the expected error equal to εa_k :

$$\frac{\|\mathbf{a}\|_1 k^{1-z}}{w} = \frac{\varepsilon \|\mathbf{a}\|_1 k^{-z}}{2}$$

and so $w = O(\frac{k}{\varepsilon})$ for $z > 1$. This improves the results in [10], which showed that for $z > \frac{1}{2}$, $O(\frac{k}{\varepsilon^2})$ space suffices with a Count sketch. In both cases, occurrences of z cancel, so there is no dependency on z provided the distribution is skewed with $z > 1$. We can set the space based on k and ε without needing to know z exactly. Further, using a CM Sketch, one can simulate the sketch of [10] by computing $(\text{count}[j, 2i] - \text{count}[j, 2i-1])$ for all $1 \leq j \leq d, 1 \leq i \leq w/2$. The converse is not possible.

6 Second Frequency Moment Estimation

The second frequency moment, and the closely related L_2 norm, have been the focus of much study in the data stream community. The work of Alon, Matias and Szegedy [4] spurred interest in the data stream model of computation. One of their results was an efficient algorithm to compute the second frequency moment of a stream of values in space $O(\frac{1}{\varepsilon^2})$. As was observed by the authors of [19], the same algorithm also allowed the L_2 difference of two streams to be computed in a very general model. The algorithm can also be viewed as a streaming implementation of the Johnson-Lindenstrauss lemma [29] with limited randomness and bounded space. The lemma states that a set of vectors in Euclidean space can each be replaced by a vector in $O(\frac{1}{\varepsilon^2})$ -dimensional space, and inter-vector distances are preserved up to a $(1 \pm \varepsilon)$ factor with high probability. This dependency on ε is essentially tight in terms of the dependency on ε for general distributions: a lower bound of $\Omega(\frac{1}{\varepsilon^2})$ has recently been shown [51]. This is problematic for applications that require a very fine quality approximation, say $\varepsilon = 1\%$ or 0.1% error, since the dependency on ε^{-2} means a high space cost. Here, we show how the CM sketch can be used to approximate this heavily studied quantity with strong guarantees of accuracy, and how, for skewed distributions, the $\Omega(\varepsilon^{-2})$ space bounds can be beaten for the first time.

6.1 Skewed Data

We describe the estimation procedure for the L_2 norm; to estimate the second frequency moment, we return the square of this value. When the distribution is skewed ($z > 1$), there are a few items with high frequency, and so a simple method to approximate the norm suffices. That is, we simply compute our estimate of the L_2 norm as

$$\min_j \left(\sum_k \text{count}[j, k]^2 \right)^{1/2}$$

which is minimum of the L_2 norm of the rows of the sketch. We refer to this method as CM^+ .

THEOREM 6.1. *This procedure estimates the L_2 norm of streams with Zipf skewness parameter $> \frac{1}{2}$, with error bounded by $\varepsilon \|a\|_2$ where $\varepsilon = O(w^{\frac{-(1+z)}{2}})$, with probability at least $1 - \delta = 1 - \frac{3}{4}^{-d}$.*

Proof. Let $m = w^{1/2}$. Then, with constant probability, in any row the largest m items fall in different buckets within the CM sketch. This follows from the pairwise independence of the hash functions used.

We compute the (squared) error in the j th estimator as $X_j = \sum_i \text{count}[j, i]^2 - \|a\|_2^2$. Consider the expectation of this quantity when the above condition holds, that is, when the m largest counts are in m distinct buckets:

$$\begin{aligned} E(X_j) &= \sum_{i=1}^U a_i^2 + \sum_{i=1}^U \sum_{j=1, j \neq i}^U a_i a_j \Pr[h(i) = h(j)] - \|a\|_2^2 \\ &\leq \|a\|_2^2 + \frac{1}{w} \left(\sum_{i=1}^U \sum_{j=1, j \neq i}^U a_i a_j - \sum_{i=1}^m \sum_{j=1, j \neq i}^m a_i a_j \right) - \|a\|_2^2 \\ &\leq \frac{\|a\|_1^2}{w} \left(2 \sum_{i=1}^m f_i \sum_{j=m+1}^U f_j + \left(\sum_{i=m+1}^U f_i \right)^2 \right) \\ &\leq 2 \frac{\|a\|_1^2}{w} \left(\sum_{i=1}^U f_i \sum_{j=m+1}^U f_j \right) \\ &\leq \frac{2 \|a\|_1^2 c_z m^{1-z}}{w(z-1)} \leq \frac{2 \|a\|_2^2 c_z (2z-1)}{c_z^2 (z-1)} w^{\frac{-(1+z)}{2}} \end{aligned}$$

This makes use of the Facts 4.2 and 4.3 to bound the sum of the tail of the distribution and to relate the L_1 norm to the L_2 norm. Note that, since $a_i = \|a\|_1 f_i$ and $\|a\|_2^2 = \sum_i a_i^2$, we can write $\|a\|_2^2 = \|a\|_1^2 \sum_i f_i^2$. We can substitute this inequality, and then use the lower bound of Fact 4.3 to rewrite $\sum_i f_i^2$ in terms of z and c_z . We set the expected squared error equal to $\varepsilon \|a\|_2^2 / 2$, which gives $w = O(\varepsilon^{\frac{2}{1+z}})$. We treat the terms polynomial in z as effectively constant.

We then apply the Markov inequality, so with probability $\frac{3}{4}$, $X_j < 2\varepsilon \|a\|_2^2$. This implies that $\|a\|_2^2 \leq X_j \leq (1 + \varepsilon)^2 \|a\|_2^2$. Taking the square root of all terms in this inequality bounds the L_2 norm of a . For each row the probability of this failing to hold is no more than $\frac{3}{4}$: $\frac{1}{2}$ for the m items not falling in different counters, $\frac{1}{4}$ from the Markov inequality. Taking the minimum of these estimates amplifies the probability of success to $1 - \frac{3}{4}^d$. \square

6.2 Moderate Skew

For the moderate skew ($z < 1$), and unskewed cases, we use the CM Sketch data structure to effectively simulate the sketch proposed by Alon Matias and Szegedy [4]. This shows the flexibility of the CM Sketch. In order for the results to be provable, we need to strengthen the hash functions used, from pairwise independent to 4-wise independent². Apart from this change, the data structure is constructed and maintained in the same way as before.

Again, let $m = w^{1/2}$; it remains the case that the m largest items will not collide, although these contribute a smaller amount to the L_2 norm. Now, compute the estimate (denoted CM^-) of the L_2 norm for each row by taking the square root of

$$Y_j = \sum_{k=1}^{w/2} (\text{count}[j, 2k] - \text{count}[j, 2k-1])^2.$$

²In [4], the authors argue that in practice, pairwise or other hash functions will often suffice.

THEOREM 6.2. *With constant probability,*

$$(1 - \varepsilon) \|\mathbf{a}\|_2^2 \leq Y_j \leq (1 + \varepsilon) \|\mathbf{a}\|_2^2$$

for $\varepsilon = w^{-\frac{(1+2z)}{4}}$.

Proof. We will define some functions derived from the hash functions, h , in order to simplify the notation and clarify the analysis. We define $g_j(x) = +1$ if $h_j(x) \equiv 0 \pmod{2}$, and -1 otherwise. We also define $h'_j(x) = \lceil h_j(x)/2 \rceil$.

First, we will show that in expectation, $E(Y_j) = \|\mathbf{a}\|_2^2$. Observe that

$$\begin{aligned} E(Y_j) &= \sum_{x,y} a_x a_y g_j(x) g_j(y) \Pr[h'_j(x) = h'_j(y)] \\ &= \sum_x a_x^2 = \|\mathbf{a}\|_2^2 \end{aligned}$$

using the pairwise independence of the function g (which follows from the pairwise independence of h). Secondly, we compute the variance of Y_j as

$$\begin{aligned} \text{Var}(Y_j) &= E(Y_j^2) - E(Y_j)^2 \\ &= \left(\sum_{i=1}^{w/2} \left(\sum_{x, h'_j(x)=i} a_x g_j(x) \right)^2 \right) - \|\mathbf{a}\|^4 \\ &\leq \sum_{v,x,y,z} 4g_j(v)g_j(x)g_j(y)g_j(z)a_v a_x a_y a_z \\ &\quad * \Pr[h'_j(v) = h'_j(x) = h'_j(y) = h'_j(z)] \\ &= 4 \sum_{x=1}^U \sum_{y=1, y \neq x}^U a_x^2 a_y^2 \Pr[h'_j(x) = h'_j(y)] \\ &= \frac{4}{w} \sum_{x=1}^U \sum_{y=1, y \neq x}^U a_x^2 a_y^2 \end{aligned}$$

This uses the 4-wise independence of the function h to imply 4-wise independence of g , and hence to show that products of 4 or fewer independent terms in g have expectation zero.

We again argue that, with probability at least $\frac{1}{2}$, the $m = w^{1/2}$ largest counts fall into different buckets. Consider the distribution of counts in the CM sketch only for such settings where this event occurs. For such distributions, then $\text{Var}(Y_j)$ is bounded as:

$$\begin{aligned} \text{Var}(Y_j) &\leq \frac{4\|\mathbf{a}\|_1^4}{w} (2 \sum_{i=1}^m f_i^2 \sum_{j=m+1}^U f_j^2 \\ &\quad + (\sum_{i=m+1}^U f_i^2)^2) \\ &\leq \frac{4\|\mathbf{a}\|_1^2}{w} (2 \sum_{i=1}^U f_i^2 \sum_{j=m+1}^U f_j^2) \\ &\leq 8\|\mathbf{a}\|_2^4 m^{1-2z}/w = 8\|\mathbf{a}\|_2^4 w^{-\frac{1-2z}{2}} \end{aligned}$$

Setting this equal to $\varepsilon^2 \|\mathbf{a}\|_2^4$ lets us apply the Chebyshev bound. This shows that $\Pr[|Y_j - \|\mathbf{a}\|_2^2| > 2\varepsilon \|\mathbf{a}\|_2^2] < \frac{1}{4}$ provided we have $\varepsilon^2 \geq 8w^{-\frac{(1+2z)}{2}}$. We can take the median of $O(\ln \frac{1}{\delta})$ independent repetitions of the estimator Y_j and apply Chernoff bounds in usual way to amplify this constant probability of success to $1 - \delta$. The space required is $O(\varepsilon^{\frac{-4}{1+2z}} \ln \frac{1}{\delta})$ for $z > \frac{1}{2}$. \square

6.3 Light Skew Case and Summary

For the case where $z \leq \frac{1}{2}$, we observe that by simply taking the variance of the CM⁻ estimator over all distributions, then it is directly bounded as $\text{Var}(Y_j) \leq 8\|\mathbf{a}\|_2^4/w$. Following the Chebyshev and Chernoff arguments results in space bounds of $O(\frac{1}{\varepsilon^2})$. This matches the space requirements for the previously best known algorithms for L_2 estimation of [4, 13, 49] up to small constant factors. Observe that

the update time is the same as the usual cost for updating a CM Sketch, which is $O(d) = O(\ln \frac{1}{\delta})$. Here we give much improved dependency on ε on space used for skewed distributions, as summarized in the table below.

Value of z	$z \leq \frac{1}{2}$	$\frac{1}{2} < z \leq 1$	$1 < z$
Space required	$O(\varepsilon^{-2})$	$O(\varepsilon^{\frac{-4}{1+2z}})$	$O(\varepsilon^{\frac{-2}{1+z}})$

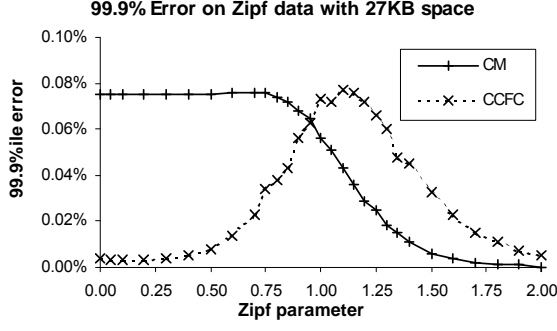
7 Experimental Study

We carried out an extensive experimental analysis of the Count Min sketch for point estimation and L_2 estimation. We made use of the public implementations of the data structure available from <http://www.cs.rutgers.edu/~muthu/massdal-code-index.html> as well as the Count sketch [10] for comparison. The Count sketch can also be used to answer point queries, and has a similar structure to the Count-Min sketch, being based around an array of counters. So in all experiments, the two methods were given exactly the same amount of space, in each case arranged as an array of counters with the same dimensions. This should give a fair comparison between the two methods. We refer to the Count-Min sketch as “CM”, and the Count sketch as “CCFC” (after the initials of its creators) for brevity. We considered synthetic datasets generated from Zipf distributions with known values of z , so that we could compare the behavior of the algorithm with that predicted by our analysis. We also considered various real data sets, two data sets in each category, text and network data.

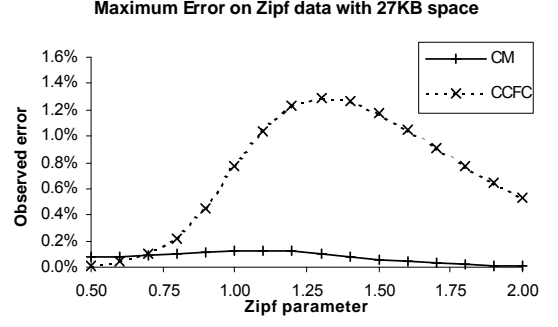
7.1 Synthetic Data

We made our synthetic data sets by using standard routines to draw values from a Zipf distribution with specified parameter z . Each experiment consisted of drawing 10^7 items from a domain of size $n = 10^6$ and computing the L_2 norm and all point queries over this domain. In evaluating the quality of our algorithms, we computed the exact solutions to all these queries, and so could compute the error in each result. For L_2 norms, we computed the fractional error as the difference between the estimated and actual value, scaled by the actual value. For simplicity, we worked with $F_2 = L_2^2$. For point queries, we computed the difference between the actual value and the estimate, and scaled by the number of items. We computed the maximum error observed, and the 99.9th percentile of the error (that is, sort the observed errors, and take the one whose rank is $\frac{999}{1000}N$). Since our algorithms give guaranteed bounds with a small probability of failure, this should test how well these bounds are met.

The first results are shown in Figure 1. These show the effect of fixing the space s for algorithms, but varying the skewness parameter of the input. Our theory predicts that the performance of the Count-Min sketch should be $\varepsilon \propto 1/s$ for $z < 1$, and $\varepsilon \propto 1/s^z$ for $z > 1$. We observe that this seems to be borne out in Figure 1(a): the error is roughly flat

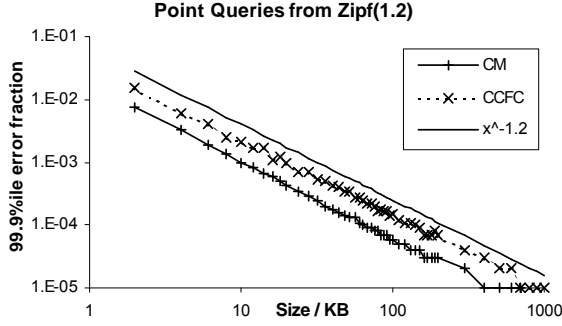


(a)

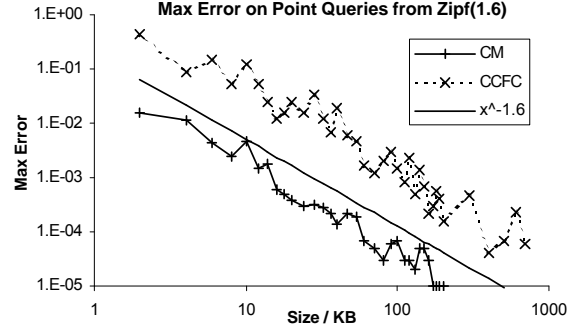


(b)

Figure 1: Testing point estimation on synthetic data



(a)



(b)

Figure 2: Testing space dependency of point estimation on synthetic data

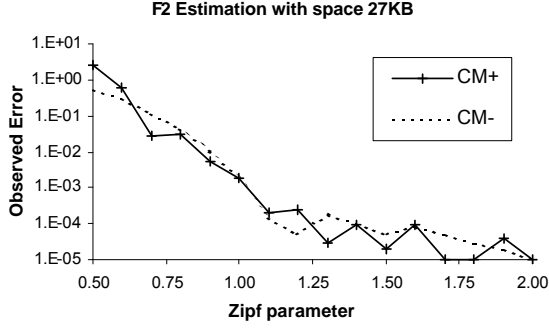
for $0.5 < z < 1$, and then falls off smoothly for larger z . We see that for $z > 1$ then the observed error is better for CM than for CCFC, justifying our analysis of the performance of these algorithms for skewed data sets. For distributions with skew $z \geq 2$, the observed error is sufficiently small to be negligible. A similar pattern of behavior is seen when we take the maximum observed error, in Figure 1(b). The main observation is that for skewed data, the largest error from the CCFC approach can become very high compared to that of CM, which is not much greater than in the previous case.³

Our theory predicts that, as space s increases, the error ε should decrease as s^{-z} . We show this to be the case in Figure 2. We plot the observed error when we fix the Zipf parameter, and increase the space for the sketch from

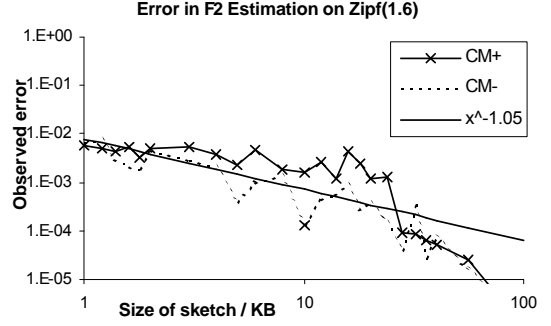
1KB to 1MB. Plotting observed error vs. space on a log-log plot should reveal a straight line with slope equal to $-z$. This is seen clearly in Figure 2(a), where we have plotted a line $y \propto x^{-1.2}$ for comparison. Note that this is a logarithmic scale plot, so the separation between the two lines is quite significant: CCFC consistently has about twice as much error. It appears to show a similar $x^{-1.2}$ dependency on size. Although the maximum error is much more variable, the same behavior occurs for $z = 1.6$ (Figure 2(b)), where CCFC has on average 10 times the error of CM, an order of magnitude worse. Several data mining problems need to manipulate item counts by summing and subtracting estimated values, so often this very fine accuracy is required, hence the need to get as good quality estimates as possible in small space.

For F_2 estimation, the results are less clearcut. We have two methods to use the Count-Min sketch in order to estimate the second frequency moment. The first, CM^+ ,

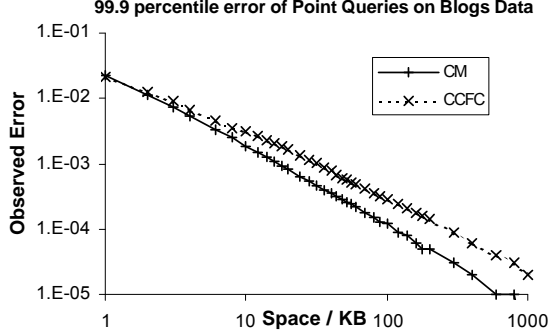
³We do not know why the CCFC algorithm appears to have a “bell-curve”-like behavior as z increases. This may be of interest for future analysis.



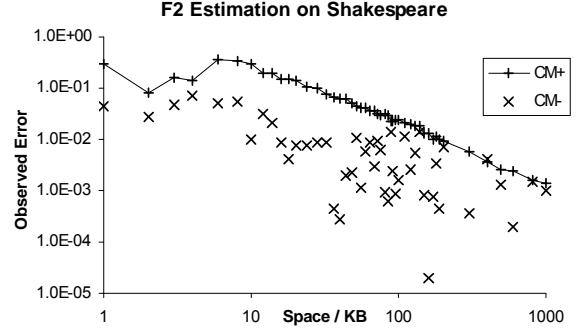
(a)



(b)

Figure 3: Testing space L_2 estimation on synthetic data

(a)



(b)

Figure 4: Results of experiments on text data

which has the best analytic results for $z > 1$, is to sum the squares of the counters. The second, CM^- , sums the squares of the differences of adjacent counters. In our experiments on synthetic data, illustrated in Figure 3, we were not able to observe a clear difference between the two approaches. As we increased z while keeping the space fixed, we saw that both methods seem to give about the same error. In Figure 3 (a), over the different values of z , CM^+ gets lower error more often than CM^- , but there is no clear trend. Figure 3 (b) shows the effect of increasing the size of the sketch for data with $z = 1.6$. Our theory predicts that the error of CM^- should behave as $s^{-\frac{1+2z}{4}} = s^{-1.05}$, and CM^+ as $s^{-\frac{1+z}{2}} = s^{-1.3}$. We have plotted the first of these on the same graph, since on this data set we can see this behavior for CM^- . The results for CM^+ are much less clear here, however when we examine real data sets we shall see the algorithms performing very closely in line with their predicted behavior.

7.2 Text Data

Zipf's law was first proposed in the context of linguistics, based on the observation that the frequency of the i th most frequent word in written text seemed to be roughly proportional to $1/i$ [53]. So it is fitting that we test our methods on mining textual data. We considered two data sources of seemingly very different nature. First, we used the complete plays of Shakespeare. This consists of 5MB of data, totaling approximately 1 million words. As a data source, it is quite 'clean', since words are spelled consistently throughout, and has been checked by many editors. Our second source of data consisted of a large amount text harvested from weblogs ("blogs"), totaling 1.5GB. This totaled over 100 million words from a large number of different authors, written in colloquial English (and some other languages mixed in), with no editing, in inconsistent styles and many errors left uncorrected. We did not attempt to clean this data, but ran our algorithms on it directly.

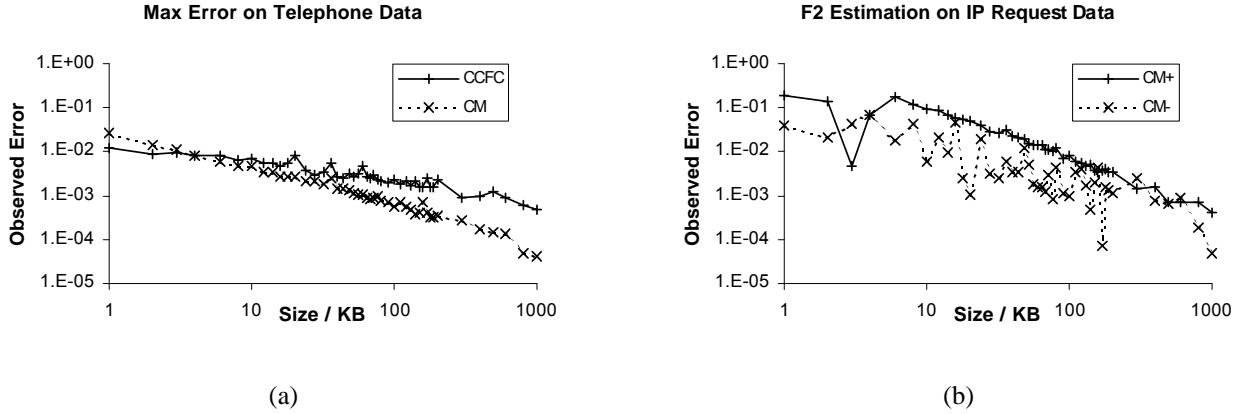


Figure 5: Results of experiments on network data

We show some of the results in Figure 4. We do not show all results for space reasons, but they are similar to those we present. On the log-log plot in Figure 4 (a) we can see a very clear linear gradient with slope approximately -1.2. This suggests that the Blog data is well-modeled by a Zipf distribution with parameter 1.2. We took the word frequencies from this data, and plotted those on a log-log chart, then computed the line of best fit; its slope was indeed approximately 1.2.⁴ For the Shakespeare dataset, we measured z as 1.2—1.3, implying that Shakespeare had similar relative frequencies of word usage as a Blog writer. In both cases, we see significant accuracy gains using CM Sketch over CCFC.

The linear behavior of the CM^+ estimation in Figure 4 (b) is quite striking. For sketch sizes 10KB—1MB, we measured a dependency of ε as $s^{-1.15}$. This implies a corresponding value of $z = 1.3$. The same trend is not obvious for the CM^- approach, but a best fit line gives the dependency $\varepsilon \propto s^{-0.85}$, which corresponds to $z = 1.2$. Recall that in both cases, we use the same sketch as the basis of the both estimation procedures (as well as for point queries). These results show us that using the CM^- estimation technique (sum of squares of differences) gives better results than the CM^+ approach (sum of squares). If the data is very skewed, or very fine accuracy is required, then CM^+ should be used, since asymptotically it has better bounds, but for this kind of data the CM^- method is preferable. The important feature is that we can make the sketch oblivious to the nature of the data, and only at query time decide which estimation technique to use, based on the observed skew.

⁴ In order to get a good fit of real data to a Zipf distribution, one typically has to drop the first few readings, and not fit the entire tail. Based on different sections of this chart, we measured Zipf parameters in the range 1.15 to 1.30, and so we conclude that 1.2 is within the bounds of uncertainty.

7.3 Network Data

We considered two types of data drawn from communication networks: a data set of 1.9 million phone calls, where we tracked the called exchange (a range of 1 million values); and a data set of Internet requests to 32-bit IP addresses, taken from the Internet Traffic Archive [38], LBL-CONN7 [46], totaling 800,000 requests. The maximum error on the phone call data is plotted in Figure 5 (a). Although it is a little fuzzier than the corresponding 99.9% error plot, the linear dependency on the log-log plot can be easily seen. The slope of the CM line is -1.16, predicting a skewness parameter of 1.16, while the slope of the CCFC line is around -0.8. Again, there is an order of magnitude improvement in the accuracy of CM over CCFC. For the Internet data, the error in point queries implies a skew of $z = 1.3$. This means that the slope for F_2 estimation should be 1.15, which is indeed what we measure on Figure 5 (b) for sketches between 10KB and 1MB using CM^+ for estimation. The slope of the CM^- line is less steep, about -0.9 as predicted, although again the observed error is less throughout most of the region of interest.

7.4 Timing Results

Since the update procedure is essentially the same for every update, the time cost is not much affected by the nature of the data. We conducted experiments on 1GHz and 2.4GHz processor machines, and observed similar update performance on each (since the algorithm is essentially bound by cache/memory access times), of about 2–3 million updates per second. By comparison, the implementation of the CCFC Count Sketch achieves a somewhat slower rate (40–50% slower), since it requires additional computation of a second hash function for every update. Greater speed can be achieved by taking advantage of the natural parallelism inherent in sketch data structures.

8 Conclusions

We have defined the problem of summarizing and mining data streams when these streams exhibit a skewed distribution. We have given practical algorithms for key post-hoc analysis problems with strong theoretical bounds of $o(1/\varepsilon)$ and $o(1/\varepsilon^2)$ where previously known results that did not exploit skew used space $\Omega(1/\varepsilon)$ and $\Omega(1/\varepsilon^2)$ respectively. In experiments, we have shown our CM sketch data structure to be a practical and flexible summary: not only does it outperform other methods for point queries and give accurate estimates for L_2 estimation, but it does this based on a simple update procedure. This approach can be employed without *a priori* knowledge of the distribution or skewness of the data: given fixed space, we can then bound the approximation quality based on the observed skew.

The two queries that we considered are fundamental to top-k items, change detection, approximate quantiles, anomaly detection and so on. Many other summarization and mining tasks can also benefit from the insight that data is rarely uniform, and realistic data is frequently highly skewed. For example, we remark that our methods in this paper will give estimates for *inner-product queries* between data streams as well in a straightforward way as an extension of [13]. This has applications to join size estimation in databases [3], to principal component analysis [31] and sparse correlation matrix estimation [32], but we do not elaborate further on this here. Likewise, the fact that skew is frequently seen at multiple levels of aggregation [33] means that our analysis can be immediately applied to *hierarchical computations*, such as computing range sums, estimating quantiles and so on (see [13] for these computations using CM sketch). With appropriate analysis and testing, methods that capitalize on data skew could improve our understanding of existing algorithms, inspire new methods, and move some tasks previously thought unachievable into the practical.

Acknowledgments We thank Yinmeng Zhang for some useful discussions, and the referees for their suggestions.

References

- [1] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R. Yan, and S. Zdonik. Aurora: a data stream management system. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 666–666, 2003.
- [2] L. Adamic. Zipf, power-law, pareto - a ranking tutorial. <http://www.hpl.hp.com/research/idl/papers/ranking/>, 2000.
- [3] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proceedings of the Eighteenth ACM Symposium on Principles of Database Systems*, pages 10–20, 1999.
- [4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996. Journal version in *Journal of Computer and System Sciences*, 58:137–147, 1999.
- [5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of ACM Principles of Database Systems*, pages 1–16, 2002.
- [6] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.
- [7] J. Baumes, M. Goldberg, M. Magdon-Ismael, and W. Wallace. Discovering hidden groups in communication networks. In *2nd NSF/NIJ Symposium on Intelligence and Security Informatics*, pages 126–137, 2004.
- [8] A. Bestavros, M. Crovella, and T. Taqqu. *Heavy-Tailed Probability Distributions in the World Wide Web*, pages 3–25. Birkhäuser, 1999.
- [9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM*, pages 126–134, 1999.
- [10] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.
- [11] G. Cormode, F. Korn, S. Muthukrishnan, T. Johnson, O. Spatscheck, and D. Srivastava. Holistic UDAFs at streaming speeds. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 35–46, 2004.
- [12] G. Cormode and S. Muthukrishnan. What’s hot and what’s not: Tracking most frequent items dynamically. In *Proceedings of ACM Principles of Database Systems*, pages 296–306, 2003.
- [13] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *Latin American Informatics*, pages 29–38, 2004.
- [14] G. Cormode and S. Muthukrishnan. What’s new: Finding significant differences in network data streams. In *Proceedings of IEEE Infocom*, 2004.
- [15] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.
- [16] A. Dobra, M. Garofalakis, J. E. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM Sigmod International Conference on Management of Data*, pages 61–72, 2002.
- [17] C. Estan and G. Varghese. Data streaming in computer networks. In *Proceedings of Workshop on Management and Processing of Data Streams*, <http://www.research.att.com/conf/mpds2003/schedule/estanV.ps>, 2003.
- [18] W. Fei, P. S. Yu, and H. Wang. Mining extremely skewed trading anomalies. In *International Conference on Extending Database Technology*, pages 801–810, 2004.
- [19] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L_1 -difference algorithm for massive data

- streams. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 501–511, 1999.
- [20] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2002.
- [21] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 389–398, 2002.
- [22] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of the International Conference on Very Large Data Bases*, pages 79–88, 2001. Journal version in *IEEE Transactions on Knowledge and Data Engineering*, 15(3):541–554, 2003.
- [23] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proceedings of the International Conference on Very Large Data Bases*, pages 454–465, 2002.
- [24] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 32(2):5–14, June 2003.
- [25] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical Report SRC 1998-011, DEC Systems Research Centre, 1998.
- [26] The himalaya project. <http://www.cs.cornell.edu/database/himalaya/Himalaya.htm>.
- [27] B. Huberman, P. Pirollo, J. Pitkow, and R. Lukose. Strong regularities in world wide web surfing. *Science*, pages 95–97, April 1998.
- [28] IBM Research — stream data mining. http://www.research.ibm.com/compsci/project_spotlight/kdd/.
- [29] W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.
- [30] NASA jet propulsion laboratory. <http://www7.nationalacademies.org/bms/BravermannasPDF.pdf>.
- [31] H. Kargupta and V. Puttagunta. An efficient randomized algorithm for distributed principal component analysis for heterogenous data. In *Workshop on high performance data mining at SIAM Intl Conf on Data mining*, 2004.
- [32] H. Kargupta and V. Puttagunta. Onboard vehicle data stream monitoring and fast computation of sparse correlation matrices. In *Workshop on data mining in resource constrained environments at SIAM Intl Conf on Data mining*, 2004.
- [33] E. Kohler, J. Li, V. Paxson, and S. Shenker. Observed structure of addresses in IP traffic. In *ACM SIGCOMM Internet Measurement Workshop*, pages 253–266, 2002.
- [34] N. Koudas and D. Srivastava. Data stream query processing: A tutorial. In *Proceedings of the International Conference on Very Large Data Bases*, page 1149, 2003.
- [35] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation and applications. In *Proceedings of the ACM SIGCOMM conference on Internet measurement*, pages 234–247, 2003.
- [36] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proceedings of the WWW Conference*, pages 568–576, 2003.
- [37] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.
- [38] Internet traffic archive. <http://ita.ee.lbl.gov/>.
- [39] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of 18th International Conference on Data Engineering*, pages 555–566, 2002.
- [40] D. Madigan. DIMACS working group on monitoring message streams. <http://stat.rutgers.edu/~madigan/mms/>, 2003.
- [41] MAIDS : Mining alarming incidents in data streams. <http://maids.ncsa.uiuc.edu/>.
- [42] G.S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the International Conference on Very Large Data Bases*, pages 346–357, 2002.
- [43] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [44] S. Muthukrishnan. Data streams: Algorithms and applications. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.
- [45] Ohio State CSE — algorithms for mining data streams. http://www.cse.ohio-state.edu/~agrawal/Research_new/mining.htm.
- [46] V. Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE ACM Transactions on Networking*, 2(4):316–336, 1994.
- [47] S. Redner. How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B*, pages 131–134, 1998.
- [48] Stanford stream data manager. <http://www-db.stanford.edu/stream/sqr>.
- [49] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 615–624, 2004.
- [50] H. Wang. Bibliography on mining data streams. <http://wis.cs.ucla.edu/~hwxwang/stream/bib.html>.
- [51] D. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 167–175, 2004.
- [52] Y. Zhu and D. Shasha. StatStream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the International Conference on Very Large Data Bases*, pages 358–369, 2002.
- [53] G. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison Wesley, 1949.

Online Analysis of Community Evolution in Data Streams

Charu C. Aggarwal, Philip S. Yu
IBM T. J. Watson Research Center
{ charu, psyu }@us.ibm.com

Abstract

This paper discusses the problem of online change detection in a large set of interacting entities. Such trends include the gradual formation and dissolution of different communities of interaction. Our results are focussed on the case where the interacting entities are received in the form of a *fast data stream* of interactions. In such cases, a user may wish to perform repeated *exploratory querying* of the data for different kinds of user-defined parameters. This is difficult to perform in a fast data stream because of the one-pass constraints on the computations. We propose an online analytical processing framework which separates out online data summarization from offline exploratory querying. The result is a method which provides the ability to perform exploratory querying without compromising on the quality of the results. The algorithms are tested over large sets of graph data streams with varying levels of evolution.

1 Introduction

The data stream has gained importance in recent years because of the greater ease in data collection methodologies resulting from advances in hardware technology. This has resulted in a host of papers on the extension of data mining techniques to the case of data streams [1, 2, 3, 8, 13]. In this paper, we discuss the problem of detecting patterns of interaction among a set of entities in a stream environment. Examples of such entities could be a set of businesses which interact with one another, sets of co-authors in a dynamic bibliography data base, or it could be the hyperlinks from web pages. In each of these cases, the interaction among different entities can rapidly evolve over time.

A convenient way to model the entity interaction relationships is to view them as graphs in which the nodes correspond to entities and the edges correspond to the interactions among the nodes. The weights on these edges represent the level of the interaction between the different participants. For example, in the case when the nodes represent interacting entities in a business environment, the weights on the edges among these entities could represent the volume of business

transactions. A *community of interaction* is defined to be a set of entities with a high degree of interaction among the participants.

The problem of finding communities in dynamic and evolving graphs been discussed in [6, 7, 9, 10, 11, 12, 14, 15, 16]. Since most of the current techniques are designed for applications such as the web, they usually assume a *gradually evolving* model for the interaction. Such techniques are not very useful for a fast stream environment in which the entities and their underlying relationships may quickly evolve over time. In addition, it is important to provide a user the *exploratory capability* to query for communities over different time horizons. Since individual points in the data streams cannot be processed more than once, we propose a framework which separates out the *offline exploratory* algorithms from the online stream processing part. The online stream processing framework creates summaries of the data which can then be further processed for exploratory querying. This paper is focussed on an Online Analytical Processing (OLAP) approach for providing offline exploratory capabilities to users in performing *change detection* across communities of interest over different time horizons.

Some examples of exploratory queries in which a user may be interested are as follows:

- (1) Find the communities with substantial increase in interaction level in the interval $(t - h, t)$. We refer to such communities as *expanding communities*.
- (2) Find the communities with substantial decrease in interaction level in the interval $(t - h, t)$. We refer to such communities as *contracting communities*.
- (3) Find the communities with the most stable interaction level in the interval $(t - h, t)$.

We note that the process of finding an emerging or contracting community needs to be carefully designed in order to normalize for the behavior of the community evolution over different time horizons. For example, consider a data stream in which two entities n_1 and n_2 share a high level of interaction in the period $(t - h, t)$. This alone does not mean that the interaction level between n_1 and n_2 is stable especially if these entities

had a even higher level of interaction in the previous period $(t - 2 \cdot h, t - h)$. Thus, a careful model needs to be constructed which tracks the behavior of the interaction graph over different time horizons in order to understand the nature of the change.

This paper is organized as follows. In the next section, we discuss notations for modelling the interaction among different entities. We also discuss methods for modelling the change in interaction among different entities. In section 4, we discuss methods to accumulate the statistics of the aggregate interaction among different entities. We also discuss methods to derive the change in interaction from these aggregate statistics. In section 5, we present the empirical results. Section 6 contains the conclusions and summary.

1.1 Contributions of this Paper This paper discusses new algorithms for online analysis of community detection in data streams. The paper discusses an OLAP-style framework in which the online preprocessing of the data stream is separated from the offline querying of the stream. Thus, the user can have the flexibility to query these summaries in an interactive way in order to find detailed information about the communities in the most relevant horizons. We design an innovative clustering algorithm which can determine clusters of interactions with the most significant change. This includes information about the disposition of the communities in terms of their expansion or contraction. Thus, the paper discusses a general framework for online analysis of the data stream.

2 Online Summarization of Graphical Data Stream

In this section, we will discuss the overall interaction model among the different entities. We will also discuss the process of online summarization of the data stream. This interaction model is stored as a graph $G = (N, A)$, in which N denotes the set of nodes, and A denotes the set of edges. Each node $i \in N$ corresponds to an entity. The edge set A consists of edges (i, j) , such that i and j are nodes drawn from N . Each edge (i, j) represents an interaction between the entities i and j . Each edge (i, j) also has a weight $w_{ij}(t)$ associated with it. This weight corresponds to the number of interactions between the entities i and j . For example, when the interaction model represents a bibliography database, the nodes could represent the authors and the weights on the edges could represent the number of publications on which the corresponding authors occur together as co-authors. As new publications are added to the database the corresponding weights on the individual edges are modified. It is also possible for new nodes to be added

to the data as new authors are added to the original mix. In this particular example, the weight on each edge increases by one, each time a new co-authorship relation is added to the database. However, in many applications such as those involving business interaction, this weight added in each iteration can be arbitrary, and in some cases even negative.

In order to model the corresponding stream for this interaction model, we assume that a current graph $G(t) = (N(t), A(t))$ exists which represents the history of interactions at time t . At time $(t + 1)$ new additions may occur to the graph $G(t)$. Subsequently, each new arrival to the stream contains two elements:

- An edge (i, j) corresponding to the two entities between whom the interaction has taken place.
- An *incremental* weight $\delta w_{ij}(t)$ illustrating the additional interaction which has taken place between entities i and j at time t .

We refer to the above pair of elements as representative of an *interaction event*. We note that the nodes i, j , or the edge (i, j) may not be present in $N(t)$ and $A(t)$ respectively. In such a case, the node set $N(t)$ and edge set $A(t)$ need to be modified to construct $N(t + 1)$ and $A(t + 1)$ respectively. In the event that a given edge does not exist to begin with, the original weight of (i, j) in $G(t)$ is assumed to be zero. Also, in such a case, the value of the edge set $A(t + 1)$ is augmented as follows:

$$(2.1) \quad A(t + 1) = A(t) \cup \{(i, j)\}$$

In the event that either the nodes i or j are not present in $N(t)$, the corresponding node set needs to be augmented with the new node(s). Furthermore, the weight of the edge (i, j) needs to be modified. If the edge (i, j) is new, then the weight of edge (i, j) in $G(t + 1)$ is set to δw_{ij} . Otherwise, we add the incremental weight δw_{ij} to the current weight of edge (i, j) in $G(t)$. Therefore, we have:

$$(2.2) \quad w_{ij}(t + 1) = w_{ij}(t) + \delta w_{ij}(t)$$

We assume that the set of interaction events received at time t are denoted by $\mathcal{E}(t)$. In each iteration, the stream maintenance algorithm adds the interaction events in $\mathcal{E}(t)$ to $G(t)$ in order to create $G(t + 1)$. We refer to the process of adding the events in $\mathcal{E}(t)$ to $G(t)$ by the \oplus operation. Therefore, we have:

$$(2.3) \quad G(t + 1) = G(t) \oplus \mathcal{E}(t)$$

At each given moment in time, we maintain the current graph of interactions $G(t)$ in main memory. In addition, we periodically store the graph of interactions

on disk. We note that the amount of disk storage available may often be limited. Therefore, it is desirable to store the graph of interactions in an efficient way during the course of the stream arrival. We will refer to each storage of the graph of interactions at a particular moment as a *frame*. Let us assume that the storage limitation for the number of frames is denoted by S . In this case, one possibility is to store the last S frames at uniform intervals of t' . The value of S is determined by the storage space available. However, this is not a very effective solution, since it means that a history of larger than $S \cdot t'$ cannot be recalled.

One solution to this problem is to recognize that frames which are more stale need not be stored at the same frequency as more recent frames. Let t_c be the current time, and t_{min} be the minimum granularity at which 0th tier snapshots are stored. We divide the set of S frames into $\theta = \log_2(t_c/t_{min})$ tiers. The i th tier contains snapshots which are separated by a distance of $t_{min} \cdot 2^{i-1}$. For each tier, we store the last S/θ frames. This ensures that the total storage requirement continues to be S . Whenever it is desirable to access the state of the interaction graph for the time t , we simply have to find the frame which is temporally closest to t . The graph from this temporally closest frame is utilized in order to approximate the interaction graph at time t . The tiered nature of the storage process ensures that it is possible to approximate recent frames to the same degree of (percentage) accuracy than less recent frames. While this means that the (absolute) approximation of stale frames is greater, this is quite satisfactory for a number of real scenarios. We make the following observations:

LEMMA 2.1. *Let h be a user-specified time window, and t_c be the current time. Then a snapshot exists at time t_s , such that $h/(1 + \theta/S) \leq t_c - t_s \leq (1 + \theta/S) \cdot h$.*

Proof. This is an extension of the result in [3]. The proof is similar.

In order to understand the effectiveness of this simple methodology, let us consider a simple example in which we store (a modest number of) $S = 100,000$ frames for a stream over 10 years, in which the minimum granularity of storage t_{min} is 1 second. We have intentionally chosen an extreme example (in terms of the time period of the stream) together with a modest storage capability in order to show the effectiveness of the approximation. In this case, the number of tiers is given by $\theta = \log_2(10 * 365 * 24 * 3600) \approx 29$. By substituting in Lemma 2.1, we see that it is possible to find a snapshot which is between 99.97% and 100.03% of the user specified value.

In order to improve the efficiency of edge storage further, we need to recognize the fact that large portions of the graph continue to be identical over time. Therefore, it is inefficient for the stream generation process to store the entire graph on disk in each iteration. Rather, we store only incremental portions of the graph on the disk. Specifically, let us consider the storage of the graph $G(t)$ for the i th tier at time t . Let the last time at which an $(i + 1)$ th tier snapshot was stored be denoted by t' . (If no snapshot of tier $(i + 1)$ exists, then the value of t' is 0. We assume that $G(0)$ is the null graph.) Then, we store the graph $F(t) = G(t) - G(t')$ at time t . We note that the graph $F(t)$ contains far fewer edges than the original graph $G(t)$. Therefore, it is more efficient to store $F(t)$ rather than $G(t)$. Another observation is that a snapshot for the i th tier can be reconstructed by summing the snapshots for all tiers larger than i .

LEMMA 2.2. *We assume that the highest tier is defined by m . Let t_i be the time at which the snapshot for tier i is stored. Let $t_{i+1}, t_{i+2} \dots t_m$ be the last time stamps of tiers $(i + 1) \dots m$ (before t_i) at which the snapshots are stored. Then the current graph $G(t_i)$ at the time stamp t_i is defined as follows:*

$$(2.4) \quad G(t_i) = \sum_{j=i}^m F(t_j)$$

Proof. This result can be proved easily by induction. We note that the definition of $F(\cdot)$ implies that:

$$\begin{aligned} G(t_i) - G(t_{i+1}) &= F(t_i) \\ G(t_{i+1}) - G(t_{i+2}) &= F(t_{i+1}) \\ &\dots \\ G(t_{m-1}) - G(t_m) &= F(t_{m-1}) \\ G(t_m) - 0 &= F(t_m) \end{aligned}$$

By summing the above equations, we obtain the desired result.

The above result implies that the graph at a snapshot for a particular tier can be reconstructed by summing it with the snapshots at higher tiers. Since there are at most $\theta = \log_2(S/t_{min})$ tiers, it implies that the graph at a given time can be reconstructed quite efficiently in a practical setting.

3 Offline Construction and Processing Of Differential Graphs

In this section, we will discuss the offline process of generating differential graphs and their application to

the evolution detection process. The differential graph is generated over a specific time horizon (t_1, t_2) over which the user would like to test the behavior of the data stream. The differential graph is defined over the interval (t_1, t_2) and is defined as a fraction of the interactions over that interval by which the level of interaction has changed during the interval (t_1, t_2) . In order to generate the differential graph, we first construct the *normalized graph* at the times t_1 and t_2 . The normalized graph $G(t) = (N(t), A(t))$ at time t is denoted by $\bar{G}(t)$, and contains exactly the same node and edge set, but with different weights. Let $W(t) = \sum_{(i,j) \in A} w_{ij}(t)$ be the sum of the weights over all edges in the graph $G(t)$. Then, the normalized weight $\bar{w}_{ij}(t)$ is defined as $w_{ij}(t)/W(t)$. We note that the normalized graph basically comprises the fraction of interactions over each edge.

Let t'_1 be the last snapshot stored just before time t_1 and t'_2 be the snapshot stored just before time t_2 . The first step is to construct the graphs $\bar{G}(t'_1)$ and $\bar{G}(t'_2)$ at time periods t'_1 and t'_2 by adding the snapshots at the corresponding tiers as defined by Lemma 2.2. Then we construct the normalized graph from the graphs at times t'_1 and t'_2 . The differential graph is constructed from the normalized graph by subtracting out the corresponding edge weights in the original normalized graphs. Therefore, the differential graph $\Delta G(t'_1, t'_2)$ basically contains the same nodes and edges as $\bar{G}(t'_2)$, except that the differential weight $\Delta w_{ij}(t'_1, t'_2)$ on the edge (i, j) is defined as follows:

$$(3.5) \quad \Delta w_{ij}(t'_1, t'_2) = \bar{w}_{ij}(t'_2) - \bar{w}_{ij}(t'_1)$$

In the event that an edge (i, j) does not exist in the graph $\bar{G}(t'_1)$, the value of $\bar{w}_{ij}(t'_1)$ is assumed to be zero. We note that because of the normalization process, the differential weights on many of the edges may be negative. These correspond to edges over which the interaction has reduced significantly during the evolution process. For instance, in our example corresponding to a publication database, when the number of jointly authored publications reduces over time, the corresponding weights in the differential graph are also negative.

Once the differential graph has been constructed, we would like to find clusters of nodes which show a high level of evolution. It is a tricky issue to determine the subgraphs which have a high level of evolution. A natural solution would be find the clustered subgraphs with high weight edges. However, in a given subgraph, some of the edges may have high positive weight while others may have high negative weight. Therefore, such subgraphs correspond to the entity relationships with high evolution, but they do not necessarily correspond

Algorithm FindEvolvingCommunities(Graph: (N, A) ,
EdgeWeights: $\Delta w_{ij}(t_1, t_2)$, NumberOfClusters: k);
begin
Randomly sample nodes $n_1 \dots n_k$ as seeds;
Let B be the bias vector of length $|N|$;
Set each position in B to 0;
while not(*termination_criterion*) **do**
 begin
 $(N_1 \dots N_k) = \text{AssignNodes}(N, B, \{n_1 \dots n_k\})$;
 $B = \text{FindBias}(N_1 \dots N_k)$;
 $(N'_1 \dots N'_k) = \text{RemoveNodes}(N_1 \dots N_k)$;
 { Assume that the removed nodes are null partitions }
 $(n_1 \dots n_k) = \text{RecenterNodes}(N_1 \dots N_k)$;
 end
end

Figure 1: Finding Evolving Communities

to entity relationships with the greatest increase or decrease in interaction level. In order to find the community of interaction with the greatest increase in interaction level, we need to find subgraphs such that most interactions within that subgraph have either a high positive or high negative weight. This is a much more difficult problem than the pure vanilla problem of finding clusters within the subgraph $\Delta G(t'_1, t'_2)$.

3.1 Finding Evolving Communities In this section, we will define the algorithm for finding evolution clusters in the interaction graph based on the user defined horizon. The process of finding the most effective clusters is greatly complicated by the fact that some of the edges correspond to an increase in the evolution level, whereas other edges correspond to a decrease in the evolution level. The edges corresponding to an increase in interaction level are referred to as the positive edges, whereas those corresponding to a reduction in the interaction level are referred to as the negative edges.

We design an algorithm which can effectively find subgraphs of positive or negative edges by using a partitioning approach which tracks the positive and negative subgraphs in a dynamic way. In this approach, we use a set of seed nodes $\{n_1 \dots n_k\}$ in order to create the clusters. Associated with each seed n_i is a partition of the data which is denoted by N_i . We also have a bias vector B which contains $|N|$ entries of $+1$, 0 , or -1 . The bias vector is an indicator of the nature of the edges in the corresponding cluster. The algorithm utilizes an iterative approach in which the clusters are constructed around these seed nodes. The overall algorithm is illustrated in Figure 1. As illustrated in Figure 1, we first pick the k seeds randomly. Next, we enter an iterative loop in which we refine the initial seeds by performing the following steps:

- We assign each nodes to one of the seeds. The process of assignment of an entity node to a given seed node is quite tricky because of the fact that we would like a given subgraph to represent either increasing or decreasing communities. Therefore, we need to choose effective algorithms which can compute the distances for each community in a different way. We will discuss this process in detail slightly later. We note that the process of assignment is sensitive to the nature of the *bias* in the node. The bias in the node could represent the fact that the cluster seeded by that node is likely to become one of the following: (1) An Expanding Community (2) A Contracting Community (3) Neutral Bias (Initial State). Therefore, we associate a *bias bit* with each seed node. This bias bit takes on the values of +1, or -1, depending upon whether the node has the tendency to belong to an expanding or contracting community. In the event that the bias is neutral, the value of that bit is set to 0. We note that an expanding community corresponds to positive edges, whereas a contracting community corresponds to negative edges. The process of assignment results in k node sets which are denoted by $N_1 \dots N_k$. In the assignment process, we compute the distance of each seed node to the different entities. Each entity is assigned to its closest seed node. The algorithm for assignment of entities to seed nodes is denoted by *AssignNodes* in Figure 1.
- Once the assignment step has been performed, we re-assess the bias of that seed node. This is denoted by *FindBias* in Figure 1. The overall approach in finding the bias is to determine whether the interactions in the community attached to that seed node represent expansion or contraction. We will discuss the details of this algorithm slightly later. The bias bit vector B is returned by this procedure.
- Some of the seed nodes may not represent a coherent community of interaction. These seed nodes may be removed by the community detection algorithm. This process is achieved by the algorithm *RemoveNodes*. The new set of nodes is denoted by $N'_1 \dots N'_k$. We note that each N'_i is either N_i or null depending upon whether or not that node was removed by the algorithm.
- The final step is to re-center the seeds within their particular subgraph. The re-centering process essentially reassigns the seed node in a given subgraph N'_i to a more central point in it. In the event that N'_i is a null partition, the recentering process simply picks a random node in the graph as the

corresponding seed. Once the recentering process is performed, we can perform the next iteration of the algorithm. The recentering procedure is denoted by *RecenterNodes* in Figure 1. The new set of seeds $n_1 \dots n_k$ are returned by this algorithm.

3.2 Subroutines for Determination of the Communities In the afore-mentioned discussion, we described the overall procedure for finding the communities of interaction. In this section, we will discuss the details of the subroutines which are required for determination of these communities. We will first discuss the procedure *AssignNodes* of Figure 1. The process of assigning the nodes to each of the centroids requires the use of the bias information stored in the bias nodes. Note that if a seed node has positive bias, then it needs to be ensured that the other nodes which are assigned to this seed node are related to it by positive interactions. The opposite is true if the bias on the seed node is negative. Finally, in the case of nodes with neutral bias, we simply need to find a path with high absolute interaction level. In this case, the positivity or negativity of the sign matters less than the corresponding absolute value. In order to achieve this goal, we define a *bias sensitive* distance function $f(n_1, n_2, b)$ between two nodes n_1 and n_2 for the bias bit b . For a given path P in the graph $\Delta G(t_1, t_2)$, we define the average interaction level $w(P)$ as the sum of the interaction levels on P divided by the number of edges on P . Therefore, we have:

$$(3.6) \quad w(P) = \sum_{(i,j) \in P} \Delta w_{ij}(t_1, t_2) / |P|$$

We note that a path with high average *positive* weight corresponds to a set of edges with increasing level of interaction. This can also be considered an expanding community. The opposite is true of a path with high average *negative* weight, which corresponds to a contracting community. Therefore, the value of $w(P)$ is equal to the weight of the path divided by the number of edges on that path. We also define the average *absolute* average interaction level $w^+(P)$ as follows:

$$(3.7) \quad w^+(P) = \sum_{(i,j) \in P} |\Delta w_{ij}(t_1, t_2)| / |P|$$

Note that the absolute interaction level does not have any bias towards a positive or negative level of interaction. Once we have set up the definitions of the path weights, we can also define the value of the interaction function $f(n_1, n_2, b)$. This interaction function is defined over all possible pairs of nodes (n_1, n_2) .

$f(n_1, n_2, b) = \text{Most positive value of } w(P) \forall \text{ paths}$

P between n_1 and n_2 if $b = 1$
 Modulus of most negative value of $w(P)$
 $\forall P$ between n_1 and n_2 if $b = -1$
 Largest value of $w^+(P) \forall$ paths
 P between n_1 and n_2 if $b = 0$

We note that the above interaction function is defined on the basis of the sum of the interaction values over a given path. In some cases, this interaction function can provide skewed results when the path lengths are long. This could result in less effective partitioning of the communities. A different interaction function is defined as the minimum interaction on the path between two entities. However, the bias of the corresponding centroid on that path is used in order to define the interaction function. This minimum interaction $w(P)$ is defined as follows:

$$\begin{aligned}
 w(P) = & \min_{(i,j) \in P} \max\{\Delta w_{ij}(t_1, t_2), 0\} \\
 & \text{if } b = 1 \\
 & \max_{(i,j) \in P} \min\{\Delta w_{ij}(t_1, t_2), 0\} \\
 & \text{if } b = -1 \\
 & \min_{(i,j) \in P} |\Delta w_{ij}(t_1, t_2)| \\
 & \text{if } b = 0
 \end{aligned}$$

We note that the above mentioned function simply finds the minimum (absolute) weight edge of the corresponding sign (depending on the bias) between the two nodes. The corresponding interaction function $f(n_1, n_2, b)$ is defined in the same way as earlier. Henceforth, we will refer to the two above-defined functions as the average-interaction function and minimum interaction function respectively. In the latter case, the interaction distance corresponds to the interaction on the weakest link between the two nodes. As our experimental results will show, we found the minimum function to be slightly more robust than the average interaction function.

During the assignment phase, we calculate the value of the function $f(n_i, n, b)$ from each node n_i to the seed node n using the bias bit b . Each node n_i is assigned to the seed node n with the largest *absolute* value of the above-mentioned function. This process ensures that nodes are assigned to seeds according to their corresponding bias. The process of computation of the interaction function will be discussed in some detail slightly later.

Next, we determine the bias of each seed node in the procedure *FindBias*. In order to do so, we calculate the bias-index of the community defined by that seed node. The bias index of the community N_i is denoted by $\mathcal{I}(N_i)$, and is defined as the edge-weight fraction of the expanding portion of N_i . In order to do, so we

divide the positive edge weights in the community by the total absolute edge weight in the same community. Therefore, we have:

$$(3.8) \quad \mathcal{I}(N_i) = \frac{\sum_{(p,q) \in N_i} \max\{0, \Delta w_{pq}(t_1, t_2)\}}{\sum_{(p,q) \in N_i} |\Delta w_{pq}(t_1, t_2)|}$$

We note that the bias index is 1 when all the edges in the community corresponding to increasing interaction, and is 0 when all the edges correspond to reducing interaction. Therefore, we define a threshold $t \in (0, 0.5)$. If the value of $\mathcal{I}(N_i)$ is less than t then the bias bit is set to -1. Similarly, if the value of $\mathcal{I}(N_i)$ is larger than $1 - t$, the bias bit is set to 1. Otherwise, the bias bit is set to zero.

Once the bias bits for the nodes have been set, we remove those seeds which have very few nodes associated with them. Such nodes usually do not correspond to a coherent community of interaction. This procedure is referred to as *RemoveNodes*. Thus, each set of nodes N_i is replaced by either itself or a null set of nodes. In order to implement this step, we use a minimum threshold on the number of nodes in a given partition. This threshold is denoted by mn_t . All partitions with less than mn_t entities are removed from consideration, and replaced by the null set.

The last step is to recenter the nodes within their corresponding partition. This denoted by the procedure *RecenterNodes* in Figure 1. The process of recentering the nodes requires us to use a process in which the central points of subgraphs are determined. In order to recenter the nodes, we determine the node which minimizes the maximum distance of any node in the cluster. This is achieved by computing the distance of all points in the cluster starting at each node, and finding the minimax distance over these different values. The process of recentering helps to adjust the centers of the nodes in each iteration such that the process of partitioning the community sets becomes more effective over time.

3.3 Approximating Interaction Distances Among Nodes

The only remaining issue is to discuss the methodology for determining the interaction distances among nodes. We would like our algorithm to be general enough to find the maximum interaction distance for general functions. It is important to understand that the problem of finding the maximum interaction distance between two nodes is NP-hard.

OBSERVATION 3.1. *The problem of determining the maximum interaction distance between two nodes is NP-hard for arbitrary interaction functions.*

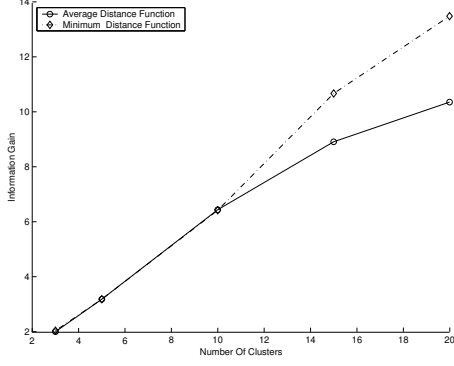


Figure 2: Information Gain with Number Of Clusters (C4.I5.D500)

This observation is easily verified by observing that the problem of finding the longest path in a graph is NP-hard [5]. Since the particular case of picking the interaction function as the edge weight is NP-hard, the general problem is NP-hard as well.

However, it is possible to approximate the interaction distance of a maximum length using dynamic programming. Let wn_{ij}^t be the maximum interaction distance between two nodes using at most t nodes on that path. Let P_{ik}^t be the maximum length path between i and k using t edges. Let PS_{ikj}^\oplus denote the path obtained by concatenating P_{ik}^t with the edge (k, j) . Then, we define wn_{ij}^t recursively as follows:

$$wn_{ij}^0 = 0;$$

$$wn_{ij}^{t+1} = \max_k \{wn_{ij}^t, w(PS_{ikj}^\oplus)\}$$

We note that this dynamic programming algorithm does not always lead to an optimal solution in the presence of cycles in the graph [5]. However, for small values of t , it approximates the optimal solution well. This is because cycles are less likely to be present in paths of smaller length. It is also important to understand that if two entities are joined only by paths containing a large number of edges, then such pairs of entities should not be regarded as belonging to the same community. Therefore, we imposed a threshold *maxthresh* on the maximum length of the (shortest interaction distance) path between two nodes for them to belong to the same community. For a pair of nodes in which the corresponding path length was exceeded, the value of the interaction distance is set to 0.

4 Empirical Results

For the purpose of testing, we generated a number of graphs with strong correlations of interaction among the different nodes. In general, it was desirable to generate

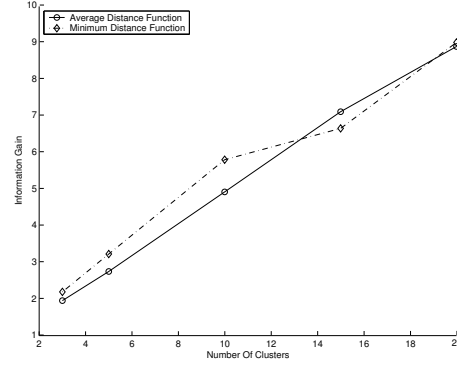


Figure 3: Information Gain with Number Of Clusters (C5.I6.D500)

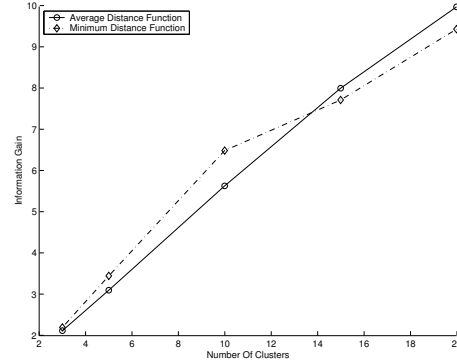


Figure 4: Information Gain with Number Of Clusters (C6.I7.D500)

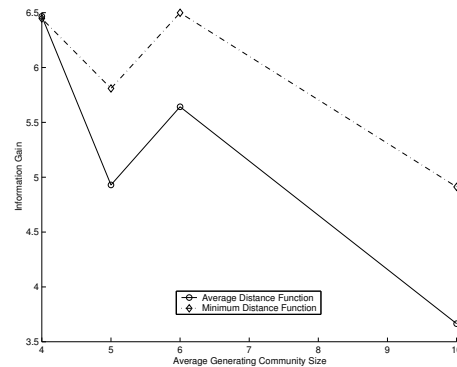


Figure 5: Information Gain with Increasing Community Set Size

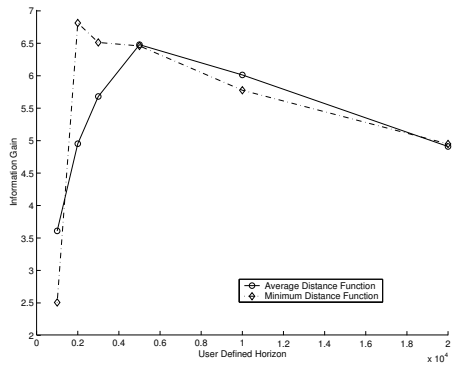


Figure 6: Information Gain with Increasing User Specified Horizon (C4.I5.D500)

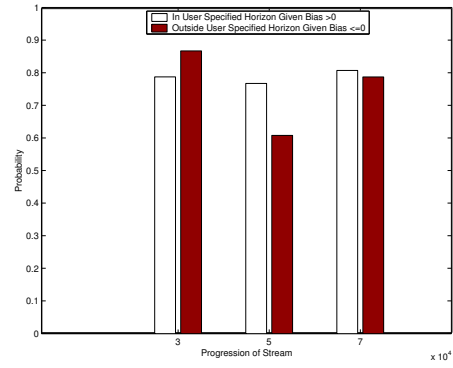


Figure 9: Effectiveness of finding expanding or contracting communities

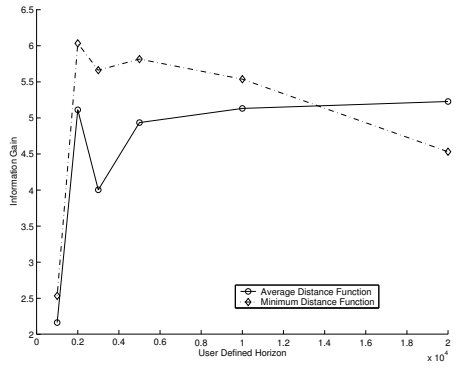


Figure 7: Information Gain with Increasing User Specified Horizon (C5.I6.D500)

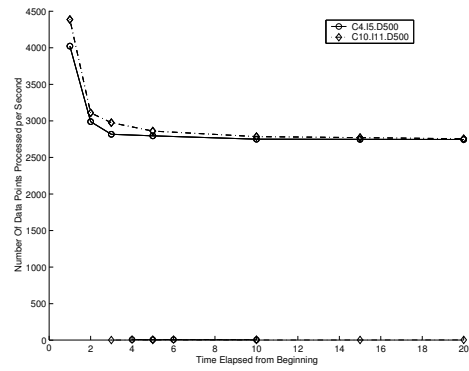


Figure 10: Stream Processing Rate

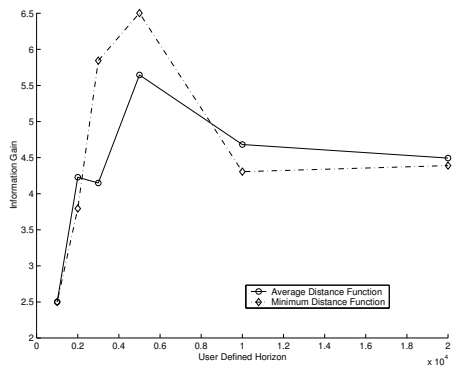


Figure 8: Information Gain with Increasing User Specified Horizon (C6.I7.D500)

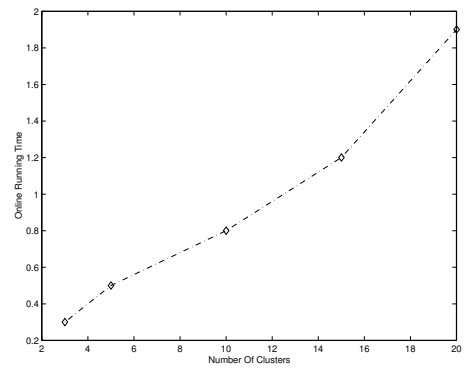


Figure 11: Interaction Times for Clustering

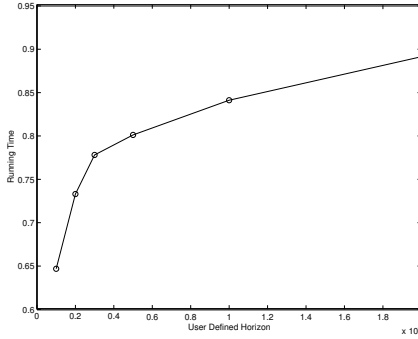


Figure 12: Interaction Time for Clustering

graphs in which the nodes showed strong interactions with each other, as well as variations in behavior over time. In order to achieve this goal, we used a technique which draws its motivation from the frequent itemset generation methodology discussed in [4]. However, the generation method was modified in order to suit the community detection problem.

The first step was to generate $L = 2000$ maximal “potential communities” over the nodes in the graph. These potential communities capture the tendencies of particular groups of entities to interact with one another. We first picked the size of a maximal potentially large community as a random variable from a poisson distribution with mean μ_L . Each successive community was generated by picking half of its items from the current community, and generating the other half randomly. This method ensures that large communities often have common entities. Each community I has a weight w_I associated with it, which is chosen from an exponential distribution with unit mean.

The maximal potential communities were then used in order to generate the *community sets* of interaction. First, the size S_T of a *current community set* was chosen as a poisson random variable with mean μ_T . Each current community set was generated by assigning maximal potentially large communities to it in succession. The community to be assigned to a community set was chosen by rolling an L sided weighted die depending upon the weight w_I assigned to the corresponding community I . If a community did not fit exactly, it was assigned to the current community set half the time, and moved to the next community set the rest of the time. In order to capture the fact that all members in a potentially large community may not always participate in the current interaction, we added some noise to the process by corrupting some of the added communities. For each community I , we decide a noise level $n_I \in (0, 1)$. We generated a geometric random variable G with param-

eter n_I . While adding a potentially large community to a community set, we dropped $\min\{G, |I|\}$ random nodes from the set. The noise level n_I for each community I was chosen from a normal distribution with mean 0.5 and variance 0.1.

We shall also briefly describe the symbols that we have used in order to annotate the data. The three primary factors which vary are the average community set size μ_T , the size of an average maximal potentially large community μ_L , and the number of community sets being considered. A data set having $\mu_T = 10$, $\mu_L = 4$, and 100 community sets is denoted by C10.I4.D100.

Next, we generate the edges of the graph stream from these community sets. For this purpose, we use two parameters which are referred to as the *epochsize* and the *maxrange*. The generation of the graph edges evolves over time and is divided into different periods which are referred to as an epoch. Within a given epoch, the edges of the graph are generated from the same distribution. At any given epoch, a set of *maxrange* contiguous community sets are used to generate the interactions among different entities. Within a given epoch, we generated *epochsize* such interactions. Each of these interactions was generated by picking two nodes from the community set, and assigning a random interaction level to this edge. Unless otherwise mentioned, we used a value of *maxrange* = 21 and *epochsize* = 500.

The interaction stream was processed in an online fashion to generate the community graph $G(t)$, which were stored using our tiered scheme. The stored summary information could then be utilized by a user to perform interactive clustering over multiple horizons. This is a more useful approach because of the potentially exploratory nature of user queries. The process of matching the output clusters to the original community sets was much more difficult since the original community sets had considerable overlaps with each other. Therefore, a given node could fall into multiple community sets, and could not be meaningfully matched with the original clusters. However, it is possible to calculate the level of correlation between the input and the output clusters using a measure called *information gain*. This information gain is defined by analyzing the level of fragmentation of the community sets among the different clusters. Ideally, we would like to have a situation in which the community sets do not get fragmented among the different clusters. Consider a community set T , which gets distributed among the k clusters such that the community i contains a fraction $p_i(T)$ of the entities in that cluster. Then, we define the gini index of

community set T as follows:

$$(4.9) \quad Gini(T) = \sum_{i=1}^k p_i(T)^2$$

We note that if the community set T occurs in only one cluster, then the value of $Gini(T)$ is 1 unit. On the other hand, when the community gets divided among the different clusters equally, the value of this function is $1/k$. Therefore, the information gain for the community set T is defined as the ratio of $Gini(T)$ to $1/k$.

$$(4.10) \quad InformationGain(T) = k \cdot Gini(T)$$

In order to calculate the overall information gain within a particular range, we need to use all the generating community sets for a given horizon. Let this generating community set be denoted by \mathcal{G} . Then, the information gain $\mathcal{IN}(\mathcal{G})$ for the community set collection \mathcal{G} is defined as follows:

$$(4.11) \quad \mathcal{IN}(\mathcal{G}) = \sum_{T \in \mathcal{G}} InformationGain(T)/|T|$$

The value of $\mathcal{IN}(\mathcal{G})$ defines an intuitive measure for the information gain of all community sets in \mathcal{G} . A value of $\mathcal{IN}(\mathcal{G})$ which is significantly larger than 1 defines a clustering which is able to distinguish between the different community sets \mathcal{G} very well. The use of information gain as a clustering measure is specially effective in situations in which there is considerable overlap between different communities. In order to test the quality of the clustering process, we computed the information gain for all the generating community sets within the currently specified user horizon.

We tested the information gain for a variety of different data sets and ranges of input/output parameters. The data sets from which the streams were generated were C4.I5.D500, C5.I6.D500, and C6.I7.D500. In the first set of results of Figures 2, 3, and 4 we have illustrated the information gain variation with the number of input clusters. The number of clusters are plotted on the X-axis, whereas the information gain is plotted on the Y-axis. The horizon was fixed at 5000 for each case. Furthermore, for each case we have illustrated the results for the use of the average distance function as well as the minimum value distance function. The general observation was that both distance functions were roughly competitive, though the minimum distance function provided superior results. A closer examination of the clusters revealed that the use of the average distance function resulted in some clusters with poor locality behavior. Such clusters did not contribute significantly to the value of the information gain. For

all cases, the information gain was significantly larger than 1, and increased with the number of input clusters. The increase in information gain with the number of input clusters is expected since a larger number of input clusters helps separate out the different overlapping community sets much more effectively.

Another observation from the results of Figures 2, 3, and 4 is that the information gain for the data in Figure 2 was much higher than the information gain in either of the other two data sets. The data in Figure 2 corresponds to the case in which the average community sizes were much smaller. In such cases, there is lower overlap among the different communities and it is possible to find more informative clusters. This also results in a higher level of information gain. In order to test the behavior with changing community size more explicitly, we varied the size of the generating community set. Specifically, we used the generating community set Cx.I(x+1).D500. The results are illustrated in Figure 5. The value of x was made to vary on the X-axis. It is clear that the information gain generally reduces with increasing community size. As in previous cases, the minimum value distance function provided more effective results than the average distance function.

We also tested the variation of the information gain with the user specified horizon. The results are illustrated in Figures 6, 7, and 8 respectively. The number of clusters were fixed at 10 in each case. The user specified horizon is plotted on the X-axis, whereas the information gain is plotted on the Y-axis. One of the interesting observations was that in each case, the information gain peaked at approximately the same value of the user specified horizon. The exact value of this user specified horizon was determined by the rate of evolution of the data stream. This rate of evolution was in turn determined by parameters such as *maxrange* and *epochsize*. In particular, the value of *epochsize* (which was fixed at 500) corresponds to the horizon in which a clear evolution of the stream can be determined. This is because within a given epoch, the base community sets do not change, but their frequency evolves over time. Correspondingly, the most distinct clusters can also be determined for this choice of the horizon. The graphs in Figures 6, 7, and 8 illustrate this trend in which the information gain peaks at approximately 500, but can also be a little different in some cases. For example, in the case of Figure 7, the clustering process does not peak at any point with the use of the average distance function. This is because of the randomness inherent in the clustering process itself. In each case, the information gain was significantly higher than the break-even value of 1. Furthermore, the behavior was quite similar over different data sets and robust over the

entire range of horizons tested. This suggests that the technique is able to find interesting and useful clusters over different values of the horizon.

While the information gain is an effective measure of the clustering process, when there are a very large number of overlapping clusters, we can get a more intuitive idea of the nature of the community clusters by using a smaller number of generating community sets with very large epoch sizes. Therefore, we used the generating community set C10.I11.D25, with *maxrange* = 2 and *epochsize* = 10,000. This effectively means that in a given epoch, 10,000 edges are generated using two overlapping communities. Then, we utilized user horizons of size 20,000 to test how the different clusters correspond to expanding or contracting communities. In order to provide a measure of an entity's interaction, we compared its level of interaction in the user specified horizon (t_h, t) to that in the previous horizon $(t - 2 \cdot h, t - h)$. For a given clustering, we separated out the nodes with positive bias from the nodes with negative or neutral bias. For each kind of node, we tested the probability whether or not its interactions in $(t - 2 \cdot h, t)$ belonged to the currently specified horizon $(t - h, t)$. The tests were performed at three different points in the stream processing. The results are illustrated in Figure 9. It is clear that a positive bias corresponds to a very high probability that nodes are present in the user specified horizon and vice-versa. This is consistent with the aim of finding expanding or contracting communities. We note that a perfect classification is not possible since in some cases, entities were present both inside and outside the horizon. Such entities could belong to either expanding or contracting communities depending upon the relative frequency in either. However, in most cases, the conditional probability of an interacting entity (not) belonging to a given horizon, given the bias bit is greater (less) than zero, is greater than 80%. Therefore, the positive and negative bias correspond to the expected phenomenon of expanding or contracting communities. It also illustrates that by using the bias bit of a cluster, it is possible to identify expanding or contracting communities in a real application.

To summarize, the following properties of the community detection algorithm were observed:

- (1) The minimum value distance function provided more robust results than the average distance function. This is especially because of the more robust computation of the minimum distance function in the case of larger clusters.
- (2) The clustering process was more robust when it was able to separate out the different (overlapping)

clusters. Larger community sets resulted in greater level of overlap among different clusters. Therefore, the information gain was much higher when the community sets were smaller.

- (3) A larger number of clusters resulted in a higher level of information gain. This is because the number of overlapping community sets were very large and increasing the number of input clusters also increased the level of information gain.

- (4) The clustering process was most effective when the horizon was chosen in a way so as to match with the evolution epoch in the data generation process. Thus, the clustering process was sensitive to the level of evolution in the data stream.

- (5) The bias bit could be used to identify expanding or contracting communities.

We tested the scalability of both the online stream processing component and the offline and interactive stream clustering component. The stream processing component is important in order to be able to process a high number of data points per second. In Figure 10, we have illustrated the scalability of the stream processing component. The results for the two data sets C4.I5.D500 and C10.I11.D500 are reported in terms of the number of data points processed per second. Therefore, the X-axis illustrates the progression of the data stream whereas the Y-axis illustrates the number of data points processed per second at that point in the stream progression. In each case, the initial processing rate was much higher than the steady state processing rate. This is because the initial processing needed to store very sparse graphs. Such graphs could be stored away relatively efficiently. We note that the data sets corresponding to C10.I11.D500 resulted in denser graphs than those generated from C4.I5.D500. The generation of such denser graphs also resulted in a lower processing rate per second. This is clear from the results illustrated in Figure 10. However, the difference between the two graphs is relatively minor. In each case, the data stream mining framework is able to process thousands of data points per second.

We also tested the interaction time scalability of the data stream. The scalability was tested with respect to the number of clusters as well as the value of the user-specified horizon. In Figure 11, we have illustrated the scalability of the interaction time results with respect to the number of the clusters in the data. The user-defined horizon was fixed at 5000 in this case. In each case, the results were averaged over 10 different queries. We found that the interaction time increased approximately linearly with the number of clusters in the data. The results with respect to the user-specified horizon are

illustrated in Figure 12. The number of clusters in the data were fixed at 10 in this case. In each case, the interaction time increased with the horizon, though the rate of increase was less than linear. This is because very small horizons lead to extremely sparse graphs which cluster very fast. With an increasing horizon, the differential graph grows more dense. This is because a larger number of edges can be included in a given horizon. A more dense differential graph also results in a greater amount of interaction time. However, the rate at which the interaction time increases with horizon is sublinear. This is because the denseness in the graph levels off after a certain point. This behavior also shows up in the interaction time. In each case, the interaction times turned out to be extremely small and were usually smaller than 1 or 2 seconds. This tends to indicate that the process can be efficiently used in order to perform online mining of the data streams effectively.

5 Conclusions and Summary

In this paper, we discussed a method for effective community detection of data streams. The approach in this paper finds communities in the graph data stream using an online approach in which we find the most relevant changes over a pre-defined horizon. The results show that the techniques can find the relevant communities in the data effectively even in case of considerable overlap among the different constituents. The quality of the clusters were defined in terms of an intuitive measure known as the *information gain*. The clusters in the data were shown to have a very high level of information gain compared to the breakeven behavior. The community detection approach separates out the online stream processing part from the offline community detection part which is based on user-defined parameters. Such an approach provides the maximum flexibility, since it is possible to process a high speed data stream without losing the ability to perform exploratory querying. The results show that the online processing part is very efficient and can process thousands of interactions per second. At the same time, the offline interaction component is able to process large community relationship sets in online interaction times. This provides a user with a comprehensive framework to query for changes in the community behavior in online interaction times.

References

- [1] C. C. Aggarwal, *A Framework for Diagnosing Changes in Evolving Data Streams*, ACM SIGMOD Conference, (2003).
- [2] C. C. Aggarwal, J. Han, J. Wang, and P. Yu, *A Framework for Clustering Evolving Data Streams*, VLDB Conference, (2003).
- [3] C. C. Aggarwal, J. Han, J. Wang, and P. Yu, *On-Demand Classification of Evolving Data Streams*, ACM KDD Conference, (2004).
- [4] R. Agrawal, and R. Srikant, *Fast Algorithms for Mining Association Rules*, VLDB Conference, (1994).
- [5] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, (1992).
- [6] C. Cortes, D. Pregibon, and C. Volinsky, *Communities of Interest*, Proceedings of Intelligent Data Analysis, (2001).
- [7] C. Cortes, D. Pregibon, and C. Volinsky, *Computational Methods for Dynamic Graphs*, Journal of Computational and Graphical Statistics, 12, (2003), pp. 950-970.
- [8] P. Domingos, and G. Hulten, *Mining High-Speed Data Streams*, ACM SIGKDD Conference, (2000).
- [9] D. Gibson, J. Kleinberg, and P. Raghavan, *Inferring Web Communities from Link Topology*, Proceedings of the 9th ACM Conference on Hypertext and Hypermedia, (1998).
- [10] D. Kempe, J. Kleinberg, and E. Tardos, *Maximizing the Spread of Influence Through a Social Network*, ACM KDD Conference, (2003).
- [11] J. Kleinberg, *Authoritative Sources in a Hyperlinked Environment*, ACM SODA Conference, (1998).
- [12] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins, *On the Bursty Evolution of Blogspace*, Proceedings of the WWW Conference, (2003).
- [13] G. Hulten, L. Spencer, and P. Domingos, *Mining Time Changing Data Streams*, ACM KDD Conference, (2001).
- [14] S. Rajagopalan, R. Kumar, P. Raghavan, and A. Tomkins, *Trawling the Web for emerging cyber-communities*, Proceedings of the 8th WWW conference, (1999).
- [15] N. Imafuji, and M. Kitsuregawa, *Finding a Web Community by Maximum Flow Algorithm with HITS Score Based Capacity*, DASFAA, (2003), pp. 101-106.
- [16] M. Toyoda, and M. Kitsuregawa, *Extracting evolution of web communities from a series of web archives*, Hypertext, (2003) pp. 28-37.

Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window

Chih-Hsiang Lin Ding-Ying Chiu Yi-Hung Wu
g914346@alumni.nthu.edu.tw dr908312@cs.nthu.edu.tw yihwu@mx.nthu.edu.tw
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.

Arbee L. P. Chen*
alpchen@cs.nccu.edu.tw
Department of Computer Science
National Chengchi University
Taipei, Taiwan, R.O.C.

Abstract

Mining frequent itemsets has been widely studied over the last decade. Past research focuses on mining frequent itemsets from static databases. In many of the new applications, data flow through the Internet or sensor networks. It is challenging to extend the mining techniques to such a dynamic environment. The main challenges include a quick response to the continuous request, a compact summary of the data stream, and a mechanism that adapts to the limited resources. In this paper, we develop a novel approach for mining frequent itemsets from data streams based on a time-sensitive sliding window model. Our approach consists of a storage structure that captures all possible frequent itemsets and a table providing approximate counts of the expired data items, whose size can be adjusted by the available storage space. Experiment results show that in our approach both the execution time and the storage space remain small under various parameter settings. In addition, our approach guarantees no false alarm or no false dismissal to the results yielded.

1 Introduction

Data items continuously flow through the Internet or sensor networks in applications like network monitoring and message dissemination. Efforts have been made at providing a data stream management system (DSMS), e.g., Telegraph [6], STREAM [25], Niagara [11], and Aurora [1]. The characteristics of data streams are as follows [4][17]:

1. **Continuity:** Data continuously arrive at a high rate.
2. **Expiration:** Data can be read only once.
3. **Infinity:** The total amount of data is unbounded.

The above leads to the following requirements:

1. **Time-sensitivity:** A model that adapts itself to the time passing of a continuous data stream is needed.
2. **Approximation:** Because the past data cannot be stored, a method for providing the approximate answers with accuracy guarantees is required.
3. **Adjustability:** Owing to the unlimited amount of data, a mechanism that adapts itself to available resources is needed.

Among the researches toward DSMS, extending mining techniques to data streams has attracted much attention [19][26][28][10][27]. In this paper, we focus on the problem of mining frequent itemsets over a data stream. In this problem, a data stream is formed by transactions arriving in series. The *support count* of an itemset means the number of transactions containing it and a *frequent* itemset means the one with a sufficient support count.

Mining frequent itemsets in static databases has been widely studied over the last decade. Many methods such as Apriori [2], FP-growth [18], and OpportuneProject [23] have been proposed. In addition, the methods that incrementally mine frequent itemsets in dynamic databases [12][22][8] have been presented as well. In these methods, all the frequent itemsets and their support counts derived from the original database are retained. When transactions are added or expired, the support counts of the frequent itemsets contained in them are recomputed. By reusing the frequent itemsets and their support counts retained, the number of candidate itemsets generated during the mining process can be reduced. All these methods have to rescan the original database because non-frequent itemsets can be frequent after the database is updated. Therefore, they cannot work without seeing the entire database and cannot be applied to data streams.

Recent works on mining frequent itemsets over data streams are classified into two groups, mining frequent items and mining frequent itemsets. Most of them [15][21][24] utilize all the data between a particular point of time (called *landmark*) and the current time for mining. The landmark usually refers to the time when the system starts. Moreover, the *support count* of an itemset in this model is the number of transactions containing it between the landmark and the current time. The landmark model is illustrated in Figure 1.

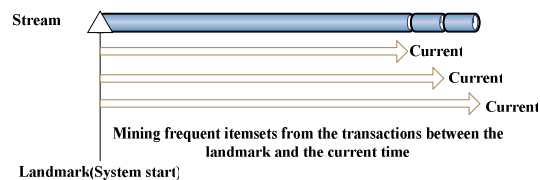


Figure 1: Landmark model

To find frequent items [15][21][24] under this model, the support count of each incoming item is

*To whom all the correspondence should be sent.

accumulated on a counter. Since the number of distinct items is often more than available counters, sampling techniques are employed to assign items to counters and then estimate the support counts of all the items.

For mining frequent itemsets, Lossy-counting [24] is the representative approach under the landmark model. It keeps monitoring the maximum possible count of each itemset in the past data, called the *maximum possible error*. Given an error tolerance parameter and a support count threshold, this approach computes the approximate count of each itemset with an accuracy guarantee and regards the itemsets whose approximate counts exceed the support count threshold as frequent. Since the approximate count of an itemset keeps growing as time goes by, the support count threshold is also increasing along the time axis.

All these approaches satisfy one requirement mentioned above – approximation. However, in many applications, new data are often more important than old ones. For example, when mining the Web click streams, the most recent data usually provides more useful information than those that arrived previously. The landmark model is not aware of time and therefore cannot distinguish between new data and old ones. To overcome this difficulty, the *time-fading* model, a variation of the landmark model, has been presented in recent works [7][13][16]. It assigns different weights to transactions such that new ones have higher weights than old ones. As shown in Figure 2, the weights are decreasing as time passes by.

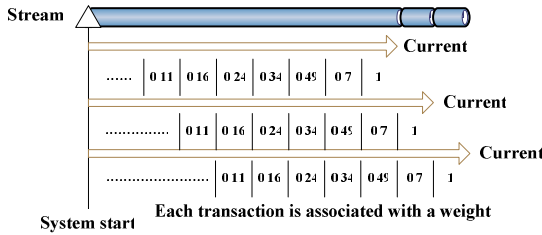


Figure 2: Time-fading model

The estDec method in [7] is proposed for mining frequent itemsets under this model. By using a decay rate, the effects of old transactions diminish as time goes by. For example, let the decay rate and the support count of itemset X be d and v , respectively. As a new transaction containing X arrives, the new support count of X is equal to $v \times d + 1$. Obviously, when d equals 1, the time-fading model becomes the landmark model. In [13], a variety of decay functions are also introduced to maintain aggregates under the time-fading models.

FP-stream approach in [16] provides a way to mine frequent itemsets under the time-fading model. Two parameters, the *minimum support count* σ and the *maximum support error* ϵ where $\sigma \geq \epsilon$, are used to classify all the itemsets into three categories:

- **Frequent:** Support count is greater than and equal

to σ .

- **Sub-frequent:** Support count falls in $[\epsilon, \sigma]$.
- **Infrequent:** Support count is smaller than ϵ .

Next, only frequent and sub-frequent itemsets are stored and organized as a *pattern tree*, a variation of the FP-tree [18]. Figure 3 shows a pattern tree, where a path starting at the root stands for an itemset. The count of each itemset is asymmetrically distributed into multiple time slots such that the recent time period is assigned more time slots than the past. The assignment of time slots is illustrated by the *tilted time window* shown in Figure 3. It is suitable for people to mine the recent data at a fine granularity while mining the long-term data at a coarse granularity.

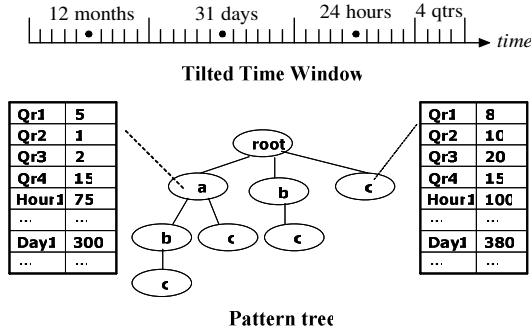


Figure 3: Pattern tree and Tilted-time window

All these approaches provide approximate answers for long-term data and adjust their storage requirement based on the available space. Therefore, they satisfy the two requirements – approximation and adjustability. However, the time-fading model (including the landmark model) has its essential limitation, i.e., the support count is computed from the entire data set between the landmark and the current time. In certain applications, users can only be interested in the data recently arriving within a fixed time period. Obviously, the models previously presented are unable to satisfy this need. On the contrary, the *sliding-window* model shown in Figure 4 achieves this goal. Given a window size W , only the latest W transactions are utilized for mining. As a transaction arrives, the oldest transaction in the sliding window is expired. Therefore, under this model, the methods for finding the expired transaction and for discounting the support counts of the itemsets involved are required.

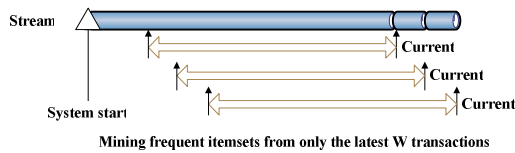


Figure 4: Sliding-window model

Babcock et al. [5] develop a mechanism, which can dynamically combine adjacent buckets in a histogram, to monitor the variance and k-medians in a sliding

window. In [9][14][20], hash-based methods are proposed to mine frequent items. In these methods, a fixed number of counters and hashing functions are used. An item is then assigned to the corresponding counters based on its hashed values. Each counter accumulates the support counts of the items with the same value hashed. In this way, the support count of an item can be estimated from the corresponding counters. Since these methods assume the expired transaction to be available, the update of these counters can be fast. However, the characteristic – expiration, said that it is not reasonable to have a chance to see the expired transaction again. In a recent work, Arasu and Manku [3] extend the Lossy-counting to the sliding-window model. Their approach can estimate the approximate counts and quantiles with certain accuracy guarantees.

Compared with the previous models considering only the insertion of transactions, the sliding-window model further considers the deletion of transactions. Therefore, if a method succeeds in the sliding-window model, it can be easily applied to the previous models. Moreover, all the previous works consider a fixed number of transactions as the basic unit for mining, which is not easy for people to specify. By contrast, it is natural for people to specify a time period as the basic unit. Therefore, in this paper, we propose the *time-sensitive sliding-window* model, which regards a fixed time period as the basic unit for mining.

Definition 1.1 Time-sensitive Sliding-window (TS)

Given a time point t and a time period p , the set of all the transactions arriving in $[t-p+1, t]$ will form a *basic block*. A data stream is decomposed into a sequence of basic blocks, which are assigned with serial numbers starting at 1. Given a window with length $|W|$, we slide it over this sequence to see a set of overlapping sequences, where each sequence is called the *time-sensitive sliding-window* abbreviated as *TS*.

Let the basic block numbered i be denoted as B_i . The number of transactions in B_i is denoted as $|B_i|$, which is not fixed due to the variable data arrival rate. For each B_i , the TS that consists of the $|W|$ consecutive basic blocks from $B_{i-|W|+1}$ to B_i is denoted as TS_i . Let the number of transactions in TS_i be denoted as Σ_i .

Definition 1.2 Frequent Itemsets in TS_i/B_i

The *support count* of an itemset in $TS_i (B_i)$ is the number of transactions in $[B_{i-|W|+1}, \dots, B_i]$ (B_i) containing it. Given the *support threshold* θ , an itemset is *frequent* in $TS_i (B_i)$ if its support count in $TS_i (B_i)$ is not smaller than $\theta \times \Sigma_i$ ($\theta \times |B_i|$).

Owing to the characteristics of data streams, it is not realistic to scan the past basic blocks again and again for mining frequent itemsets in each of the subsequent TS's. In this paper, we assume that only the summary information derived from TS_{i-1} is provided for mining frequent itemsets in TS_i . Such a scenario is illustrated

in Figure 5, where the basic unit is one day and $|W|$ is 3. As the new basic block $B_{6/21}$ comes, the oldest basic block $B_{6/18}$ in $TS_{6/20}$ is expired. To find frequent itemsets in $TS_{6/21}$, we consider three kinds of itemsets from two sources, the frequent itemsets in $TS_{6/20}$ and the frequent ones in $B_{6/21}$, as follows:

- For each frequent itemset in $TS_{6/20}$, the support count is discounted if it occurs in $B_{6/18}$ and then updated by examining $B_{6/21}$. A mechanism to keep its support count in $B_{6/18}$ and a way to find its support count in $B_{6/21}$ are needed.
- A frequent itemset in $B_{6/21}$, which is not frequent in $TS_{6/20}$, can be frequent in $TS_{6/21}$. The methods for computing its support count in $TS_{6/20}$ and for mining frequent itemsets in $B_{6/21}$ are required.
- An itemset that is not frequent in both $TS_{6/20}$ and $B_{6/21}$ cannot be frequent in $TS_{6/21}$.

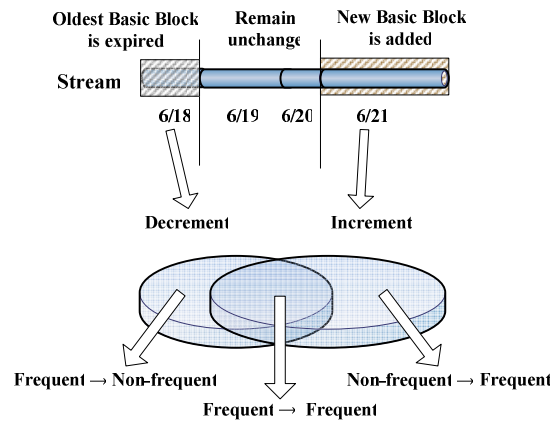


Figure 5: Time-sensitive sliding-window model

Since all the methods developed under other models accumulate the support count for each frequent itemset, no discounting information is provided. Furthermore, the hash-based approaches for mining frequent items under the sliding-window model assume that the expired transactions are available. However, for mining frequent itemsets, it is not reasonable to allow that the expired transactions can be reexamined. Therefore, in this paper, we introduce a novel approach to address the issues described above. First of all, we devise a data structure named the *discounting table* (DT) to retain the frequent itemsets with their support counts in the individual basic blocks of the current TS. Moreover, a data structure named the *Potentially Frequent-itemset Pool* (PFP) is used to keep the frequent itemsets in TS_i and the frequent ones in B_i . We include the itemsets that are frequent in B_i but not frequent in TS_{i-1} in PFP because they are possibly frequent in TS_i .

Definition 1.3 Potentially Frequent Itemset

A frequent itemset in B_i that is not frequent in TS_{i-1} is called a *potentially frequent itemset*. Since its support count in TS_{i-1} is not recorded, we estimate that as the

largest integer less than $\theta \times \sum_{i=1}^{|B_i|}$, i.e., the upper bound of its support count in $[B_{i-|W|}, \dots B_{i-1}]$. This is called the *potential count* and also recorded in PFP.

For the itemsets in PFP that are not potentially frequent, the potential counts are set to 0. In this way, each itemset in PFP is associated with the potential count and the *accumulated count*. Moreover, the sum of the two counts is regarded as the support count of this itemset in TS_i and used to determine whether it should be kept in PFP. When B_i arrives, three pieces of information are available for mining and discounting:

- DT: Frequent itemsets with support counts in each of the basic blocks $B_{i-|W|}, \dots B_{i-1}$.
- PFP: Frequent itemsets in TS_{i-1} or B_{i-1} .
- All the frequent itemsets discovered from B_i .

Mining frequent itemsets in TS_i consists of four steps. At first, the support counts of frequent itemsets in PFP are discounted according to DT and then the frequent itemsets in $B_{i-|W|}$ are removed from DT. Second, the frequent itemsets in B_i are mined by using FP-growth [18] and added into PFP with their potential counts computed. Third, for each itemset that is in PFP but not frequent in B_i , we scan B_i to accumulate its support count and then delete it from PFP if it is not frequent in TS_i . At last, two alternatives to determine the frequent itemsets for output are provided:

1. **Recall-oriented:** All the itemsets kept in PFP are output. Since all the frequent itemsets in TS_i are in PFP, it guarantees that no false dismissal occurs.
2. **Precision-oriented:** We output only those itemsets whose accumulated counts in PFP satisfy $\theta \times \sum_i$. Because for the potentially frequent itemsets, these counts are lower bounds of their support counts, it guarantees that no false alarm occurs.

In addition to the mining and discounting methods, we further design the *self-adjusting discounting table* (SDT) that can automatically adjust its size when maintaining the discounting information. Given a limitation on the size of SDT, we devise a strategy to merge the information of more than one itemset kept in SDT. The main idea is to minimize the difference between the original support count of each itemset and its approximate count after merging. The most important finding is that the two guarantees described above still hold when SDT is deployed. The following are the contributions of our paper, corresponding to the three requirements mentioned before.

- **Time-sensitive sliding-window model:** We propose a model that is sensitive to time. To our knowledge, this paper is the first one addressing the issues of mining frequent itemsets over data streams under this model.
- **Mining and discounting methods:** An approach that continuously provides frequent itemsets over data streams under our model is introduced. The

accuracy guarantees of no false dismissal or no false alarm are provided.

- **Self-adjusting discounting table:** A mechanism that is self-adjusting under the memory limitation is presented. The accuracy guarantees still hold.

The remainder of this paper is organized as follows. Section 2 details the mining and discounting methods, including the system framework and main operations. In Section 3, we present the self-adjusting discounting table. The experiment results are shown and discussed in Section 4. In Section 5, we conclude this paper.

2 Mining and Discounting

2.1 System framework

Figure 6 shows the system framework of our approach. The data stream is a series of transactions arriving continuously. Four parameters, the support threshold θ , the basic unit of time period for each basic block P , the length of TS $|W|$, and the output mode M , are given before the system starts. As Definition 1.1 states, a data stream is divided into blocks with different numbers of transactions according to P . The *buffer* continuously consumes transactions and pours them block-by-block into our system. After a basic block triggers these operations and goes through our system, it will be discarded directly.

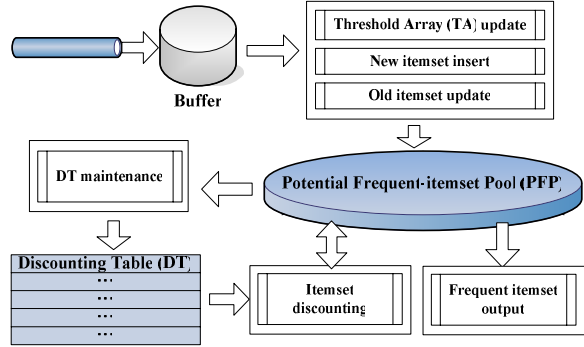


Figure 6: System framework

Because the basic blocks may have different numbers of transactions, we dynamically compute the *support count threshold* $\theta \times |B_i|$ for each basic block B_i and store it into an entry in the *threshold array* (TA), denoted as $TA[i]$. In our approach, only $|W|+1$ entries are maintained in TA. As B_i arrives, $TA[j]$ keeps the support count threshold of B_{i-j} for $1 \leq j \leq |W|+1$ and $i-j > 0$. After B_i is processed, the last entry $TA[|W|+1]$ is ignored and the others are moved to the next positions, i.e., $TA[j] \rightarrow TA[j+1]$ for $1 \leq j \leq |W|$. Finally, the support count threshold of B_i is put into $TA[1]$.

In addition to **TA_update**, the arrival of a basic block also triggers the other operations in Figure 6, which are differently executed in three cases. For each case, **Frequent_itemset_output** is used to pick up the

answers satisfying M from PFP. Figure 7 shows the main algorithm. First, as B_1 comes, two operations are executed one by one:

- **New_itemset_insertion:** An algorithm for mining frequent itemsets is applied to the transactions in the buffer. Each frequent itemset is inserted into PFP in the form of (ID, Items, Account, Pcount), recording a unique identifier, the items in it, the accumulated count, and the potential count, respectively. Since an itemset is added into PFP, Account accumulates its exact support counts in the subsequent basic blocks, while Pcount estimates the maximum possible sum of its support counts in the past basic blocks. For B_1 , Pcount is set as 0.
- **DT_maintenance:** Each itemset in PFP is inserted into DT in the form of (B_ID, ID, Bcount), recording the serial number of the current basic block, the identifier in PFP, and its support count in the current basic block, respectively. For B_1 , B_ID is set as 1.

Input: Stream S , Parameters θ , P , $|W|$, M

Output: All the frequent itemsets satisfying M

1. Let TA, PFP, and DT be empty // $\forall j, TA[j]=0$
2. While B_i comes // *from the buffer*
 - 2.1 If ($i = 1$) // B_1
 - 2.1.1 New_itemset_insertion
 - 2.1.2 DT_maintenance
 - 2.2 Else If ($i \leq |W|$) // $B_2 \dots B_{|W|}$
 - 2.2.1 New_itemset_insertion
 - 2.2.2 Old_itemset_update
 - 2.2.3 DT_maintenance
 - 2.3 Else // $B_{|W|+1} \dots$
 - 2.3.1 Itemset_discounting
 - 2.3.2 New_itemset_insertion
 - 2.3.3 Old_itemset_update
 - 2.3.4 DT_maintenance
- 2.4 TA_update // $\forall j, TA[j+1]=TA[j], TA[1]=\theta \times B_i$
- 2.5 Frequent_itemset_output

Figure 7: Main algorithm

When B_i arrives, where $1 < i \leq |W|$, three operations are executed one by one:

- **New_itemset_insertion:** In this case, we further check every frequent itemset discovered in B_i to see whether it has been kept by PFP. If it is, we increase its Account. Otherwise, we create a new entry in PFP and estimate its Pcount as the largest integer that is less than $\theta \times \sum_{i-1}$.
- **Old_itemset_update:** For each itemset that is in PFP but not frequent in B_i , we compute its support count in B_i by scanning the buffer to update its Account. After that, an itemset in PFP is deleted if its sum of Account and Pcount is less than $\theta \times \sum_i$.
- **DT_maintenance:** This operation is the same as described previously except that B_ID is set as i . At last, when B_i arrives, where $i > |W|$, the window slides and 4 operations are executed one by one. Before

that, an extra operation is executed:

- **Itemset_discounting:** Since the transactions in $B_{i-|W|}$ will be expired, the support counts of the itemsets kept by PFP are discounted accordingly. We classify the itemsets into two groups by Pcount. If it is nonzero, we repeatedly subtract the support count thresholds of the expired basic blocks from Pcount and finally set Pcount to 0. If Pcount is already 0, we subtract Bcount of the corresponding entry in DT from Account. Finally, each entry in DT where B_ID = $i - |W|$ is removed.

2.2 Main operations

Figure 8 shows the steps of **New_itemset_insertion**. First, we adopt the FP-growth algorithm to mine all the frequent itemsets from B_i . Let F_i denote this set. Next, we check each itemset in F_i to see whether it has been kept in PFP and then either update or create an entry.

Input: B_i

Output: F_i , updated PFP

1. Discover F_i from B_i
2. For each itemset f in F_i
 - 2.1 If ($f \in \text{PFP}$) Increase f .Account
 - 2.2 Else Insert f into PFP // *Estimate f .Pcount*

Figure 8: New_itemset_insertion

At Step 2.2, we need to estimate the Pcount for each itemset in F_i but not in PFP. The rationale of our estimation is as follows. Let f be such an itemset. Let S denote the sequence of basic blocks $[B_{i-|W|+1}, \dots B_{i-1}]$, which is a subsequence of TS_{i-1} , i.e., $[B_{i-|W|}, \dots B_{i-1}]$. According to **Old_itemset_update**, f is not kept by PFP only if it is non-frequent in TS_{i-1} . Therefore, the support count of f in TS_{i-1} cannot be more than $\theta \times \sum_{i-1}$. As a result, we estimate Pcount, the maximum possible count of f in S , as follows:

$$Pcount \text{ at } TS_i = \lceil \theta \times \sum_{i-1} \rceil - 1 = \left\lceil \sum_{j=1}^{|W|} TA[j] \right\rceil - 1 \quad (1)$$

Figure 9 shows the steps of **Old_itemset_update**. For each itemset g that has been kept by PFP but not in F_i , we compute its support count in B_i to increase its Account. Suppose that g was inserted into PFP when B_k comes ($k < i$). At this point, we have g .Account, the exact support count of g in $[B_k, \dots B_i]$, and g .Pcount, the maximum possible support count of g in $[B_{i-|W|+1}, \dots B_{k-1}]$. If the sum is less than the support count threshold, g must not be frequent in TS_i and can be safely deleted from PFP.

Input: F_i , B_i , PFP

Output: updated PFP

1. For each itemset g in PFP but not in F_i
 - 1.1 Increase g .Account by scanning B_i once
 - 1.2 If (g .Account + g .Pcount < $\theta \times \sum_i$)
Delete g from PFP

Figure 9: Old_itemset_update

DT_maintenance is shown in Figure 10. Each itemset in PFP is added to DT together with its support count in B_i . In this section, we assume that there is unlimited memory space utilized for DT_maintenance. The DT_maintenance under a limited memory space will be presented in the next section.

Input: PFP, DT

Output: updated DT

1. For each itemset f in PFP
Append f to DT

Figure 10: DT_maintenance

We design the steps of **Itemset_discounting** in Figure 11. At first, we classify all the itemsets in PFP into two groups by Pcount. Each itemset uses Pcount to keep its maximum possible count in the past basic blocks before it is inserted into PFP. By Formula (1), since B_i comes, Pcount is computed by including the support count threshold of an extra basic block, i.e., $B_{i-|W|}$. As B_{i+1} comes, if Pcount is nonzero, we subtract the support count threshold of $B_{i-|W|}$ from Pcount. If Pcount is smaller than the support count threshold of $B_{i-|W|+1}$, Account should have the exact support counts from $B_{i-|W|+2}$ to B_{i+1} . In this case, we set Pcount to 0. When Pcount is zero, we directly decrease its Account by its Bcount of the corresponding entry in DT.

Input: PFP, DT, i

Output: updated PFP, updated DT

1. For each itemset g in PFP
 - 1.1 If ($g.Pcount = 0$)
 - 1.1.1 Find entry h in DT where ($g.ID = h.ID$) and ($h.B_ID = i - |W|$)
 - 1.1.2 $g.Account = g.Account - h.Bcount$
 - 1.2 Else
 - 1.2.1 $g.Pcount = g.Pcount - TA[|W|+1]$
 - 1.2.2 If ($g.Pcount < TA[|W|]$)
 $g.Pcount = 0$
2. For each entry h in DT
If ($h.B_ID = i - |W|$) Remove h from DT

Figure 11: Itemset_discounting

When the first $|W|$ basic blocks come, there is no extra basic block to overestimate the value of Pcount. Therefore, Pcount is not decreased at the first time the window slides, i.e., as $B_{|W|+1}$ arrives. In this case, Step 1.2.1 has no effect since $TA[|W|+1]$ is 0. After the discounting, we can safely remove all the entries in DT belonging to $B_{i-|W|}$.

Example 2.1

Take Figure 12 as an example. Let W be 3. Assume that an itemset g is inserted into PFP in 6/18. By Formula (1), $g.Pcount$ is computed from the support count thresholds in 6/15-6/17. When the TS moves to 6/19, $g.Pcount$ is decreased and only considers 6/16-6/17. As the TS moves to 6/20, since $g.Account$ accumulates the support counts in 6/18-6/20, $g.Pcount$ is set 0. As the

TS moves to 6/21, $g.Account$ is discounted by dropping its support count in 6/18.

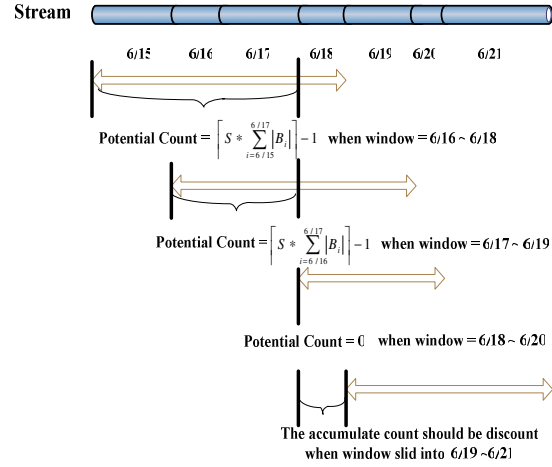


Figure 12: An example of Itemset discounting

As mentioned before, PFP keeps not only the frequent itemsets in TS_i but also the ones in B_i . In Formula (1), Pcount in PFP is overestimated. Therefore, if all the itemsets in PFP are outputted, it guarantees that no true answer is missed. It is called the *no-false-dismissal* mode (denoted as NFD). In this mode, an itemset that is frequent in B_i but not in TS_i is still outputted. Sometimes user hopes that all the itemsets outputted are real answers. Therefore, we also provide the *no-false-alarm* mode (denoted as NFA), which outputs only the itemsets with Account satisfying the support count threshold. Since Account accumulates the support counts of an itemset in the individual basic blocks after that itemset is inserted into PFP, this mode guarantees that no false answer is outputted. The steps of **Frequent_itemset_output** are shown in Figure 13.

Input: PFP

Output: The set of frequent itemsets O

1. If ($M = NFD$)
For each itemset f in PFP
 $O = O + \{f\}$
2. Else // $M = NFA$
For each itemset f in PFP
If ($f.Account \geq \theta \times \sum_i$) $O = O + \{f\}$

Figure 13: Frequent_itemset_output

Example 2.2

Let θ , $|W|$, and P be 0.4, 3, and 1 hour, respectively. Consider the stream of transactions shown in Table 1.

Table 1: A stream of transactions

	Time period	Number of transactions	Itemset (and its count)
B_1	09:00~09:59	27	a(11),b(20),c(2),ab(6)
B_2	10:00~10:59	20	a(20),c(13),ac(13)
B_3	11:00~11:59	27	a(19),b(8),c(7),ac(7)
B_4	12:00~12:59	23	a(10),c(3),d(10)

Initially, PFP and DT are empty and $TA[j] = 0$ for all j . For the 1st hour (i.e., B_1), the support count threshold is 0.4×27 (10.8). By **New_itemset_insertion**, only a and b are frequent and inserted into PFP with Pcount 0. In **DT_maintenance**, both of them are put in DT. TA is updated and the results are shown in Figure 14, where a and b are outputted for both modes.

TA	<table><tr><td>10.8</td><td>0</td><td>0</td><td>0</td></tr></table>				10.8	0	0	0								
10.8	0	0	0													
DT	<table><tr><td>B_ID</td><td>ID</td><td colspan="2">Bcount</td></tr><tr><td>1</td><td>1</td><td colspan="2">11</td></tr><tr><td>1</td><td>2</td><td colspan="2">20</td></tr></table>				B_ID	ID	Bcount		1	1	11		1	2	20	
	B_ID	ID	Bcount													
	1	1	11													
1	2	20														
PFP	(1,a,11,0) (2,b,20,0)															

Figure 14: A snapshot after B_1 passes

For the 2nd hour (B_2), the support count threshold is 0.4×20 (8). The itemsets c and ac are inserted into PFP with Pcount 10, which is the maximum possible count of a non-frequent itemset in B_1 . In addition, we accumulate the support counts of a in 2 hours to get its Account (=31). In **Old_itemset_update**, B_2 is scanned once to compute the support count of b since b is in PFP but not frequent in B_2 . Because the support count threshold is the sum of 10.8 and 8, i.e., 18.8, b is still kept in PFP. Finally, all the 4 itemsets are appended to DT and then TA is updated. The results are shown in Figure 15. Under the NFD mode, all the itemsets in PFP are outputted, while only a and b are outputted under the NFA mode.

TA	<table><tr><td>8</td><td>10.8</td><td>0</td><td>0</td></tr></table>				8	10.8	0	0																								
8	10.8	0	0																													
DT	<table><tr><td>B_ID</td><td>ID</td><td colspan="2">Bcount</td></tr><tr><td>1</td><td>1</td><td colspan="2">11</td></tr><tr><td>1</td><td>2</td><td colspan="2">20</td></tr><tr><td>2</td><td>1</td><td colspan="2">20</td></tr><tr><td>2</td><td>2</td><td colspan="2">0</td></tr><tr><td>2</td><td>3</td><td colspan="2">13</td></tr><tr><td>2</td><td>4</td><td colspan="2">13</td></tr></table>				B_ID	ID	Bcount		1	1	11		1	2	20		2	1	20		2	2	0		2	3	13		2	4	13	
	B_ID	ID	Bcount																													
	1	1	11																													
	1	2	20																													
	2	1	20																													
	2	2	0																													
	2	3	13																													
	2	4	13																													
PFP	(1,a,31,0) (2,b,20,0) (3,c,13,10) (4,ac,13,10)																															

Figure 15: A snapshot after B_2 passes

For the 3rd hour (B_3), the support count threshold is also 10.8. Since the frequent itemset a in B_3 also exists in PFP, we accumulate its support counts in 3 hours to get its Account 50. In **Old_itemset_update**, B_3 is scanned thrice to compute the support counts of b , c , and ac , respectively. For the support count threshold 29.6, we keep c and ac in PFP but delete b since its Account plus its Pcount is 28. All the 3 itemsets are appended to DT and TA is updated. Figure 16 shows the results and only a is outputted for the NFA mode.

For the 4th hour (B_4), **Itemset_discounting** is executed. First, we check and discount the itemsets in PFP. Since a .Pcount is 0, we decrease a .Account as 39 according to DT. For c and ac , we first subtract $TA[4]$ (=0) from their potential counts and then set them to 0 because they are smaller than the support count

threshold of B_1 . Secondly, all the entries of B_1 in DT are removed. After that, we repeat the remaining operations as described above. The support count threshold of B_4 is 9.2. Therefore, d is inserted into PFP with Pcount 29, while a .Account is increased as 49. During **Old_itemset_update**, for c and ac , their sums of Account and Pcount are 23 and 20, respectively. Both of them are deleted because the current support count threshold is 28 ($8+10.8+9.2$). The final results are shown in Figure 17.

TA	10.8	8	10.8	0
DT	B_ID	ID	Bcount	
	1	1	11	
	1	2	20	
	2	1	20	
	2	2	0	
	2	3	13	
	2	4	13	
	3	1	19	
	3	3	7	
	3	4	7	
PFP	(1,a,50,0) (3,c,20,10) (4,ac,20,10)			

Figure 16: A snapshot after B_3 passes

TA	<table><tr><td>9.2</td><td>10.8</td><td>8</td><td>10.8</td></tr></table>				9.2	10.8	8	10.8																																				
9.2	10.8	8	10.8																																									
DT	<table><tr><td>B_ID</td><td>ID</td><td colspan="2">Bcount</td></tr><tr><td>2</td><td>1</td><td colspan="2">20</td></tr><tr><td>2</td><td>2</td><td colspan="2">0</td></tr><tr><td>2</td><td>3</td><td colspan="2">13</td></tr><tr><td>2</td><td>4</td><td colspan="2">13</td></tr><tr><td>3</td><td>1</td><td colspan="2">19</td></tr><tr><td>3</td><td>3</td><td colspan="2">7</td></tr><tr><td>3</td><td>4</td><td colspan="2">7</td></tr><tr><td>4</td><td>1</td><td colspan="2">10</td></tr><tr><td>4</td><td>2</td><td colspan="2">10</td></tr></table>				B_ID	ID	Bcount		2	1	20		2	2	0		2	3	13		2	4	13		3	1	19		3	3	7		3	4	7		4	1	10		4	2	10	
	B_ID	ID	Bcount																																									
	2	1	20																																									
	2	2	0																																									
	2	3	13																																									
	2	4	13																																									
	3	1	19																																									
	3	3	7																																									
	3	4	7																																									
	4	1	10																																									
4	2	10																																										
PFP	(1,a,49,0) (2,d,10,29)																																											

Figure 17: A snapshot after B_4 passes

3. Self-adjusting Discounting Table

In this section, we refine **DT_maintenance** to address the issue of the limited memory space. Among the data structures maintained for mining and discounting in our approach, DT often consumes most of the memory space. When the limit is reached, an efficient way to reduce the DT size without losing too much accuracy is required. A straightforward way is to merge the entries in DT as needed. The main challenge is how to quickly select the entries for merging such that the resultant DT still performs well in discounting. In the following, a naive solution called the *naïve adjustment* is introduced first and then our proposed method named the *selective adjustment* is presented.

3.1 Naïve adjustment

Since each entry is appended to the end of DT when it shows up, we can regard DT as a list of triples (B_ID,

ID, Bcount) sorted by B_ID and ID, e.g., the figures in Example 2.2. Let the k^{th} entry in DT be DT_k . Every two adjacent entries satisfy one of the following properties:

1. $DT_k.B_ID < DT_{k+1}.B_ID$
2. $DT_k.B_ID = DT_{k+1}.B_ID$ and $DT_k.ID < DT_{k+1}.ID$

When the size of DT reaches its limit, the naïve adjustment finds the two adjacent entries satisfying the 2nd property and then merges them into one. Figure 18 illustrates the DT_maintenance with naïve adjustment, where DT_size and DT_limit respectively denote the number of entries in DT and its upper bound due to the limited memory space.

Input: PFP, DT, DT_size , DT_limit // $DT_limit > |W|$

Output: updated DT

1. For each itemset f in PFP
 - 1.1 If ($DT_size = DT_limit$) //DT is full
 - 1.1.1 $k = 2$ //naïve adjustment
 - 1.1.2 While ($DT_k.B_ID \neq DT_{k-1}.B_ID$) $k++$
 - 1.1.3 $DT_{k-1}.ID = DT_{k-1}.ID \cup DT_k.ID$
 - 1.1.4 If ($M=NFD$)
 - $DT_{k-1}.Bcount = \min\{DT_{k-1}.Bcount, DT_k.Bcount\}$
 - 1.1.5 Else //M=NFA
 - $DT_{k-1}.Bcount = \max\{DT_{k-1}.Bcount, DT_k.Bcount\}$
 - 1.1.6 Remove DT_k from DT; DT_size--
 - 1.2 Append f to DT; DT_size++

Figure 18: DT_maintenance with naïve adjustment

At the beginning of naïve adjustment, we scan DT from top to bottom and merge the first two entries having the same B_ID. Note that after Step 1.1.2, the entries DT_{k-1} and DT_k always exist so long as DT_limit is larger than $|W|$. In this way, only the adjacent entries having the same B_ID are merged. The memory space freed is immediately used for the new entry appended to DT. Except for the unchanged B_ID, we also assign ID and Bcount to the new entry after merging. Since we sort the entries having the same B_ID by ID, the ID's from the adjacent entries can be represented as a *range* of ID's, i.e., "smallest ID–largest ID". Therefore, as Step 1.1.3 indicates, we use the range covered by the ID's from DT_{k-1} and DT_k as the ID of the new DT_{k-1} .

On the other hand, the assignment of Bcount is different and depends on the output mode M given by the user. For the NFD mode, we underestimate Bcount such that the support count of an itemset is discounted as less as possible and thus overestimated. In this case, we choose the smaller Bcount between DT_{k-1} and DT_k as the Bcount of the new DT_{k-1} as Step 1.1.4 indicates. By contrast, for the NFA mode, we underestimate the support count of an itemset by discounting it as more as possible. Therefore, in Step 1.1.5, the larger Bcount between DT_{k-1} and DT_k is chosen.

Example 3.1

Suppose that the 6 itemsets in Table 2 will be inserted into DT but DT_limit is set to 4. In addition, the output mode is NFD. Initially, DT is empty. The 4 itemsets A,

B, C, and F are added one by one to form Table 3(a). Since DT is full now, the naïve adjustment is executed before the addition of AF. Specifically, the entries (1, 1, 12) and (1, 3, 13) are selected and merged into (1, 1–3, 12) as Table 3(b) shows. After that, AF is added to DT. For the addition of G, the naïve adjustment is executed again and the first 2 entries are merged into (1, 1–4, 2) as Table 3(c) indicates.

Table 2: The itemsets to be inserted

B_ID	ID	Itemset	Bcount
1	1	A	12
1	3	B	13
1	4	C	2
1	5	F	10
1	6	AF	10
1	8	G	8

Table 3: The process of the naïve adjustment

B_ID	ID	Bcount
1	1	12
1	3	13
1	4	2
1	5	10

(a)

B_ID	ID	Bcount
1	1-3	12
1	4	2
1	5	10
1	6	10

(b)

B_ID	ID	Bcount
1	1-4	2
1	5	10
1	6	10
1	8	8

(c)

The naïve adjustment is fast but provides inaccurate information for discounting when too many entries are merged together. Take (1, 1–4, 2) in Table 3(c) as an example. Due to the NFD mode, the value 2 stands for the minimum Bcount among the merged entries. When discounting itemsets A, B, and C, the errors are 10, 11, and 0, respectively. We call the sum of these errors the *merging loss*. For an entry in DT, the smaller merging loss it has, the more accurate Bcount it will provide for discounting. Next, we will introduce our method that uses the merging loss to select the entries for merging.

3.2 Selective Adjustment

In our method, each entry DT_k is in the new form of (B_ID, ID, Bcount, AVG, NUM, Loss). $DT_k.AVG$ keeps the average of support counts for all the itemsets merged into DT_k , $DT_k.NUM$ is the number of itemsets in DT_k , while $DT_k.Loss$ records the merging loss of merging DT_k with DT_{k-1} . The main idea of our method is to select the entry with the smallest merging loss,

called the *victim*, and merge it into the entry above it. $DT_1.Loss$ and $DT_k.Loss$ are set ∞ to avoid being the victim, $\forall k, DT_k.B_ID \neq DT_{k-1}.B_ID$. Since the merging loss of an entry depends on the output mode M , we formulate it as follows:

Definition 3.1 Merging loss

For $k > 1$ and $DT_k.B_ID = DT_{k-1}.B_ID$, $DT_k.Loss$ under the NFD mode is computed as follows:

$$(DT_k.NUM \times DT_k.AVG + DT_{k-1}.NUM \times DT_{k-1}.AVG) - \min\{DT_k.Bcount, DT_{k-1}.Bcount\} \times (DT_k.NUM + DT_{k-1}.NUM) \quad (2)$$

$DT_k.Loss$ under the NFA mode is computed as follows:

$$\max\{DT_k.Bcount, DT_{k-1}.Bcount\} \times (DT_k.NUM + DT_{k-1}.NUM) - (DT_k.NUM \times DT_k.AVG + DT_{k-1}.NUM \times DT_{k-1}.AVG) \quad (3)$$

According to the same reason for the assignment of Bcount in the naïve adjustment, we use the minimum (maximum) Bcount in the computation of merging loss under the NFD (NFA) mode. As a result, the smaller merging loss DT_k has, after merging, the more accurate Bcount DT_{k-1} will provide. Figure 19 illustrates the $DT_maintenance$ with selective adjustment.

Input: PFP, DT, DT_size , DT_limit

Output: updated DT

1. For each itemset f in PFP
 - 1.1 If ($DT_size = DT_limit$)
 - 1.1.1 Scan DT once to select the *victim* // DT_k
 - 1.1.2 $DT_{k-1}.ID = DT_{k-1}.ID \cup DT_k.ID$
 - 1.1.3 If ($M=NFD$)

$$DT_{k-1}.Bcount = \min\{DT_{k-1}.Bcount, DT_k.Bcount\}$$
 - 1.1.4 Else // $M=NFA$

$$DT_{k-1}.Bcount = \max\{DT_{k-1}.Bcount, DT_k.Bcount\}$$
 - 1.1.5 $DT_{k-1}.NUM = DT_{k-1}.NUM + DT_k.NUM$
 - 1.1.6 Compute $DT_{k-1}.AVG$
 - 1.1.7 If ($DT_{k-1}.Loss \neq \infty$) Recalculate $DT_{k-1}.Loss$
 - 1.1.8 Remove DT_k from DT; DT_size--
 - 1.1.9 If ($DT_{k+1}.Loss \neq \infty$) Recalculate $DT_{k+1}.Loss$
 - 1.2 Append f to DT; DT_size++

Figure 19: $DT_maintenance$ with selective adjustment

At the beginning of selective adjustment, we scan DT once to find the victim. Suppose that DT_k is the victim and will be merged into DT_{k-1} . For the new DT_{k-1} , the assignment of ID and Bcount follows the same way in the naïve adjustment. Moreover, NUM is assigned with the sum of $DT_{k-1}.NUM$ and $DT_k.NUM$, while AVG is computed as follows:

$$\frac{DT_k.AVG \times DT_k.NUM + DT_{k-1}.AVG \times DT_{k-1}.NUM}{DT_k.NUM + DT_{k-1}.NUM} \quad (4)$$

Based on the new $DT_{k-1}.Bcount$, $DT_{k-1}.AVG$, and $DT_{k-1}.NUM$, $DT_{k-1}.Loss$ can be computed by Definition 3.1. Note that if the old $DT_{k-1}.Loss$ has been set ∞ , it is unchanged. After the merging, the merging loss of the entry below the victim, i.e., $DT_{k+1}.Loss$, is also updated as Step 1.1.9 indicates.

Example 3.2

Consider Table 2, $DT_limit=4$, and $M=NFD$. Table 4(a) shows the DT as itemset A is added. Since it is the first entry, its merging loss is set ∞ . As itemset B is added, we compute its merging loss by Formula (2) and the result is shown in Table 4(b). In the same way, we add itemsets C and F to form the DT in Table 4(c). Since DT is full now, the selective adjustment is executed before the addition of AF . Specifically, the entry (1, 3, 13, 13, 1, 1) is selected as the victim and merged with (1, 1, 12, 12, 1, ∞). The result after merging forms the first entry in Table 4(d), where $DT_1.Loss$ is ∞ and $DT_1.AVG (=12.5)$ is computed by Formula (4). Notice that $DT_2.Loss$ is changed from 11 to 21 as Step 1.1.9 indicates. Finally, we add itemset G in a similar way to obtain the final result in Table 4(e).

Table 4: The process of the selective adjustment

B_ID	ID	Bcount	AVG	NUM	Loss
1	1	12	12	1	∞

(a)

B_ID	ID	Bcount	AVG	NUM	Loss
1	1	12	12	1	∞
1	3	13	13	1	1

(b)

B_ID	ID	Bcount	AVG	NUM	Loss
1	1	12	12	1	∞
1	3	13	13	1	1
1	4	2	2	1	11
1	5	10	10	1	8

(c)

B_ID	ID	Bcount	AVG	NUM	Loss
1	1-3	12	12.5	2	∞
1	4	2	2	1	21
1	5	10	10	1	8
1	6	10	10	1	0

(d)

B_ID	ID	Bcount	AVG	NUM	Loss
1	1-3	12	12.5	2	∞
1	4	2	2	1	21
1	5-6	10	10	2	16
1	8	8	8	1	4

(e)

Consider the final results in Example 3.1 and 3.2. In the former, the sum of all the merging losses is $(12-2) + (13-2) + (2-2) + (10-10) + (10-10) + (8-8) = 21$. In the latter, the sum of all the merging losses is $(12-12) + (13-12) + (2-2) + (10-10) + (10-10) + (8-8) = 1$. In this case, obviously, our method provides a higher accuracy of discounting information than a naïve solution.

4. Experiments

In this section, we will describe the experimental evaluation of our algorithms. The experimental setting

is first described and then the results are presented.

4.1 Experimental Setting

The experiments are made upon the PC with the Intel Pentium-M 1.3GHz CPU, 256 MB main memory and Microsoft Windows XP Professional. The programs are written in C and compiled using Microsoft Visual C++ 6.0. The mining algorithm we applied to find frequent itemsets in a basic block is the FP-growth. The datasets streaming into our system are synthesized via the IBM data generator [2], where we adopt most of its default values for the command options. For clarity, we name each dataset in the form of $TxxIxxDxx$, where T , I , and D mean the average transaction length, the average length of the maximum pattern, and the total number of transactions, respectively. To simulate data streams, we divide a dataset into basic blocks with equal size (i.e., 10K transactions) and feed them into the buffer. The parameter setting used in the experiments (unless explicitly specified otherwise) is shown in Table 5.

Table 5: Parameter setting

Parameter	Value
Number of distinct items	1K
DT_limit	10K
θ	0.0025
$ W $	4
T	3~7
I	4
D	150K

Two kinds of experiments have been made. First, the required execution time and memory usage are two indicators of the efficiency for mining data streams. We compare the execution times of our approach in the PFP maintenance part, the DT maintenance part, and the mining part (FP-growth). The memory usage referring to the memory space consumed for both PFP and DT is reported. In addition, the scalability of our approach for the minimum support threshold is also evaluated. On the other hand, under the NFD (NFA) mode, the number of false alarms (false dismissals) is also a good indicator of the effectiveness for mining data streams. We define measures to estimate them and to compare the two strategies for maintaining the SDT.

4.2 Efficiency on Time and Space

First, we evaluate the execution time of our approach adopting the selective adjustment for $T=3\sim 7$. Since the results show similar trends, only Figure 20 ($T=7$) is shown. The 4 curves in it refer to the execution time in 3 parts and the total, respectively. Our observations are as follows:

- The mining part dominates the total execution time. That is, **New_itemset_insertion** is the bottleneck of our algorithm, while the other operations are fast.
- The execution time of DT maintenance part is close

to zero in most cases. It verifies the feasibility of the SDT for mining data streams.

- Although the execution time of the mining part is sensitive to T , the total execution time increases slowly as the growth of T . It indicates the fast response time our approach achieved.

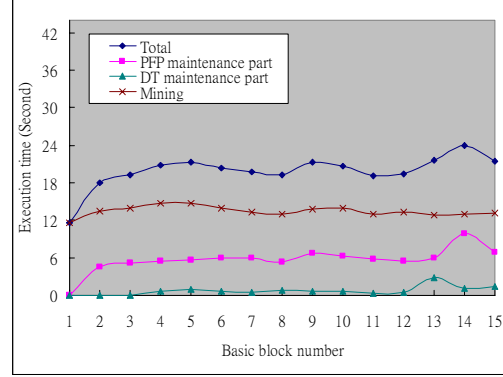


Figure 20: Execution time for $T = 7$

Figure 21 shows the memory usage under different values of T . The highest peak of memory usage during the experiment is not more than 350KB. It verifies the feasibility of the SDT, especially for the streaming environment with only a small memory. In all the figures, the curves are near smooth when the data stream flows. It implies that our approach adapts itself very well no matter how long a data stream is.

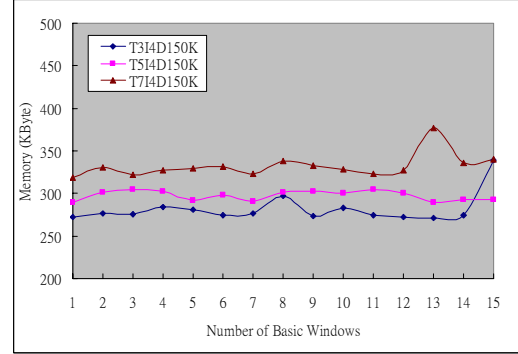


Figure 21: Memory usage for $T = 3, 5, 7$

To evaluate the scalability of our approach, we try $\theta=0.01\sim 0.002$ on the data set T7I4D150K. Intuitively, a smaller θ implies more frequent itemsets, indicating a higher execution time. As expected, in Figure 22, the average execution time grows slowly as the decreases of θ , but the curve has a sharp rise when θ is changed from 0.003 to 0.002. We found that the number of frequent itemsets for $\theta=0.002$ is six times as many as the case for $\theta=0.003$. Therefore, our approach can be stable as long as the number of frequent itemsets does not increase very dramatically.

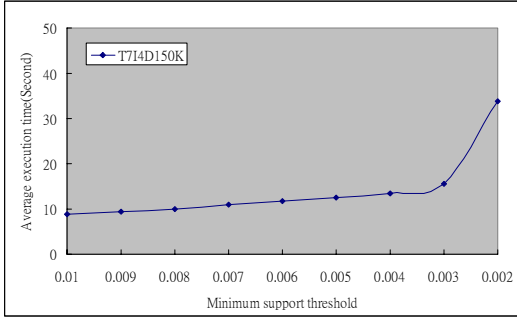


Figure 22: Average execution time for $\theta = 0.01 \sim 0.002$

4.3 Effectiveness on NFD and NFA

To evaluate the effectiveness of the proposed strategies for maintaining the SDT, we make an experiment on the data set T714D150K, where the DT_limit is varied from 4 to 12K. The experiment is performed for the NFD and NFA modes, respectively.

Under the NFD mode, we overestimate the support count of each itemset in PFP. Therefore, it guarantees no false dismissal but allows false alarms. Intuitively, the smaller the DT_limit is, the more adjustments and false alarms will occur. In our setting, the case when DT_limit=4 is the *worse case*, while the case when DT_limit=12K is the *best case*. We define a measure to evaluate the effectiveness under the NFD mode:

Definition 4.1 False Alarm Rate

The *false alarm rate* when DT_limit=M (denoted as FAR_M) can be computed as follows:

$$FAR_M = \frac{\text{The number of false alarms when DT_limit} = M}{\text{The number of false alarms in the worst case}} \quad (5)$$

The results are shown in Figure 23, where the false alarm rate shrinks as the growth of DT_limit. Consider the selective adjustment. 40% reduction of FAR is achieved when DT_limit=2K. As DT_limit=6K, more than 80% reduction of FAR can be achieved. From the two curves, it is obvious that the selective adjustment outperforms the naïve adjustment.

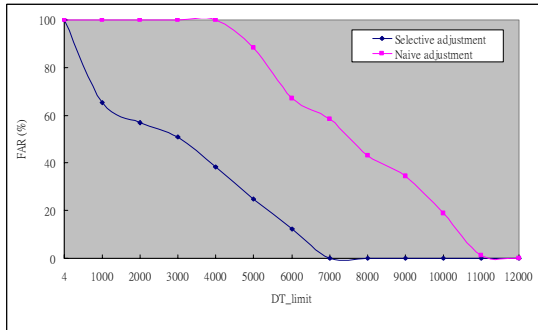


Figure 23: False alarm rate under the NFD mode

On the contrary, we underestimate the support count of each itemset in PFP under the NFA mode. Therefore, it guarantees no false alarm but allows false dismissals. A smaller DT_limit implies more false dismissals. The

worse case and the best case are the same as described above. Similarly, we define a measure to evaluate the effectiveness under the NFA mode:

Definition 4.2 False Dismissal Rate

The *false dismissal rate* when DT_limit=M (denoted as FDR_M) can be computed as follows:

$$FDR_M = \frac{\text{The number of false dismissals when DT_limit} = M}{\text{The number of false dismissals in the worst case}} \quad (6)$$

The results shown in Figure 24 also indicate that the false dismissal rate shrinks as the growth of DT_limit. The selective adjustment again outperforms the naïve adjustment. For example, the selective adjustment can achieve 60% reduction of FDR as DT_limit=4K, while the naïve adjustment cannot make it until DT_limit=9K. Our approach is both efficient and effective since it works well in the environment with a small memory and achieves fast response time without producing too many false alarms or false dismissals.

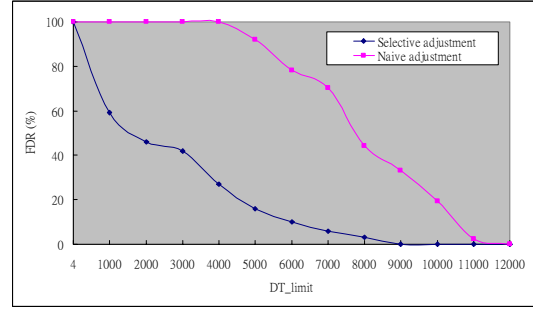


Figure 24: False dismissal rate under the NFA mode

5. Conclusion

Mining data streams is an interesting and challenging research field. The characteristics of data streams make the traditional mining algorithm unable to be applied. In this paper, we introduce an efficient algorithm for mining frequent itemsets over data streams under the time-sensitive sliding-window model. We design the data structures for mining and discounting the support counts of the frequent itemsets when the window slides. Moreover, two strategies for keeping SDT for the limited memory are proposed. Finally, the experiment results demonstrate that the execution time and required memory of our approach are acceptable under various parameter settings. On the other hand, the SDT performs well when the available memory is limited.

From this study, more topics are uncovered. First, we provide either of the two accuracy guarantees for the frequent itemsets found in this paper. However, the errors in the support counts are not precisely estimated or bounded. The error estimation can help the ranking of frequent itemsets if only the top-k frequent itemsets are needed. Second, any other type of frequent patterns such as the sequential pattern can be the next target. Finally, the constraints recently discussed in the data mining field can also be included into this study.

Acknowledgement

This work is partially supported by the National Science Council of the Republic of China under Grant No. 93-2752-E-007-004-PAE.

Reference

- [1] D.J. Abadi, D. Carney, U. Cetintemel, et al., "Aurora: A New Model and Architecture for Data Stream Management," *The VLDB Journal*, 12(2): 120-139, 2003.
- [2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. of VLDB Conf.*, pp. 487-499, 1994.
- [3] A. Arasu and G.S. Manku, "Approximate Counts and Quantiles over Sliding Windows," *Proc. of ACM PODS Symp.*, 2004.
- [4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," *Proc. of ACM PODS Symp.*, 2002.
- [5] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining Variance and k-Medians over Data Stream Windows," *Proc. of ACM PODS Symp.*, 2003.
- [6] S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," *The First Conf. on Innovative Data Systems Research (CIDR)*, 2003.
- [7] J.H. Chang and W.S. Lee, "Finding Recent Frequent Itemsets Adaptively over Online Data Streams," *Proc. of ACM SIGKDD Conf.*, pp. 487-492, 2003.
- [8] C.H. Chang and S.H. Yang, "Enhancing SWF for Incremental Association Mining by Itemset Maintenance," *Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2003.
- [9] M. Charikar, K. Chen, and M. Farach-Colton, "Finding Frequent Items in Data Streams," *Proc. of Inter'l Colloquium on Automata, Languages and Programming (ICALP)*, pp. 693-703, 2002.
- [10] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better Streaming Algorithms for Clustering Problems," *Proc. of ACM Symp. on Theory of Computing (STOC)*, pp. 30-39, 2003.
- [11] J. Chen, D.J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," *Proc. of ACM SIGMOD Conf.*, pp. 379-390, 2000.
- [12] D. Cheung, J. Han, V. Ng, and C.Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," *Proc. of ICDE Conf.*, 1996.
- [13] E. Cohen and M. Strauss, "Maintaining Time Decaying Stream Aggregates," *Proc. of ACM PODS Symp.*, 2003.
- [14] G. Cormode and S. Muthukrishnan, "What's Hot and What's Not: Tracking Most Frequent Items Dynamically," *Proc. of ACM PODS Symp.*, pp. 296-306, 2003.
- [15] E. Demaine, A. Lopez-Ortiz, and J.I. Munro, "Frequency Estimation of Internet Packet Streams with Limited Space," *Proc. of European Symp. on Algorithms (ESA)*, pp. 348-360, 2002.
- [16] C. Giannella, J. Han, J. Pei, X. Yan, and P.S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities," H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), *Next Generation Data Mining*, pp. 191-212, 2003.
- [17] L. Golab and M. Ozsü, "Issues in Data Stream Management," *ACM SIGMOD Record*, 32(2): 5-14, 2003.
- [18] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," *Data Mining and Knowledge Discovery*, 8(1): 53-87, Kluwer Academic Publishers, 2004.
- [19] G. Hulten, L. Spencer, and P. Domingos, "Mining Time Changing Data Streams," *Proc. of ACM CIKM Conf.*, pp. 97-106, 2001.
- [20] C. Jin, W. Qian, C. Sha, J.X. Yu, and A. Zhou, "Dynamically Maintaining Frequent Items over a Data Stream," *Proc. of ACM CIKM Conf.*, 2003.
- [21] R.M. Karp, C.H. Papadimitriou, and S. Shenker, "A Simple Algorithm for Finding Frequent Elements in Streams and Bags," *ACM Trans. on Database Systems (TODS)*, 28(1): 51-55, 2003.
- [22] C. Lee, C. Lin, and M. Chen, "Sliding-window Filtering: An Efficient Algorithm for Incremental Mining," *Proc. of ACM CIKM Conf.*, 2001.
- [23] J. Liu, Y. Pan, K. Wang, and J. Han, "Mining Frequent Item Sets by Opportunistic Projection," *Proc. of ACM SIGKDD Conf.*, 2002.
- [24] G. Manku and R. Motwani, "Approximate Frequency Counts over Data Streams," *Proc. of VLDB Conf.*, pp. 346-357, 2002.
- [25] R. Motwani, J. Widom, A. Arasu, and B. Babcock, "Query Processing, Approximation, and Resource Management in a Data Stream Management System," *CIDR Conf.*, 2003.
- [26] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-Data Algorithms for High-Quality Clustering," *Proc. of ICDE*, pp. 685-696, 2002.
- [27] W. Teng, M.S. Chen and P. Yu, "A Regression-Based Temporal Pattern Mining Scheme for Data Streams," *Proc. of VLDB Conf.*, pp.93-104, 2003.
- [28] Y. Zhu and D. Shasha, "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time," *Proc. of VLDB Conf.*, 2002.

On Abnormality Detection in Spuriously Populated Data Streams

Charu C. Aggarwal
IBM T. J. Watson Research Center
charu@us.ibm.com

Abstract

In recent years, advances in hardware technology have made it increasingly easy to collect large amounts of multidimensional data in an automated way. Such databases continuously grow over time, and are referred to as data streams. The behavior of such streams is sensitive to the underlying events which create the stream. In many applications, it is useful to predict abnormal events in the stream in a fast and online fashion. This is often a difficult goal in a fast data stream because of the time criticality of the detection process. Furthermore, the rare events may often be embedded with other spurious abnormalities, which affect the stream in similar ways. Therefore, it is necessary to be able to distinguish between different kinds of events in order to create a credible detection system. This paper discusses a method for supervised abnormality detection from multi-dimensional data streams, so that high specificity of abnormality detection is achieved. We present empirical results illustrating the effectiveness of our method.

1 Introduction

In recent years, the advances in hardware technology have made it possible to collect large amounts of data in many applications. Typically, such databases are created by continuous activity over long periods of time, and are therefore databases which grow without limit. The volume of such transactions may easily range in the millions on a daily basis. Examples of such domains include supermarket data, multimedia data and telecommunication applications. The volume of such data is so large that it is often not possible to store it on disk in order to apply standard algorithmic techniques. Such data are referred to as *data streams*. Algorithms which are designed on such data need to take into account the fact that it is not possible to revisit any part of the voluminous data. Thus, only a single scan of the data is allowed during stream processing.

Considerable research has been done on the data stream problem in recent years [7, 8, 9, 13, 18]. Many traditional data mining problems such as clustering

and classification have recently been re-examined in the context of the data stream environment [1, 10, 13]. In this paper, we discuss the problem of abnormality detection in data streams. In particular, we will consider the most difficult case, when such events are embedded with other similar but spurious patterns of abnormality. Some examples of applications of interest are as follows:

- In a stock market monitoring application, one may wish to find patterns in trading activity which are indicative of a possible stock market crash in an exchange. The stream of data available may correspond to the real time data available on the exchange. While a stock sell-off may be a relatively frequently occurring event which has similar effects on the stream, one may wish to have the ability to quickly distinguish the rare crash from a simple sell-off. Another example of a valuable event detection application is that of detection of particular patterns of trading activity which result in the sell-off of a particular stock, or a particular sector of stocks. A quick detection of such events is of great value to financial institutions.
- In a business activity monitoring application, it may be desirable to find particular aspects of the stream which are indicative of events of significance to the business activity. For example, certain sets of actions of competitor companies may point to the probable occurrence of significant events in the business. When such events do occur, it is important to be able to detect them very quickly, as they may be used to trigger other business-specific actions in a time-critical manner.
- In a medical application, continuous streams of data from hospitals or pharmacy stores can be used to detect any abnormal disease outbreaks or biological attacks. Certain kinds of diseases caused by biological attacks are often difficult to distinguish from other background diseases. For example, an anthrax infection has similar characteristics to the common flu attack. However, it is essential to be

able to make such distinguishing judgements in real time in order to create a credible event detection system.

Many of the events discussed above are inherently rare. For example, events such as disease outbreaks or stock market crashes may happen rarely over long periods of time. On the other hand, the value of event detection is highly dependent on the latency of the detection process. This is because most event detection systems are usually coupled with time-critical response mechanisms. Furthermore, because of efficiency considerations, it is possible to examine a data point only once throughout the entire computation. This creates an additional constraint on how abnormality detection algorithms may be designed. While event detection and anomaly detection are important problems in the data mining community [4, 6, 15, 17], these models do not address the problem in the context of predicting rare anomalies in the presence of many spurious (but similar) patterns. In order to achieve the specificities in abnormality detection, we will utilize a supervised approach in which the abnormality detection process learns from the data stream. A considerable level of specificity may be achieved by using the behavior of the combination of *multiple streams* which are able to distinguish between different kinds of seemingly similar anomalies.

Thus, the additional complexities of the generic event detection problem may be summarized as follows:

- In most real-life situations, data streams may show abnormal behavior for a wide variety of reasons. It is important for an event detection model to be *specific* in its ability to focus and learn a rare event of a particular type. Furthermore, the spurious events may be significantly more frequent than the rare events of interest. Such a situation makes the event detection problem even more difficult, since it increases the probability of a false detection.
- In many cases, even though multiple kinds of anomalous events may have similar effects on the individual dimensions, the relevant event of interest may only be distinguished by its relative effect on these dimensions. Therefore, an event detection model needs to be able to quantify such differences.
- Since the data stream is likely to change over time, not all features remain equally important for the event detection process. While some features may be more valuable to the detection of an event in a given time period, this characteristic may vary with time. It is important to be able to modify the event detection model appropriately with the evolution of the stream.

- We note that a supervised abnormality detection problem is very different from a data stream classification problem in which each record has a label attached to it. In an abnormality detection problem, individual records are unlabeled, whereas the abnormalities of importance are attached only to particular moments in time. Since individual records do not have class labels, the training of the event detection process is more constrained from the limited information availability. Furthermore, the rarity of the abnormality adds an additional level of complexity to the detection process.
- Unlike a traditional data source, a stream is a continuous process which requires simultaneous model construction and event reporting. Therefore, it is necessary for the supervision process to work with whatever information is currently available, and continue to update the abnormality detection model as new events occur.

In this paper we will design an abnormality detection algorithm which can handle the afore-mentioned complexities. Furthermore, the algorithms discussed in this paper do not require any re-scanning of the data, and are therefore useful for very fast data streams.

This paper is organized as follows. In the remainder of this section, we formalize the contributions of this paper and discuss the notations. In the next section we will discuss the algorithm for event detection. We will discuss the empirical results in section 3. Finally, section 4 contains the conclusions and summary.

1.1 Contributions of this paper This paper presents an effective method for learning rare abnormalities from fast data streams. Since a data stream may contain many different kinds of abnormalities, it is necessary to be able to distinguish their characteristic behavior. Therefore, we propose a technique which is able to distinguish particular kinds of events by learning subtle differences in how different streams are affected by different abnormalities. The algorithm performs statistical analysis on multiple dimensions of the data stream in order to perform the detection. Since the technique is tailored for fast responses to continuous monitoring of processes, the entire framework of the algorithm is constructed to facilitate online event monitoring of data streams. Therefore, the process can detect the abnormalities with any amount of historical data, but the accuracy is likely to improve with progression of the stream, as more data is received.

1.2 Notations and Definitions We discretize the points in time at which the behavior of the stream

is monitored as *ticks*. The time stamps associated with the ticks are denoted by $t(1), t(2), \dots, t(k)$. We distinguish between the ticks and time stamps, since the behavior may not necessarily be monitored at equal time intervals. It is assumed that the data points arrive only at one of these ticks or time stamps.

The total number of data streams is N , and the set of data points associated with the i th stream at tick k is denoted by $Y_i(k)$. The data points in the stream $Y_i(k)$ are denoted by $y_i(1), y_i(2), \dots, y_i(k)$. It is assumed that for each stream i , the data point $y_i(j)$ arrives at the time stamp $t(j)$. The entire feed of N streams at tick k is therefore denoted by $\mathcal{Y}(k) = \{Y_1(k) \dots Y_N(k)\}$.

We assume that the time stamps at which the rare events occur in the data stream are denoted by $T(1) \dots T(r)$. These events may either be the *primary* events that we wish to detect, or they may be spurious events in the stream. We will also refer to the spurious events as the *secondary* events. Associated with each event k at time $T(k)$, we maintain $flag(k)$ which is 1 only when this event is a primary event. In addition, we also maintain $Q(k)$, which is the time stamp of the last *reported* occurrence of any event. The value of $Q(k)$ is typically larger than the true time of event occurrence $T(k)$. This is because the value of $Q(k)$ refers to the event *report* time, whereas the value of $T(k)$ refers to the *occurrence* time. The last report time is typically larger than time of the actual event itself, since the external sources reporting the abnormality would need a lag to verify it. These external sources may use a variety of domain specific methods or simply personal observation to decide on the final verification of abnormality occurrence. It is assumed that the report of an event is an external input to the algorithm, and is available only after a reasonable lag after the actual occurrence of the event. Clearly, a *detection* algorithm is useful only if it can report events and abnormalities *before* they are independently reported and verified by external sources. Let us assume that $k(r)$ events have occurred till tick r . We denote the sequence $\{(Q(1), T(1), flag(1)) \dots (Q(k(r)), T(k(r)), flag(k(r)))\}$ until tick r by the event vector $\mathcal{E}(r)$. We note that the length of this sequence depends upon the number of events which have transpired till tick r .

The abnormality detection algorithm outputs a set of time stamps $T^*(1) \dots T^*(n)$ at which it has predicted the detection. A particular detection $T^*(i)$ is referred to as a true detection, when for some lag threshold $maxlag$, some time stamp $T(j)$ exists, such that $0 \leq T^*(i) - T(j) \leq maxlag$. Otherwise, the detection is referred to as a false positive. Clearly, there is a tradeoff between being able to make a larger number of true detections and the number of false alarms. If the

Algorithm *StreamEvent*(Initial Stream/Event History: $(\mathcal{Y}_h, \mathcal{E}_h)$,
Current Stream/Event Feeds: $(\mathcal{Y}(\cdot), \mathcal{E}(\cdot))$)

```

begin
  { Create the initial specificity model based on
    initial stream history available }
   $\mathcal{M} = LearnStream(\mathcal{Y}_h, \mathcal{E}_h)$ 
   $r = 1$ ;
  for each tick  $r$  do
    begin
       $SZ = ComputeStatisticalDeviations(\mathcal{Y}(r + 1))$ ;
       $\mathcal{AL} = PredictEvent(SZ, \mathcal{M})$ ;
      if any event has occurred on tick  $k$  then
         $\mathcal{M} = LearnStream(\mathcal{Y}(k), \mathcal{E}(k))$ 
      { This updating of the model by LearnStream is
        done as a (background) offline process }
    end
  end

```

Figure 1: The Abnormality Monitoring Algorithm

algorithm outputs a larger number of detection time stamps in order to reduce the latency, it is likely to report a greater number of false positives and vice-versa. We will discuss more about this issue in a later section.

2 The Abnormality Detection Model

The supervised abnormality detection algorithm continuously detects events utilizing the data from the history of previous event occurrences. In addition, a learning phase is triggered once after every *reported* occurrence of a primary or secondary event in order to update the model. As discussed earlier, the *reporting* of an abnormality is an independent external process and is not dependent upon the actual detection of an abnormality by the algorithm. In most practical applications, abnormalities are eventually detected and reported because of a variety of factors such as the actual practical consequences of the abnormality. These report times are often too late to be of practical use for event responses. However, they can always be used to improve the accuracy of the abnormality detection model when required.

The model is initialized at the beginning of the detection process. Both the process of initialization and updating are performed by the subroutine *LearnStream* of Figure 1. The learning phase is performed as a background offline process, whereas the abnormality detection phase is performed as an online process. Therefore, the abnormality detection phase is performed at each tick and it consists of two steps:

- Computation of abnormal statistical deviations at a given instant. This is performed by the *ComputeStatisticalDeviations* procedure of Figure 1.
- Computation of specificity of statistical deviations

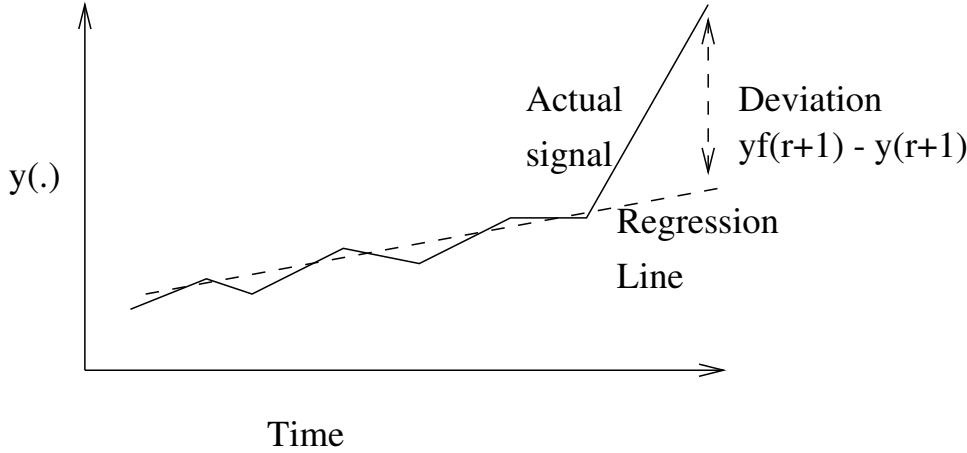


Figure 2: Deviations in data stream values

to occurrences of the primary event. This is performed by the *PredictEvent* procedure of Figure 1.

It is assumed that at the beginning of the stream monitoring process, some amount of historical data is available in order to construct an initial model of event behavior. This historical data consists of the stream and the events in the past time window at the beginning of the event monitoring process. The initial stream is denoted by \mathcal{Y}_h , and the initial set of events is denoted by \mathcal{E}_h . Once the event detector has been initialized, it then moves to an iterative phase of continuous online monitoring together with occasional offline updating.

Because of the speed of a data stream and the time-criticality of the event reporting process, it is essential that each of the above computations can be performed rapidly during real time. In the remainder of this section, we will describe each of these steps in some detail.

2.1 Detection of Rare Events in the Stream

The first step is to find deviations in the streams from the expected values based on the historical trends. The computation of the level of statistical deviations at a given instant is needed both for the alarm prediction phase as well as learning model. This process is denoted by *ComputeStatisticalDeviations* of Figure 1. For this purpose, we use a polynomial regression technique which can compute the statistically expected value of the data stream at a given moment in time.

Consider the tick r at which the points $y_i(1) \dots y_i(r)$ have already arrived for stream i . For each $l \in \{1, \dots, r\}$, the technique approximates the data point $y_i(l)$ by a polynomial in $t(l)$ or order k . In other words, we approximate the data point $y_i(l)$, by the polynomial

Algorithm *ComputeStatisticalDeviations*(Stream: $\mathcal{Y}(r+1)$)
begin
 Determine the set of points in the past window h_t together with their weights;
for each stream $i \in \{1, \dots, N\}$ **do**
 begin
 Compute coefficients $a_{i0} \dots a_{ik}$;
 Compute $Err_i(r)$ and $yf_i(r+1)$ using polynomial function $f(\cdot, \cdot)$;
 Compute $z_i(r+1) = (yf_i(r+1) - y_i(r+1))/Err_i(r)$;
 end
return($z_1(r+1) \dots z_N(r+1)$);
end

Figure 3: Computation of Statistical Deviations in the Data Stream

function $f_i(k, l)$, where:

$$(2.1) \quad f_i(k, l) = \sum_{j=0}^k a_{ij}^l \cdot t(l)^j$$

Here, the coefficients of the polynomial function are $a_{i1}^l \dots a_{ik}^l$. The values of a_{ij}^l need to be computed using the actual data points in order to find the best fit. Specifically, the data points within a maximum window history of h_t are used in order to compute the coefficients of this polynomial function. While these coefficients can be estimated quite simply by using a polynomial fitting technique, we note that not all points are equally important in characterizing this function. Those points which have arrived recently should be given greater importance in the computation. For this purpose, we use the exponential fading function $K \cdot e^{\lambda t(r)}$ in order to compute the importance of each data point. Here K and λ are constants which are specific to that particular time window. Thus, for each

data point arriving at time t , it is replaced by $\lfloor K \cdot e^{\lambda t(r)} \rfloor$ data points while computing the polynomial function. Then a polynomial fitting technique is used in order to compute the coefficients. The value of λ is chosen such that the ratio of the weight of the data point at $t(r) - h_t$ to the data point arriving at $t(r)$ is given by *maxratio*. Therefore, we have:

$$(2.2) K \cdot e^{\lambda t(r)} / (K \cdot e^{\lambda(t(r)-h_t)}) = e^{\lambda \cdot h_t} = \text{maxratio}$$

The value of K is automatically chosen that the value of the fading function is equal to 1 at the time stamp $t(r) - h_t$. Therefore, once the value of λ has been set, the value of K is chosen using the following relationship:

$$(2.3) K \cdot e^{\lambda(t(r)-h_t)} = 1$$

The two intuitive parameters which are chosen by the user are the maximum window h_t and the maximum ratio *maxratio*. The choice of h_t depends upon the amount of memory buffer available. This is because all the data points in the past window of h_t need to be held in the memory buffer while processing the stream at a given tick. It is necessary to keep such a buffer because of the large volume of data points arriving in each unit of time. At each tick, when new data points enter the system, the set of points outside the window of h_t are discarded, and the new set of points arriving at the current tick are included. We also note that while the computation of the coefficients of the polynomial function requires a matrix inversion operation [14], the order of the matrix inverted is given by the maximum order of the polynomial function. For polynomials of small order up to 2 or 3, this computation can be performed in a small constant number of operations at each tick. This is a very small overhead compared to the processing of the points in the data stream.

Once the coefficients of the polynomial function $f_i(k, r)$ has been computed, we calculate the fitted values $yf_i(r - h_t) \dots yf_i(r)$ of the points in the data stream. (The value of $yf_i(r)$ is simply the instantiation of the function $f_i(k, r)$ at the tick r .) The average standard error of fitting is given by:

$$(2.4) Err_i(r) = \sqrt{\frac{\sum_{q=r-h_t}^r e^{\lambda(q)} (y_i(q) - yf_i(q))^2}{\sum_{q=r-h_t}^r e^{\lambda(q)}}}$$

The predicted value of the data point from stream i at the tick $(r+1)$ is given by $yf_i(r+1)$. Since, the predicted value $yf_i(r+1)$ is based on the behavior of the stream i up to tick r , the true value may vary considerably from the prediction when a rare event occurs. An example of such an occurrence is illustrated in Figure 2. We quantify this deviation at tick $(r+1)$ by the

corresponding z -number of the stream:

$$(2.5) z_i(r+1) = (yf_i(r+1) - y_i(r+1)) / Err_i(r)$$

The z -number is equal to the number of standard deviations by which the true value of the stream i at tick $(r+1)$ is greater than the expected value. A high absolute magnitude of the z -number indicates significant statistical deviation from expected behavior of the stream. The process *ComputeStatisticalDeviations* of Figure 1 outputs $\mathcal{SZ} = (z_1(r+1) \dots z_N(r+1))$ which are the statistical deviations from the predicted values in each data stream. A high absolute value of these statistical deviations is indicative of the occurrence of a rare event in the streams. However, such an event could either correspond to the primary event or a secondary event. The event detector needs to distinguish between the two situations using the property that different kinds of events have a different signature as to the amount by which the events affect the different streams. The exact signature of a particular kind of event needs to be learned from the previous history of event occurrences. We will discuss this issue in the next subsection.

2.2 Learning Specific Events from the Data Stream

We note that a given data stream may have different kinds of events which have different effects on the various components. For example, consider a biological attack application in which we are tracking the number of people admitted to emergency rooms with flu like symptoms. Let us also assume that we have different data streams corresponding to adults and children. While the early phase of a large anthrax attack may be indistinguishable from a flu epidemic, the data stream corresponding to children and adults admitted is likely to be different. For both anthrax attacks and flu epidemics, the z -numbers of both streams are likely to rise. However, the z -number of the stream for children admitted is likely to be affected to a greater extent in the case of a flu epidemic, while both streams are likely to be almost equally affected in the case of an anthrax attack. How do we magnify these subtle differences in the extent to which the different curves are affected?

In order to achieve this we use the data from previous event occurrences in order to create a distinguishing model for the particular kind of event which is being tracked. This model for distinguishing different kinds of events needs modeling which is done offline. However, this modeling needs to be done only in the following cases: (1) At the very beginning of the stream monitoring process as an initialization step. It is assumed that an initial amount of event history \mathcal{E}_h and \mathcal{Y}_h is available for this purpose. (2) At the occurrence of each kind of primary event as an updating step. In this case,

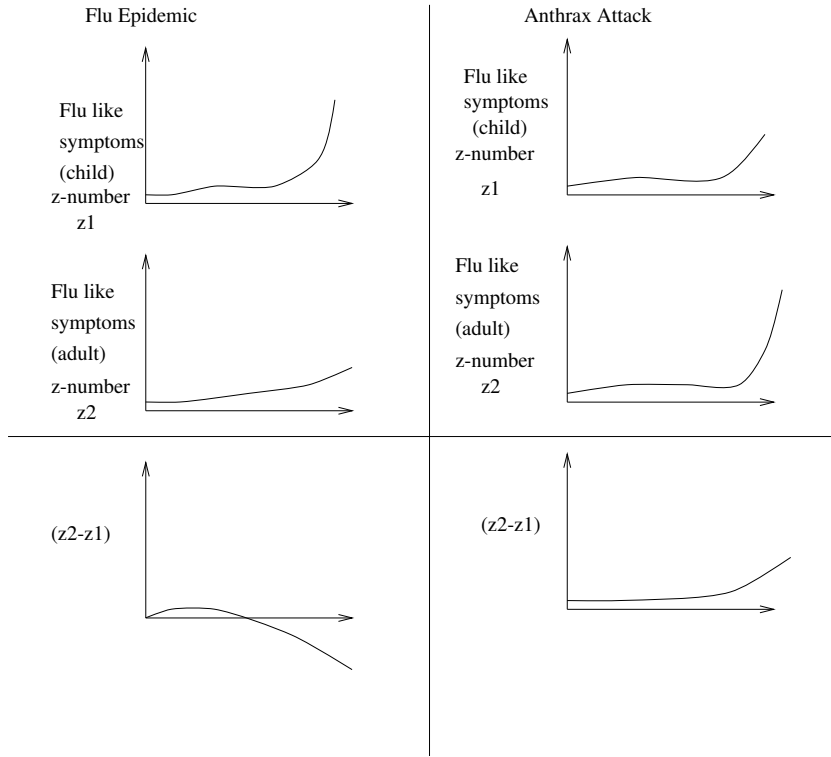


Figure 4: Distinguishing between Primary and Secondary Abnormalities

Algorithm *LearnStream*(Stream History: $\mathcal{V}(\cdot)$, Event History: $\mathcal{E}(\cdot)$)

begin

{ Let $T(1) \dots T(s)$ be the time-stamps at which the primary events have occurred }
for each time stamp $T(j) \in \{T(1) \dots T(s)\}$
and stream i find the largest value max_{ij} for the z-number of stream i in the interval $(T(j), T(j) + maxlag)$

Let \mathcal{S} be the set of streams such that max_{ij} is greater than a pre-defined threshold z_{min} for each $j \in \{1 \dots s\}$;

{ Assume that the ticks at which primary events have occurred are denoted by $j_1 \dots j_s$ and the time stamps of secondary events are $i_1 \dots i_l$; }

for each tick j_k in $\{j_1 \dots j_s\}$ and stream i
find the tick $tp_i^*(k)$ such that $tp_i^*(k) \in \{t(j_k), t(j_k) + maxlag\}$
and $z_i(\cdot)$ attains its maximum value in this time interval;

$Tp^*(k) = \sum_{i \in \mathcal{S}} tp_i^*(k) / |\mathcal{S}|$;

for each tick i_k in $\{i_1 \dots i_l\}$ and stream j
find the tick $ts_j^*(k)$ such that $ts_j^*(k) \in \{t(i_k), t(i_k) + maxlag\}$
and $z_j(\cdot)$ attains its maximum value in this time interval;

$Ts^*(k) = \sum_{j \in \mathcal{S}} ts_j^*(k) / |\mathcal{S}|$;

Define the algebraic expression $Z^s(\alpha) = \sum_{k=1}^l \sum_{i \in \mathcal{S}} \alpha_i \cdot z_i(Ts^*(k)) / l$

Define the algebraic expression $Z^p(\alpha) = \sum_{k=1}^s \sum_{j \in \mathcal{S}} \alpha_j \cdot z_j(Tp^*(k)) / s$

Find the value of the vector α which maximizes

$Z^p(\alpha) - Z^s(\alpha)$ subject to the constraints

$\|\alpha\| = 1$ and $\alpha_i = 0$ for $i \notin \mathcal{S}$

{ The gradient descent method is used for the maximization }

return(α);

end

Figure 5: Learning Specific Events from the Data Stream

the model uses the effects of the event on the stream in order to update the current model.

While the *LearnStream* procedure is triggered by the occurrence of a primary event, the iterative event monitoring process continues in the foreground. Therefore, if another primary event occurs before the *LearnStream* procedure has finished execution, the slightly stale model is always available for the detection process.

We note that in many cases, even when streams are similarly affected by different kinds of events, the relative magnitudes of different streams could vary considerably. Our aim is to create a function of the z -numbers of the different streams which is a “signature” of that particular kind of event. In order to achieve this goal, we create a new signal at each tick which is a linear combination of the signals from the different streams. Let $\alpha_1 \dots \alpha_N$ be N real coefficients. We define the following new signal $Z(r)$ in terms of the original signal and the α values:

$$(2.6) \quad Z(r) = \sum_{i=1}^N \alpha_i \cdot z_i(r)$$

For example, in the case illustrated in Figure 4, the signals for the two streams corresponding to adult and children admission to hospitals are illustrated. The signal for the adult stream is denoted by z_2 and the signal for the child stream is denoted by z_1 . A choice of $(\alpha_1, \alpha_2) = (-1, 1)$ creates the composite signal $z_2 - z_1$. In Figure 4(a), the different signals for the case of a flu epidemic are illustrated, whereas in the case of Figure 4(b) the signals for an anthrax attack are illustrated. It is clear that even though both streams are affected in a similar way by either a flu epidemic or an anthrax attack, the proportional effects are different. These effects can be magnified by the use of the composite signal $z_2 - z_1$, which is affected in a clearly different way in the two cases.

We note that many of the data streams may be noisy and will not have any correlation with the primary event. Such streams need to be discarded from the event distinguishing process. In other words, the corresponding values of α_i need to be set to zero. The first step is to identify such streams. For each of the time stamps $T(j) \in \{T(1) \dots T(s)\}$ at which an event of interest has occurred, we find the largest value¹ max_{ij} of $z_i(r)$ for each r such that $T(j) \leq t(r) \leq T(j) + maxlag$.

¹Strictly speaking, the value of max_{ij} should be based on the absolute value of the z -numbers. However, the above definition does not lose generality. For those streams in which events of interest correspond to highly negative z -numbers, the sign of the stream is flipped.

A stream i is said to be interesting to the event detector, when for each $j \in \{1 \dots s\}$ the value of max_{ij} is larger than a pre-defined threshold z_{min} .² We denote this subset of streams $\{i_1 \dots i_w\} \in \{1, \dots N\}$ by \mathcal{S} .

Once we have selected a small number of streams which are meaningful for the event detection process, we need to find the value of the *discrimination vector* α which distinguishes the primary events from other similar events. The main idea is to choose α in such a way so that the value of $Z(r)$ peaks just after the occurrence of each primary event to a much greater extent than any other event. Let us assume that the time stamps at which all secondary events which have happened within the previous history of h_t , are given by $t(i_1) \dots t(i_l)$, whereas the time stamps of the primary event are given by $\{T(1) \dots T(s)\} = \{t(j_1) \dots t(j_s)\}$. For each secondary event i_k and each stream j , we compute the maximum value of $z_j(r)$ for each value of r , such that $t(r) \in (t(i_k), t(i_k) + maxlag)$. Let the corresponding time stamp be given by $ts_j^*(k)$ for each $k \in \{1 \dots l\}$. This time stamp is then averaged over all streams which lie in \mathcal{S} . Therefore, for each secondary event k , we compute $Ts^*(k) = \sum_{i \in \mathcal{S}} ts_j^*(k) / |\mathcal{S}|$. Similarly, for each occurrence of the primary event, we can compute the average time stamp $Tp^*(k)$ for each $k \in \{1, \dots s\}$. In order for the discrimination between primary and secondary events to be as high as possible, the difference in the average value of the composite signal at the time stamps of the true and spurious events must be maximized. The average composite signal $Z^s(\alpha)$ at the time of occurrences of the true events is given by the following:

$$(2.7) \quad Z^s(\alpha) = \sum_{k=1}^l \sum_{i \in \mathcal{S}} \alpha_i \cdot z_i(Ts^*(k)) / l$$

Similarly, the average composite signal at the time of occurrence of the primary events is given by the following expression:

$$(2.8) \quad Z^p(\alpha) = \sum_{k=1}^s \sum_{i \in \mathcal{S}} \alpha_i \cdot z_i(Tp^*(k)) / s$$

For maximum discrimination between true and spurious occurrences of primary events, we must choose α in such a way that the difference $Z^p(\alpha) - Z^s(\alpha)$ is maximized. Therefore, we have:

$$\begin{aligned} & \text{Maximize } Z^p(\alpha) - Z^s(\alpha) \\ & \text{subject to:} \\ & \alpha_j = 0 \quad j \in \{i_1 \dots i_w\} \\ & \|\alpha\| = 1 \end{aligned}$$

²For a normal distribution, more than 99.9% of the points are located within 3 standard deviations from the mean.

Algorithm *PredictEvent*(Statistical Deviations: \mathcal{SZ} ,
Learned Data: \mathcal{M})
begin
{ The learned data \mathcal{M} consists of the
vector α . The deviations \mathcal{SZ}
consist of the statistical deviations at tick $(r + 1)$ }
 $ZP(r + 1) = \sum_{i=1}^N \alpha_i \cdot z_i(r + 1)$
Output event detection signature $ZP(r + 1)$;
end

Figure 6: Rare Event Prediction

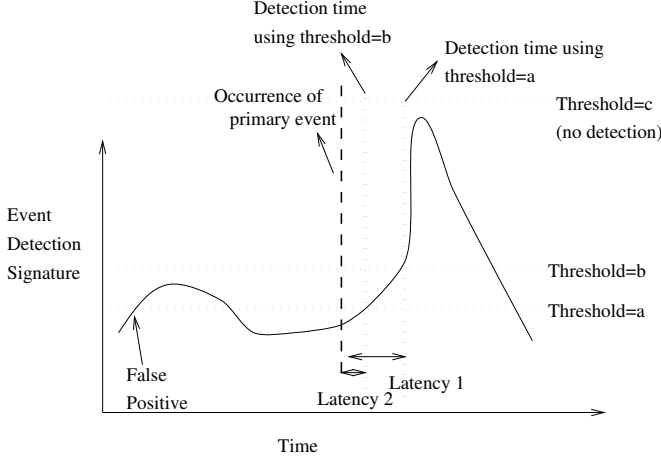


Figure 7: Illustrating the trade-off between Latency and False Positives

The first set of constraints corresponds to the fact that only the subset of streams which were determined to be significant to the event detection process are used for detection. The second constraint on α is required simply for the purpose of scaling as a boundary condition to the maximization problem. (Without the boundary condition, the maximization problem has an infinite solution.) We note that the objective function is linear in α , and all constraints are either linear or convex. Therefore, the optimum value of α can be found using a simple iterative gradient descent method which is discussed in [3].

2.3 Rare Event Prediction Process The event prediction is done by the procedure *PredictEvent* using the statistical information collected by the other procedures. The event prediction process uses the statistical deviations calculated by *ComputeStatisticalDeviations*, and the combination vector α computed by the *LearnStream* procedure. Let $\mathcal{SZ} = (z_1(r + 1) \dots z_N(r + 1))$ be the statistical deviations returned by *ComputeStatisticalDeviations* and α be the combination vector com-

puted by *LearnStream*. The *PredictEvent* procedure then computes the value $ZP(r + 1)$ which is defined as follows:

$$(2.9) \quad ZP(r + 1) = \sum_{i \in \mathcal{S}} \alpha_i \cdot z_i(r + 1)$$

This value is the signal which is specific to the primary event. The greater this value, the higher the likelihood that a primary event has indeed occurred in the stream. A primary event is predicted by using a minimum threshold on the value of $ZP(r + 1)$. Whenever the value of $ZP(r + 1)$ exceeds this threshold, a discrete signal is output which indicates that the event has indeed occurred. The use of higher thresholds on the event detection signature results in lower number of false positives, but lower detection rates as well as higher lags. For example, in Figure 7, we have illustrated the variation in latency level with time. We have illustrated two different thresholds on the event detection signature. It is clear that with the use of the lower threshold value of a , at least one false positive is created, which does not appear with the use of the higher threshold value of b . On the other hand, when the event does occur, it is detected much later with the use of the higher threshold. Therefore, the latency of detection is also much higher, when the threshold value of b is used. If the threshold level were increased even further to a value of c as indicated in Figure 7, then the algorithm misses detection completely. We will explore the effects of these trade-offs on a number of data sets in the empirical section. For the purpose of the algorithmic description of Figure 6, we output the value of $ZP(r + 1)$ as the event detection indicator.

3 Empirical Results

In this section, we will illustrate the effectiveness of the event detection system. Since many of the application-specific methods such as those in [15] were designed with the assumption of a single channel (stream) of data for classification purposes, it is difficult to make a direct comparison with any previous algorithm. Furthermore, in many cases, the algorithms were not designed to handle the computational issues arising in the context of handling the massive volumes of a fast, spuriously populated data stream. We found however, that the anomaly detection method of [15] could be adapted to the data stream environment in a relatively straightforward way. The algorithm in [15] uses a nearest neighbor classification on the previous event history in order to output the detection of an event. By calculating the distance using multiple features, the technique could be applied to the more general data stream problem discussed in this paper. We will refer to this technique as the CL

detector in the empirical results.

In order to test the algorithm, we used a number of data sets. The synthetic data sets were created by first generating each base stream j from a normal distribution with mean μ_j and standard deviation $\sigma_j = (1/3) \cdot \mu_j$. The value of μ_j for each stream is generated from a uniform distribution in the range $[0, 1]$. The events of significance are assumed to occur at w randomly distributed times $t_1 \dots t_w$ throughout the distribution of the data stream. We assume that there are f different types of events. Only one of these f different types of events corresponds to the primary event and the remaining events correspond to the secondary event. It is assumed that the occurrence of any event is equally likely to be of any type. Let us now consider a particular event at time $t_r \in \{t_1 \dots t_w\}$ which is of type $l \in \{1 \dots f\}$. The event of type l results in addition of a further signal to each data stream j . This signal is normally distributed with a mean of $\phi_{lj}(t - t_r)$ and a standard deviation of $\psi_{lj}(t - t_r)$. The value of $\phi_{lj}(s)$ was chosen to be quadratic function of s with a maxima at $s = \theta_{lj}$ and a maximum value of b_{lj} . The values of θ_{lj} and b_{lj} are chosen depending upon the event type and data stream. For each event type l and stream j , the value of θ_{lj} and b_{lj} are chosen from a uniform data distribution with ranges $[5, 50]$ and $[0, 1]$ respectively. The value of $\psi_{lj}(s)$ at each point was chosen to be $(1/3) \cdot \phi_{lj}(s)$. A value of $f = 5$ was used in each case. Two 10-dimensional synthetic data sets (for different random seeds) were generated using this methodology and are referred to as SStream-A and SStream-B respectively. In addition, two 20-dimensional data sets were generated with the same methodology. These are referred to as SStream-C and SStream-D respectively.

An interesting question which arises in the context of an empirical evaluation is that of choice of appropriate metrics. This is because a trade-off exists between the latency time and the number of false positives. Therefore, if the latency time of two algorithms is compared, it needs to be evaluated for the same number of false positives, and vice-versa. For this purpose, we need to provide a measure of the benefit that a true detection provides for a given level of detection latency. A more precise way of defining this would be with the use of a *benefit function*. While the exact function depends heavily on the application at hand, we designed a function which seems to be a reasonably intuitive metric for a variety of applications. We designated a maximum period of time latency δ_z after which detection provided no benefit. This is because anomalies of importance are often discovered in most applications through external factors such as simple human observation. There is no benefit to a detection, when the detection latency is

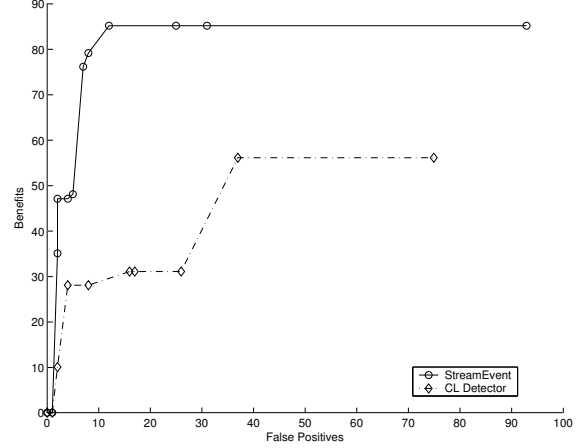


Figure 8: False Positives versus Benefits of Quick Detection (SStream-A)

larger than such external latencies. Thus, when the latency of detection is larger than δ_z , the benefit function is set to zero. For latencies less than δ_z , a linear function was used in order to characterize the benefit. Therefore, for a latency of t , the the benefit function $B(t)$ is defined as follows:

$$(3.10) \quad B(t) = \max\{0, \delta_z - t\}$$

We note that in this case, t is the latency of *first* detection of the anomaly after its actual occurrence. Similarly, the number of false positives is defined as the number of times an alarm is raised outside the δ_z interval after the true occurrence of any anomaly. The value of δ_z was chosen to be 100 in each case.

The detector proposed in this paper outputs an analogue signature level which is used for the prediction process. This analog signature can easily be converted into a binary decision by using thresholding on the intensity of the signature. When the signature level exceeds a given threshold, the detector outputs an anomaly detection indicator. The higher the threshold, the greater the latency³ but the greater number of false positives. Therefore, higher number of false positives correspond to lower values of the benefit function and and vice-versa. In order to quantify this relationship, we create a corresponding AMOC curve which plots the false positives on the X -axis and the benefits on the Y -axis. Since the CL detector also uses thresholding on a similarity measure for the classification process [15], it is also possible to create AMOC curves for that case by varying this threshold.

³We note that in many cases, higher values of the threshold result in the detector completely missing the anomaly detection. This corresponds to infinite latency, and therefore zero benefit.

Data Set	Offline Update Time	Maximum Throughput (online)
SStream-A	3 seconds	2881 per second
SStream-B	4 seconds	2755 per second
SStream-C	3 seconds	1829 per second
SStream-D	3 seconds	1954 per second

Table 1: Time Requirements of the Event Detector

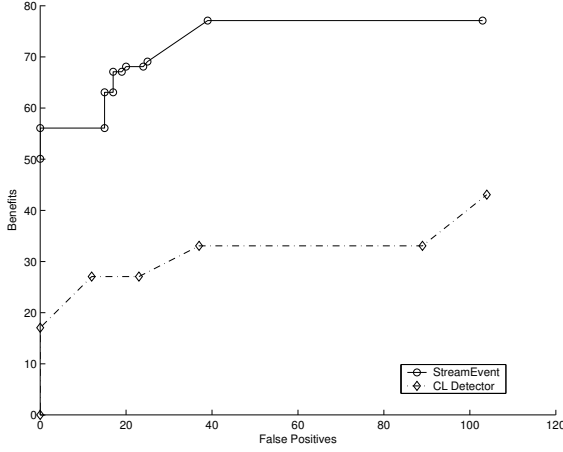


Figure 9: False Positives versus Benefits of Quick Detection (SStream-B)

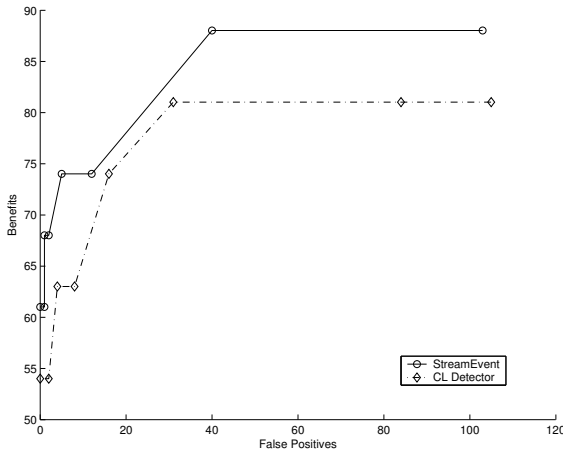


Figure 10: False Positives versus Benefits of Quick Detection (SStream-C)

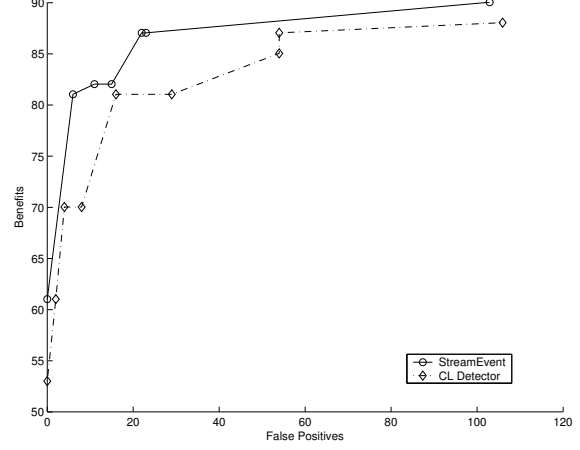


Figure 11: False Positives versus Benefits of Quick Detection (SStream-D)

The AMOC curves for both the *EventStream* and CL detector are illustrated in Figures 8, 9, 10, and 11 respectively. We note that in each case, the AMOC curve for the CL-detector algorithm was dominated by our data stream anomaly detector. In fact, in each case, there was not even one point at which the CL detector was superior to the *StreamEvent* algorithm. Furthermore, the gap between the two algorithms was significant in most cases. For example, in most cases, the *StreamEvent* algorithm was able to achieve relatively low latencies for less than 10 false positives. Modest latencies were usually achieved for only about 0-2 false positives in a majority of the cases. On the other hand, the CL Detector was never able to find the anomaly within the maximum required time latency of δ_z in the range of 10 false positives or less. Thus, the region in which the CL detector provided non-zero benefit was one in which the number of false positives was too high for the algorithm to be of practical use. As in the previous case, the *StreamEvent* algorithm significantly outperforms the CL-detector.

We tested the efficiency of the anomaly detection algorithm for processing large data streams. The algorithm was tested on 2.2 GHz laptop running Windows

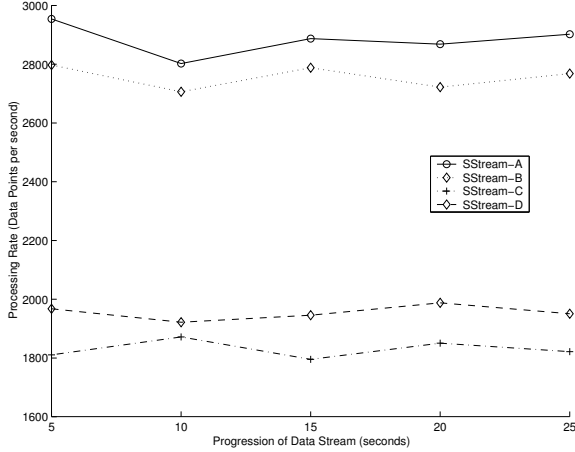


Figure 12: Processing Rate of Data Stream

XP operating system and 256 MB of main memory. We note that represents very modest hardware available today. There were two aspects which needed to be measured:

- **(1) The online efficiency:** This quantity was defined in terms of the maximum number of ticks for which a prediction could be made per second. We define this as the *maximum throughput*, since it reflects the maximum processing capabilities of the algorithm per unit of time. When the stream arrives at a faster rate than the maximum throughput, then it is necessary to use load shedding techniques in order to reduce the throughput being handled by the data stream. These load shedding techniques can be implemented in the form of sampling points from the data stream. While sampling reduces the effectiveness of the prediction process to some extent, it is an acceptable solution in many practical scenarios.
- **(2) Efficiency of the model update process:** We computed the time required by the offline component in order to update the model. As long as this average time was significantly lower than the time between the occurrence of two events, the model was always up to date at the time of prediction.

In Figure 12, we have illustrated the processing rate of the online portion of the event detection process. It is interesting to see that the processing rates for all the streams were quite stable over time and ranged in the order of thousands of data points per second. In Table 1, we have also illustrated the summary of the overall online efficiency of the algorithm for the entire data stream. While all the data sets are processed at

the rate of thousands of data points per second, the differences between the different data sets is because of the varying dimensionality of the data sets. An increased dimensionality resulted in lower processing rates, since a larger number of data points needed to be processed every second. However, since the absolute processing rates are quite high, this means that the streams can be processed efficiently using this technique.

In Table 1, we have also illustrated the time required by the offline component of the event detector. The most expensive process in the offline learning algorithm are the iterations of the gradient descent algorithm for finding the weights of the different channels. Since the gradient descent method is an iterative approach, it required a few computations in order to update the model. However, in each case, these computations require less than 5 seconds to execute. On the other hand, since the algorithm is specifically designed for detecting rare events in data streams, this time interval of a few seconds is likely to be negligible compared to the inter-arrival period between two events. Therefore, such an offline update process is easily implementable because of its relative rarity. As in the previous case, are some differences among the update times for the different data sets because of varying dimensionality of different data sets.

4 Conclusions and Summary

In this paper, we proposed a new technique for anomaly detection in massive data streams. The method is capable of fast and accurate anomaly detection in the presence of other non-relevant anomalies in the data. Therefore, the detector is able to distinguish between spurious and true anomalies. Such a system is quite unique in retaining *specificity* in anomaly detection from multi-dimensional data streams. It also has applications in many domains in which real time detection is necessary for quick response times to anomalous events. In future work, we will construct a decision support system for quick anomaly detection in massive data streams. Such a decision support system would utilize active involvement of the user in making decisions about the data.

References

- [1] C. C. Aggarwal, *A Framework for Diagnosing Changes in Evolving Data Streams*, Proceedings of the ACM SIGMOD Conference, (2003).
- [2] R. Abbott, and H. Garcia-Molina, *Scheduling real-time transactions with disk resident data*, Proceedings of the VLDB Conference, (1989).
- [3] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, 2nd Edition, (1999).

- [4] M. Berndtsson, and J. Hansson, *Issues in Active Real-Time Databases*, Active and Real-Time Databases, (1995), pp. 142–157.
- [5] D. Bonachea, K. Fisher, A. Rogers, and F. Smith, *Hancock: A language for processing very large data*, USENIX 2nd Conference on Domain-Specific Languages, (1999), pp. 163–176.
- [6] H. Branding, A. Buchmann, T. Kudrass, and J. Zimmermann, *Rules in an open system: The reach rule system*, First Workshop of Rules in Database Systems, (1993).
- [7] J. Feigenbaum, S. Kannan, M. Strauss, and M. Vishwanathan, *Testing and spot-checking of data streams*, Proceedings of the ACM SODA Conference, (2000).
- [8] J. Fong, and M. Strauss, *An approximate L^p -difference algorithm for massive data streams*, Annual Symposium on Theoretical Aspects in Computer Science, (2000).
- [9] C. Cortes, K. Fisher, D. Pregibon, A. Rogers, and F. Smith, *Hancock: A Language for Extracting Signatures from Data Streams*, Proceedings of the ACM KDD Conference, (2000).
- [10] P. Domingos, and G. Hulten, *Mining High-Speed Data Streams*, Proceedings of the ACM KDD Conference, (2000).
- [11] V. Ganti, J. Gehrke, and R. Ramakrishnan, *Mining Data Streams under block evaluation*, ACM SIGKDD Explorations, Vol. 3(2), (2002).
- [12] J. Gehrke, F. Korn, and D. Srivastava, *On Computing Correlated Aggregates over Continual Data Streams*, Proceedings of the ACM SIGMOD Conference, (2001).
- [13] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan, *Clustering Data Streams*, Proceedings of the IEEE FOCS Conference, (2000).
- [14] R. A. Johnson, and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Fourth Edition, Prentice Hall, Upper Saddle River, NJ, (1999).
- [15] T. Lane, and C. E. Brodley, *An Application of Machine Learning to Anomaly Detection*, Proceedings of the 20th National Information Systems Security Conference, (1997), pp. 366–380.
- [16] W. Labio, and H. Garcia-Molina, *Efficient Snapshot Differential Algorithms for Data Warehousing*, Proceedings of the VLDB Conference, (1996).
- [17] W. Lee, S. J. Stolfo, and P. K. Chan, *Learning Patterns from Unix Process Execution Traces for Intrusion Detection*, AAAI Workshop: AI Approaches to Fraud Detection and Risk Management, (1997), pp. 50–56.
- [18] B-K Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris, *Online Data Mining for Co-Evolving Time Sequences*, Proceedings of the ICDE Conference, (2000).

Privacy-Preserving Classification of Customer Data without Loss of Accuracy*

Zhiqiang Yang¹ Sheng Zhong^{1,2} Rebecca N. Wright¹

¹Computer Science Department, Stevens Institute of Technology, Hoboken, NJ 07030

²DIMACS Center, Rutgers University, Piscataway, NJ 08854

Abstract

Privacy has become an increasingly important issue in data mining. In this paper, we consider a scenario in which a data miner surveys a *large* number of customers to learn classification rules on their data, while the sensitive attributes of these customers need to be protected. Solutions have been proposed to address this problem using randomization techniques. Such solutions exhibit a tradeoff of accuracy and privacy: the more each customer's private information is protected, the less accurate result the miner obtains; conversely, the more accurate the result, the less privacy for the customers.

In this paper, we propose a simple cryptographic approach that is efficient even in a many-customer setting, provides *strong privacy* for each customer, and *does not lose any accuracy as the cost of privacy*. Our key technical contribution is a privacy-preserving method that allows a data miner to compute frequencies of values or tuples of values in the customers' data, without revealing the privacy-sensitive part of the data. Unlike general-purpose cryptographic protocols, this method requires no interaction between customers, and each customer only needs to send a single flow of communication to the data miner. However, we are still able to ensure that nothing about the sensitive data beyond the desired frequencies is revealed to the data miner.

To illustrate the power of our approach, we use our frequency mining computation to obtain a privacy-preserving naive Bayes classifier learning algorithm. Initial experimental results demonstrate the practical efficiency of our solution. We also suggest some other applications of privacy-preserving frequency mining.

1 Introduction

The rapid growth of the Internet makes it easier than ever to collect data on a large scale. Data mining, with its promise to efficiently discover valuable knowledge

from vast amounts of data, is playing an increasingly important role in the current world. However, data mining, given its power in knowledge discovery, can be misused to violate people's privacy if proper measures are not taken. Privacy concerns have become one of the top priorities in technical, business, and political discussions of data mining. Due to these concerns, when customer data is collected from a large population, many people may decide to give false information, or simply decline to provide any information. Applying data-analysis tools and knowledge-discovery tools to data collected in this manner can therefore produce results with unacceptably low accuracy. This is in spite of the fact that, assuming privacy is properly protected, many people say they are willing to provide correct information in exchange for certain benefits the miner gives based on the result of mining [Wes99]. For example, a survey conducted to understand Internet customers' attitudes towards privacy showed that only 27% respondents say they are willing to provide their data without protection of privacy [Cra99]. Consequently, providing privacy protection measures may be critical to the success of data collection, and to the success of the entire task of data mining.

In this paper, we consider a "fully distributed" setting, in which each of many customers hold their own data. Existing solutions [AS00, AA01, ESAG02, EGS03, RH02, DZ03] to the privacy problem in this setting depend on randomization of each customer's data, which induces a tradeoff between privacy and accuracy: the more privacy of each customer has, the more accuracy the miner loses in his result. While in some cases the tradeoff may simultaneously offer sufficient privacy and sufficient accuracy, it is more desirable to have solutions that are both fully private and fully accurate. It has been demonstrated that in many cases random data distortion preserves very little privacy [KDWS03]. Related work on confidentiality in statistical databases [AW89, DN03, DN04] and random response techniques [War65, AJL04] also give nice dis-

*This work was supported by the National Science Foundation under grant number CCR-0331584.

cussions of randomization of data.

In contrast, cryptographic solutions to privacy-preserving data mining that provide strong privacy have been proposed [LP02, VC02, VC03, KC02, WY04], but these tend to do so at a high performance cost. In particular, in order to have efficiency that is reasonable (say, linear in the size of the “virtual” database), these solutions require the data to be shared among only a small number of parties, often just two parties. In addition, these solutions typically require multiple rounds of communication between the participants.

In this work, we consider the fully distributed setting described above. Each customer maintains her own data. This can be thought of a horizontally partitioned database in which each transaction is owned by a different customer. A data miner wishes to compute some data mining results based on the customer data. In some applications, the data miner may also have some additional information about each customer. Note that the fully distributed setting can inherently provide more customer privacy than a horizontal partition into a small number of partitions, as each customer retains full control of her data in the fully distributed setting.

We propose a new approach that provides cryptographically strong privacy for each customer, but does not lose any accuracy as a cost of privacy, and is very efficient even in the fully distributed setting. In particular, we consider a scenario in which a miner surveys a large number of customers to learn classification rules by mining the set of collected data. In this scenario, we assume each customer is willing to provide her data to the miner as long as the privacy-sensitive part of her data is protected. By adding the efficient cryptographic operations we propose, we can prove that each customer’s privacy is protected in a strong cryptographic sense.

1.1 Related Work We note that our scenario is similar to the scenario of electronic voting [BT94]. However, e-voting systems usually assume that the voting authority consists of a group of servers that are threshold-trusted, or that another authority independent from the voting authority also participates in the protocol. Both of these possibilities are not justifiable in our scenario, however. Another difference is that in our setting, additional specialized concerns in e-voting such as receipt-freedom of the protocol are not an issue.

We also note that [AJL04] uses cryptography in the randomization setting to ensure that participants properly follow their specified instructions about randomization. However, their solution still has a privacy/accuracy tradeoff; in contrast, we use cryptography to “break” the privacy/accuracy tradeoff. Yet another piece of related work is [KV02], which proposes

a general architecture for privacy-preserving data mining. But their approach needs multiple service providers that do not collude with each other, which may not be available in practice.

In the context of privacy-preserving data mining, recent work gives privacy-preserving solutions for mining a naive Bayes classifier across a database horizontally or vertically partitioned into a small number of partitions [KV03, VC04]. In contrast, our solution provides a privacy-preserving method for mining a naive Bayes classifier in a fully distributed setting where each customer holds one record.

1.2 Our Contribution Our key technical contribution is a privacy-preserving method that allows a data miner to compute frequencies of values or tuples of values in the customers’ data, without revealing the privacy-sensitive part of the data. Technically, this can be reduced to a problem of securely summing private inputs and thus in theory it can be solved by either a general-purpose protocol for secure multi-party computation, e.g. [Yao86, GMW87, BGW88], or a special-purpose protocol for the secure sum problem, e.g. [Sch96, Chapter 6]. However, our solution has a number of advantages over the existing general-purpose protocols:

- Our solution is very efficient, while the existing general-purpose protocols are prohibitively expensive.
- Our solution does not assume any communication channels between customers and so is superior in the fully distributed scenario, where it would be infeasible to require such channels.
- In our solution, only one round of interaction is needed between the miner and each customer. This brings great convenience to the data mining task, since the customers can “submit data and go.”

To illustrate the power of our proposed approach, we take naive Bayes classification as an example and enable a privacy-preserving learning algorithm to protect customers’ privacy using our privacy-preserving frequency mining computation. We also suggest other privacy-preserving algorithms that are enabled by joint frequencies, such as decision trees and association rule mining.

Cryptographic techniques are often dismissed as being too expensive to use in practical data mining applications. Indeed, performance measured from implementations of cryptographic data mining protocols have often corroborated this impression [SWY04]. However, some cryptographic solutions are very efficient, and

their performance may be good enough to be useful in practice. We have implemented our privacy-preserving frequency mining algorithm and our privacy-preserving naive Bayes classifier learning algorithm, and show that both have very reasonable overhead.

We present our privacy-preserving joint frequency algorithm, including experimental results, in Section 2. We apply joint frequencies to naive Bayes classification in Section 3, again including experimental results. We discuss the application of the frequency algorithm to decision trees and association rule mining in Section 4, and we conclude in Section 5.

2 Privacy-Preserving Frequency Mining

This section describes a privacy-preserving primitive for frequency mining. In Section 2.1, we formulate the problem of frequency mining. We state our privacy definition in Section 2.2, and present our privacy-preserving protocol for frequency mining in Section 2.3. We give a security proof for this protocol in Section 2.4 and provide experimental results in Section 2.5.

2.1 Problem Formulation We consider a very basic problem: there are n customers U_1, \dots, U_n ; each customer U_i has a Boolean value d_i ; the miner would like to find out how many d_i 's are 1's and how many are 0's.

This problem essentially amounts to computing the sum $d = \sum_{i=1}^n d_i$ without revealing each d_i . However, there are a few important restrictions:

- Each customer only sends one flow of communication to the miner; there is no further interaction between the customer and the miner.
- No customer communicates with other customers.

We call this the *reduced interaction* model. Solutions in this model are highly practical because they do not need communication channels between different customers or multi-round interaction between any customer and the miner.

In addition, we require that each customer's d_i is protected as defined in Section 2.2.

2.2 Definition of Privacy In the context of the randomization approach of protecting data privacy, there are two approaches to quantify the privacy-preserving property of a randomization methods. One approach relies on *information theory* [AA01], the other approach is based on the notion of *privacy breaches* [ESAG02, EGS03]. In the context of the cryptographic approach, the definition of privacy can be derived from the general definition of security in multi-party computations [Gol04]. Our definition of privacy given below, for example, can be viewed as a simplification of

the general definition in the *semi-honest* model, where the simplification results from our reduced interaction model. In our definition, we consider the possibility that some corrupted customers might share their information with the miner in order to help derive the private information of honest customers. We require that no extra information about the honest customers' values be leaked even if the miner above receives such help from corrupted customers. In the following definition, we do not consider problem of customers sharing their information with each other since, as we discuss after the definition, this will not give them any additional information in the reduced interaction model.

DEFINITION 1. Assume that each customer U_i has private keys x_i, y_i and public keys X_i, Y_i . A protocol for the above defined mining problem protects each customer's privacy against the miner and t corrupted users in the semi-honest model if, $\forall I \subseteq \{1, \dots, n\}$ such that $|I| = t$, there exists a probabilistic polynomial-time algorithm M such that

$$(1) \quad \{M(d, [d_i, x_i, y_i]_{i \in I}, [X_j, Y_j]_{j \notin I})\} \stackrel{c}{=} \{\text{view}_{\text{miner}, \{U_i\}_{i \in I}}([d_i, x_i, y_i]_{i=1}^n)\}.$$

Here, $\stackrel{c}{=}$ denotes *computational indistinguishability* (see a standard book of cryptography, e.g., [Gol04], for a definition), and $\{\text{view}_{\text{miner}, \{U_i\}_{i \in I}}([d_i, x_i, y_i]_{i=1}^n)\}$ is the joint *view* (again, see [Gol04] for a precise definition) of the miner and the t corrupted customers, $\{U_i\}_{i \in I}$. Intuitively, this definition states that a polynomial-time algorithm M , called a *simulator*, can simulate what the miner and the corrupted customers have observed in the protocol using only the final result d , the corrupted users' knowledge, and the public keys. Therefore, the miner and the corrupted customers jointly learn nothing beyond d .

Definition 1 only addresses privacy in the semi-honest model [Gol04] (which assumes that all parties follow the protocol). However, in our reduced interaction model, a protocol protecting customer privacy in the semi-honest model actually also protects customer privacy even when the miner and the corrupted customers are fully malicious (i.e., may deviate arbitrarily from the protocol). This is because these malicious parties cannot have any influence on the honest customers due to the restrictions of the reduced interaction model. In this sense, our solution provides privacy against malicious parties "for free". We note, however, that the correct completion of the protocol cannot be guaranteed with malicious parties, as they may send garbage or refuse to send anything at all.

2.3 Protocol Our protocol design is based on the additively homomorphic property of a variant of ElGamal encryption, which has been used in, e.g., [HS00]. The privacy of our protocol is based on the believed computational difficulty of the *discrete logarithm* problem, and the related ElGamal cryptosystem, which we will describe in more detail in Section 2.4. The protocol itself uses the mathematical properties of exponentiation, which allows the miner to combine encrypted results received from the customers into the desired sums.

Let G be a group in which discrete logarithm is hard, and let g be a generator of G . All computations in this section are carried out in the group G . Suppose that each customer U_i has two pairs of keys: $(x_i, X_i = g^{x_i}), (y_i, Y_i = g^{y_i})$. Define

$$(2) \quad X = \prod_{i=1}^n X_i$$

$$(3) \quad Y = \prod_{i=1}^n Y_i$$

The values x_i and y_i are private keys (i.e., each x_i and y_i is known only to customer U_i); X_i and Y_i are public keys (i.e., they can be publicly known). In particular, the protocol requires that all customers know the values X and Y . In addition, each customer knows the group G and the common generator g .

Recall that each customer U_i holds the Boolean value d_i , and the miner's goal is to learn $d = \sum_{i=1}^n d_i$. The privacy-preserving protocol for the miner to learn d is detailed in Figure 1.

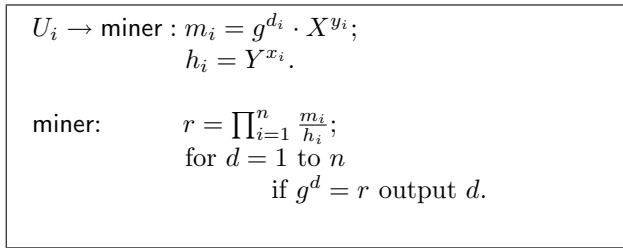


Figure 1: Privacy-Preserving Protocol for Frequency Mining.

THEOREM 2.1. *The protocol for frequency mining presented in Figure 1 correctly computes the sum of all customers' inputs.*

Proof: We show that, in the protocol, when the miner finds $g^d = r$, the value d is the desired sum. Suppose

that $g^d = r$. Then:

$$\begin{aligned}
g^d &= r \\
&= \prod_{i=1}^n \frac{m_i}{h_i} \\
&= \prod_{i=1}^n \frac{g^{d_i} \cdot X^{y_i}}{Y^{x_i}} \\
&= \prod_{i=1}^n g^{d_i} \cdot \prod_{i=1}^n \frac{X^{y_i}}{Y^{x_i}} \\
&= g^{\sum_{i=1}^n d_i} \cdot \frac{\prod_{j=1}^n X_j^{y_i}}{\left(\prod_{j=1}^n Y_j\right)^{x_i}} \\
&= g^{\sum_{i=1}^n d_i} \cdot \frac{\left(g^{\sum_{j=1}^n x_j}\right)^{y_i}}{\left(g^{\sum_{j=1}^n y_j}\right)^{x_i}} \\
&= g^{\sum_{i=1}^n d_i} \cdot \frac{g^{\sum_{i=1}^n \sum_{j=1}^n x_j y_i}}{g^{\sum_{i=1}^n \sum_{j=1}^n y_j x_i}} \\
&= g^{\sum_{i=1}^n d_i}
\end{aligned}$$

Thus, $g^d = g^{\sum_{i=1}^n d_i}$, and therefore $d = \sum_{i=1}^n d_i$, as desired. ■

2.4 Privacy Analysis Next, we establish our privacy guarantee based on a standard cryptosystem—the ElGamal encryption scheme. In this encryption scheme, to encrypt a message α using public key X , we compute

$$C = (\alpha X^k, g^k),$$

where k is chosen uniformly at random in $[0, q - 1]$. It has been shown in [TY98] (under standard complexity-theoretic assumptions) the ElGamal encryption scheme is secure in the sense of *semantic security* (see, e.g., [GM84] for the definition of semantic security). Informally, the notion of *semantic security* means that whatever the attacker could compute from the ciphertext the attacker could compute without ciphertext, then attacker learns nothing new by seeing ciphertext.

Our protocol makes use of a *homomorphic* property of a modified version of ElGamal. Specifically, if the message α is changed to g^α before encrypting, then from encryptions $(g^{\alpha_1} X^{k_1}, g^{k_1})$ and $(g^{\alpha_2} X^{k_2}, g^{k_2})$ of α_1 and α_2 , respectively, then it is possible to derive an encryption of $\alpha_1 + \alpha_2$, as $(g^{\alpha_1} X^{k_1} \cdot g^{\alpha_2} X^{k_2}, g^{k_1} \cdot g^{k_2}) = (g^{(\alpha_1 + \alpha_2)} X^{(k_1 + k_2)}, g^{(k_1 + k_2)})$.

In our protocol, the message m_i sent by customer U_i is equivalent to the first part of an ElGamal encryption of d_i under a private key $(\sum x_i) y_i$, while the message h_i is the second part is part of an ElGamal encryption of d_i under private key $(\sum y_i) x_i$. Together, all the customer

messages are combined by the miner to be an encryption of g^d . In order to undo the resulting encryption, the miner must first, in its first step, decrypt to learn r , which is g^d , and then since even the miner cannot take discrete logarithms, the miner must use trial and error to learn d . Since the range of possible values of d is not too large, this use of trial and error is feasible.

In the following, we show that our protocol protects each customer's privacy (even if there are up to $n - 2$ users colluding with the miner) as long as the ElGamal encryption scheme is secure. Throughout the rest of this paper, we take the semantic security of ElGamal encryption as an assumption.

THEOREM 2.2. *Assuming that all keys have been distributed properly when the protocol starts, the protocol for mining frequency presented in Figure 1 protects each honest customer's privacy against the miner and up to $n - 2$ corrupted customers.*

Proof: For *honest* customers' privacy, clearly it suffices to consider the case with the maximum number ($n - 2$) of corrupted customers. Since our protocol is symmetric in customer indices, without loss of generality we assume that $I = \{3, 4, \dots, n\}$. Recall that, to prove the protocol protects customer privacy, we need to construct a simulator M that can generate an ensemble indistinguishable from the miner and the corrupted customers' view using only the final result d , the corrupted users' knowledge, and the public keys. Given this simulator algorithm, we can state that the miner and the corrupted customers jointly learn nothing beyond d .

Instead of describing the entire simulator in detail, we give an algorithm that computes the view of the miner and the corrupted customers in polynomial time using only d , corrupted customers' knowledge, public keys, and some ElGamal encryptions. Under the assumption that the ElGamal encryption is semantically secure, we already know that each ElGamal ciphertext can be simulated. Therefore, combining our algorithm with a simulator for ElGamal ciphertexts, we obtain a complete simulator.

Below is the algorithm that computes the view of the miner and the corrupted customers. It takes four encryptions as its input: $(u_{11}, v_{11}) = (g^{d_1} \cdot g^{x_1 y_1}, g^{x_1})$, $(u_{12}, v_{12}) = (g^{d_1} \cdot g^{x_2 y_1}, g^{x_2})$, $(u_{21}, v_{21}) = (g^{d_2} \cdot g^{x_1 y_2}, g^{x_1})$, $(u_{22}, v_{22}) = (g^{d_2} \cdot g^{x_2 y_2}, g^{x_2})$. Then it computes m_1, m_2 by the computation:

$$(4) \quad m'_1 = u_{11} u_{12} Y_1^{\sum_{i \in I} x_i};$$

$$(5) \quad m'_2 = u_{21} u_{22} Y_2^{\sum_{i \in I} x_i}.$$

It computes h_1, h_2 by the computation:

$$(6) \quad h'_1 = u_{11} u_{21} X_1^{\sum_{i \in I} y_i} / g^{d - \sum_{i \in I} d_i};$$

$$(7) \quad h'_2 = u_{12} u_{22} X_2^{\sum_{i \in I} y_i} / g^{d - \sum_{i \in I} d_i},$$

completing the proof. ■

We note that the security of ElGamal encryption depends on new random values being used for each encryption. In our setting, this means that the x_i and y_i values, and associated X and Y , cannot be reused in different uses of the protocol. However, since these parameters do not depend on the actual data values, they can in general be precomputed off-line before the protocol starts. In particular, if the protocol is to be run many times, many sets of values could be precomputed in advance so that only a single phase of key distribution is required.

2.5 Experimental Results of Frequency Mining

We implemented our privacy-preserving frequency mining algorithm in C, using the OpenSSL libraries for the cryptographic operations. We ran a series of experiments on a PC with a 1GHz processor and 512MB memory under NetBSD. In our experiments, the length of each cryptographic key is 512 bits. We measured the computing time of the privacy-preserving frequency mining protocol for different numbers of customers, from 2,000 to 10,000.

To set up for the privacy-preserving frequency mining protocol, the key-generation time for each customer is 4.2 seconds. Computing the protocol parameters X and Y for 10,000 customers takes 140 milliseconds. As previously noted, these values can be precomputed off-line before the protocol starts.

In the privacy-preserving frequency mining protocol, it takes each customer to prepare her message to the miner only 1 millisecond, as it requires just a single modular exponentiation. The miner's computation is somewhat longer, but still quite efficient. Figure 2 shows the times the miner uses to compute one frequency for different numbers of customers. For example, for 10,000 customers, the miner's computation takes 146 milliseconds. As these results demonstrate, the privacy-preserving frequency mining protocol is very efficient.

3 Protecting Customer Privacy in Learning Naive Bayes Classifier

The primitive of frequency mining is simple, but is very useful in data mining applications. Correspondingly, our privacy-preserving frequency mining solution is also

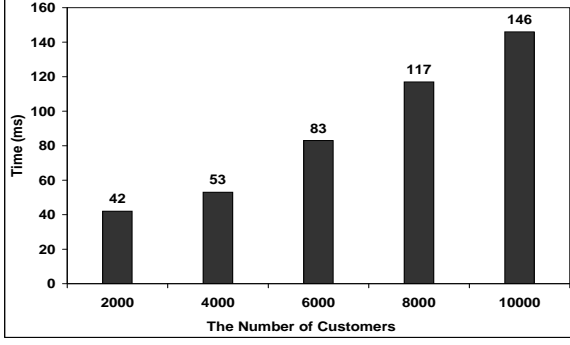


Figure 2: Server's Computation Time for a Single Frequency Calculation

quite simple, but is potentially useful whenever privacy is a top concern. In this section, we demonstrate the power of our primitive by showing a privacy-preserving naive Bayes classifier computation in the fully distributed setting (which can be thought of a horizontally partitioned database in which *each* record is held by a different party).

3.1 Naive Bayes Learning with Privacy Concerns Naive Bayes classifiers have been used in many practical applications. They greatly simplify the learning task by assuming that attributes are independent given the class. Although independence of attributes is an unrealistic assumption, naive Bayes classifiers often compete well with more sophisticated models, even if there is modest correlation between attributes. Naive Bayes classifiers have significant advantages in terms of simplicity, learning speed, classification speed, and storage space. They have been used, for example, in text classification and medical diagnosis [DP97, Mit97].

In (non-private) naive Bayes learning, the miner is given a set of training examples. We assume that each example is an attribute vector of a customer together with her class label. From these examples the miner learns a classifier that can be used to classify new instances.

In this paper, we consider the following scenario: there are m attributes, (A_1, A_2, \dots, A_m) , and one class attribute V . Without loss of generality, we assume that each attribute A_i ($1 < i < m$) has a domain of $\{a_i^{(1)}, \dots, a_i^{(d)}\}$ and the class attribute V has a domain of $\{v^{(1)}, \dots, v^{(p)}\}$. We also assume that there are n customers (U_1, \dots, U_n) , where each customer U_j has a vector denoted by $(a_{j1}, \dots, a_{jm}, v_j)$. In the customer's vector, $(a_{j1}, \dots, a_{jm}, v_j)$ is an instance of the attributes vector (A_1, \dots, A_m) and v_j is U_j 's class label. In our problem, these data are the training samples from which

the miner learns the classifier without learning the samples themselves. The miner surveys all customers for values based on their data, and constructs a classifier to classify a new instance by selecting the most likely class label v :

$$(8) \quad v = \operatorname{argmax}_{v^{(\ell)} \in V} \Pr(v^{(\ell)}) \prod_{i=1}^m \Pr(a_i | v^{(\ell)}),$$

where (a_1, \dots, a_m) is the attributes vector of the new instance.

To learn the naive Bayes classifier, traditionally the miner collects all customers' data into one central site, and then learns the classifier at that central site. In our setting, there is a set S of privacy-sensitive attributes where $S \subseteq A$. Formally, for any $j \in \{1, \dots, n\}$, U_j is not willing to reveal any information about $(a_{ji})_{i \in S}$ to the miner; but she is willing to reveal to the miner all the remaining non-sensitive attribute values $(a_{ji})_{i \in \{1, \dots, n\} - S}$. To protect customers' privacy and also enable learning classifier, we design a privacy-preserving protocol for naive Bayes learning.

3.2 A Privacy-preserving Protocol for Naive Bayes Learning We use the primitive for frequency mining in Section 2 as a building block to design a privacy-preserving protocol for naive Bayes learning. In this protocol the miner knows the schema of customer data. Without loss of generality, we assume all customers' sensitive attributes need to be protected. We have two goals to achieve:

- *Correctness*: the miner learns the naive Bayes classifier accurately.
- *Privacy*: the miner learns nothing about each customer's sensitive data except the knowledge derivable from the naive Bayes classifier itself.

To achieve these two goals, we first rewrite (8) as:

$$(9) \quad v = \operatorname{argmax}_{v^{(\ell)} \in V} \#(v^{(\ell)}) \prod_{i=1}^m \frac{\#(a_i, v^{(\ell)})}{\#(v^{(\ell)})},$$

where $\#(v^{(\ell)})$ ($\#(a_i, v^{(\ell)})$, resp.) denotes the frequency, or number of occurrences, of attribute value $v^{(\ell)}$ (attribute pair value $(a_i, v^{(\ell)})$, resp.) in all customers' data. To learn the classifier, all the miner needs to do is to learn $\#(v^{(\ell)})$ and $\#(a_i^{(k)}, v^{(\ell)})$ for each $i \in S$, each k , and each ℓ . Since the occurrence of $v^{(\ell)}$ or of the pair $(a_i^{(k)}, v^{(\ell)})$ can be denoted by a Boolean value, we can use the primitive presented in Section 2. A detailed specification of the protocol is given in Figure 3.

$\triangleright a_{ji}^{(k,\ell)} \stackrel{\text{def}}{=} 1$ if $(a_{ji}, v_j) = (a_i^{(k)}, v^{(\ell)})$;
 0 otherwise.
 $\triangleright U_j$'s private keys:
 $(x_{ji}^{(k,\ell)})_{i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p}, (y_{ji}^{(k,\ell)})_{i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p}$.
 \triangleright Public keys: $X_{ji}^{(k,\ell)} = g^{x_{ji}^{(k,\ell)}}; Y_{ji}^{(k,\ell)} = g^{y_{ji}^{(k,\ell)}}$.
 $(1 \leq j \leq n, i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p)$
 $X_i^{(k,\ell)} = \prod_{1 \leq j \leq m} X_{ji}^{(k,\ell)}; Y_i^{(k,\ell)} = \prod_{1 \leq j \leq m} Y_{ji}^{(k,\ell)}$.
 $(1 \leq j \leq n, i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p)$

 U_j : for $i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p$
 $\bar{a}_{ji}^{(k,\ell)} = g^{a_{ji}^{(k,\ell)}} \cdot (X_i^{(k,\ell)})^{y_{ji}^{(k,\ell)}}$;
 $h_{ji}^{(k,\ell)} = (Y_i^{(k,\ell)})^{x_{ji}^{(k,\ell)}}$.

 $U_j \rightarrow \text{miner} : (\bar{a}_{ji}^{(k,\ell)}, h_{ji}^{(k,\ell)})_{i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p};$
 $(a_{ji})_{i \notin S}, v_j$.

 miner: for $i \in S, 1 \leq k \leq d, 1 \leq \ell \leq p$
 $r_i^{(k,\ell)} = \prod_{j=1}^n \frac{\bar{a}_{ji}^{(k,\ell)}}{h_{ji}^{(k,\ell)}}$;
 for $\#(a_i^{(k)}, v^{(\ell)}) = 1$ to n
 if $g^{\#(a_i^{(k)}, v^{(\ell)})} = r_i$ break.
 for $i \notin S, 1 \leq k \leq d, 1 \leq \ell \leq p$
 count $\#(a_i^{(k)}, v^{(\ell)})$.
 for $1 \leq \ell \leq p$
 count $\#(v^{(\ell)})$.
 output classifier.

Figure 3: Privacy-Preserving Protocol for Naive Bayes Learning.

3.3 Protocol Analysis In the following theorem, we implicitly assume that the output classifier is encoded in such a way that it contains the frequencies $\#(v^{(\ell)})$ and $\#(a_i^{(k)}, v^{(\ell)})$ for all (i, k, ℓ) .

THEOREM 3.1. *The protocol for naive Bayes learning presented in Figure 3 protects each customer's sensitive data against the miner and up to $n - 2$ corrupted customers.*

Proof: Since all the frequency computations are done independently, the theorem follows immediately from Theorem 2.2. ■

For accuracy, we compare our privacy-preserving protocol with a traditional naive Bayes learning algorithm running on all customers' data without protection of privacy. Suppose that the learning algorithm without privacy protection outputs a classifier c and that our privacy-preserving protocol outputs c' . We claim $c = c'$, which means our protocol does not lose any accuracy as a cost of privacy.

THEOREM 3.2. *The protocol for naive Bayes learning presented in Figure 3 does not lose accuracy.*

Proof: This is straightforward from our protocol specification. Our protocol counts each $\#(v^{(\ell)})$ and each $\#(a_i^{(k)}, v^{(\ell)})$ for $i \notin S$ directly. It uses the privacy-preserving method we presented in Section 2 to count each $\#(a_i^{(k)}, v^{(\ell)})$ for $i \in S$. Since the method in Section 2 computes frequencies precisely, our protocol outputs exactly the same classifier as a non-privacy-preserving naive Bayes algorithm. ■

Overhead Analysis Recall that there are n customers and m attributes and that each (non-class) attribute has a domain of size d , and the class label has a domain of size p . Also recall that the set of privacy-sensitive attributes is S . Assume $s = |S|$ is the number of sensitive attributes. It is easy to see that each customer has a computational overhead—as compared to a non-private solution—of dps ElGamal encryptions. In data mining applications, we usually have $n \gg dps$; thus the computational overhead for each customer is small. The computation overhead for the miner is $O(dpsn)$ modular exponentiations, which is also reasonable. The communication overhead is $dpsn$ ciphertexts.

3.4 Experimental Results of Bayes Classifier Learning

The basic experimental set-up is the same here as described in Section 2.5. The Bayes classifier learning algorithm is implemented in C, and uses the frequency mining implementation as a subroutine. Again, we ran a series of experiments on a PC with a 1GHz processor and 512MB memory under NetBSD, using 512 bit cryptographic keys. For the Bayes classifier

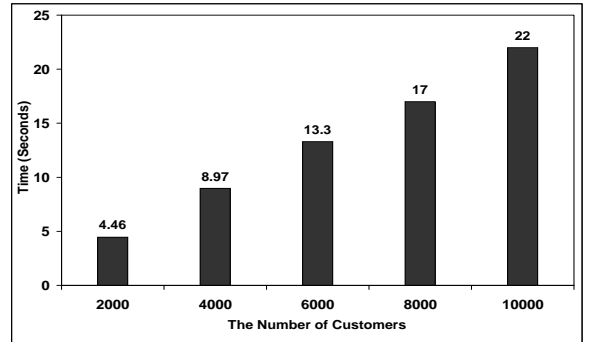


Figure 4: Server's Learning Time for Naive Bayes Classifier vs. Number of Customers

experiments, we assumed that each customer has ten attributes, that each attribute has eight nominal values, and that there are two classes. We measured the

computation time of each customer and the miner in our privacy-preserving protocol for Bayes classifier learning. Our results show that each customer needs only 120 milliseconds to compute her message flow to the miner. Figure 4 shows the computation times the miner needs to learn a naive Bayes classifier for different numbers of customers. For example, when the number of customers is 10,000, the miner’s computation requires only 22 seconds.

Figure 5 further studies how the server’s learning time changes when both the customer number and the attribute number vary. In this experiment, we fix the domain size of each non-class attribute to four and the domain size of the class attribute to two.

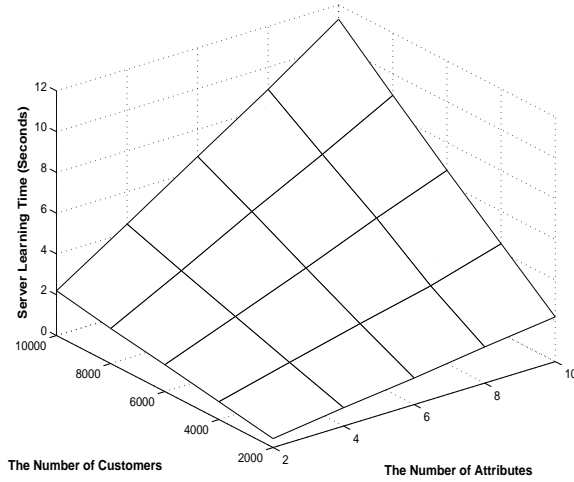


Figure 5: Server’s Learning Time for Naive Bayes Classifier vs. Number of Customers and Number of Attributes

4 Extension to Other Data Mining Algorithms

We describe how our privacy-preserving frequency computation can be used for other important data mining algorithms in the fully distributed model. In Section 4.1, we describe a privacy-preserving ID3 decision tree learning algorithm. In Section 4.2, we sketch a privacy-preserving association rule mining algorithm. Both algorithms are in the fully distributed model without loss of accuracy. Both of them leak no information about the customer data beyond the computed frequencies. Both can be efficient even if the number of customers (transactions) is very large. However, both require certain parameters (such as the number of attributes) are small, as they require exponential computation in those parameters, as we discuss further below.

4.1 Privacy-Preserving Learning of ID3 Trees

Using our privacy-preserving frequency primitive, we can learn ID3 trees in the fully distributed setting without loss of accuracy. Solutions such as [AS00] can be used in the fully distributed setting, but they lose some accuracy as cost of privacy; solutions such as [LP02] do not lose accuracy for privacy, but they do not work in the fully distributed setting.

The miner’s algorithm has the same complexity as the original ID3 tree algorithm, except for an additional linear overhead factor whose value is determined by the ElGamal key size used. However, the computation time of each customer is exponential on the domain size of her attribute. Therefore, our algorithm is efficiently applicable only if the attribute domains are small.

We define the problem of privacy-preserving learning of ID3 decision trees as *PPLID3*:

DEFINITION 2. *PPLID3: A data miner queries n customers and learns an ID3 tree based on the customers’ responses. The miner should not be able to derive any information about each customer’s data beyond the computed frequencies.*

In the *PPLID3* problem, each customer has a $(m + 1)$ -tuple of data in which there are m non-class attributes and one class attribute. The schema of customer data and the domain of each attribute are assumed to be publicly known. Our solution is in the reduced interaction model: each customer sends only a single message flow to the data miner, and there is no further communication. Hence, if customer i sends E_i to the miner, the *security goal* is that the miner learns nothing (beyond the computed frequencies) about customer i ’s data from E_i .

First, we give a brief review of ID3 decision trees. (See [Mit97] for additional details.) An ID3 tree is a rooted tree containing nodes and edges. Each internal node is a test node and corresponds to an attribute. The edges going out of a node correspond to the possible values of that attribute. The ID3 algorithm works as follows. The tree is constructed top-down in a recursive fashion. At the root, each attribute is tested to determine how well it alone classifies the samples. The “best” attribute is then chosen and the samples are partitioned according to this attribute. The ID3 algorithm is then recursively called for each child of this node, using the corresponding subset of data.

Next, we review how the ID3 algorithm chooses the best attribute for a node. We use the following notation. Let $A = \{A_i \mid 1 \leq i \leq m\}$ be the set of (non-class) attributes, V the class attribute, and T the set of samples (or records). For simplicity, we assume that all attributes have the same domain size d :

$A_i = \{a_i^{(j)} \mid 1 \leq j \leq d\}$. The set of possible values of the class attribute is $V = \{v^{(i)} \mid 1 \leq i \leq p\}$. Let $T(v^{(i)})$ be the set of samples with class $v^{(i)}$. Then the entropy is:

$$H_V(T) = \sum_{i=1}^p -\frac{|T(v^{(i)})|}{|T|} \log \frac{|T(v^{(i)})|}{|T|}.$$

Consider an attribute $A_t = \{a_t^{(j)} \mid 1 \leq j \leq d\}$. The conditional information of T given A_t is:

$$H_V(T \mid A_t) = \sum_{i=1}^d \frac{|T(a_t^{(i)})|}{|T|} H_V(T(a_t^{(i)})).$$

For each attribute A_t , the information gain is defined by:

$$\text{gain}(A_t) = H_V(T) - H_V(T \mid A_t)$$

The chosen attribute A_t is the attribute that can achieve the maximum information gain at each node. Clearly, the problem of choosing the best attribute can be reduced to computing entropies. Accordingly, the *PPLID3* problem can be reduced to a problem in which the miner computes entropies while no sensitive data of customers are revealed to her. Again, we use our

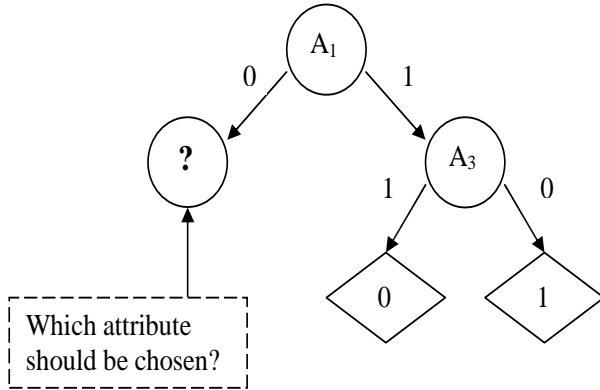


Figure 6: ID3 example

technique of privacy-preserving frequency mining to solve this problem. The solution is showed in Figure 6. Suppose that there are n customers, each holding a record with three Boolean attributes A_1, A_2 and A_3 . The class value is also Boolean. Figure 6 shows an intermediate state of the ID3 algorithm in which the algorithm needs to choose an attribute for the node “?”. To compute the information gain $H_V(A_i)$ (for $i = 2$ and 3 , resp.), we need to compute $H_V(T)$ and $H_V(T \mid A_i)$:

$$H_V(T) = -\frac{|T(V=0)|}{|T|} \log \frac{|T(V=0)|}{|T|} - \frac{|T(V=1)|}{|T|} \log \frac{|T(V=1)|}{|T|},$$

$$H_V(T \mid A_i) = \frac{|T(A_i=0)|}{|T|} H_V(T(A_i=0)) + \frac{|T(A_i=1)|}{|T|} H_V(T(A_i=1))$$

The above formulae involve several frequencies: $|T|$, $|T(V=0)|$, $|T(V=1)|$, $|T(A_i=0)|$, $|T(A_i=1)|$, etc. All these frequencies can be computed using our privacy-preserving frequency mining protocol.

The general protocol is sketched in Figure 7, where A is the attribute set, V the class and T the set of all customers’ data.

ID3 (A, V, T)

1. If A is empty, the miner returns a leaf node with the dominating class value in T .
2. Use the privacy-preserving method to count the number of records with each class label. If T consists of records which have the same class label v , return a leaf node with v .
3. Otherwise:
 - (a) Determine the best attribute A_i for T using the privacy-preserving method.
 - (b) For $A_i = \{a_1, \dots, a_d\}$, let $T(a_1), \dots, T(a_d)$ be a partition of T s.t. every record in $T(a_j)$ has attribute value a_j .
 - (c) Return a tree whose root is labeled A_i ; the root has outgoing edges labeled a_1, \dots, a_d s.t. each edge a_j goes to the tree $\text{ID3}(A - A_i, V, T(a_j))$.

Figure 7: Privacy-preserving Protocol for Learning ID3 Tree

Unlike naive Bayes classification, which assumes independence between non-class attributes given the class label, attributes are interdependent with ID3. If the class attribute has p class labels, and each of the m non-class attributes has a domain of size d , then the number of joint frequencies that need to be counted is exponential in m . In some cases we have small m and d and thus we can still achieve reasonable overhead. For example, the data set of Car Evaluation Database from UCI repository [BM98] has six nominal attributes: buying, maint, doors, persons, lug_boot and safety, and the class attribute has a domain of size four. For such

a scenario, we estimate that each customer needs only one minute to compute her message flow to the miner. Another example [Mit97, Ch. 3] is a weather data set containing four data attributes: **outlook**, **temperature**, **humidity**, and **windy**, and a class attribute, **play**. If each customer holds one weather record, we estimate that each customer needs only about 0.5 seconds to compute her message flow to the miner.

4.2 Association Rule Mining The mining of association rules can be formulated as follows. Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items. Let D denote a set of n transactions (or records), where each transaction T is a subset of D . Associated with each transaction is a unique identifier. We say that a transaction T contains P if $P \subset T$. An association rule is an implication of the form $P \Rightarrow Q$, where $P \subset I$, $Q \subset I$, and $P \cap Q = \emptyset$. The rule $P \Rightarrow Q$ has *support* s in D if $s\%$ of the transactions in D contain $P \cap Q$. $P \Rightarrow Q$ holds with c *confidence* if $c\%$ transactions in D that contain P also contain Q .

Association rule mining can be reduced to computing the frequency of a number of particular sets. Using the privacy-preserving frequency mining protocol in Section 2, we enable the miner to compute the frequency of any set of items from customer response. From these, the miner can learn all association rules. Note that this leaks *all* the computed frequencies to the miner, rather than only revealing the actual frequent itemsets.

In the fully distributed setting, each customer has a transaction. The miner wants to learn the association rules based the customer data, but each customer does not want to reveal her data to the miner. Consider a small example in which $P = \{p_1, p_2, p_3\}$ and $Q = \{q_1, q_2\}$. We describe how to compute confidence and support for the rule $P \Rightarrow Q$. Each customer sends the encryptions of the occurrences of $\{p_1, p_2, p_3, q_1, q_2\}$, $\{p_1, p_2, p_3\}$ and $\{q_1, q_2\}$. By applying the privacy-preserving frequency-mining protocol, the miner can easily learn the confidence and support from customers' output. To learn all association rules with threshold of c and s on D , each customer must compute all possible combinations of the occurrence of sets of items, which is of course exponential on the size m of the item domain. Hence, the solution is only practical if $m \ll n$.

5 Conclusion

In this paper, we proposed a privacy-preserving method of frequency mining and applied it to naive Bayes learning in a fully distributed setting. If this problem is solved using randomization techniques, then there is a trade-off between privacy and accuracy. However, our proposed solution enjoys cryptographically strong privacy without losing any accuracy as cost of privacy.

Furthermore, both theoretical analysis and experimental results show that the method itself and its application to naive Bayes learning are very efficient. Our work assumes that the data are horizontally partitioned such that each customer holds a row. Therefore, an immediate open question is whether a solution similar to ours can be found for the case in which the data are fully vertically partitioned among different parties.

We also discussed other possible applications of the proposed privacy-preserving frequency mining: privacy-preserving learning of ID3 trees and association rule mining. These are very practical in some cases, but rely on certain parameters being small. A second open question is whether those applications can be made as efficient as our privacy-preserving protocol for naive Bayes learning. We conjecture that additional techniques are needed to make such protocols efficient.

A third open question is whether it is possible to combine our method with randomization techniques to further improve efficiency. For example, is there a protocol for frequency mining or naive Bayes learning that is more efficient than the one presented in this paper, but still enjoys full privacy and does not lose any accuracy? Alternately, it may be possible to combine some randomization techniques with some cryptographic techniques to further improve the efficiency while the resulting loss of privacy can be quantified and thus limited to an acceptable extent. We believe these questions are worth further investigation.

Acknowledgement

We thank the anonymous reviewers for their insightful comments.

References

- [AA01] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255. ACM Press, 2001.
- [AJL04] A. Ambainis, M. Jakobsson, and H. Lipmaa. Cryptographic randomized response techniques. In *Proc. of the 2004 International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, pages 425–438. Springer, 2004.
- [AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
- [AW89] N. Adam and J. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

- [BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.
- [BM98] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.
- [BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. of the 26th Annual ACM symposium on Theory of Computing*, pages 544–553. ACM Press, 1994.
- [Cra99] L. Cranor, editor. *Comm. ACM* 42(2), *Special Issue on Internet Privacy*, 1999.
- [DN03] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210. ACM Press, 2003.
- [DN04] C. Dwork and K. Nissim. Privacy-preserving data-mining on vertically partitioned databases. In *Advances in Cryptology - Proceedings of CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, Santa Barbara, California, August 2004.
- [DP97] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.
- [DZ03] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505–510. ACM Press, 2003.
- [EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222. ACM Press, 2003.
- [ESAG02] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228. ACM Press, 2002.
- [GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Computer and System Sciences*, 28:270–299, 1984.
- [GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th Annual ACM Conference on Theory of Computing*, pages 218–229. ACM Press, 1987.
- [Gol04] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.
- [HS00] Martin Hirt and Kazuo Sako. Efficient receipt-free voting based on homomorphic encryption. *Lecture Notes in Computer Science*, 1807:539+, 2000.
- [KC02] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD’02)*, pages 24–31, June 2002.
- [KDWS03] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *The Third IEEE International Conference on Data Mining*, 2003.
- [KV02] M. Kantarcioglu and J. Vaidya. An architecture for privacy-preserving mining of client information. In *IEEE ICDM Workshop on Privacy, Security and Data Mining*, pages 37–42, 2002.
- [KV03] M. Kantarcioglu and J. Vaidya. Privacy preserving naive Bayes classifier for horizontally partitioned data. In *IEEE Workshop on Privacy Preserving Data Mining*, 2003.
- [LP02] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.
- [Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [RH02] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proc. of the 28th VLDB Conference*, 2002.
- [Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, second edition, 1996.
- [SWY04] H. Subramaniam, R. N. Wright, and Z. Yang. Experimental analysis of privacy-preserving statistics computation. In *Proc. of the VLDB Workshop on Secure Data Management*, pages 55–66, August 2004.
- [TY98] Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In *Public Key Cryptography’98*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134, 1998.
- [VC02] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM Press, 2002.
- [VC03] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proc. of the Ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM Press, 2003.
- [VC04] J. Vaidya and C. Clifton. Privacy preserving naive Bayes classifier on vertically partitioned data. In *2004 SIAM International Conference on Data Mining*, 2004.
- [War65] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
- [Wes99] A. Westin. Freebies and privacy: What net users think. Technical report, Opinion Research Corporation, 1999.
- [WY04] R. N. Wright and Z. Yang. Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–718. ACM Press, 2004.
- [Yao86] A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

Privacy-Aware Market Basket Data Set Generation: A Feasible Approach for Inverse Frequent Set Mining

Xintao Wu Ying Wu Yongge Wang
UNC at Charlotte
9201 Univ. City Blvd
Charlotte, NC 28223
{xwu,ywu,yonwang}@uncc.edu

Yingjiu Li
Singapore Management University
469 Bukit Timah Road
Singapore 259756
yjli@smu.edu.sg

Abstract

Association rule mining has received a lot of attention in the data mining community and several algorithms were proposed to improve the performance of association rule or frequent itemset mining. The IBM Almaden synthetic data generator has been commonly used for performance evaluation. One recent work shows that the data generated is not good enough for benchmarking as it has very different characteristics from real-world data sets. Hence there is a great need to use real-world data sets as benchmarks. However, organizations hesitate to provide their data due to privacy concerns. Recent work on privacy preserving association rule mining addresses this issue by modifying real data sets to hide sensitive or private rules. However, modifying individual values in real data may impact on other, non-sensitive rules. In this paper, we propose a feasible solution to the NP-complete problem of inverse frequent set mining. Since solving this problem by linear programming techniques is very computationally prohibitive, we apply graph-theoretical results to divide the original itemsets into components that preserve maximum likelihood estimation. We then use iterative proportional fitting method to each component. The technique is experimentally evaluated with two real data sets and one synthetic data set. The results show that our approach is effective and efficient for reconstructing market basket data set from a given set of frequent itemsets while preserving sensitive information.

1 Introduction

Since its introduction in [1], association rule mining has received a lot of attention in the data mining community. Several algorithms were proposed to improve performance of association rule or frequent itemset mining. The algorithms developed typically only show the performance advantage using synthetic data sets (e.g., the market basket data generator provided by IBM Almaden). Recently, Zheng et al. compared five well-known association rule algorithms using three real-world

data sets and the artificial data set from IBM Almaden [22]. One interesting result is that the artificial data sets from IBM Almaden have very different characteristics from the real-world data sets and hence there is a great need to use real-world data sets as benchmarks.

However, organizations hesitate to provide their real-world data sets as benchmarks due to the potential disclosure of private information or commercial secret. Some recent work [3, 6, 9, 10, 14, 16, 10, 17] on privacy preserving association rule mining considers how much information can be inferred or computed from large data made available through data mining algorithms and looks for ways to minimize the leakage of information. Most ideas are to modify the given database (e.g., randomization, hiding, replacing individual values with unknowns) so that the support of a given set of sensitive rules decreases below some minimum support thresholds. The problem here is modifying the individual values of original databases to limit the disclosure of sensitive rules may impact on other, non-sensitive frequent itemsets.

The frequent sets and their supports (defined as the number of transactions in the database that contain the items) can be considered to be a reasonable summary of the original data set. A natural problem is inverse frequent set mining, i.e., to find a binary data set that is compatible with frequent set mining results. The inverse frequent set mining is related to the questions of how well privacy is preserved in the frequent sets and how well the frequent sets characterize the original data set. The authors, in [13, 5], investigate the problem whether there exists a data set that is consistent with the given frequent itemsets and frequencies and show this problem is NP-Complete.

In this paper, we propose a feasible approach on inverse frequent itemset mining. Here the given frequent itemsets and their frequencies were discovered from the original one, hence there must exist at least one compat-

ible data set. Our focus here is how to generate one effectively and efficiently. The frequency of each frequent itemset is taken as a constraint over the original data set. The problem of inverse frequent set mining then can be translated to a linear constraint problem. Linear programming problems can be commonly solved today in hundreds or thousands of variables and constraints. However, the number of variables and constraints in our scenario is far beyond hundreds or thousands (e.g., 2^d , where d is the number of items). Hence it is impractical to apply linear programming techniques directly. In this paper, we propose a feasible approach using graphical decomposition techniques to reconstruct market basket data sets given a set of frequent itemsets and their supports.

The constructed market basket data set can be used as benchmarks for evaluating various different frequent itemset mining algorithms. Ramesh et al. recently investigated the relation between the distribution of discovered frequent set and the performance of association rule mining [15]. It suggests that the performance of association rule mining method using the original data set should be very similar to that using the synthetic one compatible with the same frequent set mining results.

Furthermore, our approach could achieve better privacy preserving as it only needs to access the summary information (i.e., frequencies of given frequent itemsets) instead of the individual values of the original data set. If frequencies of some frequent itemsets are considered confidential information by user, they can be removed directly from the released list. However, though those frequencies of the released frequent itemsets are not confidential themselves, attackers may still be able to derive some confidential or private information (e.g., frequencies of some confidential or sensitive itemsets) from the released information. We will also address this issue by using the *Frechet Bounds* to analyze the potential disclosure of our approach.

The rest of the paper is organized as follows. In Section 2 we review the related work. In Section 3 we formalize the problems and in Section 4 present our method in detail. Experimental results are discussed in Section 5. In Section 6 we draw conclusions and describe directions for future work.

2 Related Work

Privacy preserving data mining considers how much information can be inferred or computed from large data made available through data mining algorithms and looks for ways to minimize the leakage of information. Recent research has been focused on the information leakage in association rule mining. In [3, 6], the authors considered the problem of limiting disclosure of sensitive

rules, aiming at selectively hiding some frequent itemsets from large databases with as little impact on other, non-sensitive frequent itemsets as possible. The idea was to modify a given database so that the support of a given set of sensitive rules decreases below the minimum support value. Similarly, the authors in [17] presented a method for selectively replacing individual values with unknowns from a database to prevent the discovery of a set of rules, while minimizing the side effects on non-sensitive rules. The authors studied the impact of hiding strategies in the original data set by quantifying how much information is preserved after sanitizing a data set [14]. [16, 10] studied the problem of mining association rules from transactions in which the data has been randomized to preserve privacy of individual transactions. One problem is it may introduce some false association rules.

The authors, in [15], proposed a method to generate market basket data set for benchmarking when the length distributions of frequent and maximal frequent itemset collections are available. Though the generated synthetic data set preserves the length distributions of frequent patterns, one serious limitation is that the size of transactions generated is much larger than that of original database while the number of items generated is much smaller. We believe the sizes of items and transactions are two important parameters as they may significantly affect the performance of association rule mining algorithms. In our paper, we will apply graphical decomposition techniques to estimate the cell frequencies of the contingency table which is then used to construct market basket data set. Graphical modeling and decomposition techniques are well studied in statistics [12, 19] and were recently used for screening and interpreting multi-item associations in [20].

A number of researchers from statistics field have recently been working on the problem of determining upper and lower bounds on the cells of the cross-classification given a set of margins [8, 7]. Upper and lower bounds induced by some fixed set of marginal on the cell entries of a contingency table are of great importance in measuring the disclosure risk associated with the release of these marginal totals.

Wu et al. have proposed a general framework for privacy preserving database application testing by generating synthetic data sets based on some a-priori knowledge about the production databases [21]. The general a-priori knowledge such as statistics and rules can also be taken as constraints of the underlying data records. The problem investigated in this paper can be thought as a simplified problem where data set here is binary one and constraints are frequencies of given frequent itemsets. However, the techniques developed

in [21] are infeasible here as the number of items are much larger than the number of attributes in general data sets.

3 Problem Formulation

For market basket data, we define each transaction, such as list of items purchased, as a subset of all possible items.

DEFINITION 3.1. Let $\mathcal{I} = \{I_1, \dots, I_d\}$ be a set of d boolean variables called items. Then a set of transactions $\mathcal{T} = \{t_1, \dots, t_N\}$ is a collection of N d -tuples from $\{\text{True}, \text{False}\}^d$ which represent a collection of value assignments to the d items. The support of an itemset s over \mathcal{I} , denoted $\text{support}(s, \mathcal{T})$, is defined as the number of the transactions that contain s . The frequency of an itemset s , denoted $\text{freq}(s, \mathcal{T})$, is defined as $\text{support}(s, \mathcal{T})/N$.

The well known frequent itemset mining problem is defined as:

PROBLEM 1. Frequent Itemsets Mining. Given a transaction database \mathcal{T} over \mathcal{I} and a threshold $\tau \in [0, 1]$, find all frequent itemsets FS such that $\text{freq}(FS, \mathcal{T}) \geq \tau$.

In our paper, we focus on the inverse frequent itemset mining defined as:

PROBLEM 2. Inverse Frequent Itemsets Mining. Given a finite set $\mathcal{FS} = \{FS_1, \dots, FS_n\}$ together with their frequencies $\{\text{freq}(FS_1), \dots, \text{freq}(FS_n)\}$ discovered from the original database D , construct a transaction database \hat{D} such that

- 1) \hat{D} satisfies \mathcal{FS} and their frequencies and
- 2) D and \hat{D} are over the same set \mathcal{I} and have the same number of transactions.

DEFINITION 3.2. We define CT_s as a k -dimensional contingency table for itemset $s \subseteq \mathcal{I}$ where $|s| = k$. Each dimension has two values (True or False) and each cell contains the number (or the frequency) of transactions located in that cell.

The market basket data set in Definition 3.1 can be easily transformed into one (and only) d -dimensional contingency table, and vice versa. Table 1 shows a market basket data set with 9 transactions and three items and its 3-dimensional contingency table. Note the contingency table shown in Table 1(c) is equivalent to that shown in Table 1(b) when the number of transactions is fixed. The $\text{cell}(ABC)$ contains the number (or frequency) of those transactions which buy item ABC together (i.e., t8 and t9) while $\text{cell}(\bar{A}BC)$

contains the number (or frequency) of those transactions which buy BC but do not buy A (i.e., t6). Hence we can construct its contingency table by one scan of the original basket data set if the contingency table is fitted in memory. On the other hand, we can construct exactly one market basket data set by scanning each cell value of the contingency table¹. For example, from $\text{cell}(ABC) = 2$, we generate two transactions which contain item AB but not C.

As market basket data set is equivalent to its contingency table, Problem 2 can be mapped to the contingency table reconstruction problem, i.e., construct a feasible contingency table CT_I which satisfies the given marginal frequencies $\{\text{freq}(FS_1), \dots, \text{freq}(FS_n)\}$. Here each given frequent itemset and its frequency can be taken as a constraint or a marginal as shown in Example 1.

EXAMPLE 1. For the data set shown in Table 1, the frequent itemsets are A, B, C, AB, AC and their frequencies are 6/9, 7/9, 5/9, 4/9, 4/9 respectively when support threshold is set as 4/9. Each frequency can be taken as a constraint:

$$\begin{aligned}
 \text{freq}(ABC) + \text{freq}(\bar{A}\bar{B}\bar{C}) + \text{freq}(AB\bar{C}) + \text{freq}(\bar{A}BC) &= \frac{6}{9} \\
 \text{freq}(ABC) + \text{freq}(\bar{A}BC) + \text{freq}(AB\bar{C}) + \text{freq}(\bar{A}\bar{B}\bar{C}) &= \frac{7}{9} \\
 \text{freq}(ABC) + \text{freq}(\bar{A}\bar{B}C) + \text{freq}(\bar{A}BC) + \text{freq}(\bar{A}\bar{B}\bar{C}) &= \frac{5}{9} \\
 \text{freq}(ABC) + \text{freq}(AB\bar{C}) &= \frac{4}{9} \\
 \text{freq}(ABC) + \text{freq}(\bar{A}BC) &= \frac{4}{9}
 \end{aligned}
 \tag{3.1}$$

It is straightforward to see we can apply linear programming techniques to compute frequencies of cells (e.g., $\text{freq}(ABC)$, $\text{freq}(\bar{A}\bar{B}\bar{C})$, etc.) and then generate market basket data using frequencies computed. However, the number of variables and constraints in our scenario is too large to be handled by the current linear programming packages.

4 Our Method

Figure 1 illustrates our method. Generally it involves grouping frequent itemsets into disjoint clusters, decomposing each cluster into components that are mutually independent, deriving or computing the cell frequency of

¹Here we assume transactions in market basket data are unordered.

Table 1: An example of data set with three items

TID	List of items
t1	A,B
t2	B
t3	B
t4	A,B
t5	A,C
t6	B,C
t7	A,C
t8	A,B,C
t9	A,B,C

(a) market basket data

	\tilde{B}		B	
	\tilde{A}	A	\tilde{A}	A
\tilde{C}	0	0	2	2
C	0	2	1	2

(b) contingency table

	\tilde{B}		B	
	\tilde{A}	A	\tilde{A}	A
\tilde{C}	0	0	2/9	2/9
C	0	2/9	1/9	2/9

(c) contingency table with cell value as frequency

the contingency table built from each component, computing the cell frequency of the contingency table for each cluster, and generating transactions by merging the contingency table of each cluster.

First we decompose d items into m disjoint item clusters $\mathcal{S} = \{s_1, \dots, s_m\}$, where $|s_i| = k_i$, $s_i \cap s_j = \emptyset$ and $\bigcup_{i=1, \dots, m} s_i = \mathcal{I}$, according to the given frequency sets \mathcal{FS} (line 2). it works as follows. We take each item as one node and add an edge between two nodes if both nodes are contained in some frequent itemset. After processing all frequent itemsets, each disconnected subgraph forms one cluster. For example, if the released frequency sets are $\{AB\}$, $\{BC\}$, $\{DE\}$, $\{BF\}$ for 6 items, we merge to get two clusters $\{ABCF\}$ and $\{DE\}$. As item clusters here are exclusive disjoint, we can generate \hat{D}_s independently (line 3-9) and join them to get \hat{D} finally (line 10). Here each \hat{D}_s is a vertical partition where each row contains those items in s .

If the frequency of item cluster s is already contained in the given frequency set, we can generate contingency table directly without further processing (line 4 and 5). For example, assume $s = ABC$ and $freq(ABC)$ is known, then $freq(AB)$ and $freq(AC)$ must also be available since the frequency of any set $s' \subset s$ is available when the frequency s is available. Hence we can compute $f_q(AB\bar{C}) = freq(AB) - freq(ABC)$, $f_q(A\bar{B}C) = freq(AC) - freq(ABC)$ and so on. If s is not contained in the given frequency sets, we apply a divide and conquer approach and decompose s into components recursively (line 11-21). During this process, we apply graphical decomposition techniques to decompose the independence graph G_s into subgraphs while keeping the maximum likelihood estimation unchanged. We leave the discussion of this part in Section 4.2. When the component can not be further decom-

posed and it is not included in the frequent itemsets, we apply Iterative Proportional Fitting (IPF) method to estimate its contingency table. We present how to use IPF to generate contingency table in Section 4.1.

4.1 Iterative Proportional Fitting In this section we briefly review the iterative proportional fitting method. The IPF has been well studied in the statistical literature. It is an iterative algorithm that converges to the maximum likelihood estimation. In its simplest form, the algorithm provides a method of adjusting one two-way contingency table to conform to the margins. It begins by scaling the rows of the first table to have the correct row margins, then it scales the resulting table to have the correct column margins, then it scales the resulting table to have the correct row margins, and so on, iterating through the cycle of rows and columns, until convergence is reached.

In our scenario, the underlying contingency table is d dimensions (d is the number of items contained in s). IPF starts the reconstruction by initializing each cell c with frequency $1/2^d$ and computes its marginals specified by each constraint. At each following iteration, IPF loops over all constraints and all cells c involved in each constraint, and adjusts the values of the cells according to the formula:

$$\hat{f}_q(c)^{(t+1)} = \hat{f}_q(c)^{(t)} \frac{freq(FS_i)}{\hat{freq}^{(t)}(FS_i)}$$

here we denote as $\hat{f}_q(c)^{(t)}$ the estimate of the value of cell c during the t -th iteration of the algorithm and denote as $\hat{freq}^{(t)}(FS_i)$ the frequency of marginal cell FS_i which is computed from estimated values of cells $(\hat{f}_q(c)^{(t)})$ in the t -th iteration of the algorithm.

```

InvseFS(FS, freq(FS), d, N)
input  FS, a given set of frequent itemsets
        freq(FS), frequencies of given FS
        d, number of items
        N, number of transactions
output  $\hat{D}$ , generated data set
BEGIN
1    $\hat{D} = \phi$ 
2    $\mathcal{S} = \text{GenItemCluster}(\text{FS})$ 
3   For each cluster  $s \in \mathcal{S}$  do
4       If  $s \in \mathcal{FS}$ 
5            $CT_s = \text{GenContingencyTable}(s)$ 
6       Else
7            $G_s = \text{GenIndependenceGraph}(s)$ 
8            $CT_s = \text{Decompose}(s, G_s)$ 
9       Generate data set  $\hat{D}_s$  from  $CT_s$ 
10   $\hat{D} = \hat{D} \cup \hat{D}_s$ 
END

Decompose ( $s, G_s$ )
input     $s$ , a given itemset
           $G_s$ , independence graph of  $s$ 
output    $CT_s$ , contingency tables of  $s$ 
BEGIN
11  If  $s \in \mathcal{FS}$ 
12       $CT_s = \text{GenContingencyTable}(s)$ 
13  Else
14      If  $s$  is irreducible
15           $CT_s = \text{IPF}(s)$ 
16      Else
17          there exists a partition of  $G_s$ 
            into  $(G_{s_a}, G_{s_b}, G_{s_c})$  such that
             $s_a \perp s_b \mid s_c$  and  $G_{s_c}$  is a clique
18           $\text{Decompose}(s_{a \cup c}, G_{s_{a \cup c}})$ 
19           $\text{Decompose}(s_{b \cup c}, G_{s_{b \cup c}})$ 
20           $\text{Decompose}(s_c, G_{s_c})$ 
21           $CT_s = \frac{CT_{s_a \cup s_c} \times CT_{s_b \cup s_c}}{CT_{s_c}}$ 
END

```

Figure 1: Inverse Frequent Set Generation Algorithm

The estimates converge in a monotonically decreasing fashion, and we commonly choose to terminate the iterations when the change in each cell estimate becomes smaller than some user specified threshold value.

As we stated in the introduction, we cannot apply IPF over the very large contingency table that contains many items. Usually the contingency table with many items tends to be sparse, which affects the accuracy of IPF method. Besides, even if the contingency table is dense, the complexity of IPF algorithm is generally iterative oriented and thus computationally expensive for large tables. In next section, we discuss how to decompose into subsets and apply IPF only on those irreducible components without losing any significant information.

4.2 Graphical Decomposition The graphical decomposition involves two steps: 1) building one *independence graph* for each cluster; 2) applying graph-theoretical results to decompose the graph into irreducible components.

4.2.1 Building Independence Graph from Frequent Itemsets The independence graph is defined by making every vertex of the graph correspond to a discrete random variable, and the edges denoting the dependency of the two variables linked. A missing edge in the graph represents the conditional independence of the two variables associated with the two vertices. To build the independence graph, we need to test conditional independence for every pair of variables, controlling for the other variables in the same cluster. There are several approaches to test conditional independence (See [2]). In our paper, we build the independence graph by applying the Cochran-Mantel-Haenszel test.

Here we first assume the contingency table of s is known and describe how the Cochran-Mantel-Haenszel test works. For any pair of two items I_i, I_j from item set $s \subseteq I$ ($|s| = k$), we derive one partial 2×2 contingency table (stratum) for each possible value from set $s \setminus \{I_i, I_j\}$. Hence we can have L ($L = 2^{k-2}$) strata. For each stratum l , we need to compute the marginal totals $\{n_{\cdot 0}^{(l)}, n_{\cdot 1}^{(l)}, n_{0 \cdot}^{(l)}, n_{1 \cdot}^{(l)}\}$, where a dot “.” denotes a sum along that dimension (e.g., $n_{\cdot 0}^{(l)} = n_{00}^{(l)} + n_{10}^{(l)}$). Table 2(a) shows the stratum form for item variable A and B while Table 2(b) shows one stratum ($C = 1$). Equation 4.2 shows the summary statistics where $m_{11}^{(l)}$ and $V(n_{11}^{(l)})$ is mean and variance respectively.

$$m_{11}^{(l)} = E(n_{11}^{(l)}) = \frac{n_{1 \cdot}^{(l)} n_{\cdot 1}^{(l)}}{n_{\cdot \cdot}^{(l)}}$$

Table 2: A 2×2 contingency table for variable A and B

	B	\tilde{B}	
A	n_{11}	n_{10}	$n_{1.}$
\tilde{A}	n_{01}	n_{00}	$n_{0.}$
	$n_{.1}$	$n_{.0}$	$n_{..}$

(a) stratum form

	B	\tilde{B}	
A	2	2	4
\tilde{A}	1	0	1
	3	2	5

(b) example of
one stratum
C=1

$$V(n_{11}^{(l)}) = \frac{n_{1.}^{(l)} n_{0.}^{(l)} n_{.1}^{(l)} n_{.0}^{(l)}}{n_{..}^{(l)} n_{..}^{(l)} (n_{..}^{(l)} - 1)}$$

$$(4.2) \quad M^2 = \frac{(|\sum n_{11}^{(l)} - \sum m_{11}^{(l)}| - 0.5)^2}{\sum V(n_{11}^{(l)})}$$

The summary statistics M^2 has approximately a chi-squared distribution with d.f. = 1 under the null hypothesis of conditional independence. Hence, if $M^2 > P_\alpha$, we can reject the null hypothesis of conditional independence and include the edge of I_i and I_j in the interaction graph.

However, the contingency table of s is unknown in our scenario. We apply a modified summary statistics version. From frequency itemsets FS , extract $FS^s = \{FS_{i_1}, \dots, FS_{i_k}\}$ such that $s \subset FS_{i_k}$. For each FS_{i_k} , we build its stratum and compute its summary statistics. If all summary statistics are larger than P_α , we can reject the null hypothesis of conditional independence and include the edge of I_i and I_j in the interaction graph.

4.2.2 Decomposing Independence Graph

THEOREM 4.1. *Graph Factorization ([12, 19]). If there exists a decomposition of independence graph G_s into $(G_{s_a}, G_{s_b}, G_{s_c})$ such that 1) $s_a \perp s_b \mid s_c$ (i.e., the variables in S_a are conditionally independent of those in S_b given the variables in S_c) and neither G_{s_a} nor G_{s_b} empty; and 2) the subgraph G_{s_c} is a clique², the joint density of s admits the factorization*

$$f_s = \frac{f_{s_a \cup s_c} f_{s_b \cup s_c}}{f_{s_c}}$$

²A clique is a subset of vertices which induce a complete subgraph for which the addition of any further vertex renders the induced subgraph incomplete. A graph is complete if all vertices are joined with undirected edges. In other words, the clique is maximally complete.

The theory may be interpreted by the following way: if two disjoint subsets of vertices s_a and s_b are separated by a subset s_c in the sense that all paths from s_a to s_b go through s_c , then the variables in s_a are conditionally independent of those in s_b given the variables in s_c . In other words, the maximum likelihood estimations (MLEs) for the parameters of the model can easily be derived by combining the estimates of the models on the lower dimensional tables represented by the simpler subgraphs. The subgraphs $G_{s_a \cup s_c}$ and $G_{s_b \cup s_c}$ may be further decomposed into subgraphs recursively until the graph is decomposed into basic, irreducible components. Hence, applying a divide-and-conquer approach based on the decompositions will make the procedure applicable to much larger tables.

EXAMPLE 2. *To clarify the concepts and the results presented so far, we use an example shown in Figure 2. The graph in Figure 2(a) has 7 vertices and 11 edges. The edge $\{A, C\}$ is a separator for $\{A, C, G\}$ and $\{A, B, C, D, E, F\}$. The former is a triangle, hence cannot be further decomposed. Similarly, $\{B, D\}$ separates $\{A, B, C, D\}$ and $\{B, D, E, F\}$. Both are a prime subgraph, therefore we have finished the decomposition. From Theorem 4.1, we have*

$$f_{ABCDEFGF} = \frac{f_{ACG} f_{ABCDEF}}{f_{AC}} = \frac{f_{ACG} f_{ABCD} f_{BDEF}}{f_{AC} f_{BD}}$$

. The cell values in the original 7-dimensional contingency table (i.e., $ABCDEFGF$) can be computed from the low dimensional contingency tables explicitly.

In each step, we search the clique separators of a graph. If there is no clique to be found, it means we get irreducible components and we apply IPF to estimate. Please note if the frequency of any component is already contained in the given frequency set, we can generate its contingency table directly without further decomposition (line 11-12 in Figure 1). An algorithm with a complexity of $O(ne + n^2)$ to find the clique separators of a graph or to find the vertex-sets of the irreducible components of the graphs was presented in [18], where n is the number of vertices and e is the number of edges. We have implemented the decomposition algorithm in our system. Note that decomposition step is determined by the size of independence graph (i.e., the number of variables n and the number of edges e). Our implementation is able to handle clusters with larger number of variables³.

³A widely used graph decomposition package is CoCo [4]. However, it can not decompose a graph with more than 128 variables.

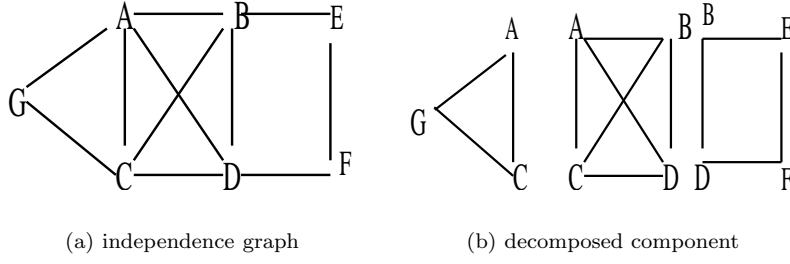


Figure 2: Composition of independence graph

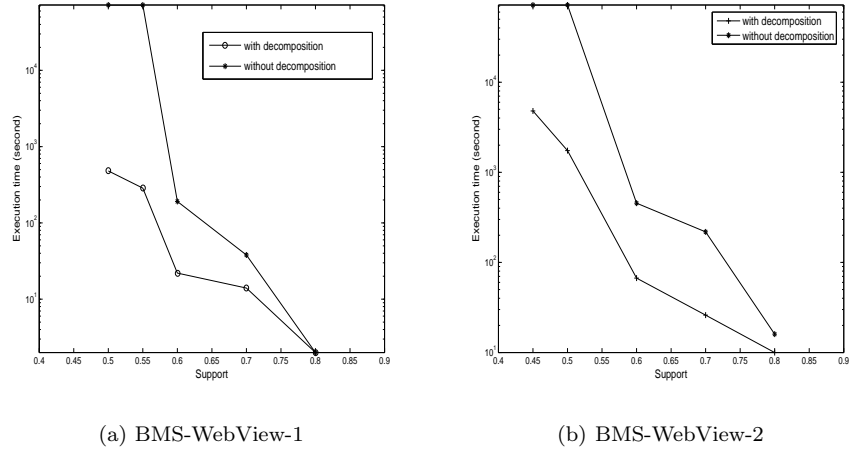


Figure 3: Performance vs. Varying Supports over BMS-WebView1 and BMS-WebView2

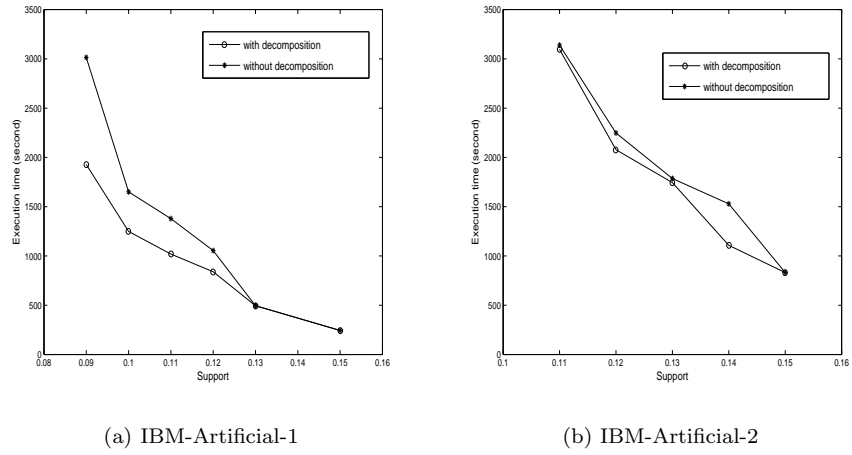


Figure 4: Performance vs. Varying Supports over two IBM Artificial Datasets

4.3 Privacy Aware Generation We have presented how our approach can effectively generate contingency table given the marginal frequencies. It is no wonder there may exist many data sets which satisfy the given frequent itemsets. IPF method itself generates a data set which converges to the maximum likelihood estimation. All the information contained in the generated synthetic data set is from the frequencies of the released frequent itemsets. Though the released frequent itemsets are not considered as confidential information, attackers may still be able to derive or estimate frequencies of some confidential sets.

For example, in the table which records the number of patients visiting physicians to receive treatments, the frequencies on Patient-Doctor and Doctor-Treatment are not sensitive and can be released. However, the Patient-Treatment information is sensitive, and so, confidential. Disclosure detection addresses to what extent attackers can infer cell entry values for the Patient-Treatment table. This entailment of frequent itemsets problem is defined as:

PROBLEM 3. Entailment of Frequent Itemsets. Given a finite set $\mathcal{FS} = \{FS_1, \dots, FS_n\}$ together with their frequencies $\{\text{freq}(FS_1), \dots, \text{freq}(FS_n)\}$ discovered from the original database D , plus a target private itemset ps with its original frequency $\text{freq}(ps)$, compute the bound of $\text{freq}(ps)$ entailed from \mathcal{FS} .

If the induced upper and lower bounds are too tight or too close to the actual sensitive value in a cell entry, the information associated with the transactions classified in that cell may be disclosed. The bound or feasibility interval can be obtained by solving the corresponding linear programming problem generally. The problem of determining sharp upper and lower bounds for the cell entries subject to some linear constraints expressed in this form is known to be NP-hard. Recently, Dobra and Fienberg have developed generalized Frechet Bounds for reducible log-linear models with any number of minimal sufficient statistics and have shown the upper and lower bounds can be expressed as explicit functions of marginal totals.

THEOREM 4.2. Generalized Frechet Bounds for Reducible Models⁴ [8]. Assume that the released set of marginals is the set of minimum sufficient statistics of a reducible loglinear model. Then the upper bounds for the cell entries in the initial table are the minimum of upper bounds of relevant components, while the lower

bounds are the maximum of zero, or sum of the lower bounds of relevant components minus the separators.

EXAMPLE 3. Using the example shown in Example 2, we have

$$f_{ABCDEFGF} = \frac{f_{ACG}f_{ABCDEF}}{f_{AC}} = \frac{f_{ACG}f_{ABCD}f_{BDEF}}{f_{AC}f_{BD}}$$

where we have three components $\{ACG\}$, $\{ABCD\}$, $\{BDEF\}$ and two separators $\{AC\}$, $\{BD\}$. Using Theorem 4.2, we can compute the bounds of cell values in the original 7-dimensional contingency table (i.e., $ABCDEFGF$) from the low dimensional contingency tables explicitly. If $\{ACG\}$, $\{ABCD\}$ and $\{BDEF\}$ are contained in the released frequency itemsets, we see the upper bounds for the cell entries in $\{ABCDEFGF\}$ induced by $\{ACG\}$, $\{ABCD\}$ and $\{BDEF\}$ are the minimum of the corresponding entries in the fixed marginals, while the lower bounds are the sum of the same entries minus the sum of corresponding entries in the marginals associated with the separators, i.e., $\{AC\}$ and $\{BD\}$. If not all components are contained in the released frequency itemsets, e.g., $\{ABCD\}$ is not contained but its subsets $\{AB\}$, $\{AC\}$, $\{AD\}$, $\{BC\}$, $\{BD\}$, and $\{CD\}$ are contained in the released frequency itemsets, we calculate bounds for the cell entries in the marginal $\{ABCD\}$ given the marginals $\{AB\}$, $\{AC\}$, $\{AD\}$, $\{BC\}$, $\{BD\}$, and $\{CD\}$ and then calculate bounds for the cell entries in the marginal $\{ABCDEFGF\}$ given the marginals $\{ACG\}$, $\{BDEF\}$ and the computed marginals $\{ABCD\}$.

In our scenario, customers may specify a list of private itemsets \mathcal{PS} together with the set of released frequent itemsets \mathcal{FS} . Each ps in \mathcal{PS} may be associated with a secure interval which customers do not want attackers to derive the estimation of ps in this interval. Though we can apply Frechet Bounds to compute the upper bound and lower bound of ps given the released \mathcal{FS} , however, we do not know which frequencies should be removed from release list when the computed bounds are contained in its secure interval. Hence our aim here is to find $\hat{\mathcal{FS}}$ such that 1) $\hat{\mathcal{FS}} \subseteq \mathcal{FS}$ and 2) no ps in \mathcal{PS} can be entailed from $\hat{\mathcal{FS}}$ within a given bound. Our heuristic method is sketched as follows. For each private itemset $ps \in \mathcal{PS}$, decompose ps and compute its Frechet Bounds from the current \mathcal{FS} . If its bound exceeds the tolerated threshold, remove its largest component from \mathcal{FS} and recompute the bound until the bound fits in the threshold. We repeat this process for all private itemset and the reduced \mathcal{FS} can be released as $\hat{\mathcal{FS}}$ finally.

⁴An independence graph that is not necessarily decomposable, but still admits a proper decomposition, is called reducible. Our approach assumes reducible models implicitly.

Table 3: IBM Artificial data sets parameter

	ntrans	nitems	tlen	npats	patlen	corr	conf
IBM-Artificial-1	1M	100,000	10	10,000	4	0.25	0.75
IBM-Artificial-2	1M	100,000	12	10,000	6	0.25	0.75

Table 4: Data set characteristics

	trans.	items	max. trans. size	avg. trans. size
BMS-WebView-1	59,602	497	267	2.5
BMS-WebView-2	77,512	3,340	161	5.0

Table 5: The number of clusters and the maximum cluster size at different minimum support levels on four data sets

data set	support(%)	frequent itemset	clusters	max. cluster size
BMS-WebView-1	0.08	9,934	7	231
	0.1	3,648	4	205
	0.2	530	8	86
	0.4	105	5	39
	0.6	32	3	17
	0.8	17	3	9
BMS-WebView-2	0.08	39,500	66	231
	0.1	22,161	52	140
	0.2	3,110	20	77
	0.4	443	14	27
	0.6	130	7	15
IBM-Artificial-1	0.08	17362	2305	19
	0.09	11324	1970	13
	0.10	9080	1629	12
	0.11	6190	1381	12
	0.12	4561	1098	12
	0.13	3342	938	10
	0.15	1222	608	6
IBM-Artificial-2	0.10	15637	1919	18
	0.11	8788	1587	13
	0.12	5521	1235	13
	0.13	3935	1011	12
	0.14	3197	793	12
	0.15	2866	646	11

5 Empirical Evaluation

In this section we show the experiment results with both IBM synthetic data sets and two real data sets. IBM-Artificial-1 and IBM-Artificial-2 are generated using IBM Almaden association rule synthetic data generator. The parameters used are shown in Table 3. Please note all parameters of IBM-Artificial-1 data set are default setting. We also used two real data sets, *BMS-WebView-1*, *BMS-WebView-2* in our experiments. Each transaction in these data sets is a web session consisting of all the product detail pages viewed in that session. These two data sets can be found from KDD-CUP 2000 [11] and were used for performance evaluation of various association rule mining algorithms in [22]. Table 4 characterizes these data sets in terms of the number of transactions, the number of distinct items, the maximum transaction size, and the average transaction size. Our focus here is to show the feasibility of our approach. For each original data set, we first run Apriori algorithm using different support thresholds and extract a set of frequent itemsets (together with their frequencies) for each support threshold value. We then apply our approach to generate a data set using each set of frequent itemsets (together with their frequencies). In section 5.1, we show efficiency of our method by comparing against a method that does not use graphical decomposition. In section 5.2, we show effectiveness of our approach from two aspects. First, mining results (i.e., frequent itemsets) from original and generated one should be similar for certain thresholds. Second, the performance of mining algorithms (e.g., Apriori) on two sets should be similar. Please note we did not introduce any private itemsets in this experiment. Comparing with existing privacy preserving association rule mining approaches when private itemsets are introduced will be our future work. The experiments were conducted in a DELL Dimension 8100, with one 1.7G processor, and 640 Mbytes of RAM.

5.1 Performance Analysis For each data set, we run Apriori algorithm to generate frequent itemsets. Our first step is to merge frequent itemsets into disjoint clusters. Table 5 shows the number of frequent itemsets, the number of disjoint clusters, the average cluster size, and the maximum cluster size by varying support values on each data set. We observe that the items are grouped into a reasonable number of disjoint clusters for all four data sets. Each disjoint cluster contains a subset of items varying dozens to hundreds in most cases. This validates our strategy that we can generate transactions for each cluster and merge them later.

We believe that most real world market basket datasets display this characteristics, i.e., disjoint item

clusters exist. However, we did encounter one real data set, BMS-POS [11], with all items forming one cluster. This data set contains several years of point-of-sale data from an electronics retailer. The transaction in this data set is a customer’s purchase transaction consisting of all the product categories purchased at one time. The reason that all itemsets form one cluster is the BMS-POS data set was transformed from the original point-of-sale data by grouping products into category. Each item thus represents a category, rather than an individual product.

Though each disjoint cluster contains much fewer items, it may still be infeasible or impractical to apply IPF or other linear programming techniques directly on each cluster. So our following steps are to decompose each cluster into components, apply IPF to estimate cell frequencies of contingency table corresponding to each component, and finally calculate cell frequencies of contingency table corresponding to the original cluster. As the complexity of graphical decomposition is $O(ne + n^2)$ where n is the number of vertices and e is the number of edges, it is significantly lower than the complexity of IPF. Hence applying this divide and conquer strategy can significantly improves the performance when there are large clusters which can be decomposed.

Figure 3 shows the execution time of our method of inverse frequent itemset mining with graphical decomposition on BMS-WebView data sets. We also include the execution time of method without graphical decomposition in Figure 3. Under all support values, our method is significantly better (2 or 3 orders) than method without decomposition. For example, under support threshold 0.55, our method uses 286 seconds to generate data while the method without decomposition needs more than 20 hours.

Figure 4 shows the comparison on IBM-Artificial data sets. Though our method is still with better performance, the difference is not as significant as BMS-WebView datasets. The reason is that there are very few clusters which contain more than 10 items. In fact, more than 90 % clusters are single or two items clusters in these two data sets. When the number of items contained in one cluster is less than 10, the time of applying IPF method for each component is trivial. So graphical decomposition does not improve performance significantly as there are very few clusters involving decomposition. This experiment also validates the result discovered in [22], i.e., the artificial data sets generated using IBM Almaden data generator have very different characteristics from real-world data sets.

5.2 Accuracy Analysis As we discussed before, there may exist many data sets which satisfy the given

Table 6: Similarity of mining results on original vs. generated

BMS-Web	support	Jaccard	Dice	Overlap
View-1				
s = 0.7	0.3	0.367	0.537	0.964
	0.4	0.507	0.673	0.986
	0.5	0.689	0.816	0.985
	0.6	0.817	0.899	0.985
	0.7	0.940	0.969	0.992
	0.8	0.883	0.938	0.934
	0.9	0.893	0.944	0.954
	1.0	0.887	0.940	0.959
View-2				
s = 0.6	0.6	0.696	0.768	1
	0.7	0.708	0.739	0.964
	0.8	0.710	0.830	0.928
	0.9	0.722	0.838	0.976
	1.0	0.701	0.824	0.910
	1.1	0.704	0.826	0.962
	1.2	0.702	0.825	1

frequent itemsets. During data generation, using IPF method simply can generate a data set which converges to the maximum likelihood estimation and graphical decomposition does not lose any information. Hence we can expect to get the same frequent itemsets (and also same frequency values) when we mine from generated data with support values greater or equal to support threshold used to extract frequent itemsets. However, our method introduce errors when we apply graphical decomposition. Recall statistics we used to build independence graph is not complete as we can not compute statistics over all stratum (we can not access the original data set).

$$\begin{aligned}
 D_{jaccard}(\mathcal{FS}^0, \mathcal{FS}^f) &= \frac{|\mathcal{FS}^0 \cap \mathcal{FS}^f|}{|\mathcal{FS}^0 \cup \mathcal{FS}^f|} \\
 D_{dice}(\mathcal{FS}^0, \mathcal{FS}^f) &= \frac{2 \times |\mathcal{FS}^0 \cap \mathcal{FS}^f|}{|\mathcal{FS}^0| + |\mathcal{FS}^f|} \\
 D_{overlap}(\mathcal{FS}^0, \mathcal{FS}^f) &= \frac{|\mathcal{FS}^0 \cap \mathcal{FS}^f|}{\min(|\mathcal{FS}^0|, |\mathcal{FS}^f|)}
 \end{aligned}
 \tag{5.3}$$

We use Jaccard, dice and overlap coefficients to measure the similarity between frequent itemsets (\mathcal{FS}^0) mined from original data and frequent itemsets (\mathcal{FS}^f) mined from data set generated using our approach. Results on two IBM-BMS-View data sets are shown in

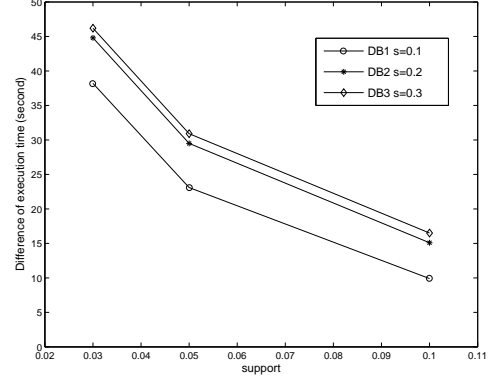


Figure 5: Performance of the Apriori method on three data sets generated using different support values (s=0.1, 0.2, 0.3) from IBM-Artificial-1

Table 6. For each data, we only show result between original and one data set generated due to space limitation. We can see \mathcal{FS}^0 and \mathcal{FS}^f are very similar in terms of three measures when support threshold values used for mining are greater than or equal to support threshold used for data generation. On the other hand, when support threshold values used for mining are less than that used for data generation, \mathcal{FS}^0 and \mathcal{FS}^f are dissimilar as we expected.

Figure 5 shows difference of performance of Apriori mining algorithm when running on original IBM-Artificial-1 and three data sets generated by our approach using support $s = 0.1, 0.2$ and 0.3 respectively. As we expected, the execution time on the first data set DB1 generated using $s = 0.1$ has the smallest difference with that on original one while the execution time on the third data set DB3 generated using $s = 0.3$ has the largest difference. Please note we used IBM-Artificial data instead of BMS-Webview data sets for this experiment because those two real data sets are relatively small.

6 Conclusions and Future Work

In this paper we presented a feasible solution to the NP-Complete problem of inverse frequent set mining. The approach can effectively and efficiently generate a synthetic market basket data set from the released frequent itemsets and their supports. We also presented a heuristic method to screen out confidential frequent itemsets from frequent itemsets used for data generation. The generated synthetic data set can preserve most frequent itemset patterns as the original one without disclosing confidential information. Hence it can be used for benchmarking frequent item set mining algo-

rithms.

There are some aspects of this work that merit further research. Among them, we are trying to figure out better solution for the entailment of frequent itemsets problem, i.e., given a set of frequency constraints plus a set of private itemsets, how to screen those frequency constraints to get the best list of released frequency constraints while guaranteeing no precise information of private itemsets can be derived.

Another aspect that merits further research is how to generate market basket data when only frequency bounds of frequent itemsets are available. In some scenario, customers would only provide several sequences of frequent itemsets with different support thresholds, rather than exact frequency of each frequent itemset. Finally, we will compare our approach with current privacy preserving association rule mining approaches and investigate the tradeoff between privacy and performance for each approach.

7 Acknowledgments

This work was supported in part by U.S. National Science Foundation CCR-0310974. The authors would like to thank Christian Borgelt for providing his implementation of the Apriori algorithm. We would also like to thank IBM Almaden Quest group for providing the market basket data generator.

References

- [1] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Database*, pages 207–216, 1993.
- [2] A. Agresti. *Categorical data analysis*. Wiley, 1990.
- [3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop*, pages 45–52, Nov 1999.
- [4] J. Badsberg. An environment for graphical models. Ph.D. Thesis, Aalborg University, Demark, 1995.
- [5] T. Calders. Computational complexity of itemset frequency satisfiability. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database System*, 2004.
- [6] E. Dasseni, V. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *Proceedings of the 4th International Information Hiding Workshop*, pages 369–383. Pittsburg,PA, April 2001.
- [7] A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables given marginal totals and decomposable graphs. *PNAS*, 97(22):11885–11892, 2000.
- [8] A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation. *Statistical Journal of the United Nations ECE*, 18:363–371, 2001.
- [9] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd Symposium on Principles of Database Systems*, pages 211–222, 2003.
- [10] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228. Edmonton, Canada, July 2002.
- [11] KDDCUP2000. <http://www.ecn.purdue.edu/KDDCUP>.
- [12] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.
- [13] T. Mielikainen. On inverse frequent set mining. In *Proceedings of the 2nd Workop on Privacy Preserving Data Mining*, Nov 2003.
- [14] S. Oliveira and O. Zaiane. Protecting sensitive knowledge by data sanitization. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 211–218. Melbourne, Florida, Nov 2003.
- [15] G. Ramesh, W. Maniatty, and M. Zaki. Feasible itemset distributions in data mining: theory and application. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 284–295. San Diego, CA, June 2003.
- [16] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 682–693, August 2002.
- [17] Y. Saygin, V. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *Sigmod Record*, 30(4):45–54, Dec 2001.
- [18] R. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.
- [19] J. Whittaker. *Graphical Models in Applied Mathematical Multivariate Statistics*. Wiley, 1990.
- [20] X. Wu, D. Barbará, and Y. Ye. Screening and interpreting multi-item associations based on log-linear modeling. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 276–285. Washinton D.C, August 2003.
- [21] X. Wu, Y. Wang, and Y. Zheng. Privacy preserving database application testing. In *Proceedings of the ACM Workshop on Privacy in Electronic Society*, pages 118–128, 2003.
- [22] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the 7th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406. San Francisco, CA, August 2001.

On Variable Constraints in Privacy Preserving Data Mining

Charu C. Aggarwal, Philip S. Yu
IBM T. J. Watson Research Center
{ charu, psyu }@us.ibm.com

Abstract

In recent years, privacy preserving data mining has become an important problem because of the large amount of personal data which is tracked by many business applications. In many cases, users are unwilling to provide personal information unless the privacy of sensitive information is guaranteed. A recent framework performs privacy preserving data mining by using a condensation based approach. In this framework, the privacy of all records is treated homogeneously. It is therefore inefficient to design a system with a uniform privacy requirement over all records. We discuss a new framework for privacy preserving data mining, in which the privacy of all records is not the same, but can vary considerably. This is often the case in many real applications, in which different groups of individuals may have different privacy requirements. We discuss a condensation based approach for privacy preserving data mining in which an efficient method is discussed for constructing the condensation in a heterogeneous way. The heterogeneous condensation is capable of handling both static and dynamic data sets. We present empirical results illustrating the effectiveness of the method.

1 Introduction

Privacy preserving data mining has become an important problem in recent years, because of the large amount of consumer data tracked by automated systems on the internet. The proliferation of electronic commerce on the world wide web has resulted in the storage of large amounts of transactional and personal information about users. In addition, advances in hardware technology have also made it feasible to track information about individuals from transactions in everyday life. In many cases, users are not willing to supply such personal data unless its privacy is guaranteed. Therefore, in order to ensure effective data collection, it is important to design methods which can mine the data with a guarantee of privacy. Some interesting discourses on the nature of privacy in the context of recent trends in information technology may be found in [6, 9, 10]. The recent focus on privacy in data collec-

tion has resulted to a considerable amount of research on the subject [1, 2, 3, 4, 5, 7, 11, 12, 15, 16]. A recent approach to privacy preserving data mining has been a *condensation-based* technique [2]. This technique essentially creates condensed groups of records which are then utilized in one of two ways:

- The statistical information in the pseudo-groups can be utilized to generate a new set of pseudo-data which can be utilized with a variety of data mining algorithms.
- The condensed pseudo-groups can be utilized directly with minor modifications of existing data mining algorithms.

The condensation approach of [2] is also referred to as the *k*-indistinguishability model. A record is said to be *k*-indistinguishable, when there are at least *k* other records in the data (including itself) from which it cannot be distinguished. Clearly, when a record is 1-indistinguishable, it has no privacy. The *k*-indistinguishability of a record is achieved by placing it in a group with at least (*k*−1) other records. This model shares a number of conceptual characteristics with the *k*-anonymity model [18], though the algorithms for doing so are quite different. Another important difference between the two schemes is that the former does not rely on domain specific hierarchies (as in the case of the *k*-anonymity model). The *k*-indistinguishability model can also work effectively in a dynamic environment such as that created by data streams.

In the model discussed in [2], it was assumed that all records have the same privacy requirement. This is also the case for the *k*-anonymity model in which the level of privacy is fixed a-priori. In most practical applications, this is not be a reasonable assumption. For example, when a data repository contains records from heterogeneous data sources, it is rarely the case that each repository has the same privacy requirement. Similarly, in an application tracking the data for brokerage customers, the privacy requirements of retail investors are likely to be different from those of institutional investors. Even among a particular class of customers, some customers

(such as high net-worth individuals) may desire a higher level of privacy than others. In general, we would like to associate a different privacy level with each record in the data set.

Let us assume that we have a database \mathcal{D} containing N records. The records are denoted by $\overline{X_1} \dots \overline{X_N}$. We denote this desired privacy level for record $\overline{X_i}$ by $p(i)$. The process of finding condensed groups with varying level of point specific privacy makes the problem significantly more difficult from a practical standpoint. This is because it is not advisable to pre-segment the data into different privacy levels before performing the condensation separately for each segment. When some of the segments contain very few records, such a condensation may result in an inefficient representation of the data. In some cases, the number of records for a given level of privacy k' may be lower than k' . Clearly, it is not even possible to create a group containing only records with privacy level k' , since the privacy level of the entire group would then be less than k' . Therefore, it is not possible to create an efficient (and feasible) system of group condensation without mixing records of different privacy levels. This leads to a number of interesting trade-offs between information loss and privacy preservation. We will discuss these trade-offs and the algorithms to optimize them.

In many cases, the data may be available at one time or it may be available in a more dynamic and incremental fashion. We discuss two cases for our algorithm:

- We discuss an algorithm to perform the condensation when the entire data is available at one time.
- We discuss an algorithm for the case when the data is available incrementally. This is a more difficult case because it is often not possible to design the most effective condensation at the moment the data becomes available.

We will show that in most cases, the algorithm for performing the dynamic group construction is able to achieve results which are comparable to the algorithm for static group construction.

This paper is organized as follows. In the next section, we will discuss some notations and definitions and also introduce the locality sensitive condensation approach. We will first discuss the simple case in which an entire data set is available for application of the privacy preserving approach. This approach will be extended to incrementally updated data sets in section 3. The empirical results are discussed in section 4. Finally, section 5 contains the conclusions and summary.

2 The Condensation Approach

In this section, we will discuss the condensation approach for privacy preserving data mining. Before describing details of the algorithm, we will discuss some notations and definitions. We assume that we have a set of N records, each of which contain d dimensions. We also assume that associated with each data point i , we have a corresponding privacy level $p(i)$. The overall database is denoted by \mathcal{D} whereas the database corresponding to the privacy level p is denoted by \mathcal{D}_p . The privacy level for a record is defined as follows:

DEFINITION 2.1. *The privacy level for a given record is defined as the minimum number of other records in the data from which it cannot be distinguished.*

In the condensation based approach, the data is partitioned into groups of records. Records within a given group cannot be distinguished from one another. For each group, we maintain certain summary statistics about the records. This summary statistics provides the ability to apply data mining algorithms directly to the condensed groups of records. This information also suffices to preserve information about the mean and correlations across the different dimensions. The size of the groups may vary, but its size is at least equal to the desired privacy level of each record in that group. Thus, a record with privacy level equal to $p(i)$ may be condensed with records of privacy levels different from $p(i)$. However, the size of that group must at least be equal to the maximum privacy level of any record in that group.

Each group of records is referred to as a condensed unit. Let \mathcal{G} be a condensed group containing the records $\{\overline{X_1} \dots \overline{X_k}\}$. Let us also assume that each record $\overline{X_i}$ contains the d dimensions which are denoted by $(x_i^1 \dots x_i^d)$. The following information is maintained about each group of records \mathcal{G} :

- For each attribute j , we maintain the sum of corresponding values. The corresponding value is given by $\sum_{i=1}^k x_i^j$. We denote the corresponding first-order sums by $Fs_j(\mathcal{G})$. The vector of first order sums is denoted by $\overline{Fs(\mathcal{G})}$.
- For each pair of attributes i and j , we maintain the sum of the product of corresponding attribute values. The corresponding sum is given by $\sum_{t=1}^k x_t^i x_t^j$. We denote the corresponding second order sums by $Sc_{ij}(\mathcal{G})$. The vector of second order sums is denoted by $\overline{Sc(\mathcal{G})}$.
- We maintain the sum of the privacy levels of the records in the group. This number is denoted by $Ps(\mathcal{G})$.

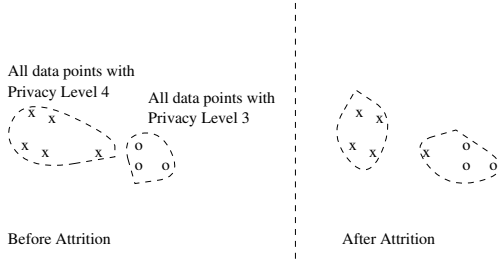


Figure 1: The efficiency of Mixing Different Privacy Levels

- We maintain the total number of records k in that group. This number is denoted by $n(\mathcal{G})$.

The following facts are true about the records in a given group.

OBSERVATION 2.1. *The mean value of attribute j in group \mathcal{G} is given by $Fs_j(\mathcal{G})/n(\mathcal{G})$.*

OBSERVATION 2.2. *The covariance between attributes i and j in group \mathcal{G} is given by $Sc_{ij}(\mathcal{G})/n(\mathcal{G}) - Fs_i(\mathcal{G}) \cdot Fs_j(\mathcal{G})/n(\mathcal{G})^2$.*

We note that the algorithm for group construction must try to put each record in a group which is at least equal to the maximum privacy level of any record in the group. A natural solution is to first classify the records based on their privacy levels and then independently create the groups for varying privacy levels. Unfortunately, this does not lead to the most efficient method for packing the sets of records into different groups. This is because the most effective method for constructing the groups may require us to combine records from different privacy levels. For example, a record with a very low privacy requirement may sometimes naturally be combined with a group of high privacy records in its locality. An attempt to construct a separate group of records with a low privacy requirement may lead to an even higher loss of information. In order to illustrate this point better, we will provide an example.

Consider the set of records illustrated in Figure 1. In this case, there are 3 records with privacy level 3 and 5 records with privacy level 4. One way of grouping the records is to place all the records of privacy level 3 in one group and all records with privacy level 4 in the other. Unfortunately, the group corresponding to privacy level 4 turns out to be ineffective in representing the data. The condensed group utilized from this set of records has poor statistical characteristics, since one of the data points is far removed from the group. Since the condensed statistics of the group does not represent the variations within it, this can lead to an

Algorithm *ConstructGroups*(Level: *MaxPrivacyLevel*, Database: \mathcal{D});

```

begin
   $p = 2$ ;
   $\mathcal{H}_1 = \text{Groups from singleton points in } \mathcal{D}_1$ ;
  while ( $p \leq \text{MaxPrivacyLevel}$ ) do
    begin
       $\mathcal{H}_p = \text{Segment}(\mathcal{D}_p, p)$ ;
       $(\mathcal{H}_{p-1}, \mathcal{H}_p) = \text{Cannibalize}(\mathcal{H}_{p-1}, \mathcal{H}_p)$ ;
       $(\mathcal{H}_{p-1}, \mathcal{H}_p) = \text{Attrition}(\mathcal{H}_{p-1}, \mathcal{H}_p)$ ;
       $\mathcal{H}_p = \mathcal{H}_p \cup \mathcal{H}_{p-1}$ ;
       $p = p + 1$ ;
    end;
  end

```

Figure 2: The Process of Group Construction for Privacy Preserving Data Mining

Algorithm *Segment*(Database: \mathcal{D}_p , Privacy level: p)

```

begin
  while  $\mathcal{D}_p$  contains at least  $p$  data points;
    begin
      Sample a data point  $\bar{X}$  from  $\mathcal{D}_p$ ;
      Find the  $(p - 1)$  data points closest to  $\bar{X}$  in  $\mathcal{D}_p$ ;
      Create a group  $\mathcal{G}$  of  $p$  data points comprising  $\bar{X}$ 
        and the  $p - 1$  other closest data points;
      Add  $\mathcal{G}$  to the set of groups  $\mathcal{H}$ ;
    end
  Assign remaining data points in  $\mathcal{D}_p$  to closest
  groups;
end

```

Figure 3: Group Segmentation

inefficient representation in many cases. In the situation illustrated in Figure 1, it is better to place the outlying record of privacy level 4 into the group with privacy level 3. We also note that it may not be possible to place this outlying record in a group with only two pre-existing members, because of the higher privacy requirement of the record.

First, we need a measure to quantify the effectiveness of a given condensation based approach. In general, this effectiveness is related to the level of compactness with which we can partition the data into different groups. As a goal, this compactness is not very different from the aim of most clustering algorithms. However, the difference here is that there are several constraints on the cardinality of the data points in each group as

Algorithm *Cannibalize*(Groups: $\mathcal{H}_{p-1}, \mathcal{H}_p$);
begin
 for each group $\mathcal{G} \in \mathcal{H}_{p-1}$ **do**
 begin
 for each point in \mathcal{G} perform temporary
 assignment to closest group in \mathcal{H}_p ;
 if (SSQ of temporary assignment is lower) or
 (\mathcal{H}_{p-1} contains fewer than $(p-1)$ members),
 then make assignment permanent
 else keep old assignment;
 end
 end
end

Figure 4: Cannibalization Algorithm

Algorithm *Attrition*(Groups: $\mathcal{H}_{p-1}, \mathcal{H}_p$,
 Privacy Level: p);
begin
for each data point \bar{X} in \mathcal{H}_p **do**
 begin
 $Distc(\bar{X}, p)$ = Distance of \bar{X} to
 centroid of its *current* group in \mathcal{H}_p ;
 $Disto(\bar{X}, p-1)$ = Distance of \bar{X} to
 centroid of its *closest* viable group in \mathcal{H}_{p-1} ;
 $Improve(\bar{X}) = Distc(\bar{X}, p) - Disto(\bar{X}, p-1)$;
 end;
for each group in \mathcal{H}_p with
 at least $p' > p$ points **do**
 begin
 find (if any) the at most $(p' - p)$ data points
 with largest value of $Improve(\cdot)$ function
 which is larger than 0;
 Assign these at most $(p' - p)$ points to their
 corresponding closest groups in \mathcal{H}_{p-1} ;
 end

Figure 5: Attrition Algorithm

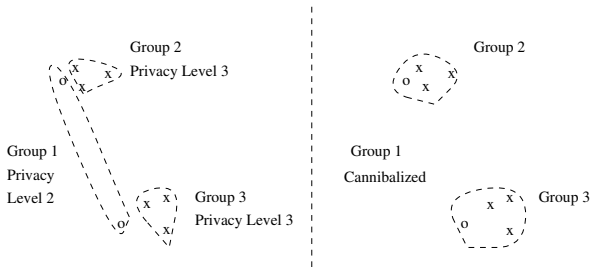


Figure 6: An example of Cannibalization

well as the identity of the data points which can be added to a group with given cardinality. Thus, for the process of quantification of the condensation quality, we simply use the square sum error of the data points in each group. While the privacy level of a group is determined by the number of records in it, the information loss is defined by the average variance of the records about their centroid. We will refer to this quantity as the Sum Squared Error (SSQ).

The method of group construction is different depending upon whether an entire database of records is available or whether the data records arrive in an incremental fashion. We will discuss two approaches for construction of class statistics. The first approach is utilized for the case when the entire database of records is available. The second approach is utilized in an incremental scheme in which the data points arrive one at a time. First, we will discuss the static case in which the entire database of records is available.

The essence of the static approach is to construct the groups using an iterative method in which the groups are processed with increasing privacy level. The overall process of group construction is illustrated in Figure 2. The input to the algorithm is the database \mathcal{D} and the maximum privacy level which is denoted by $MaxPrivacyLevel$. We assume that the segment of the database with privacy level requirement of p is denoted by \mathcal{D}_p . We also assume that the set of groups with privacy level of p is denoted by \mathcal{H}_p . We note that the database \mathcal{D}_1 consists of the set of points which have no privacy constraint at all. Therefore, the group \mathcal{H}_1 comprises of the singleton items from the database \mathcal{D}_1 .

Next, we construct the statistics of the groups in \mathcal{H}_p using an iterative algorithm. In each iteration, we increase the privacy level p by 1, and construct the condensed groups \mathcal{H}_p which have privacy level p . The first step is to construct the group \mathcal{H}_p by using a purely segmentation based process. This process is denoted by *Segment* in Figure 2. This segmentation process is a straightforward iterative approach. In each iteration, a record \bar{X} is sampled from the database \mathcal{H}_p . The closest $(p-1)$ records to this individual record \bar{X} are added to this group. Let us denote this group by \mathcal{G} . The statistics of the p records in \mathcal{G} are computed. Next, the p records in \mathcal{G} are removed from \mathcal{D}_p . The process is repeated iteratively, until the database \mathcal{D}_p is empty. We note that at the end of the process, it is possible that between 1 and $(p-1)$ records may remain. These records can be added to their nearest sub-group in the data. Thus, a small number of groups in the data may contain larger than p data points. The segmentation procedure is illustrated in Figure 3.

Once the segmentation procedure has been performed, we apply the process of *Attrition* and *Cannibalize* in order to further reduce the level of information loss without compromising on the privacy requirements. The purpose of the *Cannibalize* procedure is slightly different. In this procedure, we intend to cannibalize some of the groups in \mathcal{H}_{p-1} and reassign their data points to better fitting groups in \mathcal{H}_p . Consider the example illustrated in Figure 6. In this case, we have illustrated three groups. One of the groups (containing two points) has privacy level of two, and another group (containing three points) has privacy level of three. However, the group with privacy level two does not form a natural cluster of data points. In such a case, it may be desirable to break up the group with privacy level 2 and assign one point each to the groups with privacy level 3. Thus, cannibalization is performed when the group $\mathcal{G} \in \mathcal{H}_{p-1}$ does not form a natural cluster. In such cases, it is more effective to cannibalize the group \mathcal{G} and reassign its group members to one or more clusters in \mathcal{H}_p . Another example of a situation when cannibalization is desirable is when \mathcal{H}_{p-1} has fewer than $(p - 1)$ members. Such a situation arises in situations in which there are very few records for a given privacy level. Consequently, it is not possible to create a group containing only the points at a particular privacy level. We refer to this test for cannibalization as the *numerical test*.

If the group passes the numerical test, we perform an additional *qualitative* test to see if cannibalization should be performed. In order to test whether the cannibalization procedure should be performed, we calculate the SSQ of the regrouping when a temporary assignment of the data points in \mathcal{G} is performed to one or more groups in \mathcal{H}_p . If the SSQ of the resulting assignment is lower, then we make this assignment permanent. The pseudo-code for the cannibalization process is illustrated in Figure 4. By performing this operation, the appropriate privacy level of all data points is maintained. This is because the cannibalization process only assigns data points to groups with higher privacy level. Therefore, the assigned data points find themselves in a group with at least their corresponding required privacy.

We note that some groups in \mathcal{H}_p may sometimes contain more than p data points. This is due to the effects of the *Segment* and *Cannibalize* procedures discussed earlier. The idea in the *Attrition* procedure is to move these excess points to a better fitting group in \mathcal{H}_{p-1} . The movement of these excess points is likely to improve the quality of data representation in terms of reducing the level of information loss. An example of such a case is illustrated in Figure 1. In this case, the group with five data points contains one record which

does not fit very well with the rest of the group. In such a case, the reassignment of the data point to a group with privacy level 3 results in a more compact representation. We note that the reassigned data point has privacy level 4. However, the reassignment process results in the group with privacy level 3 containing 4 data points. Therefore, even though the data point with privacy level 4 was assigned to a group with lower privacy level, the resulting group continues to maintain the desired level of privacy for the reassigned data point. For this purpose, during the attrition process we consider only those groups which are *viable* for reassignment. For a group to be considered viable, it must contain at least as many data points as the privacy level (after the assignment). Furthermore, for a group \mathcal{G} containing p' data points and with privacy level p , we can remove at most $(p' - p)$ data points from it without disturbing the privacy level of the remaining group. In order to perform the actual reassignment, we calculate a function called $Improve(\bar{X})$ for each data point $\bar{X} \in \mathcal{G}$. The value of $Improve(\bar{X})$ is defined to be difference between the distance of \bar{X} from its closest viable centroid and the distance from its current centroid. Clearly, the reassignment of the data point \bar{X} to another group is useful only when the value of $Improve(\bar{X})$ is larger than 0. We re-assign the at most $(p' - p)$ data points with largest value of $Improve(\cdot)$, provided that the value of $Improve(\cdot)$ for each of these data points is larger than 0. The overall attrition procedure is illustrated in Figure 5.

The processes of segmentation, cannibalization and attrition are applied iteratively to the segment \mathcal{D}_p of the database for each value of the privacy level p . The value of p is incremented by 1 in each iteration up to the maximum privacy level. The set of groups constructed at this point are returned as the final condensation.

Once the condensed statistics have been constructed, anonymized data can be generated as discussed in [2]. The anonymized data is generated using the statistical properties which can be derived from the group. While this new set of points resembles the original data distribution, it maintains the privacy of the data. The process of anonymized group construction is achieved by first constructing a $d * d$ covariance matrix for each group \mathcal{G} . This matrix is denoted by $C(\mathcal{G})$. The ij th entry of the co-variance matrix is the co-variance between the attributes i and j of the set of records in \mathcal{G} . The eigenvectors of this co-variance matrix are determined by decomposing the matrix $C(\mathcal{G})$ in the following form:

$$(2.1) \quad C(\mathcal{G}) = P(\mathcal{G}) \cdot \Delta(\mathcal{G}) \cdot P(\mathcal{G})^T$$

The columns of $P(\mathcal{G})$ are the eigenvectors of $C(\mathcal{G})$. The diagonal entries $\lambda_1(\mathcal{G}) \dots \lambda_d(\mathcal{G})$ of $\Delta(\mathcal{G})$ represent

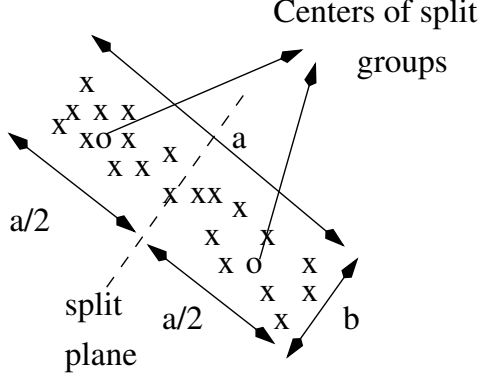


Figure 7: Splitting Group Statistics (Illustration)

the corresponding eigenvalues. It can be shown that the eigenvectors of a covariance matrix form an ortho-normal axis system. This ortho-normal axis-system represents the directions along which the second order correlations are zero. If the data were represented using this ortho-normal axis system, then the covariance matrix would be the diagonal matrix corresponding to $\Delta(\mathcal{G})$. The diagonal entries of $\Delta(\mathcal{G})$ represent the variances along the individual dimensions in this new axis system. We can assume without loss of generality that the eigenvalues $\lambda_1(\mathcal{G}) \dots \lambda_d(\mathcal{G})$ are ordered in decreasing magnitude. The corresponding eigenvectors are denoted by $e_1(\mathcal{G}) \dots e_d(\mathcal{G})$. The anonymized data for each group is reconstructed assuming that the data within each group is independently and uniformly distributed along the different eigenvectors. Furthermore, the variance of the distribution along each eigenvector is equal to the corresponding eigenvalue. These approximations are reasonable when only a small spatial locality is used.

3 Dynamic Maintenance of Groups

The process of dynamic maintenance of groups is useful in a variety of settings such as that of data streams. In the process of dynamic maintenance, the points in the data stream are processed incrementally. It is assumed that a set \mathcal{S} of the data points (denoted by *InitNumber*) are available at the beginning of the process. The static process *ConstructGroups* is applied to this set \mathcal{S} . Once the initial groups have been constructed, a dynamic process of group maintenance is applied in order to maintain the condensed groups of varying privacy levels.

The incremental algorithm works by using a nearest neighbor approach. When an incoming data point \bar{X}_i is received, we find the closest cluster to it using the

distance of the data point \bar{X}_i to the different centroids. While it is desirable to add \bar{X}_i to its closest centroid, we cannot add \bar{X}_i to a given cluster which has fewer than $p(i) - 1$ data points in it. Therefore, the data point \bar{X}_i is added to the closest cluster which also happens to have at least $p(i) - 1$ data points inside it.

In general, it is not desirable to have groups with high sizes compared to their constituent privacy levels. When such a situation arises, it effectively means that a higher level of representational inaccuracy is created than is really necessary with the privacy requirements of the points within the group. The average privacy level of the group \mathcal{G} can be computed from the condensed statistics. This number is equal to $Ps(\mathcal{G})/n(\mathcal{G})$. This is because $Ps(\mathcal{G})$ is equal to the sum of the privacy levels of the data points in the group.

The split criterion used by our algorithm is that a group is divided when the number of items in the group is more than twice the average privacy level of the items in the group. Therefore, the group is split when the following holds true:

$$(3.2) \quad n(\mathcal{G}) \geq 2 \cdot Ps(\mathcal{G})/n(\mathcal{G})$$

As in the case of anonymized data construction, we utilize the uniformity assumption in order to split the group statistics. In each case, the group is split along the eigenvector with the largest eigenvalue. This also corresponds to the direction with the greatest level of variance. This is done in order to reduce the overall variance of the resulting clusters and ensure the greatest compactness of representation. An example of this case is illustrated in Figure 7. We assume without loss of generality that the eigenvector \bar{e}_1 with the lowest index is the chosen direction the split. The corresponding eigenvalue is denoted by λ_1 . Since the variance of the data along \bar{e}_1 is λ_1 , then the range (a) of the corresponding uniform distribution along \bar{e}_1 is given¹ by $a = \sqrt{12 \cdot \lambda_1}$.

In such a case, the original group of size $2 \cdot k$ is split into two groups of equal size. We need to determine the first order and second order statistical data about each of the split groups \mathcal{M}_1 and \mathcal{M}_2 . We assume that the privacy component $Ps(\mathcal{G})$ is also equally divided between the two groups. We first derive the centroid and eigenvector directions for each group. These values are sufficient to reconstruct the values of $Fs_i(\mathcal{G})$ and $Sc_{ij}(\mathcal{G})$ about each group.

Assume that the centroid of the unsplit group \mathcal{M} is denoted by $\bar{Y}(\mathcal{M})$. This centroid can be computed

¹This calculation was done by using the formula for the standard deviation of a uniform distribution with range a . The corresponding standard deviation is given by $\sqrt{a/12}$.

from the first order values $\overline{Fs(\mathcal{M})}$ as follows:

$$(3.3) \quad \overline{Y(\mathcal{M})} = (Fs_1(\mathcal{M}), \dots, Fs_d(\mathcal{M}))/n(\mathcal{G})$$

Once the centroid has been computed, those of each of the split groups can be computed as well. From Figure 7, it is easy to see that the centroids of each of the split groups \mathcal{M}_1 and \mathcal{M}_2 are given by $\overline{Y(\mathcal{M})} - (a/4) \cdot \overline{e_1}$ and $\overline{Y(\mathcal{M})} + (a/4) \cdot \overline{e_1}$ respectively. By substituting $a = \sqrt{12 \cdot \lambda_1}$, it is easy to see that the new centroids of the groups \mathcal{M}_1 and \mathcal{M}_2 are given by $\overline{Y(\mathcal{M})} - (\sqrt{12 \cdot \lambda_1}/4) \cdot \overline{e_1}$ and $\overline{Y(\mathcal{M})} + (\sqrt{12 \cdot \lambda_1}/4) \cdot \overline{e_1}$ respectively.

We will now discuss how to compute the second order statistical values. The first step is the determination of the covariance matrix of the split groups. Let us assume that the ij th entry of the co-variance matrix for the group \mathcal{M}_1 is given by $C_{ij}(\mathcal{M}_1)$. We also note that the eigenvectors of \mathcal{M}_1 and \mathcal{M}_2 are identical to the eigenvectors of \mathcal{M} , since the directions of zero correlation remain unchanged by the splitting process. Therefore, we have:

$$\begin{aligned} e_1(\mathcal{M}_1) &= e_1(\mathcal{M}_2) = e_1(\mathcal{M}) \\ e_2(\mathcal{M}_1) &= e_2(\mathcal{M}_2) = e_2(\mathcal{M}) \\ e_3(\mathcal{M}_1) &= e_3(\mathcal{M}_2) = e_3(\mathcal{M}) \\ &\dots \\ e_d(\mathcal{M}_1) &= e_d(\mathcal{M}_2) = e_d(\mathcal{M}) \end{aligned}$$

The eigenvalue (in the split groups \mathcal{M}_1 and \mathcal{M}_2) corresponding to $\overline{e_1}(\mathcal{M})$ is equal to $\lambda_1/4$. This is because the splitting process along $\overline{e_1}$ reduces the corresponding variance by a factor of 4. Other eigenvalues remain unchanged. Let $P(\mathcal{M})$ represent the eigenvector matrix of \mathcal{M} , and $\Delta(\mathcal{M})$ represent the corresponding diagonal matrix. Then, the new diagonal matrix $\Delta(\mathcal{M}_1) = \Delta(\mathcal{M}_2)$ of \mathcal{M}_1 can be derived by dividing the entry $\lambda_1(\mathcal{M})$ by 4. Therefore, we have:

$$\lambda_1(\mathcal{M}_1) = \lambda_1(\mathcal{M}_2) = \lambda_1(\mathcal{M})/4$$

The other eigenvalues of \mathcal{M}_1 and \mathcal{M}_2 remain the same:

$$\begin{aligned} \lambda_2(\mathcal{M}_1) &= \lambda_2(\mathcal{M}_2) = \lambda_2(\mathcal{M}) \\ \lambda_3(\mathcal{M}_1) &= \lambda_3(\mathcal{M}_2) = \lambda_3(\mathcal{M}) \\ &\dots \\ \lambda_d(\mathcal{M}_1) &= \lambda_d(\mathcal{M}_2) = \lambda_d(\mathcal{M}) \end{aligned}$$

Thus, the (identical) co-variance matrixes of \mathcal{M}_1 and \mathcal{M}_2 may be determined as follows:

$$C(\mathcal{M}_1) = P(\mathcal{M}_1) \cdot \Delta(\mathcal{M}_1) \cdot P(\mathcal{M}_1)^T$$

From Observation 2.2, it is clear that the second order statistics of \mathcal{M}_1 may be determined as follows:

$$\begin{aligned} Sc_{ij}(\mathcal{M}_1) &= \\ k \cdot C_{ij}(\mathcal{M}_1) &+ Fs_i(\mathcal{M}_1) \cdot Fs_j(\mathcal{M}_1)/k \end{aligned}$$

An important observation is that even though the covariance matrices of \mathcal{M}_1 and \mathcal{M}_2 are identical, the values of $Sc_{ij}(\mathcal{M}_1)$ and $Sc_{ij}(\mathcal{M}_2)$ are different because of different first order aggregates substituted in the above formula for $Sc_{ij}(\mathcal{M}_1)$. The overall process for splitting the group statistics is illustrated in Figure 7. Another interesting point to be noted is that the entire purpose of splitting is to keep groups sizes sufficiently compact for data mining algorithms. The process of splitting itself can *never* result in the violation of the privacy condition, since the split group is based on a split of the *statistics*, but not of the data points themselves. In order to understand this point, let us consider the following “example” of a case where the split condition seems to violate privacy. Consider a group having 5 tuples, the privacy constraints of the tuples being 2, 2, 2, 3, 5 respectively. The group does not split because $5 < 2 * 14/5$. Now, if a new tuple having privacy constraint 3 wants to join the group, the splitting condition is satisfied since $6 > 2 * 17/6$. Hence each of the split group corresponds to statistics of 3 data points. Therefore, it would apparently seem that the privacy of the tuple with requirement 5 has been violated. This is *not* the case since we split the *statistics* into two *pseudo-groups* of 3 points each, rather than actually split the *points* themselves. The process of performing the split partitions the statistics based on a *probability distribution assumption* (uniform distribution) rather than using the actual points themselves (which have already been lost in the merged statistics). The tuple with privacy condition 5 may contribute to the statistics of both groups, when the splitting condition is used. Each pseudo-group thus has a privacy level as high as the unsplit group, from the perspective of the *old data points* in it, but at the same time we would need to use the size of the group while considering the addition of *further data points* into the smaller pseudo-groups.

In order to test the quality of our results we applied our approach to a nearest neighbor classifier. In the classification process, the condensation process was performed separately for each class. In the next section, we will discuss the behavior of this nearest neighbor classifier.

4 Empirical Results

We tested the privacy preserving approach over a wide range of data sets and metrics. An important question which arises in the context of a privacy preserving

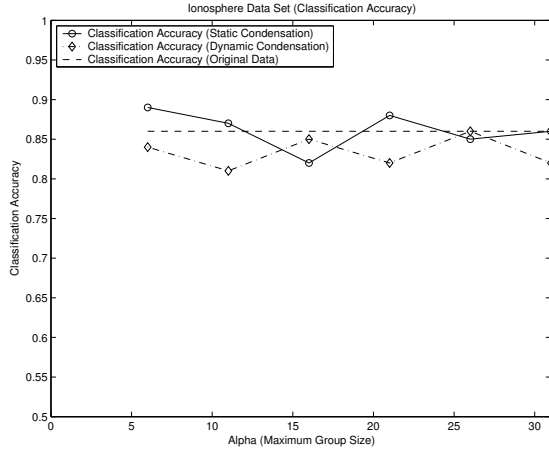


Figure 8: Accuracy of Classifier with Increasing Privacy Level (Ionosphere Data Set)

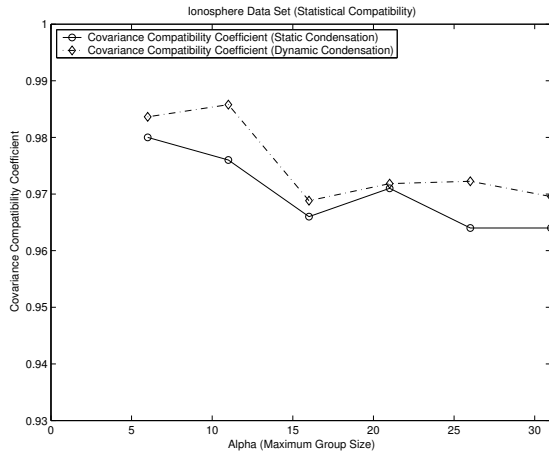


Figure 9: Covariance Compatibility of Condensed Data Set with Increasing Privacy Level (Ionosphere Data Set)

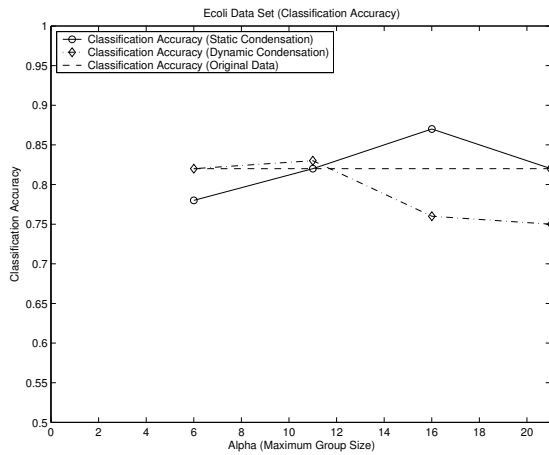


Figure 10: Accuracy of Classifier with Increasing Privacy Level (Ecoli Data Set)

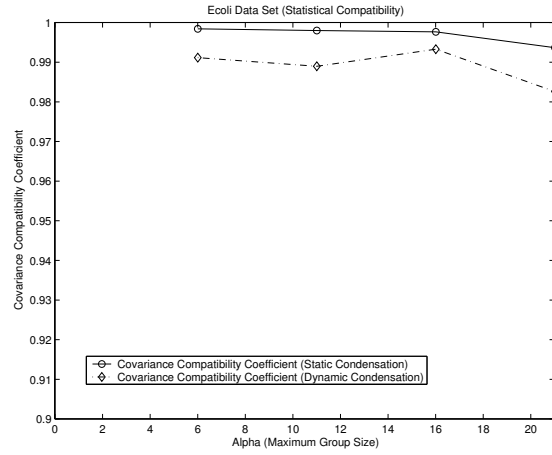


Figure 11: Covariance Compatibility of Condensed Data Set with Increasing Privacy Level (Ecoli Data Set)

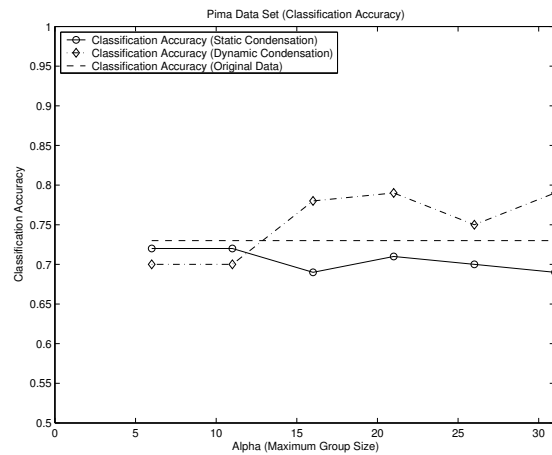


Figure 12: Accuracy of Classifier with Increasing Privacy Level (Pima Indian Data Set)

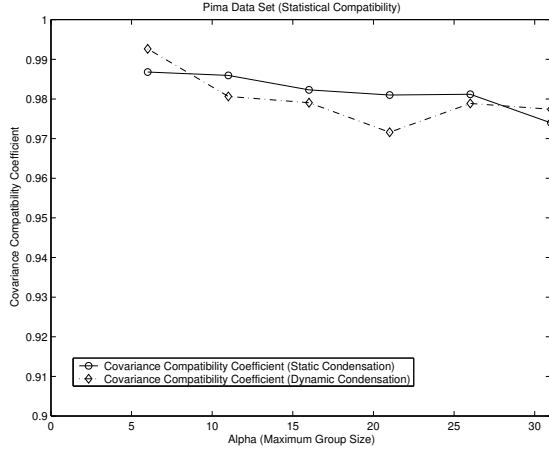


Figure 13: Covariance Compatibility of Condensed Data Set with Increasing Privacy Level (Pima Indian Data Set)

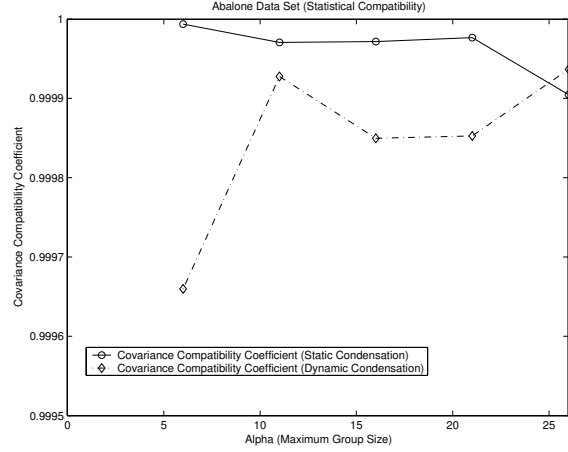


Figure 15: Covariance Compatibility of Condensed Data Set with Increasing Privacy Level (Abalone Data Set)

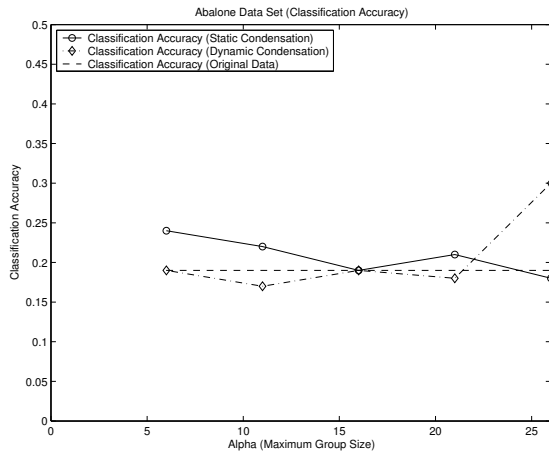


Figure 14: Accuracy of Classifier with Increasing Privacy Level (Abalone Data Set)

approach is the nature of the metric to be used in order to test the quality of the approach. The first step is to test the nature of the tradeoff between increased levels of privacy, and the resulting information loss. While the level of privacy is controlled by the average condensed group size, the information loss is measured indirectly in terms of the effect of the perturbation on the quality of data mining algorithms. We tested the accuracy of a simple k -nearest neighbor classifier with the use of different levels of privacy. The minimum privacy level of each data point was generated from a (discrete) uniform distribution in the range $[\alpha - \beta, \alpha]$. By changing the value of α it is possible to vary the level of privacy during the condensation process. The aim of our approach is to show that a high level of privacy can be achieved without significantly compromising accuracy.

Another useful metric for testing the quality of the privacy preserving process arises from the level of matching between the original and perturbed data. This provides insight into the nature of the relationship between the original data set and perturbed data set. The first step is therefore to identify the statistics used for testing the effectiveness of the perturbation process. One simple method is to test how the covariance structure of the perturbed data set matched with the original data set. This is because the covariance structure of the data identifies the essential data properties up to a second order approximation. If the newly created data set has very similar data characteristics to the original data set, then the condensed data set is a good substitute for most data mining algorithms. For each dimension pair (i, j) , let the corresponding entries in the covariance matrix for the original and the perturbed data be denoted

by o_{ij} and p_{ij} respectively. We computed the statistical coefficient of correlation between the data entry pairs (o_{ij}, p_{ij}) . Let us denote this value by μ . When the two matrices are identical, the value of μ is 1. On the other hand, when there is perfect negative correlation between the entries, the value of μ is -1 .

A number of real data sets from the UCI machine learning repository² were used for the testing. We used the Ionosphere, Ecoli, Pima Indian and Abalone data sets. The last data set was a regression modeling problem, and therefore the classification measure needed to be redefined. For this problem, the classification accuracy measure used was the percentage of the time that the age was predicted within an accuracy of less than one year by the nearest neighbor classifier. In many cases, the number of data points for a given privacy level for lower than the numerical value of the privacy level itself. In such cases, the mixing of data points for different privacy levels is inevitable. Thus, the condensation process could not have been performed for such cases using the homogeneous k -anonymity model or k -indistinguishability model [2, 18].

The results on classification accuracy for the Ionosphere, Ecoli, Pima Indian, and Abalone data sets are illustrated in Figures 8, 10, 12 and 14 respectively. The value of β was fixed to 4, whereas the value of α was varied over the different data sets. The range of values of α is determined by the number of data points in the particular data set at hand. This value of α is illustrated on the X-axis. On the Y-axis, we have plotted the classification accuracy of the nearest neighbor classifier, when the condensation technique was used. For each graph, we have illustrated the results using both static and dynamic condensation. In addition, a baseline is marked on each graph. This baseline is a horizontal line on the graph which shows the classification accuracy using the original data. It is clear that in most cases, the accuracy of classification reduced with increasing group size. This is a natural tradeoff because a greater amount of privacy is achieved with larger groups sizes. At the same time, it leads to a higher amount of information loss.

In many cases, the quality of the classification improved because of the condensation process. In most cases. While the aim of our approach was to provide a high level of privacy without losing information, it appears that the process of condensation itself actually helped in removing the anomalies in the data for the purpose of classification. This phenomenon is likely to be helpful over a number of different data mining problems in which the aggregate behavior of the data is exposed by the condensation process.

Furthermore, the static condensation approach provided higher quality results than the dynamic technique. This is because the splitting algorithm of the dynamic condensation process introduced an additional level of approximation into the data representation. The splitting procedure assumed a uniform distribution of the data within a condensed group of data points. The accuracy of this approximation reduces when group sizes are small. In such cases, there are simply too few data points to make an accurate estimation of the values of split group statistics. Thus, the use of the uniform distribution approximation reduces the quality of the covariance statistics in the split groups for small group sizes. For this reason, the dynamic condensation process was sometimes less effective than the static condensation approach. However, in all cases, the dynamic condensation approach worked almost as effectively as the classifier on the original data. One notable exception to the general advantage of the static condensation process was the behavior on the Pima Indian data set. In this case, the dynamic condensation process provided results of higher quality for larger group sizes. The reason for this was that the splitting process seemed to improve the quality of the classification. The data set seemed to contain a number of anomalies. These anomalies were removed by the splitting process. This resulted in a higher classification accuracy of the dynamic approach.

We also compared the covariance characteristics of the data sets. The results are illustrated in Figures 9, 11, 13 and 15 respectively. For most data sets, the value of the statistical correlation is almost perfect. This corresponds to the fact that the correlation values was larger than 0.95 in most cases. For some examples such as the Abalone data set (illustrated in Figure 15), the covariance compatibility value was larger than 0.99. These results emphasize the fact that the perturbed data is similar to the original data in terms of its statistical structure. As in the previous case, the results for the case of static condensation were better than those for dynamic condensation. This is again because of the additional inaccuracy introduced by the splitting process. In all cases, the absolute correlation provided by the scheme was very high. In the dynamic case, the correlation coefficient tended to drop for small group sizes. The only exception to this general rule was the ionosphere data set in which the covariance compatibility values were slightly lower for the static case. The covariance compatibility also reduced for extremely large group sizes. This is because in such a case, the pseudo-data no longer represents a particular data locality well. Thus, the covariance compatibility was highest in those cases in which the data contained tight clusters comprising a relatively modest number of

²<http://www.ics.uci.edu/~mllearn>

data points. This is because of the following reasons:

- When the number of points in each cluster were large, the accuracy of the uniform distribution assumption during the splitting process is maintained.
- When the clusters are tight, these data points represent a small spatial locality with respect to the rest of the data set. An approximation in a small spatial locality does not significantly affect the overall correlation structure.

We note that the process of representing a small spatial locality in a group and that of representing a larger number of data points in a group are two competing and contradictory goals. It is important to pick a balance between the two, since this tradeoff defines the quality of performance on the underlying data mining algorithm. This balance is externally defined, since the average group size is determined by the privacy requirements of the users. In general, since our approach continued to be as effective as the base classification accuracy over a wide range of group sizes, this illustrates the effectiveness of our methodology in most practical scenarios.

5 Conclusions and Summary

In this paper, we discussed a scheme for privacy preserving data mining in which the data points are allowed to have variable privacy levels. This is useful in a number of applications in which different records have inherently different privacy requirements. We propose a method for privacy protection in a data stream environment using condensed statistics of the data set. These condensed statistics can either be generated statically or they can be generated dynamically in a data stream environment. We tested our results on a number of real data sets from the UCI machine learning repository. The results show that our method produces data sets which are quite similar to the original data in structure, and also exhibit similar accuracy results.

References

- [1] C. C. Aggarwal, and S. Parthasarathy, *Mining Massively Incomplete Data Sets by Conceptual Reconstruction*, Proceedings of the ACM KDD Conference, (2001), pp. 227–232.
- [2] C. C. Aggarwal, and P. S. Yu, *A Condensation Based Approach to Privacy Preserving Data Mining*, Proceedings of the EDBT Conference, (2004), pp. 183–199.
- [3] D. Agrawal, and C. C. Aggarwal, *On the Design and Quantification of Privacy Preserving Data Mining Algorithms*, Proceedings of the ACM PODS Conference, (2002).
- [4] R. Agrawal, and R. Srikant, *Privacy Preserving Data Mining*, Proceedings of the ACM SIGMOD Conference, (2000).
- [5] P. Benassi, *Truste: An online privacy seal program*, Communications of the ACM, 42(2), (1999), pp. 56–59.
- [6] C. Clifton, and D. Marks, *Security and Privacy Implications of Data Mining*, ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, (1996), pp. 15–19.
- [7] J. Vaidya, and C. Clifton, *Privacy Preserving Association Rule Mining in Vertically Partitioned Data*, ACM KDD Conference, (2002).
- [8] T. M. Cover, J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., New York, (1991).
- [9] Cranor L. F. (Ed.) *Special Issue on Internet Privacy*, Communications of the ACM, 42(2), (1999).
- [10] The Economist, *The End of Privacy*, (1999).
- [11] V. Estivill-Castro, and L. Brankovic, *Data Swapping: Balancing privacy against precision in mining for logic rules*, Data Warehousing and Knowledge Discovery, Springer-Verlag, Lecture Notes in Computer Science 1676, (1999), pp. 389–398.
- [12] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, *Privacy Preserving Mining Of Association Rules*, ACM KDD Conference, (2002).
- [13] A. Hinneburg, and D. A. Keim, *An Efficient Approach to Clustering in Large Multimedia Databases with Noise*, ACM KDD Conference, (1998).
- [14] V. S. Iyengar, *Transforming Data To Satisfy Privacy Constraints*, ACM KDD Conference, (2002).
- [15] C. K. Liew, U. J. Choi, and C. J. Liew, *A data distortion by probability distribution*, ACM TODS Journal, (1985), 10(3) pp. 395–411.
- [16] T. Lau, O. Etzioni, and D. S. Weld, *Privacy Interfaces for Information Management*, Communications of the ACM, 42(10), (1999), pp. 89–94.
- [17] S. Murthy, *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey*, Data Mining and Knowledge Discovery, 2, (1998), pp. 345–389.
- [18] P. Samarati, and L. Sweeney, *Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement Through Generalization and Suppression*. Proceedings of the IEEE Symposium on Research in Security and Privacy, (1998).
- [19] S. L. Warner, *Randomized Response: A survey technique for eliminating evasive answer bias*, Journal of the American Statistical Association, 60(309), (1965), pp. 63–69.

Clustering with Model-Level Constraints

David Gondek *

Shivakumar Vaithyanathan [†]

Ashutosh Garg [‡]

Abstract

In this paper we describe a systematic approach to uncovering multiple clusterings underlying a dataset. In contrast to previous approaches, the proposed method uses information about structures that are *not desired* and consequently is very useful in an exploratory datamining setting. Specifically, the problem is formulated as constrained model-based clustering where the constraints are placed at a model-level. Two variants of an EM algorithm, for this constrained model, are derived. The performance of both variants is compared against a state-of-the-art information bottleneck algorithm on both synthetic and real datasets.

1 Introduction

Clustering is a form of unsupervised learning which groups instances based on their similarity. Most clustering algorithms either optimize some objective [15] or make use of a similarity/distance function between instances [13]. In real applications there may be multiple clusterings underlying the data and the user may typically not know a priori which clustering is of interest. A commonly identified liability of clustering is the large degree to which the solution returned is dictated by assumptions inherent in the choice of similarity of objective function [12]. To address this, a number of *semi-supervised clustering* approaches have been proposed. They all share a dependency on user-supplied knowledge about the structure of the desired solution. They differ, however, on how this knowledge is employed. Approaches vary from enforcing constraints directly within the algorithm [22], seeding the initialization of clusters [1], learning distance functions based on user input as in [23] and [12], or even a combination of these approaches such as in [3, 2] which learns a distance function *and* enforces constraints within the algorithm.

The problem with these approaches is that the user may be unable to define a useful similarity function or

specify prior knowledge of a desired solution. For example, consider the “Yahoo! problem” of [6]: you are given 100,000 text documents and asked to group them into categories. You are not given any information in advance about what categories to use or which documents are related. The goal is then to create a categorization which can be browsed and accessed efficiently. In [6], they propose an iterative solution which supplies clusterings to the user and receives feedback according to three forms. The user may specify “This document doesn’t belong in here,” “Move this document to that cluster,” and “These two documents shouldn’t (or should) be in the same cluster.” We note that all three forms of feedback require the user to have information about a desired clustering. Furthermore, with 100,000 documents, a user may be able to inspect only a small percentage of the documents. Feedback obtained may be appropriate for the sample but misleading for the overall dataset. This will have the effect of biasing the search and suppressing other interesting clusterings of which the user is not aware.

We propose as an alternate approach to provide the user with a series of high-quality non-redundant clusterings. In our case, feedback would not require positive knowledge about which instances should be assigned to which cluster. Instead, the user’s interaction would be to specify, “Find another clustering.” With this in mind, we develop a mechanism to systematically search through the clusterings underlying the data. A formal framework to incorporate this non-redundancy as constraints in model-based clustering, as well as associated algorithms for obtaining high-quality solutions, is the focus of this paper.

2 Problem Definition

Unsupervised clustering algorithms obtain the *natural* grouping underlying the data¹. In some cases, prior knowledge about a desired clustering can be incorporated using a variety of constrained clustering techniques [22] [14] [23]. Briefly, these techniques require the analyst to provide explicit knowledge of the target clustering. E.g. in [22]: *must-link* constraints

*Department of Computer Science, Brown University, Providence, RI 02912, dgc@cs.brown.edu

[†]IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120, shiv@almaden.ibm.com

[‡]Google, Inc., 1600 Amphitheater Parkway, Mountain View, CA 94043, ashutosh@google.com

¹The natural clustering is dependent upon the choice of objective function.

require two instances to be assigned the same cluster and *cannot-link* constraints require two instances to be assigned to different clusters.

In contrast, we consider the less-studied setting in which the knowledge available pertains to particular solutions which are *not desired*. Our goal is to use this knowledge to discover new clusterings in a systematic fashion. In this paper this knowledge can be expressed in the following two ways:

[a.] **Known clusterings** One or more clusterings are given.

[b.] **Negative features** A set of “negative” features is specified and clusterings associated with these features are undesired.

In case [a.] above, the goal is to find newer clusterings which do not resemble the known clusterings. In case [b.], the goal is to find clusterings not associated with the negative features.

Table 1: Example: 5-dimensional Dataset

i	remaining features			negative features	
	$y_{.1}^+$	$y_{.2}^+$	$y_{.3}^+$	$y_{.1}^-$	$y_{.2}^-$
1	1	1	1	1	0
2	1	1	1	1	0
3	1	0	0	0	1
4	1	0	0	0	1
5	0	1	1	1	0
6	0	1	1	1	0
7	0	1	0	0	1
8	0	0	0	0	1

To understand the implications of negative features, consider the example in Table 1. Each instance has been separated into “negative” and “remaining” features. Clustering only on the negative features produces clusters $\{1, 2, 5, 6\}$ and $\{3, 4, 7, 8\}$. We desire clusterings that are different from this grouping of the data. The naive solution of simply ignoring $y_{.1}^-$ and $y_{.2}^-$ is not sufficient since $y_{.1}^-$ and $y_{.2}^-$ are correlated with $y_{.2}^+$ and $y_{.3}^+$. Consequently the naive solution will still result in finding $\{1, 2, 5, 6\}$ and $\{3, 4, 7, 8\}$ ². However, a more intelligent search may find $\{1, 2, 3, 4\}$ and $\{5, 6, 7, 8\}$ which is a different and meaningful grouping of the data. What complicates matters is that in general there may be many natural clusterings and negative features may have more than one associated clustering. The task at hand is then to find a meaningful grouping of the data

²Note that even repeated runs of clustering techniques which depend on random initializations, such as k-means [15], will typically return this dominant clustering.

which is different from all clusterings associated with the negative features.

Techniques have been derived from the Information Bottleneck framework[20] for related problems. The Information Bottleneck with Side Information[5] may be applied to clustering and we will use it as a baseline in our experiments. In [9], the Conditional Information Bottleneck was proposed as a technique to find a non-redundant clustering in a dataset given one known clustering. It is not, however, designed for the setting in which negative features may be associated with several known clusterings.

3 Constrained Model-Based Clustering

Mathematically stated, we have a n -instance dataset $\mathcal{Y} = \{\mathbf{y}_i : i = 1 \dots n\}$. Each instance, \mathbf{y}_i is assumed to have negative \mathbf{y}_i^- and remaining \mathbf{y}_i^+ features where m^- is the number of negative features and m^+ is the number of remaining features. y_{ij}^- and y_{ij}^+ represent the j th negative/remaining feature of the i th instance. We use $y_{.j}^-$ and $y_{.j}^+$ to represent the j th negative/remaining feature for a generic instance. All features are assumed to be *binary*, i.e., $\mathbf{y}_i^- \in Y^-$ where Y^- is the set of all such vectors $\{0, 1\}^{m^-}$ and $\mathbf{y}_i^+ \in Y^+$ where Y^+ is the set of all such vectors $\{0, 1\}^{m^+}$. The negative features \mathbf{y}_i^- are assumed to take one of the following forms:

Known clusterings One or more clusterings are given. In this case \mathbf{y}_i^- is represented by a vector where the entries for the clusters to which i is assigned are set to 1 and all other entries are 0.

Negative features A set of m^- negative features, $\{\nu(1) \dots, \nu(m^-)\}$, are specified. In this case \mathbf{y}_i^- is represented as the vector, $\mathbf{y}_i^- = [y_{i\nu(1)}, \dots, y_{i\nu(m^-)}]$.

3.1 Model Recall that our interest is in grouping the instances to maximize their similarity in \mathbf{y}_i^+ while simultaneously searching for structure that is not already present in \mathbf{y}_i^- . Intuitively, this suggests a model where \mathbf{y}_i^- and the cluster, c_k , are independent given \mathbf{y}_i^+ . This captures that the \mathbf{y}_i^- for an instance should be a consequence of its \mathbf{y}_i^+ features. Thus, \mathbf{y}_i^+ are assumed to be drawn from K clusters, c_1, \dots, c_K , while the \mathbf{y}_i^- are drawn conditioned on \mathbf{y}_i^+ . Now, assuming \mathcal{Y} is independently drawn gives the following log-likelihood, $l(\mathcal{Y})$, for the dataset:

$$(3.1) \quad l(\mathcal{Y}) = \sum_{i=1}^n \log \sum_{k=1}^K p(\mathbf{y}_i^- | \mathbf{y}_i^+) p(\mathbf{y}_i^+ | c_k) p(c_k).$$

3.2 Objective Function for CMBC The unconstrained task would be to find a clustering which max-

imizes (3.1). Here we extend (3.1) to incorporate the prior knowledge as model-level constraints. The term *model-level constraints* refers to restrictions on the space of clustering. We begin by representing the prior knowledge as constraints.

3.2.1 Representing Knowledge as Constraints

Recall from above that both forms of knowledge may be expressed via \mathbf{y}_i^- . A natural way to express the constrained problem is [note an implicit assumption that cluster probabilities are equal]:

$$(3.2) \quad \begin{aligned} \max \quad & l(\mathcal{Y}) \\ \text{s.t.} \quad & p(c_j|\mathbf{y}_i^-) = p(c_k|\mathbf{y}_i^-) \quad \forall i, j, k \\ & \text{where } \mathbf{y}_i^- \text{ is negative information.} \end{aligned}$$

The intuition behind the constraints is to enforce the requirement that the value taken by \mathbf{y}_i^- should not influence the cluster assignments.

The constraints in (3.2) may be too restrictive and allow only degenerate solutions. To soften the constraints, we replace them with the requirement that the conditional entropy, $H(C|Y^-)$, be high, where $H(C|Y^-) = -\sum_{k,i} p(c_k|\mathbf{y}_i^-) \log p(c_k|\mathbf{y}_i^-)$. Note that $H(C|Y^-)$ is maximized when the constraints in (3.2) are met. This results in the following objective function:

$$(3.3) \quad \mathcal{L} = (1 - \gamma)l(\mathcal{Y}) + \gamma H(C|Y^-).$$

where γ acts as a tradeoff between the loglikelihood and the penalty terms. Intuitively, as γ is increased the resulting solution is farther away from the clusterings associated with Y^- .

3.2.2 Approximating the Objective Function

(3.3) Ultimately, we wish to derive methods to maximize the objective function (3.3) and in the next section, we will derive an EM algorithm. Before that, however, it is necessary to approximate (3.3) to facilitate easier computation in the EM algorithm. We perform two approximations: a. bounding the $H(C|Y^-)$ term and b. empirical approximation.

Bounding $H(C|Y^-)$ The objective in (3.3) is problematic in its current form because it contains $H(C|Y^-)$, which is unwieldy to compute from the parameters $p(\mathbf{y}^-|c)$ and results in an expression which is difficult to optimize. We can, however, exploit the fact that $H(C|Y^-) = H(C, Y^-) - H(Y^-)$ to rewrite (3.3) as:

$$(3.4) \quad \mathcal{L} = (1 - \gamma)l(\mathcal{Y}) + \gamma (H(C, Y^-) - H(Y^-)).$$

Then, both terms $H(C, Y^-)$ and $H(Y^-)$ may be

computed easily from $p(\mathbf{y}_i^-|c_k)$ according to: (3.5)

$$(3.6) \quad \begin{aligned} H(C, Y^-) &= -\sum_{\mathbf{y}_i^- \in Y^-} \sum_{k=1}^K p(\mathbf{y}_i^-|c_k) p(c_k) \log(p(\mathbf{y}_i^-|c_k) p(c_k)), \\ H(Y^-) &= -\sum_{\mathbf{y}_i^- \in Y^-} \sum_{k=1}^K p(\mathbf{y}_i^-|c_k) p(c_k) \log\left(\sum_{k=1}^K p(\mathbf{y}_i^-|c_k) p(c_k)\right). \end{aligned}$$

The remaining hurdle is that each term contains a log of sums, which prevents us from deriving closed-form update equations in an EM algorithm: the $H(Y^-)$ sums over k within the log term. Further, as will be seen in **Appendix A**, the $p(\mathbf{y}_i^-|c_k)$, which occurs in the log terms of both $H(Y^-)$ and $H(C, Y^-)$, also requires performing a sum. We address this issue by replacing these entropy terms with quadratic bounds.

The Shannon entropies in (3.5) and (3.6) can be approximated by the commonly-used Havrda-Charvát structural δ -entropy[11] to obtain quadratic expressions, however this approximation can be quite loose³. Instead we make use of quadratic bounds recently presented in [10] and [21]. These lower and upper bounds are built around the *index of coincidence (IC)*, the probability of drawing the same element in two independent trials. From [21], we obtain the lower bound which we denote by $H^l(X)$:

$$(3.7) \quad H(X) \geq H^l(X) \doteq \delta_d - \beta_d \sum_x p(x)^2,$$

where d is the number of elements x , used to define:

$$(3.8) \quad \delta_d = \ln(d+1) + d \ln(1 + \frac{1}{d}),$$

$$(3.9) \quad \beta_d = (d+1)d \ln(1 + \frac{1}{d}).$$

From [10], we use the upper bound on $H(X)$, which we denote as $H^u(X)$:

$$(3.10) \quad H(X) \leq H^u(X) \doteq (\ln d) \cdot \left(1 - \frac{1}{1 - \frac{1}{d}} \left(\sum_x p(x)^2 - \frac{1}{d}\right)\right).$$

Applying (3.7) to $H(Y^-, C)$ and (3.10) to $H(Y^-)$ in our objective function (3.4) we obtain:

$$(3.11) \quad \mathcal{L} \geq (1 - \gamma)l(\mathcal{Y}) + \gamma (H^l(C, Y^-) - H^u(Y^-))$$

Empirical Approximation We describe how to compute $p(\mathbf{y}_i^-|c_k)$ and terms in $l(\mathcal{Y})$ in **Appendix A**.

³We attempted to use this approximation in our experiments but found the approximations to differ substantially from the Shannon entropies and result in very poor performance.

For this paper we assume the $p(c_k)$ are constant. From the computation in (A.5), it is clear that $p(\mathbf{y}_i^-|\mathbf{y}_i^+)$ is constant for a given dataset and so can be ignored in $l(\mathcal{Y})$. Also, from (A.5), we see the $p(\mathbf{y}_i^-|\mathbf{y}_h^+)$ can be estimated only for those \mathbf{y}_i^- in the dataset \mathcal{Y}^- , which requires that the summation in the entropy terms $H^l(C, Y^-)$ and $H^u(Y^-)$ be restricted to those $\mathbf{y}_i^- \in \mathcal{Y}^-$. We denote these empirical approximations as $\tilde{H}^l(C, Y^-)$ and $\tilde{H}^u(Y^-)$ which are defined as:

$$(3.12) \quad \tilde{H}^l(C, Y^-) = \left(\delta_{|Y^-|} - \beta_{|Y^-|} \sum_{\mathbf{y}_i^- \in \mathcal{Y}^-} \sum_{k=1}^K (p(\mathbf{y}_i^-|c_k)p(c_k))^2 \right),$$

$$(3.13) \quad \tilde{H}^u(Y^-) = (\log |Y^-|) \cdot \left(1 - Q \sum_{\mathbf{y}_i^- \in \mathcal{Y}^-} \left(\sum_{k=1}^K p(\mathbf{y}_i^-|c_k)p(c_k) \right)^2 \right),$$

where $Q = \frac{1}{1-1/|Y^-|}$ and where $|Y^-|$ is the number of unique \mathbf{y}^- which occur in the dataset. The $H^l(C, Y^-)$ and $H^u(Y^-)$ terms of (3.11) are replaced with $\tilde{H}^l(C, Y^-)$ and $\tilde{H}^u(Y^-)$ to produced the final approximation of the objective function, $\tilde{\mathcal{L}}$:

$$(3.14) \quad \tilde{\mathcal{L}} = l(\mathcal{Y}^+) + \gamma(\tilde{H}^l(C, Y^-) - \tilde{H}^u(Y^-)).$$

4 EM Algorithm

Select annealing rate $\alpha < 1$. Initialize T to be high. Randomly initialize $p(y_{.j}^+|c_k)$. Loop until hard-assignment obtained:

- Loop until converged:
 - For $j = \{1 \dots m^+\}$
 - For $k = \{1 \dots K\}$:
 - E-Step Compute assignment expectations:
 - for all instances $i = 1 \dots n$:

$$(4.15) \quad q(c_k|\mathbf{y}_i) \propto p(c_k)p(\mathbf{y}_i^+|c_k)^{1/T}$$

- M-step Maximize for $p(y_{.j}^+|c_k)$ by solving:
 - (4.16)

$$F_3 p(y_{.j}^+|c_k)^3 + F_2 p(y_{.j}^+|c_k)^2 + F_1 p(y_{.j}^+|c_k) + F_0 = 0$$

- Decrease temperature $T \leftarrow \alpha T$

Figure 1: CMBC Algorithm: Partial Maximization (CMBCpm)

The E-step [shown in (4.15)] is unaffected by the

Select annealing rate $\alpha < 1$. Initialize T to be high. Randomly initialize $p(y_{.j}^+|c_k)$.

Loop until hard-assignment obtained:

1. Loop until converged:

E-Step Compute assignment expectations:

• for all instances $i = 1 \dots n$:

$$(4.17) \quad q(c_k|\mathbf{y}_i) \propto p(c_k)p(\mathbf{y}_i^+|c_k)^{1/T}$$

M-step For $k = \{1 \dots K\}, j = \{1 \dots m^+\}$:

• Maximize for $p(y_{.j}^+|c_k)$ by solving:

$$(4.18) \quad F_3 p(y_{.j}^+|c_k)^3 + F_2 p(y_{.j}^+|c_k)^2 + F_1 p(y_{.j}^+|c_k) + F_0 = 0$$

2. Decrease temperature $T \leftarrow \alpha T$

Figure 2: CMBC Algorithm: Batch Update (CMBCbu)

second and third terms from (3.14)⁴. The M-step finds the maximum likelihood estimates for the model parameters, $p(y_{.j}^+|c_k)$, given the $q(c_k|\mathbf{y}_i)$ from (4.15). The derivation, as described in **Appendix B**, produces the cubic equations in (4.16) and (4.18) and has a closed-form solution due to [4], however in our experiments for ease of implementation we use a numerical method to obtain solutions. We may either perform partial maximization in each iteration using the optimization method of [17] as in Figure 1 or approximate this using a faster batch update as shown in Figure 2. For both approaches, we use a deterministic annealing algorithm[18] which obtains hard-assignments.

5 Information Bottleneck with Side Information [IBwS]

A state-of-the-art existing approach, the IBwS algorithm [5], provides a convenient and elegant way of introducing model-level constraints in optimizing the following objective function:

$$(5.19) \quad \min_{p(c_k|x_i)} I(C; X) - \beta (I(C; Y^+) - \gamma I(C; Y^-)).$$

$I(C; X)$ measures the compactness of the representation of instances X by C . The $I(C; Y^+)$ term measures how well the representation retains information about Y^+ and the $I(C; Y^-)$ term penalizes information which C conveys about Y^- . The value for γ controls the trade-off between retaining information about Y^+ and penalizing information about Y^- . Following an annealing approach as suggested in [20], β may be increased until a hard clustering is found.

⁴The derivation is simple and we have omitted it in the interests of space

6 Experiments

We report experiments on both synthetic and real datasets using CMBCbu [Fig. 1], CMBCpm [Fig. 2] and a deterministic annealing version of IBwS (dIBwS). Results from all experiments are evaluated using the normalized mutual information (NMI) [8], which for clustering C and true labeling L measures how much information C captures about L . It is computed as $NMI(C, L) = \frac{I(C, L)}{H(L)}$ and ranges from 0 [no information] to 1 [$C = L$].

6.1 Synthetic Data We generate synthetic datasets which contain multiple clusterings to test the ability of the algorithms to find those clusterings. In particular, we generate data with 4 underlying clusterings, $\{Q^{(1)}, Q^{(2)}, Q^{(3)}, Q^{(4)}\}$. The strength of the clustering is controlled by the number of features associated with it; 6 features are associated with $Q^{(1)}$, 5 with $Q^{(2)}$, 4 with $Q^{(3)}$ and 3 with $Q^{(4)}$, resulting in an 18-dimensional set. The resultant dataset contains 4 underlying clusterings ordered according to decreasing strength: $Q^{(1)} \succ Q^{(2)} \succ Q^{(3)} \succ Q^{(4)}$. Each $Q^{(i)}$ groups the data into 2 clusters where each cluster has a representative binary vector. Drawing an instance now consists of, for each of $\{Q^{(1)}, Q^{(2)}, Q^{(3)}, Q^{(4)}\}$, randomly selecting one of the two clusters and assigning the binary vector for that cluster. Noise is introduced by randomly flipping each feature with some probability p_{noise} . We have divided the experiments according to the two forms of prior knowledge.

6.1.1 Known Clusterings The first experiment evaluates the ability of the algorithms to find successive clusterings and is divided into three sessions. For Session 1, we assume that one clustering, $Q^{(1)}$, is known, for Session 2 we assume that $Q^{(1)}$ and $Q^{(2)}$ are known, and for Session 3, we assume that $Q^{(1)}, Q^{(2)}$, and $Q^{(3)}$ are known. In each session we consider datasets with p_{noise} ranging from 0.1 to 0.3. The value of γ for each of the CMBC and IBwS algorithms is independently optimized over 20 possible settings for baseline setting $p_{noise} = 0.1$ and this value is retained for all other p_{noise} settings. Setting γ in this manner allows us to investigate the robustness with respect to γ of the algorithms if applied to different datasets. We will later investigate the ranges of effective γ for each of the p_{noise} . We compare performance against a deterministic annealing version of Expectation Maximization (EM) [7], which does not incorporate any prior knowledge. Results are shown in Table 2.

Uncovering Underlying Structure We first evaluate the algorithms over all three sessions for the baseline setting where $p_{noise} = 0.1$. Here we expect the

best performance as this is the setting for which the algorithms have been tuned. The EM algorithm does not incorporate any prior knowledge and so obtains the same solution across sessions. It typically discovers the most prominent clustering, $Q^{(1)}$. Of the 100 datasets, EM obtains a solution with $NMI(C, Q^{(1)}) > 0.75$ for 87 sets and $NMI(C, Q^{(2)}) > 0.75$ for 13. In none of the trials does EM obtain solutions with $NMI(C, Q^{(3)}) > 0.75$ or $NMI(C, Q^{(4)}) > 0.75$ which demonstrates the failure of random initialization to discover less-prominent clusterings. Performance among the CMBC and IBwS algorithms is approximately the same. While the performance of CMBCbu at finding the next most prominent clustering lags somewhat in Session 1, where there is a single known clustering, it improves relative to the other two algorithms in Sessions 2 and 3, where there are multiple known clusterings. The lower score of CMBCbu at discovering $Q^{(2)}$ in Session 1 occurs because in several of the trials, it instead discovers $Q^{(3)}$. This is not a failure of the algorithm; it is successfully accomplishing the task of discovering a novel clustering, however it is not discovering the next most prominent clustering. We will discuss this phenomenon further when looking at other noise settings.

Examining results for higher noise settings, we find the performance of dIBwS drops dramatically. For example, consider Session 2 with $p_{noise} = 0.20$. CMBCbu finds solutions that average $NMI(C, Q^{(3)}) = 0.4782$, and CMBCpm that average $NMI(C, Q^{(3)}) = 0.5007$ whereas for dIBwS, $NMI(C, Q^{(3)}) = 0.0008$. The dIBwS algorithm is finding solutions similar to known clustering $Q^{(2)}$. This behaviour is consistent for the higher noise ($p_{noise} \geq 0.20$) settings throughout Sessions 2 and 3 where dIBwS fails to discover unknown clusterings. Interestingly, in these cases, dIBwS seems to consistently discover the weakest of the known clusterings, as can be seen by looking at Sessions 2 and 3. In Session 2, where $Q^{(2)}$ is the weakest known clustering, $NMI(C, Q^{(2)})$ is 0.8404 and 0.8601 for p_{noise} set to 0.20 and 0.30. In Session 3, where $Q^{(3)}$ is the weakest known clustering, $NMI(C, Q^{(3)})$ is 0.9800 and 0.7270 for p_{noise} set to 0.20 and 0.30. For all of these settings, the solutions obtained by CMBCbu and CMBCpm are most like the target clustering, whereas dIBwS largely fails to discover the target clustering.

In comparing CMBCbu and CMBCpm, there is not a clear winner. As we saw in the baseline setting of $p_{noise} = 0.10$, CMBCbu’s performance relative to CMBCpm and dIBwS generally improves across sessions as there are more known clusterings. There does not, however, appear to be a clear trend within a given session as the noise is increased.

Finally, in Sessions 1 and 2, where there are multi-

Table 2: Mean NMI for 100 synthetic sets generated with 1000 instances according to the procedure in Section 6.1. Parameter settings are: $\alpha = 0.5$, $\gamma^{CMBCbu} = .97$, $\gamma^{CMBCpm} = .95$, $\gamma^{IBwS} = 2.5714$.

Session 1: $Q^{(1)}$ is known. Goal is discovery of $Q^{(2)}$, $Q^{(3)}$ or $Q^{(4)}$.

Session 1: $\mathbf{y}^- = Q^{(1)}$					
p_{noise}	Algorithm	$NMI(C, Q^{(1)})$	$NMI(C, Q^{(2)})$	$NMI(C, Q^{(3)})$	$NMI(C, Q^{(4)})$
0.10	CMBCbu	0.0007	0.8653	0.0591	0.0006
	CMBCpm	0.0008	0.9297	0.0008	0.0006
	dIBwS	0.0008	0.9072	0.0173	0.0007
	EM	0.8089	0.1217	0.0007	0.0008
0.20	CMBCbu	0.0007	0.6136	0.0531	0.0008
	CMBCpm	0.0007	0.6599	0.0163	0.0007
	dIBwS	0.0107	0.6595	0.0114	0.0006
	EM	0.6443	0.0429	0.0012	0.0015
0.30	CMBCbu	0.0006	0.2546	0.0553	0.0048
	CMBCpm	0.0006	0.2796	0.0279	0.0025
	dIBwS	1.0000	0.0008	0.0006	0.0008
	EM	0.3129	0.0430	0.0022	0.0015

Session 2: $Q^{(1)}$ and $Q^{(2)}$ are known. Goal is discovery of $Q^{(3)}$ or $Q^{(4)}$.

Session 2: $\mathbf{y}^- = Q^{(1)}, Q^{(2)}$					
p_{noise}	Algorithm	$NMI(C, Q^{(1)})$	$NMI(C, Q^{(2)})$	$NMI(C, Q^{(3)})$	$NMI(C, Q^{(4)})$
0.10	CMBCbu	0.0006	0.0008	0.8336	0.0009
	CMBCpm	0.0005	0.0006	0.8300	0.0008
	dIBwS	0.0006	0.0008	0.8235	0.0085
	EM	0.8089	0.1217	0.0007	0.0008
0.20	CMBCbu	0.0006	0.0008	0.4782	0.0498
	CMBCpm	0.0005	0.0007	0.5007	0.0317
	dIBwS	0.0108	0.8404	0.0833	0.0006
	EM	0.6443	0.0429	0.0012	0.0015
0.30	CMBCbu	0.0110	0.0009	0.1958	0.0205
	CMBCpm	0.0006	0.0006	0.1520	0.0283
	dIBwS	0.1407	0.8601	0.0008	0.0007
	EM	0.3129	0.0430	0.0022	0.0015

Session 3: $Q^{(1)}$, $Q^{(2)}$ and $Q^{(3)}$ are known. Goal is discovery of $Q^{(4)}$.

Session 3: $\mathbf{y}^- = Q^{(1)}, Q^{(2)}, Q^{(3)}$					
p_{noise}	Algorithm	$NMI(C, Q^{(1)})$	$NMI(C, Q^{(2)})$	$NMI(C, Q^{(3)})$	$NMI(C, Q^{(4)})$
0.10	CMBCbu	0.0009	0.0006	0.0006	0.8176
	CMBCpm	0.0008	0.0006	0.0005	0.7997
	dIBwS	0.0007	0.0006	0.0005	0.8068
	EM	0.8089	0.1217	0.0007	0.0008
0.20	CMBCbu	0.0009	0.0005	0.0006	0.5220
	CMBCpm	0.0009	0.0005	0.0005	0.4351
	dIBwS	0.0006	0.0208	0.9800	0.0006
	EM	0.6443	0.0429	0.0012	0.0015
0.30	CMBCbu	0.0086	0.0013	0.0068	0.2107
	CMBCpm	0.0047	0.0008	0.0101	0.1160
	dIBwS	0.0393	0.1985	0.7270	0.0006
	EM	0.3129	0.0430	0.0022	0.0015

ple unknown clusterings, dIBwS almost always finds the next most prominent clustering whereas CMBCbu and CMBCpm occasionally discover less prominent clusterings (e.g. $Q^{(3)}$ and $Q^{(4)}$ in Sessions 1 and 2 in Table 2.) In general, the CMBC algorithms were more sensitive to initialization than dIBwS which often obtains the same solution regardless of initialization. This is despite the fact that all three algorithms are using a deterministic annealing framework.

Robustness of γ Parameter. While the performance of dIBwS deteriorates for $p_{noise} > 0.10$, performance can be improved by re-tuning γ . This was tested on Session 3 with results shown in Figures 4 and 3. It can easily be seen in Figure 3 that, with CMBCbu, an optimal γ value for one p_{noise} setting is successful on other p_{noise} settings. For example, $\gamma = .95$ is optimal for $p_{noise} = 0.10$, but also scores well for $p_{noise} = 0.20$ and $p_{noise} = 0.30$. On the other hand, the fact that the curves are not nested in Figure 4 indicates that dIBwS is more sensitive to the value of γ . For example, the optimal $\gamma = 2.6667$ on $p_{noise} = 0.10$ fails completely on $p_{noise} = 0.20$ and $p_{noise} = 0.30$.

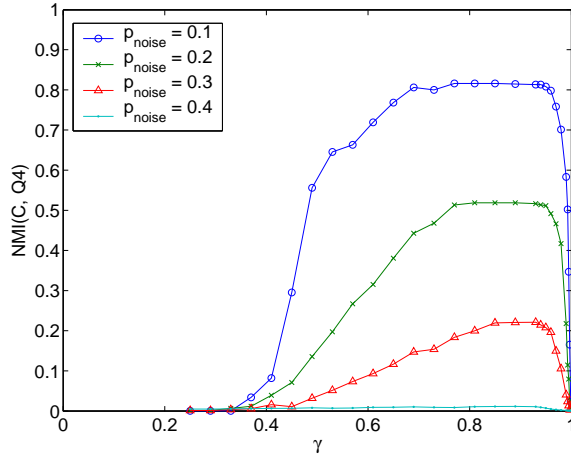


Figure 3: Similarity to target clustering $Q^{(4)}$ in Session 3 for varying γ , CMBCbu: optimal γ for one p_{noise} works for other p_{noise}

6.1.2 Negative Features Table 3 shows the results of a similar set of experiments using negative features instead of known clusterings. Half of the features associated with each clustering were set to be in \mathbf{y}^- . The use of multiple negative features means information which was previously part of the \mathbf{y}^+ is now instead part of the \mathbf{y}^- . For example, consider Session 1 where in the previous experiments \mathbf{y}^+ contained 6 features associated with the known clustering. In these experiments, 3 of those features are instead part of \mathbf{y}^- meaning \mathbf{y}^+

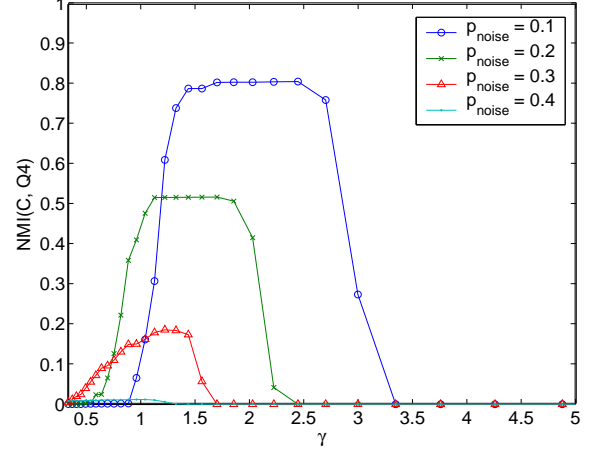


Figure 4: Similarity to target clustering $Q^{(4)}$ in Session 3 for varying γ , p_{noise} . dIBwS: optimal γ for one p_{noise} fails for other p_{noise}

now contains only 3 remaining features associated with the known clustering. Examining the baseline settings ($p_{noise} = 0.10$), the algorithms appear competitive except for Session 3, where CMBCpm substantially underperforms CMBCbu and dIBwS. As in the previous set of experiments, we note for Session 1 that the CMBC techniques do occasionally discover the less prominent clusterings, as evidenced by the fact that $NMI(C, Q^{(3)})$ is 0.0333 and 0.0253 for CMBCbu and CMBCpm whereas it is 0.0008 for dIBwS. This phenomenon is consistent across noise settings for Session 1, however in Session 2, the results are mixed. In Session 2, dIBwS also discovers less prominent clusterings.

The most striking difference between these results and the results in the previous section is that the performance of dIBwS does not deteriorate strongly as the noise increases. In fact, in Sessions 1 and 3, the dIBwS continues to outperform CMBCpm and CMBCbu as the noise increases, e.g. in Session 1, $p_{noise} = 0.30$, the $NMI(C, Q^{(2)})$ is 0.3245 for dIBwS while for CMBCpm and CMBCbu it is 0.3064 and 0.2148. In Session 3, $p_{noise} = 0.30$, $NMI(C, Q^{(3)})$ is 0.1488 for dIBwS while for CMBCbu and CMBCpm it is 0.1221 and 0.0578. In these experiments where noise features are given, dIBwS does not share the same sensitivity to parameter settings as in the known clustering experiments.

6.2 Real Data In this section we report our experiments on a sub-set of the RCV-1 corpus [19]. We define three collections, each with two different clusterings based on *Topic* and *Region*. E.g. in reut2x2a [described below], documents can be clustered into *Topic* cate-

Table 3: Mean NMI for 100 synthetic sets generated with 1000 instances according to the procedure in Section 6.1. Parameter settings are: $\alpha = 0.5$, $\gamma^{CMBCbu} = .9$, $\gamma^{CMBCpm} = .9$, $\gamma^{IBwS} = 1.2222$.

Session 1: Some features associated with $Q^{(1)}$ are known. Goal is discovery of $Q^{(2)}$, $Q^{(3)}$ or $Q^{(4)}$.

Session 1: $\mathbf{y}^- = 50\%$ of the features associated with $Q^{(1)}$					
p_{noise}	Algorithm	$NMI(C, Q^{(1)})$	$NMI(C, Q^{(2)})$	$NMI(C, Q^{(3)})$	$NMI(C, Q^{(4)})$
0.10	CMBCbu	0.0008	0.8923	0.0333	0.0006
	CMBCpm	0.0008	0.9012	0.0253	0.0007
	dIBwS	0.0008	0.9289	0.0008	0.0007
0.20	CMBCbu	0.0007	0.6554	0.0204	0.0007
	CMBCpm	0.0032	0.6473	0.0155	0.0030
	dIBwS	0.0008	0.6799	0.0006	0.0006
0.30	CMBCbu	0.0025	0.3064	0.0229	0.0012
	CMBCpm	0.0062	0.2148	0.0391	0.0089
	dIBwS	0.0009	0.3245	0.0044	0.0006

Session 2: Some features associated with $Q^{(1)}$ and $Q^{(2)}$ are known. Goal is discovery of $Q^{(3)}$ or $Q^{(4)}$.

Session 2: $\mathbf{y}^- = 50\%$ of the features associated with $Q^{(1)}, Q^{(2)}$					
p_{noise}	Algorithm	$NMI(C, Q^{(1)})$	$NMI(C, Q^{(2)})$	$NMI(C, Q^{(3)})$	$NMI(C, Q^{(4)})$
0.10	CMBCbu	0.0007	0.0009	0.8302	0.0006
	CMBCpm	0.0007	0.0009	0.8115	0.0167
	dIBwS	0.0007	0.0008	0.8224	0.0085
0.20	CMBCbu	0.0010	0.0011	0.5104	0.0269
	CMBCpm	0.0323	0.0042	0.4453	0.0299
	dIBwS	0.0007	0.0008	0.5269	0.0112
0.30	CMBCbu	0.0063	0.0012	0.2115	0.0103
	CMBCpm	0.0212	0.0017	0.1367	0.0215
	dIBwS	0.0007	0.0006	0.1153	0.0191

Session 3: Some features associated with $Q^{(1)}$, $Q^{(2)}$ and $Q^{(3)}$ are known. Goal is discovery of $Q^{(4)}$.

Session 3: $\mathbf{y}^- = 50\%$ of the features associated with $Q^{(1)}, Q^{(2)}, Q^{(3)}$					
p_{noise}	Algorithm	$NMI(C, Q^{(1)})$	$NMI(C, Q^{(2)})$	$NMI(C, Q^{(3)})$	$NMI(C, Q^{(4)})$
0.10	CMBCbu	0.0008	0.0006	0.0006	0.8175
	CMBCpm	0.0459	0.0220	0.0128	0.6931
	dIBwS	0.0008	0.0006	0.0006	0.8047
0.20	CMBCbu	0.0173	0.0005	0.0006	0.5026
	CMBCpm	0.0979	0.0137	0.0119	0.3017
	dIBwS	0.0035	0.0006	0.0006	0.5100
0.30	CMBCbu	0.0711	0.0030	0.0017	0.1221
	CMBCpm	0.0637	0.0125	0.0145	0.0578
	dIBwS	0.0184	0.0006	0.0037	0.1488

gories [MCAT, ECAT] or into *Region* categories [UK, India].

reut2x2a: *Topic* = [MCAT (Markets), GCAT (Government/Social)], *Region* = [UK, India], 3400 documents, 2513 terms.

reut2x2b: *Topic* = [I35 (Motor Vehicles and Parts), I81 (Banking and Financial Services)], *Region* = [Germany, USA], 1547 documents, 2235 terms.

reut2x2c: *Topic* = [GPOL (Domestic Politics), GDIP (International Relations)], *Region* = [France, UK], 800 documents, 2194 terms.

For each collection, stopwords were removed and a frequency cutoff was used to filter out the low frequency terms. We then evaluated the algorithms according to two scenarios. Each scenario consists of assuming one of either *Topic* or *Region* clusterings is known and then evaluating the algorithms on their ability to find the other clustering. Tables 4 and 5 show the best of 10 random initializations for each of the algorithms.

The experiments on real data confirmed the patterns observed with synthetic data. Specifically, CMBCbu and CMBCpm outperform dIBwS on all experiments except for the reut2x2b set in Scenario 2 where CMBCpm and dIBwS tie and dIBwS scores better than CMBCbu by 1.28x. For all other experiments the median performance gain with respect to dIBwS is 1.54x for CMBCpm and 1.66x for CMBCbu. Performance of CMBCbu and CMBCpm is considerably stronger than dIBwS in Scenario 2 - reut2x2a where CMBCbu has a NMI of 0.7576 and CMBCpm has a NMI of 0.7621 but dIBwS has a NMI of only 0.1600. In Scenario 1 - reut2x2c, CMBCbu and CMBCpm outperform dIBwS and find reasonably high quality solutions with NMIs of 0.3162 and 0.3257 for CMBCbu and CMBCpm respectively while dIBwS scores 0.1903, further showing that the outperformance of CMBCbu and CMBCpm with respect to dIBwS can be substantial.

We now consider the only experiment where CMBCbu and CMBCpm did not outperform dIBwS. On Scenario 2 - reut2x2b, dIBwS, with a NMI of 0.0934 outscores CMBCbu (NMI = 0.0729) and ties with CMBCpm, also with a NMI of 0.0934. In fact, dIBwS obtains the same clustering in both scenarios, highlighting a difficulty with using extrinsic measures like NMI to evaluate success. Namely, there may be other prominent structure in the dataset which is not associated with our known categorizations. Then, the algorithm may discover this unassociated structure in both scenarios.

The presence of other prominent structures is a possible explanation for the low NMI scores in Scenario

1 - reut2x2a where CMBCbu (NMI = 0.0189) and CMBCpm (NMI = 0.0014) beats dIBwS (NMI = 0.0002). Clearly the algorithms are not discovering the clustering we expect. indicating that the “Topic” clustering we have in mind may not be prominent in the data. This hypothesis is a reasonable explanation for this dataset which includes substantially more documents (n=3400) than the other two (n=1547 and n=800.) We intend to study these effects further to better determine the causes of the poor performance.

Finally, these experiments confirm there is no penalty associated with using the batch update approach of CMBCbu instead of the partial maximization approach of CMBCpm. This is an advantageous finding as the runtime of the CMBCbu algorithm is comparable with dIBwS whereas CMBCpm incurs a greater computational cost.

7 Conclusions

In this paper we have addressed the problem of systematically uncovering multiple clusterings underlying a dataset. We have proposed a constrained likelihood model and an associated EM algorithm with two variants [CMBCbu and CMBCpm]. These algorithms, along with an existing state-of-the-art information bottleneck variant [dIBwS], have been evaluated on both synthetic and real data in their ability to systematically uncover new clusterings. On synthetic data, all algorithms showed the ability to uncover multiple underlying clusterings even under noise. The performance of CMBCbu and CMBCpm was more robust to noise with respect to parameter settings than dIBwS. With increasing amounts of information the performance of the dIBwS algorithm was comparable to that of CMBCbu and CMBCpm.

The experiments on real data confirmed the patterns observed with synthetic data. In the high-dimensional, high-noise and minimal-information setting of real data, the CMBC methods outperformed dIBwS, where only in one case did dIBwS not come in last. However, the results of CMBCpm come at a higher computational cost while the CMBCbu algorithm, on the other hand, is comparable in runtime to the dIBwS algorithm. Notably, there appeared to be no penalty for using the batch update approach of CMBCbu instead of partial maximization as in CMBCpm.

The CMBC methods rely on several assumptions, namely: clusters have roughly equal probability, the prior knowledge is generated according to a mixture of relevant features, and features are binary. It remains to be seen whether these methods may be modified to relax these assumptions. In general, however, the results are very encouraging and this direction of research certainly

Table 4: Scenario 1: Topic clustering is known. NMI of best of 10 random initializations on the Reuters RCV1 sets. The goal is to find the Region clustering. We use $\delta = 0.5$ and γ is tuned for each set. Highest scores are in bold.

dataset	Algorithm	$\mathbf{y}^- = Topic$	
		$NMI(C; Topic)$	$NMI(C; Region)$
reut2x2a	CMBCbu($\gamma = .997$)	0.0020	0.0189
	CMBCpm($\gamma = .99$)	0.0151	0.0014
	dIBwS ($\gamma = 4.8824$)	0.0150	0.0002
reut2x2b	CMBCbu($\gamma = .99$)	0.0086	0.1245
	CMBCpm ($\gamma = .95$)	0.0047	0.1155
	dIBwS ($\gamma = 3.1667$)	0.0934	0.0748
reut2x2c	CMBCbu($\gamma = .99$)	0.0019	0.3162
	CMBCpm ($\gamma = .95$)	0.0031	0.3257
	dIBwS ($\gamma = 24$)	0.0879	0.1903

Table 5: Scenario 2: Region clustering is known. NMI of best of 10 random initializations on the Reuters RCV1 sets. The goal is to find the Topic clustering. We use $\delta = 0.5$ and γ is tuned for each set. Highest scores are in bold.

dataset	Algorithm	$\mathbf{y}^- = Region$	
		$NMI(C; Topic)$	$NMI(C; Region)$
reut2x2a	CMBCbu($\gamma = .997$)	0.7576	0.0001
	CMBCpm($\gamma = .99$)	0.7621	0.0001
	dIBwS ($\gamma = 4.8824$)	0.1600	0.0000
reut2x2b	CMBCbu($\gamma = .99$)	0.0729	0.0082
	CMBCpm($\gamma = .95$)	0.0934	0.0748
	dIBwS ($\gamma = 4.8824$)	0.0934	0.0748
reut2x2c	CMBCbu($\gamma = .997$)	0.4382	0.0001
	CMBCpm($\gamma = .99$)	0.4570	0.0001
	dIBwS ($\gamma = 4.8824$)	0.4294	0.0017

warrants further study.

References

- [1] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 27–34. Morgan Kaufmann Publishers Inc., 2002.
- [2] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM Press, 2004.
- [3] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Twenty-first international conference on Machine learning*. ACM Press, 2004.
- [4] G. Cardano. *Ars Magna*. Nurnberg, 1545.
- [5] G. Chechik and N. Tishby. Extracting relevant structures with side information. In *Advances in Neural Information Processing Systems 15*, 2002.
- [6] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback, 2003.
- [7] A. Dempster, N. Laird, and D. B. Rubin. Maximum likelihood from incomplete data. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- [8] X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *The 20th International Conference on Machine Learning*, 2003.
- [9] D. Gondek and T. Hofmann. Non-redundant data clustering. In *4th IEEE International Conference on Data Mining*, 2004.
- [10] P. Harremoës and F. Topsøe. Details for inequalities between entropy and index of coincidence derived from information diagrams. *IEEE Transactions on Information Theory*, 47:2944–2960, 2001.
- [11] J. H. Havrda and F. Charvát. Quantification methods of classification processes: Concepts of structural entropy. In *Kybernetika*, 3, 1967.
- [12] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *Twenty-first international conference on Machine learning*. ACM Press, 2004.

- [13] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.
- [14] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.
- [15] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967.
- [16] R. Neal and G. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.
- [17] M. J. D. Powell. *A Fortran Subroutine for Solving Systems of Nonlinear Algebraic Equations*, chapter 7. 1970.
- [18] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems, 1998.
- [19] T. Rose, M. Stevenson, and M. Whitehead. The Reuters corpus volume 1 – from yesterday’s news to tomorrow’s language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002.
- [20] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.
- [21] F. Topsøe. Entropy and index of coincidence, lower bounds. preprint, 2003.
- [22] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. of the 17th International Conference on Machine Learning*, pages 1103–1110, 2000.
- [23] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, 15, 2003.

A Computation of $p(\mathbf{y}_i^+ | c_k)$, $p(\mathbf{y}_i^- | \mathbf{y}_h^+)$ and $p(\mathbf{y}_i^- | c_k)$

A.1 Computation of $p(\mathbf{y}_i^+ | c_k)$ Assuming class-conditional independence of the binary features:

$$(A.1) \quad p(\mathbf{y}_i^+ | c_k) = \prod_{j=1}^{m^+} p(y_{ij}^+ | c_k)$$

$$(A.2) \quad = \prod_{j=1}^{m^+} p(y_j^+ | c_k)^{y_{ij}^+} (1 - p(y_j^+ | c_k))^{1 - y_{ij}^+}.$$

A.2 Computation of $p(\mathbf{y}_i^- | \mathbf{y}_h^+)$ Computing $p(\mathbf{y}_i^- | c_k)$ [described below] requires computing $p(\mathbf{y}_i^- | \mathbf{y}_h^+)$. To address the sparsity of \mathbf{y}_h^+ in the

observed data and to ensure tractability, we assume \mathbf{y}_i^- is generated according to a mixture of the features y_{hj}^+ where we assume that $p(y_{hj}^+)$ is uniform:

(A.3)

$$(A.4) \quad \begin{aligned} p(\mathbf{y}_i^- | \mathbf{y}_h^+) &= \sum_{j=1}^{m^+} p(\mathbf{y}_i^- | y_{hj}^+) p(y_{hj}^+) = \frac{1}{m^+} \sum_{j=1}^{m^+} p(\mathbf{y}_i^- | y_{hj}^+) \\ &= \frac{1}{m^+} \sum_j p(\mathbf{y}_i^- | y_j^+ = 1)^{y_{hj}^+} p(\mathbf{y}_i^- | y_j^+ = 0)^{1 - y_{hj}^+}, \end{aligned}$$

where (A.4) follows from the fact that the data is binary. The $p(\mathbf{y}_i^- | y_j^+ = 1)$ and $p(\mathbf{y}_i^- | y_j^+ = 0)$ are estimated from data using

(A.5)

$$p(\mathbf{y}_i^- | y_j^+ = b) = \frac{|\{\mathbf{y}_t \in \mathcal{Y} : \mathbf{y}_t^- = \mathbf{y}_i^- \text{ and } y_{tj}^+ = b\}|}{|\{\mathbf{y}_t \in \mathcal{Y} : y_{tj}^+ = b\}|},$$

for $b \in \{0, 1\}$.

E.g. for the data in Table 1,

$$\begin{aligned} p([1 \ 0] | y_1^+ = 1) &= 2/4, \\ p([1 \ 0] | y_2^+ = 1) &= 4/5, \\ p([1 \ 0] | y_3^+ = 1) &= 4/4. \end{aligned}$$

A.3 Computation of $p(\mathbf{y}_i^- | c_k)$ The $p(\mathbf{y}_i^- | \mathbf{y}_i^+)$ are fixed, so the $p(\mathbf{y}_i^- | c_k)$ are consequences of the choice of $p(y_j^+ | c_k)$ as can be seen in Lemma A.1.

LEMMA A.1. *The $p(\mathbf{y}_i^- | c_k)$ may be expanded as:*

$$(A.6) \quad p(\mathbf{y}_i^- | c_k) = \frac{1}{m^+} \sum_j \sum_{y_j^+ \in \{0,1\}} p(\mathbf{y}_i^- | y_j^+) p(y_j^+ | c_k).$$

Sketch of proof By marginalizing over all possible \mathbf{y}_h^+ we obtain: $p(\mathbf{y}_i^- | c_k) = \sum_{\mathbf{y}_h^+ \in \mathcal{Y}^+} p(\mathbf{y}_i^- | \mathbf{y}_h^+) p(\mathbf{y}_h^+ | c_k)$. Substituting the definitions from (A.2) and (A.4), restating the summation as used in (A.6), grouping terms, and simplifying the expression produces the final result.

B Derivation of Class-Conditional M-Step Equations

LEMMA B.1. *The $p(y_j^+ | c_k)$ are maximized for solutions to the cubic equation:*

$$F_3 p(y_h^+ | c_k)^3 + F_2 p(y_h^+ | c_k)^2 + F_1 p(y_h^+ | c_k) + F_0 = 0.$$

where

$$\begin{aligned}
F_3 &= \frac{-2\gamma p(c_k)^2}{m^{+2}} (Q \ln |Y^-| - \alpha_{|Y^-|}) A_2(h, h), \\
F_2 &= \frac{2\gamma p(c_k)^2}{m^{+2}} (Q \ln |Y^-| - \alpha_{|Y^-|}) A_2(h, h) \\
&\quad + \frac{2 \ln |Y^-| \cdot Q p(c_k)}{m^{+2}} \left(\sum_{r,l:(r,l) \neq (k,h)} p(c_r) A_2(h, l) \theta_{lk} \right. \\
&\quad \left. + A_1(h) \right) \\
&\quad - \frac{2 \ln |Y^-| \cdot \beta_{|Y^-|} p(c_k)^2}{m^{+2}} \left(\sum_{l:l \neq h} A_2(h, l) \theta_{lk} + A_1(h) \right), \\
F_1 &= -(1 - \gamma) \sum_{\mathbf{y}_i \in \mathcal{Y}} q(c_k | \mathbf{y}_i) \\
&\quad + \frac{2 \ln |Y^-| \cdot Q p(c_k)}{m^{+2}} \left(\sum_{r,l:(r,l) \neq (k,h)} p(c_r) A_2(h, l) \theta_{lk} \right. \\
&\quad \left. + A_1(h) \right) \\
&\quad - \frac{2 \ln |Y^-| \cdot \beta_{|Y^-|} p(c_k)^2}{m^{+2}} \left(\sum_{l:l \neq h} A_2(h, l) \theta_{lk} + A_1(h) \right), \\
F_0 &= (1 - \gamma) \sum_{\mathbf{y}_i \in \mathcal{Y}} q(c_k | \mathbf{y}_i) y_{ih}^+.
\end{aligned}$$

The A terms are defined as:

$$\begin{aligned}
A_2(j, l) &= \sum_{\mathbf{y}_i^- \in \mathcal{Y}^-} (p(\mathbf{y}_i^- | y_j^+ = 1) - p(\mathbf{y}_i^- | y_j^+ = 0)) \\
&\quad \cdot (p(\mathbf{y}_i^- | y_l^+ = 1) - p(\mathbf{y}_i^- | y_l^+ = 0)), \\
A_1(j) &= \sum_{\mathbf{y}_i^- \in \mathcal{Y}^-} p(\mathbf{y}_i^- | y_l^+ = 0) \\
&\quad \cdot (p(\mathbf{y}_i^- | y_j^+ = 1) - p(\mathbf{y}_i^- | y_j^+ = 0)), \\
A_0 &= \sum_{\mathbf{y}_i^- \in \mathcal{Y}^-} \sum_{j,l=1}^{m^+} p(\mathbf{y}_i^- | y_j^+ = 0) p(\mathbf{y}_i^- | y_l^+ = 0).
\end{aligned}$$

B.1 Expanding the $p(\mathbf{y}_i^- | c_k)$ terms in the $\tilde{H}^1(C, Y^-)$ term from (3.14) Recall that $\tilde{H}^1(C, Y^-)$ is defined in (3.13) in terms of $p(\mathbf{y}_i^- | c_k)$. The parameters for the model, however, are $p(\mathbf{y}_i^+ | c_k)$. We now expand the $p(\mathbf{y}_i^- | c_k)$ according to (A.6) in order to obtain expressions explicitly in terms of $p(\mathbf{y}_i^+ | c_k)$. We will need

the following quantity which appears in $\tilde{H}^1(C, Y^-)$:

$$\begin{aligned}
(B.1) \quad & \sum_{\mathbf{y}_i^- \in Y^-} \sum_{k=1}^K p(\mathbf{y}_i^-, c_k) \cdot p(\mathbf{y}_i^-, c_k) \\
&= \sum_{\mathbf{y}_i^- \in Y^-} \sum_{k=1}^K p(c_k)^2 \frac{1}{m^{+2}} \sum_{j=1}^{m^+} \sum_{l=1}^{m^+} [(p_{ij}^1 - p_{ij}^0)(p_{il}^1 - p_{il}^0) \\
&\quad \cdot \theta_{jk} \theta_{lk} + (p_{il}^0(p_{ij}^1 - p_{ij}^0)) \theta_{jk} + (p_{ij}^0(p_{il}^1 - p_{il}^0)) \theta_{lk} \\
&\quad + (p_{ij}^0 p_{il}^0)] \\
&\quad \text{where we have substituted } p_{ij}^0 = p(\mathbf{y}_i^- | y_j^+ = 0), p_{ij}^1 = p(\mathbf{y}_i^- | y_j^+ = 1) \text{ and } \theta_{jk} = p(y_j^+ | c_k) \text{ for simplification. To further simplify the expression, we introduce constants } A_0, A_1, A_2 \text{ according to the definitions in (B.1) and obtain the expanded expression for } \tilde{H}^1(C, Y^-):
\end{aligned}$$

$$\begin{aligned}
(B.2) \quad \tilde{H}^1(C, Y^-) &= \alpha_{|Y^-|} - \beta_{|Y^-|} \sum_{k=1}^K p(c_k)^2 \frac{1}{m^{+2}} \\
&\quad \cdot \left\{ \sum_{j,l} A_2(j, l) \theta_{jk} \theta_{lk} + 2 \sum_j A_1(j) \theta_{jk} + A_0 \right\}.
\end{aligned}$$

B.2 Expanding the $p(\mathbf{y}_i^- | c_k)$ terms in the $H(Y^-)$ term of (3.14) In a similar fashion, one obtains the expanded version of $\tilde{H}^u(Y^-)$ from (3.13):

$$\begin{aligned}
(B.3) \quad \tilde{H}^1(Y^-) &= (\ln |Y^-|) \cdot \left(1 - Q \sum_{k=1}^K \sum_{r=1}^K p(c_k) p(c_r) \frac{1}{m^{+2}} \right. \\
&\quad \cdot \left\{ \sum_{j,l} A_2(j, l) \theta_{jk} \theta_{lr} + 2 \sum_j A_1(j) \theta_{jk} + A_0 \right\} - \frac{Q}{|Y^-|} \Bigg).
\end{aligned}$$

B.3 Finding $p(\mathbf{y}_i^+ | c_k)$ which maximize (3.14) The critical points of (3.14) may be obtained as follows: First, the expanded versions given in (B.2) and (B.3) are substituted into (3.14). Next, the standard variational approximation for EM is applied [for details, see [16]]. Finally, the result is differentiated with respect to θ_{hk} and equated to zero. The result is a cubic equation. This is because the $p(\mathbf{y}_i^+ | c_k)$ are linear in the derivatives of (B.2) and (B.3). The y_{ij}^+ appear as the exponent of $p(\mathbf{y}_i^+ | c_k)$ in (A.2) which itself appears in the first term of (3.14). Since the y_{ij}^+ are binary, we are guaranteed that after differentiating we obtain a cubic equation. Using the fact that $A_2(j, l) = A_2(l, j)$, and grouping terms results in the cubic equation in Lemma B.1.

Clustering With Constraints: Feasibility Issues and the k -Means Algorithm

Ian Davidson*

S. S. Ravi†

Abstract

Recent work has looked at extending the k -Means algorithm to incorporate background information in the form of instance level must-link and cannot-link constraints. We introduce two ways of specifying additional background information in the form of δ and ϵ constraints that operate on all instances but which can be interpreted as conjunctions or disjunctions of instance level constraints and hence are easy to implement. We present complexity results for the *feasibility* of clustering under each type of constraint individually and several types together. A key finding is that determining whether there is a feasible solution satisfying *all* constraints is, in general, **NP**-complete. Thus, an iterative algorithm such as k -Means should not try to find a feasible partitioning at each iteration. This motivates our derivation of a new version of the k -Means algorithm that minimizes the *constrained vector quantization error* but at each iteration does not attempt to satisfy all constraints. Using standard UCI datasets, we find that using constraints improves accuracy as others have reported, but we also show that our algorithm reduces the number of iterations until convergence. Finally, we illustrate these benefits and our new constraint types on a complex real world object identification problem using the infra-red detector on an Aibo robot.

Keywords: k -Means clustering, constraints.

1 Introduction and Motivation

The k -Means clustering algorithm is a ubiquitous technique in data mining due to its simplicity and ease of use (see for example, [6, 10, 16]). It is well known that k -Means converges to a local minimum of the vector quantization error and hence must be restarted many times, a computationally very expensive task when dealing with the large data sets typically found in data mining problems.

Recent work has focused on the use of background

information in the form of instance level must-link and cannot-link constraints. A must-link constraint enforces that two instances must be placed in the same cluster while a cannot-link constraint enforces that two instances must not be placed in the same cluster. We can divide previous work on clustering under constraints into two types: 1) Where the constraints help the algorithm learn a distortion/distance/objective function [4, 15] and 2) Where the constraints are used as “hints” to guide the algorithm to a useful solution [18, 19]. Philosophically, the first type of work makes the assumption that points surrounding a pair of must-link/cannot-link points should be close to/far from each other [15], while the second type just requires that the two points be in the same/different clusters. Our work falls into the second category. Recent examples of the second type of work include ensuring that constraints are satisfied at each iteration [19] and initializing algorithms so that constraints are satisfied [3]. The results of this type of work are quite encouraging; in particular, Wagstaff et al. [18, 19] illustrate that for simple classification tasks, k -Means with constraints obtains clusters with a significantly better purity (when measured on an extrinsic class label) than when not using constraints. Furthermore, Basu et al. [5] investigate determining the most informative set of constraints when the algorithm has access to an Oracle.

In this paper we carry out a formal analysis of clustering under constraints. Our work makes several pragmatic contributions.

- We introduce two new constraint types which act upon *groups* of instances. Roughly speaking, the ϵ -constraint enforces that each instance x in a cluster must have another instance y in the same cluster such that the distance between x and y is at most ϵ . We shall see that this can be used to enforce prior information with respect to how the data was collected. The δ -constraint enforces that every instance in a cluster must be at a distance of at least δ from every instance in every other cluster. We can use this type of constraint to specify background information on the minimum distance between the clusters/objects we hope to discover.
- We show that these two new constraints can be eas-

*Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: davidson@cs.albany.edu.

†Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: ravi@cs.albany.edu.

ily represented as a disjunction and conjunction of must-link constraints, thus making their implementation easy.

- We present complexity results for the feasibility problem under each of the constraints individually and in combination. The polynomial algorithms developed in this context can be used for initializing the k -Means algorithm. We show that in many situations, the feasibility problem (i.e., determining whether there is a solution that satisfies all the constraints) is **NP**-complete. Therefore, it is not advisable to attempt to find a solution that satisfies all the constraints at each iteration of a clustering algorithm.
- To overcome this difficulty, we illustrate that the k -Means clustering algorithm can be viewed as a two step algorithm with one step being to take the derivative of its error function and solving for the new centroid positions. With that in mind, we propose a new differentiable objective function that incorporates constraint violations and rederive a new constrained k -Means algorithm. This algorithm, like the original k -Means algorithm, is designed to monotonically decrease its error function.

We empirically illustrate our algorithm’s performance on several standard UCI data sets and show that adding must-link and cannot-link constraints not only helps accuracy but can help improve convergence. Finally, we illustrate the use of δ and ϵ constraints in clustering distances obtained by an Aibo robot’s infra-red detector for the purpose of object detection for path planning. For example, we can specify the width of the Aibo robot as δ ; that is, we are only interested in clusters/objects that are more than δ apart, since the robot can only physically move between such objects.

The outline of the paper is as follows. We begin by considering the feasibility of clustering under all four types constraints individually. The next section builds upon the feasibility analysis for combinations of constraints. A summary of the results for feasibility can be found in Table 1. We then discuss the derivation of our constrained k -Means algorithm. Finally, we present our empirical results.

2 Definitions of Constraints and the Feasibility Problem

Throughout this paper, we use the terms “instances” and “points” interchangeably. Let $S = \{s_1, s_2, \dots, s_n\}$ denote the given set of points which must be partitioned into K clusters, denoted by S_1, \dots, S_K . For any pair of points s_i and s_j in S , the distance between them is

denoted by $d(s_i, s_j)$. The distance function is assumed to be symmetric so that $d(s_i, s_j) = d(s_j, s_i)$. We consider the problem of clustering the set S under the following types of constraints.

(a) Must-Link Constraints: Each must-link constraint involves a pair of points s_i and s_j ($i \neq j$). In any feasible clustering, points s_i and s_j must be in the same cluster.

(b) Cannot-Link Constraints: Each cannot-link constraint also involves a pair of distinct points s_i and s_j . In any feasible clustering, points s_i and s_j *must not* be in the same cluster.

(c) δ -Constraint (or Minimum Separation Constraint): This constraint specifies a value $\delta > 0$. In any solution satisfying this constraint, the distance between any pair of points which are in two different clusters must be at least δ . More formally, for any pair of clusters S_i and S_j ($i \neq j$), and any pair of points s_p and s_q such that $s_p \in S_i$ and $s_q \in S_j$, $d(s_p, s_q) \geq \delta$. Informally, this constraint requires that each pair of clusters must be well separated. As will be seen in Section 3.4, a δ -constraint can be represented by a conjunction of instance level must-link constraints.

(d) ϵ -Constraint: This constraint specifies a value $\epsilon > 0$ and the feasibility requirement is the following: for any cluster S_i containing two or more points and for any point $s_p \in S_i$, there must be another point $s_q \in S_i$ such that $d(s_p, s_q) \leq \epsilon$. Informally, this constraint requires that in any cluster S_j containing two or more points, each point in S_j must have another point within a distance of at most ϵ . As will be seen in Section 3.5, an ϵ -constraint can be represented by a disjunction of instance level must-link constraints, with the proviso that when none of the must-link constraints is satisfied, the point is in a singleton cluster by itself. The ϵ -constraint is similar in principle to the ϵ +minpts criterion used in the DB-SCAN algorithm [11]. However, in our work, the aim is to minimize the constrained vector quantization error subject to the ϵ -constraint, while in DB-SCAN, their criterion is central to defining a cluster and an outlier.

Although the constraints discussed above provide a useful way to specify background information to a clustering algorithm, it is natural to ask whether there is a feasible clustering that satisfies all the given constraints. The complexity of satisfying the constraints will determine how to incorporate them into existing clustering algorithms.

The feasibility problem has been studied for other types of constraints or measures of quality [14]. For example, the clustering problem where the quality is measured by the maximum cluster diameter can be trans-

formed in an obvious way into a constrained clustering problem. Feasibility problems for such constraints have received a lot of attention in the literature (see for example [12, 13, 14]).

Typical specifications of clustering problems include an integer parameter K that gives the number of required clusters. We will consider a slightly more general version, where the problem specification includes a lower bound K_ℓ and an upper bound K_u on the number of clusters rather than the exact number K . Without upper and lower bounds, some feasibility problems may admit trivial solutions. For instance, if we consider the feasibility problem for a collection of must-link constraints (or for a δ -constraint) and there is no lower bound on the number of clusters, a trivial feasible solution is obtained by having all the points in a single cluster. Likewise, when there is no upper bound on the number of clusters for the feasibility problem under cannot-link constraints (or an ϵ -constraint), a trivial feasible solution is to make each point into a separate cluster. Obviously, K_ℓ and K_u must satisfy the condition $1 \leq K_\ell \leq K_u \leq n$, where n denotes the total number of points.

For the remainder of this paper, a feasible clustering is one that satisfies all the given constraints and the upper and lower bounds on the number of clusters. For problems involving must-link or cannot-link constraints, it is assumed that the collection of constraints $C = \{C_1, C_2, \dots, C_m\}$ containing the m constraints is given, where each constraint $C_j = \{s_{j_1}, s_{j_2}\}$ specifies a pair of points. For problems involving ϵ and δ constraints, it is assumed that the values of ϵ and δ are given. For convenience, the feasibility problems under constraints (a) through (d) defined above will be referred to as **ML-feasibility**, **CL-feasibility**, **δ -feasibility** and **ϵ -feasibility** respectively.

3 Complexity of Feasibility Problems

3.1 Overview In this section, we investigate the complexity of the feasibility problems by considering each type of constraint separately. In Section 4, we examine the complexity of feasibility problems for combinations of constraints. Our polynomial algorithms for feasibility problems do not require distances to satisfy the triangle inequality. Thus, they can also be used with nonmetric distances. On the other hand, the **NP**-completeness results show that the corresponding feasibility problems remain computationally intractable even when the set to be clustered consists of points in \mathbb{R}^2 .

The feasibility algorithms presented in Sections 3 and 4 focus on determining whether there is a partition that satisfies all the given constraints; they do not attempt to optimize any objective. Thus, these algorithms

may produce solutions with singleton clusters when the constraints under consideration permit such clusters.

3.2 Feasibility Under Must-Link Constraints

Klein et al. [15] showed that the ML-feasibility problem can be solved in polynomial time. They considered a more general version of the problem, where the goal is to obtain a new distance function that satisfies the triangle inequality when there is a feasible solution. In our definition of the ML-feasibility problem, no distances are involved. Therefore, a straightforward algorithm whose running time is linear in the number of points and constraints can be developed as discussed below.

As is well known, must-link constraints are transitive; that is, must-link constraints $\{s_i, s_j\}$ and $\{s_j, s_k\}$ imply the must-link constraint $\{s_i, s_k\}$. Thus, the two constraints can be combined into a single must-link constraint, namely $\{s_i, s_j, s_k\}$. Thus, a given collection C of must-link constraints can be transformed into an equivalent collection $M = \{M_1, M_2, \dots, M_r\}$ of constraints, by computing the transitive closure of C . The sets in M are pairwise disjoint and have the following interpretation: for each set M_i ($1 \leq i \leq r$), the points in M_i must all be in the same cluster in any feasible solution. For feasibility purposes, points which are not involved in any must-link constraint can be partitioned into clusters in an arbitrary manner. These facts allow us to obtain a straightforward algorithm for the ML-feasibility problem. The steps of the algorithm are shown in Figure 1. Whenever a feasible solution exists, the algorithm outputs a collection of K_ℓ clusters. The only situation in which the algorithm reports infeasibility is when the lower bound on the number of clusters is too high.

The transitive closure computation (Step 1 in Figure 1) in the algorithm can be carried out as follows. Construct an undirected graph G , with one node for each point appearing in the constraint sets C , and an edge between two nodes if the corresponding points appear together in a must-link constraint. Then, the connected components of G give the sets in the transitive closure. It can be seen that the graph G has n nodes and $O(m)$ edges. Therefore, its connected components can be found in $O(n + m)$ time [8]. The remaining steps of the algorithm can be carried out in $O(n)$ time. The following theorem summarizes the above discussion.

THEOREM 3.1. *Given a set of n points and m must-link constraints, the ML-feasibility problem can be solved in $O(n + m)$ time.* ■

3.3 Feasibility Under Cannot-Link Constraints

Klein et al. [15] mention that the CL-feasibility problem is **NP**-complete but omit the proof. Since we

Note: Whenever a feasible solution exists, the following algorithm outputs a collection of K_ℓ clusters satisfying all the must-link constraints.

1. Compute the transitive closure of the constraints in C . Let this computation result in r sets of points, denoted by M_1, M_2, \dots, M_r .
2. Let $S' = S - \bigcup_{i=1}^r M_i$. (S' denotes the subset of points that are not involved in any must-link constraint.)
3. **if** $r \geq K_\ell$ **then**
 - (a) Let $A = (\bigcup_{i=K_\ell}^r M_i) \cup S'$.
 - (b) Output $M_1, \dots, M_{K_\ell-1}, A$.
- else**
 - if** $|S'| < K_\ell - r$ **then**

Output “There is no solution.”
 - else**
 - (a) Let $t = K_\ell - r$. Partition S' into t clusters A_1, \dots, A_t arbitrarily.
 - (b) Output $M_1, \dots, M_r, A_1, \dots, A_t$.

Figure 1: Algorithm for the ML-Feasibility Problem

wish to draw some additional conclusions from the proof, we have included it in the appendix. The proof uses a straightforward reduction from the GRAPH K -COLORABILITY problem (K -COLOR).

It is known that the K -COLOR problem is **NP**-complete even for graphs in which the number of edges is linear in the number of nodes [12]. This fact in conjunction with the proof in the appendix implies that the CL-feasibility problem is computationally intractable even when the number of constraints is linear in the number of points. Further, the K -COLOR problem is known to be **NP**-complete for every *fixed* value of $K \geq 3$. From this fact, it follows that the CL-feasibility problem is also **NP**-complete when the lower bound on the number of clusters is 1 and the upper bound is fixed at any value ≥ 3 .

While **NP**-completeness result indicates that the CL-feasibility problem is at least as hard as the K -COLOR problem, it can also be shown that the feasibility problem is no harder than the K -COLOR problem as follows. For each point s_i , create a graph node v_i , and for each cannot-link constraint $\{s_i, s_j\}$, create the undirected edge $\{v_i, v_j\}$. It is easy to verify that the

-
1. **for** each point s_i **do**
 - (a) Determine the set $X_i \subseteq S - \{s_i\}$ of points such that for each point $x_j \in X_i$, $d(s_i, x_j) < \delta$.
 - (b) For each point $x_j \in X_i$, create the must-link constraint $\{s_i, x_j\}$.
 2. Let C denote the set of all the must-link constraints created in Step 1. Use the algorithm for the ML-feasibility problem (Figure 1) with point set S , constraint set C and the values K_ℓ and K_u .

Figure 2: Algorithm for the δ -Feasibility Problem

resulting graph is K -colorable iff there is a solution to the feasibility problem with at least 1 and at most K clusters. This reduction to the coloring problem points out that in practice, one can use known heuristics for graph coloring in choosing the number of clusters.

Although the coloring problem is known to be hard to approximate in the worst-case, heuristics that work well in practice are known (see for example [1, 7]). The reduction also shows that when the upper bound on the number of clusters is two, the CL-feasibility problem can be solved in polynomial time. This is because the problem of determining whether an undirected graph can be colored using at most two colors can be solved efficiently [8].

3.4 Feasibility Under δ -Constraint In this section, we show that the δ -feasibility problem can be solved in polynomial time. The basic idea is simple: in any feasible solution, every pair of points s_i and s_j for which $d(s_i, s_j) < \delta$, must be in the same cluster. Thus, given the value of δ , we can create a collection of appropriate must-link constraints and use the algorithm for the ML-feasibility problem. This shows that a δ -constraint can be replaced by a conjunction of must-link constraints. The steps of the resulting algorithm are shown in Figure 2.

The running time of the algorithm for δ -feasibility is dominated by the time needed to complete Step 1, that is, the time to compute the set of must-link constraints. Clearly, this step can be carried out in $O(n^2)$ time, and the number of must-link constraints generated is also $O(n^2)$. Thus, the overall running time of the algorithm is $O(n^2)$. The following theorem summarizes the above discussion.

THEOREM 3.2. *For any $\delta > 0$, the feasibility problem*

under the δ -constraint can be solved in $O(n^2)$ time, where n is the number of points to be clustered. ■

3.5 Feasibility Under ϵ -Constraint Let the set S of points and the value $\epsilon > 0$ be given. For any point $s_p \in S$, the set Γ_p of ϵ -neighbors is given by

$$\Gamma_p = \{s_q : s_q \in S - \{s_p\} \text{ and } d(s_p, s_q) \leq \epsilon\}.$$

Note that a point is *not* an ϵ -neighbor of itself. Two distinct points s_p and s_q are ϵ -neighbors of each other if $d(s_p, s_q) \leq \epsilon$. The ϵ -constraint requires that in any cluster containing two or more points, each point in the cluster must have an ϵ -neighbor within the same cluster. This observation points out that an ϵ -constraint corresponds to a disjunction of must-link constraints. For example, if $\{s_{i_1}, \dots, s_{i_r}\}$ denote the ϵ -neighbors of a point s_i , then satisfying the ϵ -constraint for point s_i means that either one or more of the must-link constraints $\{s_i, s_{i_1}\}, \dots, \{s_i, s_{i_r}\}$ are satisfied or the point s_i is in a singleton cluster by itself. In particular, any point in S which does not have an ϵ -neighbor must form a singleton cluster.

So, to determine feasibility under an ϵ -constraint for the set of points S , we first find the subset S_1 containing each point which does not have an ϵ -neighbor. Let $|S_1| = t$, and let C_1, C_2, \dots, C_t denote the singleton clusters formed from S_1 . To cluster the points in $S_2 = S - S_1$ (i.e., the set of points each of which has an ϵ -neighbor), it is convenient to use an auxiliary undirected graph defined below.

DEFINITION 3.1. *Let a set of points S and a value $\epsilon > 0$ be given. Let $Q \subseteq S$ be a set of points such that for each point in Q , there is an ϵ -neighbor in Q . The **auxiliary graph** $G(V, E)$ **corresponding to** Q is constructed as follows.*

- (a) *The node set V has one node for each point in Q .*
- (b) *For any two nodes v_p and v_q in V , the edge $\{v_p, v_q\}$ is in E if the points in Q corresponding to v_p and v_q are ϵ -neighbors.*

Let $G(V, E)$ denote the auxiliary graph corresponding to S_2 . Note that each connected component (CC) of G has at least two nodes. Thus, the CCs of G provide a way of clustering the points in S_2 . Let X_i , $1 \leq i \leq r$, denote the cluster formed from the i^{th} CC of G . The t singleton clusters together with these r clusters would form a feasible solution, provided the upper and lower bounds on the number of clusters are satisfied. So, we focus on satisfying these bounds.

If $S_2 = \emptyset$, then the minimum number of clusters is $t = |S_1|$, since each point in S_1 must be in a separate

singleton cluster. Otherwise, we need at least one additional cluster for the points in S_2 ; that is, the minimum number of clusters in that case is $t + 1$. Thus, the minimum number of clusters, denoted by N^* , is given by $N^* = t + \min\{1, r\}$. If $K_u < N^*$, there is no feasible solution. We may therefore assume that $K_u \geq N^*$.

As mentioned above, there is a solution with $t + r$ clusters. If $t + r \geq K_u$, then we can get K_u clusters by simply merging (if necessary) an appropriate number of clusters from the collection X_1, X_2, \dots, X_r into a single cluster. Since each CC of G has at least two points, this merging step will not violate the ϵ -constraint.

The only remaining possibility is that the value $t + r$ is smaller than the lower bound K_ℓ . In this case, we can increase the number of clusters to K_ℓ by splitting some of the clusters X_1, X_2, \dots, X_r to form more clusters. One simple way to increase the number of clusters by one is to create a new singleton cluster by taking one point away from some cluster X_i with two or more points. To facilitate this, we construct a spanning tree for each CC of G . The advantage of having trees is that we can remove a leaf node from a tree and make the point corresponding to that node into a new singleton cluster. Since each tree has at least two leaves and removing a leaf will not disconnect a tree, this method will increase the number of clusters by exactly one at each step. Thus, by repeating the step an appropriate number of times, the number of clusters can be made equal to K_ℓ .

The above discussion leads to the feasibility algorithm shown in Figure 3. It can be seen that the running time of the algorithm is dominated by the time needed for Steps 1 and 2. Step 1 can be implemented to run in $O(n^2)$ time by finding the ϵ -neighbor set for each point. Since the number of ϵ -neighbors for each point in S_2 is at most $n - 1$, the construction of the auxiliary graph and finding its CCs (Step 2) can also be done in $O(n^2)$ time. So, the overall running time of the algorithm is $O(n^2)$. The following theorem summarizes the above discussion.

THEOREM 3.3. *For any $\epsilon > 0$, the feasibility problem under the ϵ -constraint can be solved in $O(n^2)$ time, where n is the number of points to be clustered. ■*

4 Feasibility Under Combinations of Constraints

4.1 Overview In this section, we consider the feasibility problem under combinations of constraints. Since the CL-feasibility problem is **NP-hard**, the feasibility problem for any combination of constraints involving cannot-link constraints is, in general, computationally

-
1. Find the set $S_1 \subseteq S$ such that no point in S_1 has an ϵ -neighbor. Let $t = |S_1|$ and $S_2 = S - S_1$.
 2. Construct the auxiliary graph $G(V, E)$ for S_2 (see Definition 3.1). Let G have r connected components (CCs) denoted by G_1, G_2, \dots, G_r .
 3. Let $N^* = t + \min\{1, r\}$. (**Note:** To satisfy the ϵ -constraint, at least N^* clusters must be used.)
 4. **if** $N^* > K_u$ **then** Output “No feasible solution” and **stop**.
 5. Let C_1, C_2, \dots, C_t denote the singleton clusters corresponding to points in S_1 . Let X_1, X_2, \dots, X_r denote the clusters corresponding to the CCs of G .
 6. **if** $t + r \geq K_u$
 - then** /* We may have too many clusters. */
 - (a) Merge clusters $X_{K_u-t}, X_{K_u-t+1}, \dots, X_r$ into a single new cluster X_{K_u-t} .
 - (b) Output the K_u clusters $C_1, C_2, \dots, C_t, X_1, X_2, \dots, X_{K_u-t}$.
 - else** /* We have too few clusters. */
 - (a) Let $N = t + r$. Construct spanning trees T_1, T_2, \dots, T_r corresponding to the CCs of G .
 - (b) **while** $(N < K_\ell)$ **do**
 - (i) Find a tree T_i with at least two nodes. If no such tree exists, output “No feasible solution” and **stop**.
 - (ii) Let v be a leaf in tree T_i . Delete v from T_i .
 - (iii) Delete the point corresponding to v from cluster X_i and form a new singleton cluster X_{N+1} containing that point.
 - (iv) $N = N + 1$.
 - (c) Output the K_ℓ clusters $C_1, C_2, \dots, C_t, X_1, X_2, \dots, X_{K_\ell-t}$.

Figure 3: Algorithm for the ϵ -Feasibility Problem

intractable. So, we need to consider only the combinations of must-link, δ and ϵ constraints. We show that the feasibility problem remains efficiently solvable when both a must-link constraint and a δ constraint are considered together as well as when δ and ϵ constraints are considered together. When must-link constraints are considered together with an ϵ constraint, we show that the feasibility problem is **NP**-complete. This result points out that when must-link, δ and ϵ constraints are all considered together, the resulting feasibility problem is also **NP**-complete in general.

4.2 Combination of Must-Link and δ Constraints We begin by considering the combination of must-link constraints and a δ constraint. As mentioned in Section 3.4, the effect of the δ -constraint is to contribute a collection of must-link constraints. Thus, we can merge these must-link constraints with the given must-link constraints, and then ignore the δ -constraint. The resulting feasibility problem involves only must-link constraints. Hence, we can use the algorithm from Section 3.2 to solve the feasibility problem in polynomial time. For a set of n points, the δ constraint may contribute at most $O(n^2)$ must-link constraints. Further, since each given must-link constraint involves two points, we may assume that the number of given must-link constraints is also $O(n^2)$. Thus, the total number of must-link constraints due to the combination is also $O(n^2)$. Thus, the following result is a direct consequence of Theorem 3.1.

THEOREM 4.1. *Given a set of n points, a value $\delta > 0$ and a collection C of must-link constraints, the feasibility problem for the combination of must-link and δ constraints can be solved in $O(n^2)$ time.* ■

4.3 Combination of Must-Link and ϵ Constraints Here, we show that the feasibility problem for the combination of must-link and ϵ constraints is **NP**-complete. To prove this result, we use a reduction from the following problem which is known to be **NP**-complete [9].

PLANAR EXACT COVER BY 3-SETS (PX3C)

Instance: A set $X = \{x_1, x_2, \dots, x_n\}$, where $n = 3q$ for some positive integer q and a collection $T = \{T_1, T_2, \dots, T_m\}$ of subsets of X such that $|T_i| = 3$, $1 \leq i \leq m$. Each element $x_i \in X$ appears in at most three sets in T . Further, the bipartite graph $G(V_1, V_2, E)$, where V_1 and V_2 are in one-to-one correspondence with the elements of X and the 3-element sets in T respectively, and an edge $\{u, v\} \in E$ iff the element corresponding to u appears in the set corresponding to v , is planar.

Question: Does T contain a subcollection $T' = \{T_{i_1}, T_{i_2}, \dots, T_{i_q}\}$ with q sets such that the union of the sets in T' is equal to X ?

For reasons of space, we have included only a sketch of this **NP**-completeness proof.

THEOREM 4.2. *The feasibility problem under the combination of must-link and ϵ constraints is **NP**-complete.*

Proof sketch: It is easy to verify the membership of the problem in **NP**. To prove **NP**-hardness, we use a reduction from PX3C. Consider the planar (bipartite) graph G associated with the PX3C problem instance. From the specification of the problem, it follows that each node of G has a degree of at most three. Every planar graph with N nodes and maximum node degree three can be embedded on an orthogonal $N \times 2N$ grid such that the nodes of the graph are grid points, each edge of the graph is a path along the grid, and no two edges share a grid point except for the grid points corresponding to the graph vertices. Moreover, such an embedding can be constructed in polynomial time [17]. The points and constraints for the feasibility problem are created from this embedding.

Using suitable scaling, assume that the each grid edge is of length 1. Note that each edge e of G joins a set node to an element node. Consider the path along the grid for each edge of G . Introduce a new point in the middle of each grid edge in the path. Thus, for each edge e of G , this provides the set S_e of points in the grid path corresponding to e , including the new middle point for each grid edge. The set of points in the feasibility instance is the union of the sets S_e , $e \in E$. Let S'_e be obtained from S_e by deleting the point corresponding to the element node of the edge e . For each edge e , we introduce a must-link constraint involving all the points in S'_e . We also introduce a must-link constraint involving all the points corresponding to the elements nodes of G . We choose the value of ϵ to be $1/2$. The lower bound on the number of clusters is set to $m - n/3 + 1$ and the upper bound is set to m . It can be shown that there is a solution to the PX3C problem instance if and only if there is a solution to the feasibility problem. ■

4.4 Combination of δ and ϵ Constraints In this section, we show that the feasibility problem for the combination of δ and ϵ constraints can be solved in polynomial time. It is convenient to consider this problem under two cases, namely $\delta \leq \epsilon$ and $\delta > \epsilon$. For reasons of space, we will discuss the algorithm for the first case and mention the main idea for the second case.

For the case when $\delta \leq \epsilon$, our algorithm is based

on the following simple observation: Any pair of points which are separated by a distance less than δ are also ϵ -neighbors of each other. This observation allows us to reduce the feasibility problem for the combination to one involving only the ϵ constraint. This is done as follows. Construct the auxiliary undirected graph $G(V, E)$, where V is in one-to-one correspondence with the set of points and an edge $\{x, y\} \in E$ iff the distance between the points corresponding to nodes x and y is less than δ . Suppose C_1, C_2, \dots, C_r denote the connected components (CCs) of G . Consider any CC, say C_i , with two or more nodes. By the above observation, the ϵ -constraint is satisfied for all the points corresponding to the nodes in C_i . Thus, we can “collapse” each such set of points into a single “super point”. Let $S' = \{P_1, \dots, P_r\}$ denote the new set of points, where P_i is a super point if the corresponding CC C_i has two or more nodes, and P_i is a single point otherwise. Given the distance function d for the original set of points S , we define a new distance function d' for S' as follows:

$$d'(P_i, P_j) = \min\{d(s_p, s_q) : s_p \in P_i, s_q \in P_j\}.$$

With this new distance function, we can ignore the δ constraint. We need only check whether there is a feasible clustering of the set S' under the ϵ -constraint using the new distance function d' . Thus, we obtain a polynomial time algorithm for the combination of ϵ and δ constraints when $\delta \leq \epsilon$. It is not hard to see that the resulting algorithm runs in $O(n^2)$ time.

For the case when $\delta > \epsilon$, any pair of ϵ -neighbors must be in the same cluster. Using this idea, it is possible to construct a collection of must-link constraints corresponding to the given δ and ϵ constraints, and solve the feasibility problem in $O(n^2)$ time.

The following theorem summarizes the above discussion.

THEOREM 4.3. *Given a set of n points and values $\delta > 0$ and $\epsilon > 0$, the feasibility problem for the combination of δ and ϵ constraints can be solved in $O(n^2)$ time. ■*

The complexity results for various feasibility problems are summarized in Table 1. For each type of constraint, the table indicates whether the feasibility problem is in **P** (i.e., efficiently solvable) or **NP**-complete. Results for which no references are cited are from this paper.

5 A Derivation of a Constrained k -Means Algorithm

In this section we derive the k -Means algorithm and then derive a new constrained version of the algorithm

Constraint	Complexity
Must-Link	P [15]
Cannot-Link	NP-Complete [15]
δ -constraint	P
ϵ -constraint	P
Must-Link and δ	P
Must-Link and ϵ	NP-complete
δ and ϵ	P

Table 1: Results for Feasibility Problems

from first principles. Let C_j be the centroid of the j^{th} cluster and Q_j be the set of instances that are closest to the j^{th} cluster.

It is well known that the error function of the k -Means *problem* is the vector quantization error (also referred to as the distortion) given by the following equations.

$$(5.1) \quad VQE = \sum_{j=1}^k VQE_j$$

$$(5.2) \quad VQE_j = \frac{1}{2} \sum_{s_i \in Q_j} (C_j - s_i)^2$$

The k -Means algorithm is an iterative algorithm which in every step attempts to further minimize the distortion. Given a set of cluster centroids, the algorithm assigns instances to their nearest centroid which of course minimizes the distortion. This is step 1 of the algorithm. Step 2 is to recalculate the cluster centroids so as to minimize the distortion. This can be achieved by taking the first order derivative of the error (Equation (5.2)) with respect to the j^{th} centroid and setting it to zero. A solution to the resulting equation gives us the k -Means centroid update rule as shown in Equation (5.3).

$$(5.3) \quad \frac{d(VQE_j)}{d(C_j)} = \sum_{s_i \in Q_j} (C_j - s_i) = 0$$

$$(5.4) \quad C_j = \sum_{s_i \in Q_j} s_i / |Q_j|$$

Recall that Q_j is the set of points closest to the centroid of the j^{th} cluster.

Iterating through these two steps therefore monotonically decreases the distortion. However, the algorithm is prone to getting stuck in local minima. Nevertheless, good pragmatic results can be obtained, hence

the algorithm's popularity. We acknowledge that a more complicated analysis of the algorithm exists, namely an interpretation of the algorithm as analogous to the Newton-Raphson algorithm [2]. Our derivation of a new constrained version of k -Means is not inconsistent with these other explanations.

The key step in deriving a constrained version of k -Means is to create a new *differentiable* error function which we call the *constrained* vector quantization error. Consider a collection of r must-link constraints and s cannot-link constraints. We can represent the instances affected by these constraints by two functions $g(i)$ and $g'(i)$. These functions return the *cluster index* (i.e., a value in the range 1 through k) of the closest cluster to the 1^{st} and 2^{nd} instances governed by the i^{th} constraint. For clarity of notation, we assume that the instance associated with the function $g'(i)$ violates the constraint if at all. The new error function is shown in Equation (5.5).

$$(5.5) \quad CVQE_j = \frac{1}{2} \sum_{s_i \in Q_j} T_{j,1} + \frac{1}{2} \sum_{l=1, g(l)=j}^{s+r} (T_{j,2} \times T_{j,3})$$

where

$$\begin{aligned} T_{j,1} &= (C_j - s_i)^2 \\ T_{j,2} &= [(C_j - C_{g'(l)})^2 \neg \Delta(g'(l), g(l))]^{m_l} \\ T_{j,3} &= [(C_j - C_{h(g'(l))})^2 \Delta(g(l), g'(l))]^{1-m_l}. \end{aligned}$$

Here $h(i)$ returns the index of the cluster (other than i) whose centroid is closest to cluster i 's centroid and Δ is the Kronecker Delta Function defined by $\Delta(x, y) = 1$ if $x = y$ and 0 otherwise. We use $\neg \Delta$ to denote the negation of the Delta function.

The first part of the new error function is the regular distortion. The remaining terms are the errors associated with the must-link ($m_l=1$) and cannot-link ($m_l=0$) constraints. In future work we intend to allow the parameter m_l to take any value in $[0, 1]$ so that we can allow for a continuum between must-link and cannot-link constraints. We see that if a must-link constraint is violated then the cost is equal to the distance between the cluster centroids containing the two instances that should have been in the same cluster. Similarly, if a cannot-link constraint is violated the cost is the distance between the cluster centroid both instances are in and the nearest cluster centroid to one of the instances. Note that both violation costs are in units of distance, as is the regular distortion.

The first step of the constrained k -Means algorithm must minimize the new constrained vector quantization error. This is achieved by assigning instances so as to minimize the new error term. For instances that are not part of constraints, this involves as before, performing a nearest cluster centroid calculation. For pairs of instances in a constraint, for each possible combination of cluster assignments, the $CVQE$ is calculated and the instances are assigned to the clusters that minimally increases the $CVQE$.

The second step is to update the cluster centroids so as to minimize the constrained vector quantization error. To achieve this we take the first order derivative of the error, set to zero, and solve. By setting the appropriate values of m_l we can derive the update rules (Equation (5.6)) for the must-link and cannot-link constraint violations. The resulting equations are shown below.

$$\begin{aligned} \frac{d(CVQE_j)}{d(C_j)} &= \sum_{s_i \in Q_j} (C_j - s_i) + \\ &\quad \sum_{l=1, g(l)=j, \Delta(g(l), g'(l))=0}^s (C_j - C_{g'(l)}) + \\ &\quad \sum_{l=s+1, g(l)=j, \Delta(g(l), g'(l))=1}^{s+r} (C_j - C_{h(g'(l))}) \\ &= 0 \end{aligned}$$

Solving for C_j , we get

$$(5.6) \quad C_j = Y_j / Z_j$$

where

$$\begin{aligned} Y_j &= \sum_{s_i \in Q_j} s_i + \sum_{l=1, g(l)=j, \Delta(g(l), g'(l))=0}^s C_{g'(l)} + \\ &\quad \sum_{l=s+1, g(l)=j, \Delta(g(l), g'(l))=1}^{s+r} C_{h(g'(l))} \end{aligned}$$

and

$$\begin{aligned} Z_j &= |Q_j| + \sum_{g(l)=j, l=1}^s (1 - \Delta(g(l), g'(l))) + \\ &\quad \sum_{g(l)=j, l=s+1}^{s+r} \Delta(g(l), g'(l)) \end{aligned}$$

The intuitive interpretation of the centroid update rule is that if a must-link constraint is violated, the cluster centroid is moved towards the other cluster containing the other point. Similarly, the interpretation of

the update rule for a cannot-link constraint violation is that cluster centroid containing both constrained instances should be moved to the nearest cluster centroid so that one of the instances eventually gets assigned to it, thereby satisfying the constraint.

6 Experiments with Minimization of the Constrained Vector Quantization Error

In this section we compare the usefulness of our algorithm on three standard UCI data sets. We report the average over 100 random restarts (initial assignments of instances). As others have reported [19] for $k=2$ the addition of constraints improves the purity of the clusters with respect to an extrinsic binary class not given to the clustering algorithm. We observed similar behavior in our experiments. We are particularly interested in using our algorithm for more traditional unsupervised learning where $k > 2$. To this end we compare regular k -Means against constrained k -Means with respect to the number of iterations until convergence and the number of constraints violated. For the PIMA data set (Figure 4) which contains two extrinsic classes, we created a random combination of 100 must-link and 100 cannot-link constraints between the same and different classes respectively. Our results show that our algorithm converged on average in 25% fewer iterations while satisfying the vast majority of constraints.

Similar results were obtained for the BreastCancer data sets (Figure 6) where 25 must-link and 25 cannot-link constraints were used and Iris (Figure 8) where 13 must-link and 12 cannot-link constraints were used.

7 Experiments with the Sony Aibo Robot

In this section we describe an example with the Sony Aibo robot to illustrate the use of our newly proposed δ and ϵ constraints.

The Sony Aibo robot is effectively a walking computer with sensing devices such as a camera, microphone and an infra-red distance detector. The on-board processor has limited processing capability. The infra-red distance detector is connected to the end of the head and can be moved around to build a distance map of a scene. Consider the scene (taken from a position further back than the Aibo was for clarity) shown in Figure 5. We wish to cluster the distance information from the infra-red sensor to form objects/clusters (spatially adjacent points at a similar distance) which represent solid objects that must be navigated around.

Unconstrained clustering of this data with $k=9$ yields the set of significant (large) objects shown in Figure 7.

However, this result does not make use of important background information. Firstly, groups of clusters

Figure 4: Performance of regular and constrained k -Means on the UCI PIMA dataset

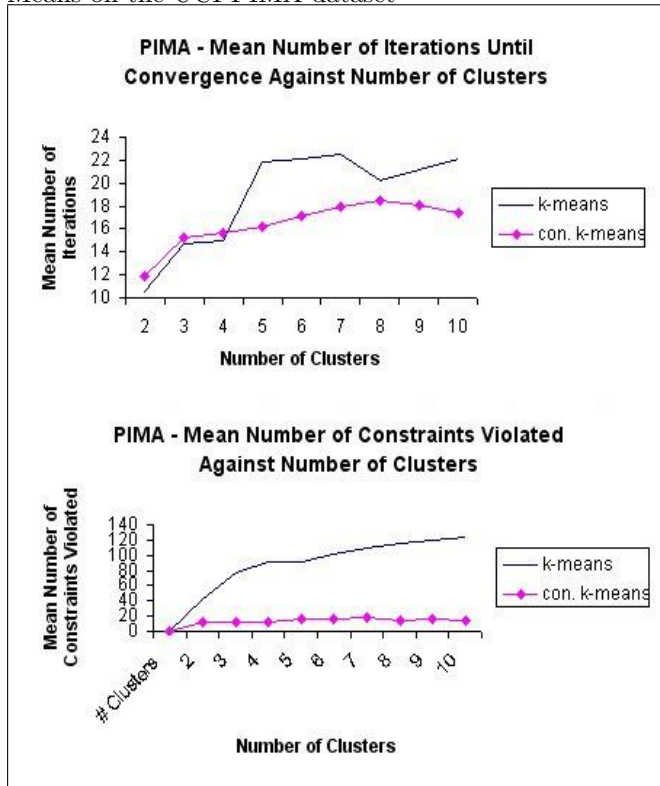


Figure 6: Performance of regular and constrained k -Means on the UCI Breast Cancer dataset

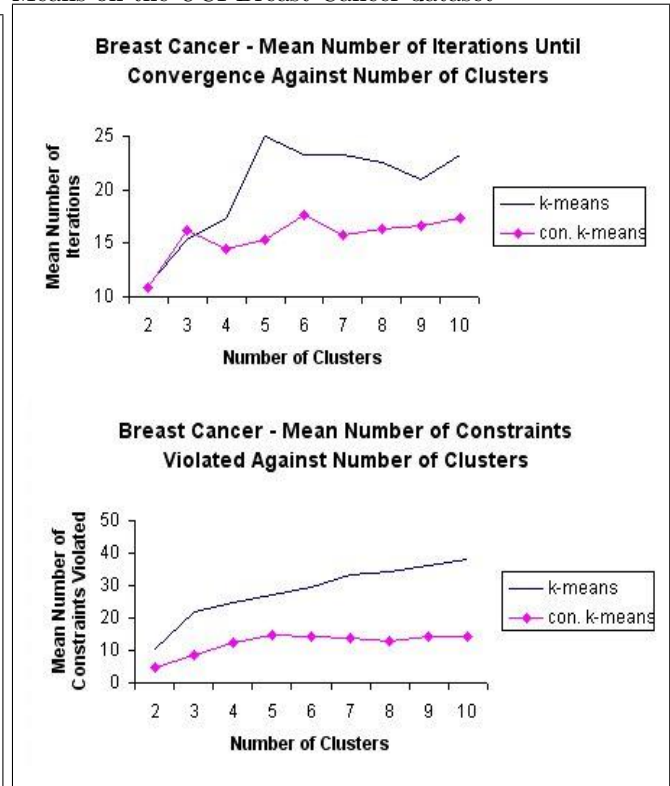


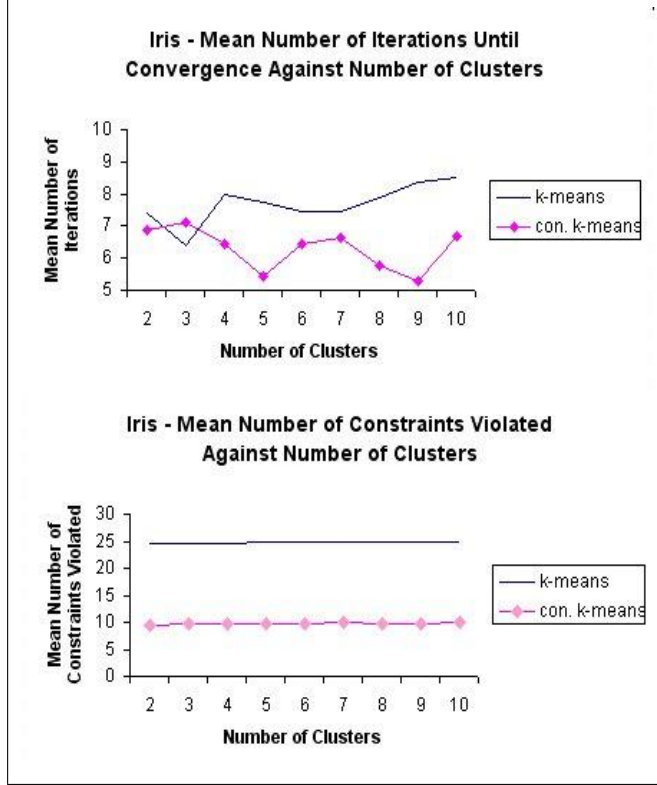
Figure 5: An image of the scene to navigate



Figure 7: Unconstrained clustering of the distance map using $k=9$. The approximate locations of significant (large) clusters are shown by the ellipses.



Figure 8: Performance of regular and constrained k -Means on the UCI Iris dataset



(such as the people) that are separated by a distance less than one foot can effectively be treated as one big cluster since the Aibo robot cannot easily walk through a gap smaller than one foot. Secondly, often one contiguous region is split into multiple objects due to errors in the infra-red sensor. These errors are due to poor reception of the incoming signal or the inability to reflect the outgoing signal which occurs in the wooden floor region of the scene. These errors are common in the inexpensive sensor on the Aibo robot.

Since we know the accuracy of the Aibo head movement we can determine the distance between the furthest (about three feet) adjacent readings which determines a lower bound for ϵ (namely, $3 \times \tan 1^\circ$) so as to ensure a feasible solution. However, if we believe that it is likely that one but unlikely that two incorrect adjacent mis-readings occur, then we can set ϵ to be twice this lower bound to overcome noisy observations.

Using these values for our constraints we can cluster the data under our background information and obtain the clustering results shown in Figure 9. We see that the clustering result is now more useful for our intended purpose. The Aibo can now move towards the area/cluster that represents an open area.

Figure 9: Constrained clustering of the distance map using $k=9$. The approximate locations of significant (large) clusters are shown by the ellipses.



8 Conclusion and Future Work

Clustering with background prior knowledge offers much promise with contributions using must-link and cannot-link instance level constraints having already been published. We introduced two additional constraints: ϵ and δ . We studied the computational complexity of finding a *feasible* solution for these four constraint types individually and together. Our results show that in many situations, finding a feasible solution under a combination of constraints is **NP**-complete. Thus, an iterative algorithm should not try to find a feasible solution in each iteration.

We derived from first principles a constrained version of the k -Means algorithm that attempts to minimize the proposed constrained vector quantization error. We find that the use of constraints with our algorithm results in faster convergence and the satisfaction of a vast majority of constraints. When constraints are not satisfied it is because it is less costly to violate the constraint than to satisfy it by assigning two quite different (i.e. far apart in Euclidean space) instances to the same cluster in the case of must-link constraints. Future work will explore modifying hierarchical clustering algorithms to efficiently incorporate constraints.

Finally, we showed the benefit of our two newly proposed constraints in a simple infra-red distance clustering problem. The δ constraint allows us to specify the minimum distance between clusters and hence can encode prior knowledge regarding the spatial domain. The ϵ constraint allows us to specify background information with regard to sensor error and data collection.

References

- [1] A. Hertz and D. de Werra, "Using Tabu Search Tech-

- niques for Graph Coloring”, *Computing*, Vol. 39, 1987, pp. 345–351.
- [2] L. Bottou and Y. Bengio, “Convergence Properties of the K -Means Algorithms”, *Advances in Neural Information Processing Systems*, Vol. 7, Edited by G. Tesauro and D. Touretzky and T. Leen, MIT Press, Cambridge, MA, 1995, pp. 585–592.
- [3] S. Basu, A. Banerjee and R. J. Mooney, “Semi-supervised Learning by Seeding”, *Proc. 19th Intl. Conf. on Machine Learning (ICML-2002)*, Sydney, Australia, July 2002, pp. 19–26.
- [4] S. Basu, M. Bilenko and R. J. Mooney, “A Probabilistic Framework for Semi-Supervised Clustering”, *Proc. 10th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD-2004)*, Seattle, WA, August 2004.
- [5] S. Basu, M. Bilenko and R. J. Mooney, “Active Semi-Supervision for Pairwise Constrained Clustering”, *Proc. 4th SIAM Intl. Conf. on Data Mining (SDM-2004)*.
- [6] P. S. Bradley and U. M. Fayyad, “Refining initial points for K -Means clustering”, *Proc. 15th Intl. Conf. on Machine Learning (ICML-1998)*, 1998, pp. 91–99.
- [7] G. Campers, O. Henkes and J. P. Leclercq, “Graph Coloring Heuristics: A Survey, Some New Propositions and Computational Experiences on Random and Leighton’s Graphs”, in *Proc. Operational Research ’87*, Buenos Aires, 1987, pp. 917–932.
- [8] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, Cambridge, MA, 2001.
- [9] M. E. Dyer and A. M. Frieze, “Planar 3DM is NP-Complete”, *J. Algorithms*, Vol. 7, 1986, pp. 174–184.
- [10] I. Davidson and A. Satyanarayana, “Speeding up K -Means Clustering Using Bootstrap Averaging”, *Proc. IEEE ICDM 2003 Workshop on Clustering Large Data Sets*, Melbourne, FL, Nov. 2003, pp. 16–25.
- [11] M. Ester, H. Kriegel, J. Sander and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining (KDD-96)*, Portland, OR, 1996, pp. 226–231.
- [12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.
- [13] T. F. Gonzalez, “Clustering to Minimize the Maximum Intercluster Distance”, *Theoretical Computer Science*, Vol. 38, No. 2-3, June 1985, pp. 293–306.
- [14] P. Hansen and B. Jaumard, “Cluster Analysis and Mathematical Programming”, *Mathematical Programming*, Vol. 79, Aug. 1997, pp. 191–215.
- [15] D. Klein, S. D. Kamvar and C. D. Manning, “From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering”, *Proc. 19th Intl. Conf. on Machine Learning (ICML 2002)*, Sydney, Australia, July 2002, pp. 307–314.
- [16] D. Pelleg and A. Moore, “Accelerating Exact k -means Algorithms with Geometric Reasoning”, *Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, Aug. 1999, pp. 277–281.
- [17] R. Tamassia and I. Tollis, “Planar Grid Embedding in Linear Time”, *IEEE Trans. Circuits and Systems*, Vol. CAS-36, No. 9, Sept. 1989, pp. 1230–1234.
- [18] K. Wagstaff and C. Cardie, “Clustering with Instance-Level Constraints”, *Proc. 17th Intl. Conf. on Machine Learning (ICML 2000)*, Stanford, CA, June–July 2000, pp. 1103–1110.
- [19] K. Wagstaff, C. Cardie, S. Rogers and S. Schroedl, “Constrained K -means Clustering with Background Knowledge”, *Proc. 18th Intl. Conf. on Machine Learning (ICML 2001)*, Williamstown, MA, June–July 2001, pp. 577–584.

9 Appendix

Here, we show that the feasibility problem for cannot-link constraints (CL-feasibility) is **NP**-complete using a reduction from the GRAPH K -COLORABILITY problem (K -COLOR) [12].

GRAPH K -COLORABILITY (K -COLOR)

Instance: Undirected graph $G(V, E)$, integer $K \leq |V|$.

Question: Can the nodes of G be colored using at most K colors so that for every pair of adjacent nodes u and v , the colors assigned to u and v are different?

THEOREM 9.1. *The CL-feasibility problem is **NP**-complete.*

Proof: It is easy to see that the CL-feasibility problem is in **NP**. To prove **NP**-hardness, we use a reduction from the K -COLOR problem. Let the given instance I of K -COLOR problem consist of undirected graph $G(V, E)$ and integer K . Let $n = |V|$ and $m = |E|$. We construct an instance I' of the CL-feasibility problem as follows. For each node $v_i \in V$, we create a point s_i , $1 \leq i \leq n$. (The coordinates of the points are not specified as they play no role in the proof.) The set S of points is given by $S = \{s_1, s_2, \dots, s_n\}$. For each edge $\{v_i, v_j\} \in E$, we create the cannot-link constraint $\{s_i, s_j\}$. Thus, we create a total of m constraints. We set the lower and upper bound on the number clusters to 1 and K respectively. This completes the construction of the instance I' . It is obvious that the construction can be carried out in polynomial time. It is straightforward to verify that the CL-feasibility instance I' has a solution if and only if the K -COLOR instance I has a solution. ■

A Cutting Algorithm for the Minimum Sum-of-Squared Error Clustering

Jiming Peng*

Yu Xia†

Abstract

The minimum sum-of-squared error clustering problem is shown to be a concave continuous optimization problem whose every local minimum solution must be integer. We characterize its local minima. A procedure of moving from a fractional solution to a better integer solution is given. Then we adapt Tuy's convexity cut method to find a global optimum of the minimum sum-of-squared error clustering problem. We prove that this method converges in finite steps to a global minimum. Promising numerical examples are reported.

1 Introduction.

Clustering (or *cluster analysis*) is one of the basic tools in data analysis. In this paper, we consider clustering based on minimum within-group sum-of-squared error criterion.

Many early studies on minimum sum-of-squared error clustering (or MSSC in brief) were focused on the well-known K-means algorithm [5, 13, 15] and its variants (see [12] for a survey). Usually, these methods can only reach a local solution, not a global minimum of the distortion function. From a theoretical viewpoint, the minimum sum-of-squared error clustering problem can be formulated as a nonlinear integer programming model. In [7], Hansen and Jaumard give a review on optimization approaches to some general clustering problems. There are some attempts to solve the MSSC problem exactly through mathematical programming; however, only numerical examples on data sets with less than 200 samples are reported.

Next, we will briefly describe some mathematical models for the minimum sum-of-squared error clustering

problem and introduce our algorithm.

1.1 Problem description. To partition n entities into k groups, people usually cast an entity into a vector in a Euclidean space: $\mathbf{a}_i = ((a_i)_1, \dots, (a_i)_d)^T \in \mathbb{R}^d$ ($i = 1, \dots, n$), where d is the number of attributes of the entity. Although coordinates of different points may be the same, we assume that there are at least $k + 1$ different points; otherwise, one only needs to group points with same coordinates together. Below are some mathematical programming models for the minimum sum-of-squared error clustering problem.

Bi-level program The objective of MSSC can be described as finding k representatives of the clusters \mathbf{c}_i ($i = 1, \dots, k$), and an assignment of the n entities to the k representatives such that the total sum-of-squared errors within each cluster, i.e. the sum of squared Euclidean distance from each point to its cluster representative, is minimum. This problem can be represented as the following bi-level programming model (see for instance [14]).

$$(1.1) \quad \min_{\mathbf{c}_1, \dots, \mathbf{c}_k} \sum_{i=1}^n \min\{\|\mathbf{a}_i - \mathbf{c}_1\|_2^2, \dots, \|\mathbf{a}_i - \mathbf{c}_k\|_2^2\}.$$

This model is not easy to solve.

Mixed integer program The bi-level program is equivalent to partitioning the n points into k groups, and then for each group finding a representative such that the total within-group sum-of-squared Euclidean distances is minimized. Define the assignment matrix $X = [x_{ij}] \in \mathbb{R}^{n \times k}$ as

$$x_{ij} \stackrel{\text{def}}{=} \begin{cases} 1 & \mathbf{a}_i \text{ assigned to } j\text{th group;} \\ 0 & \text{otherwise.} \end{cases}$$

Then (1.1) can be transformed to

*Advanced optimization Lab, Department of Computing and Software McMaster University, Hamilton, Ontario L8S 4K1, Canada (pengj@mcmaster.ca). Research partially supported by the grant # RPG 249635-02 of the National Sciences and Engineering Research Council of Canada (NSERC) and a PREA award. This research was also supported by the MITACS project "New Interior Point Methods and Software for Convex Conic-Linear Optimization and Their Application to Solve VLSI Circuit Layout Problems".

†The Institute of Statistical Mathematics, 4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569, Japan (yuxia@ism.ac.jp). Research supported in part through JSPS (Japan Society for the Promotion of Science).

(1.2a)

$$\min_{x_{ij}, \mathbf{c}_j} \sum_{j=1}^k \sum_{i=1}^n x_{ij} \|\mathbf{a}_i - \mathbf{c}_j\|_2^2$$

$$(1.2b) \quad \text{s.t.} \quad \sum_{j=1}^k x_{ij} = 1 \quad (i = 1, \dots, n)$$

$$(1.2c) \quad \sum_{i=1}^n x_{ij} \geq 1 \quad (j = 1, \dots, k)$$

$$(1.2d) \quad x_{ij} \in \{0, 1\} \quad (i = 1, \dots, n; j = 1, \dots, k).$$

The constraint (1.2b) ensures that each point \mathbf{a}_i is assigned to one and only one group. It can be replaced by

$$(1.3) \quad \sum_{j=1}^k x_{ij} \geq 1 \quad (i = 1, \dots, n),$$

since the objective is minimization. And (1.2c) ensures that there are exactly k clusters. We will prove that this constraint is redundant later. In addition, for a cluster j , given x_{ij} ($i = 1, \dots, n$) — x_{ij} 's are not necessarily integer — the distortion function $\sum_{i=1}^n x_{ij} \|\mathbf{a}_i - \mathbf{c}_j\|_2^2$ is convex in \mathbf{c}_j , and attains its global minimum at the arithmetical mean of the entities in the cluster, i.e.,

$$\mathbf{c}_j^* = \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i}{\sum_{i=1}^n x_{ij}}.$$

Therefore, (1.1) is equivalent to

(1.4a)

$$\min_{x_{ij}} \sum_{j=1}^k \sum_{i=1}^n x_{ij} \left\| \mathbf{a}_i - \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i}{\sum_{i=1}^n x_{ij}} \right\|_2^2$$

$$(1.4b) \quad \text{s.t.} \quad \sum_{j=1}^k x_{ij} = 1 \quad (i = 1, \dots, n)$$

$$(1.4c) \quad x_{ij} \in \{0, 1\} \quad (i = 1, \dots, n; j = 1, \dots, k).$$

This is the nonlinear integer programming model of the minimum sum-of-squared error clustering problem. Note that its objective is not convex in x_{ij} and the constraints (1.2d) are discrete. This makes the problem very hard to solve. There is no evidence that the standard IP techniques can be applied to MSSC on large data sets.

Continuous relaxation Another way to deal with (1.4) is to relax the integer constraints (1.2d) to

$$x_{ij} \in [0, 1].$$

$$(1.5) \quad \begin{aligned} & \min_{x_{ij}} \sum_{j=1}^k \sum_{i=1}^n x_{ij} \left\| \mathbf{a}_i - \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i}{\sum_{i=1}^n x_{ij}} \right\|_2^2 \\ & \text{s.t.} \quad \sum_{j=1}^k x_{ij} = 1 \quad (i = 1, \dots, n) \\ & \quad \sum_{i=1}^n x_{ij} \geq 1 \quad (j = 1, \dots, k) \\ & \quad x_{ij} \geq 0 \quad (i = 1, \dots, n; j = 1, \dots, k) \end{aligned}$$

To our knowledge, this is first considered by Gordon and Henderson [6], who have further proved that at a global minimizer of (1.5), all the variables x_{ij} 's must have values 0 or 1. This indicates that the global minima of (1.5) are exactly those of (1.4). Although that statement is correct, the proof in [6] is not rigorous and not easy to follow. In this paper, we will prove that every local minimum of (1.5) is integer; furthermore, our proof gives a procedure of moving from a fractional solution of (1.5) to an integer solution with better objective value.

1.2 Our algorithm. In [19], Selim and Ismail have proved that a class of distortion functions used in K-means-type clustering are essentially concave functions of the assignment variables. Specializing their conclusion to MSSC, we can claim that the objective function (1.2a) is concave in the feasible domain of the continuous relaxation of (1.2). In this paper, we will give a direct but alternative proof of the concavity of the function given in (1.4a) under certain conditions. Although a concave function achieves its local minima at some extreme points of its feasible region, unless the function is strictly convex, it is not always true that all of its local minimizers are extreme points. For (1.5), we will show that at any of its local minimum, the assignment variables x_{ij} 's are either 0 or 1, even though (1.5) is not a strictly concave program. Thus, we can safely work on (1.5) instead of (1.4).

Since a concave objective function may have many local minima, the global minima to (1.5) is still very hard to locate. Neither [6] nor [19] addresses how to find such a global solution. Until recently there have been only a few works on finding the exact solutions of the MSSC problem. In [21], Tuy, Bagirov and Rubinov cast some clustering problems to d.c.¹ programs. And branch-and-bound methods are suggested to solve the resulting d.c. program. Only numerical results on

¹Here, d.c. stands for the difference of convex functions.

small-size data sets are reported. In [2], Merle et al consider the Lagrangian relaxation of (1.4)

$$(1.6) \quad \max_{\lambda_i \geq 0} \left\{ \sum_{j=1}^k \min_{x_{ij} \in \{0,1\}} \left[\sum_{i=1}^n x_{ij} \left\| \mathbf{a}_i - \frac{\sum_{l=1}^n x_{lj} \mathbf{a}_l}{\sum_{l=1}^n x_{lj}} \right\|^2 - \left(\sum_{i=1}^n \lambda_i x_{ij} \right) \right] + \sum_{i=1}^n \lambda_i \right\}.$$

It is proved in [2] that there is no duality gap between (1.4) and its Lagrangian relaxation. Further, the authors of [2] propose to use Dinkelbach's nonlinear fractional programming method ([1]) combined with some heuristics to solve the subproblem of (1.6) for temporarily fixed λ_i . It should be noted that Dinkelbach's method is convergent only when the numerator is convex and the denominator is positive, which is not satisfied by the function defined in (1.6). This partially explains why the pure Dinkelbach's method takes a longer time than some heuristic method to solve the subproblem of (1.6), as is observed in [2]. Moreover, the Lagrangian relaxation of the integer programming model deals directly with the objective, the total within-group sum-of-squared error, not the partitioning itself. Therefore, extra effort is needed to recover the optimal partition. Again, only numerical experiments on data sets having up to 150 samples are reported. To summarize, it is fair to claim that although the MSSC problem is an important problem that has attracted many researchers' attentions, no globally convergent and efficient algorithms have been reported in the literature, in particular for moderately large data sets.

The main target of this paper is to propose a globally convergent and efficient algorithm for the minimum sum-of-squared error clustering problem. For this purpose, we use the fact that the MSSC problem can be formulated as a concave minimization problem over a polyhedron. For self-completeness, we give a detailed and direct proof of the concavity of the function (1.4a). We also characterize the local optimality of (1.5) and (1.4). It should be mentioned that in [19], Selim and Ismail have discussed the local optimality conditions for a class of optimization problems based on K-means-type clustering. However, the theoretical framework in [19] focuses on the mathematical essence of the local optimality for several different clustering approaches, while our analysis is emphasized on the difference between the stop criteria of the K-means algorithm and the local optimality conditions of (1.4). We also compare the local optimality conditions of the continuous model (1.5) with those of its discrete counterpart (1.4). As we shall see later, our discussion can help us skip a local minimum and further improve the objective value.

Many methods for concave minimization have been proposed in the optimization community. Those include cone splitting ([20, 11]), successive underestimation ([3]) and branch and bound ([9]) etc. Among them, Tuy's convexity cut method ([10]) is particularly interesting because the complexity of each step is very low. Note that all the above-mentioned methods are designed for full dimensional feasible region; thus are not directly applicable to (1.5). In this paper, we will adapt Tuy's cut method to solve the concave minimization problem derived from MSSC and prove its finite convergence. Promising numerical results will be provided.

The rest of the paper is organized as follows. In §2, we will prove that the minimum sum-of-squared error clustering problem can be formulated as a concave minimization problem and give a constructive proof showing that each local optimum of (1.5) is integer. We will also give local optimality conditions for (1.5) and compare it with that of (1.4). In §3, we will adapt Tuy's concavity cuts to find a global minimum of (1.5). Preliminary numerical examples will be given in §4.

Few words about notations throughout this paper. We use bold lower case letters for column vectors; lower case letters for scalars; capital letters for matrices. Superscript T is used to represent the transpose of a matrix or vector.

2 Characteristics of the integer formulation for MSSC and its continuous relaxation.

In this section, we will give some optimal conditions for (1.4) and its continuous relaxation (1.5) for the purpose of algorithm design. To describe the common characteristics of (1.4) and (1.5), we assume $x_{ij} \in [0, 1]$ ($i = 1, \dots, n; j = 1, \dots, k$) is a continuous variable in this section.

Let $\mathbf{x}_j \stackrel{\text{def}}{=} (x_{1j}, \dots, x_{nj})^T$ ($j = 1, \dots, k$) denote the assignment vector for the j th cluster.

Represent the within-group sum-of-squared error for the j th group by the function:

$$s_j(\mathbf{x}_j) \stackrel{\text{def}}{=} \sum_{i=1}^n x_{ij} \sum_{p=1}^d \left[(a_i)_p - \frac{\sum_{i=1}^n x_{ij} (a_i)_p}{\sum_{i=1}^n x_{ij}} \right]^2.$$

The total within-group sum-of-squared error is denoted as:

$$s(X) \stackrel{\text{def}}{=} \sum_{j=1}^k s_j(\mathbf{x}_j).$$

Denote the difference from entity \mathbf{a}_l to the centroid of the j th group as $\mathbf{v}_{lj} \in \mathbb{R}^d$. Here, \mathbf{a}_l is not necessarily assigned to the j th group.

$$(2.7) \quad \mathbf{v}_{lj} \stackrel{\text{def}}{=} \mathbf{a}_l - \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i}{\sum_{i=1}^n x_{ij}} = \mathbf{a}_l - \mathbf{c}_j.$$

As we have mentioned in the introduction, the concavity of (1.2a) in the feasible domain of the continuous relaxation of (1.2) can follow from the conclusions in [19]. Below, we give a direct proof of the concavity of (1.4a) under certain conditions, since our algorithm is based on (1.4a) instead of (1.2a). Another reason for our proof is that some technical results in our proof will be used in our later discussion.

PROPOSITION 2.1. *The objective function (1.4a) is concave whenever $\sum_{i=1}^n x_{ij} > 0$ ($j = 1, \dots, k$).*

Proof. We give the gradient and Hessian of $s(X)$ to verify that the objective function of (1.5) is concave in its feasible region.

$$\begin{aligned} \frac{\partial s(X)}{\partial x_{lj}} &= \left\| \mathbf{a}_l - \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i}{\sum_{i=1}^n x_{ij}} \right\|_2^2 \\ &\quad + 2 \sum_{i=1}^n x_{ij} \left(\frac{\sum_{p=1}^n x_{pj} \mathbf{a}_p}{\sum_{p=1}^n x_{pj}} - \mathbf{a}_i \right)^T \\ &\quad \left(\frac{1}{\sum_{p=1}^n x_{pj}} \mathbf{a}_l - \frac{\sum_{p=1}^n x_{pj} \mathbf{a}_p}{\left(\sum_{p=1}^n x_{pj} \right)^2} \right) \end{aligned}$$

Since $\sum_{i=1}^n x_{ij} \left(\frac{\sum_{p=1}^n x_{pj} \mathbf{a}_p}{\sum_{p=1}^n x_{pj}} - \mathbf{a}_i \right) = \mathbf{0}$ and $\left(\frac{1}{\sum_{p=1}^n x_{pj}} \mathbf{a}_l - \frac{\sum_{p=1}^n x_{pj} \mathbf{a}_p}{\left(\sum_{p=1}^n x_{pj} \right)^2} \right)$ is independent of i , the second term vanishes. Therefore,

$$(2.8) \quad \frac{\partial s(X)}{\partial x_{lj}} = \left\| \mathbf{a}_l - \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i}{\sum_{i=1}^n x_{ij}} \right\|_2^2.$$

Let V_j represent the matrix whose l th row is the vector \mathbf{v}_{lj} . Let \mathbf{e} denote the vector of all 1's.

Then for any $l, g \in \{1, \dots, n\}$ and $j \neq m \in \{1, \dots, k\}$:

$$\begin{aligned} \nabla^2 s_j &= -\frac{2}{\mathbf{x}_j^T \mathbf{e}} A A^T + \frac{2}{(\mathbf{x}_j^T \mathbf{e})^2} (A A^T \mathbf{x}_j \mathbf{e}^T + \mathbf{e} \mathbf{x}_j^T A^T A) \\ &\quad - \frac{2 \mathbf{x}_j^T A A^T \mathbf{x}_j}{(\mathbf{x}_j^T \mathbf{e})^3} \mathbf{e} \mathbf{e}^T = -\frac{2}{\sum_{i=1}^n x_{ij}} V_j V_j^T, \\ \frac{\partial^2 s}{\partial x_{lj} \partial x_{gm}} &= 0 \quad (j \neq m). \end{aligned}$$

Denote $\mathbf{x} \stackrel{\text{def}}{=} (\mathbf{x}_1^T, \dots, \mathbf{x}_n^T)$. It follows that

$$\nabla^2 s(\mathbf{x}) = \text{Diag}(\nabla^2 s_1(\mathbf{x}_1), \dots, \nabla^2 s_k(\mathbf{x}_k)).$$

Hence $-\nabla^2 s(\mathbf{x})$ is positive semidefinite whenever $\sum_{i=1}^n x_{ij} > 0$ ($j = 1, \dots, k$). This implies that the objective function of (1.5) is concave in its feasible region.

Observe that $s(x)$ is not strictly concave under the above assumptions. Next, we will show that each local minimizer of (1.5) is integer, so that we can work on (1.4) instead of (1.4). We will also discuss the optimal conditions for both (1.4) and (1.5).

First, we give a proposition that will be used to prove some properties of local minima and construct our algorithm.

PROPOSITION 2.2. *Perturb x_{lj} by Δx_{lj} . Then the new within-group sum-of-squared distances about the new centroid \mathbf{c}'_j for the j th group, $s_j(\mathbf{x}_j + \Delta \mathbf{x}_j)$, is*

$$(2.9) \quad \begin{cases} s_j(\mathbf{x}_j) + \Delta x_{lj} \|\mathbf{v}_{lj}\|_2^2 - \frac{(\Delta x_{lj})^2 \|\mathbf{v}_{lj}\|_2^2}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} & \sum_{i=1}^n x_{ij} + \Delta x_{lj} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The difference from \mathbf{a}_l to \mathbf{c}'_j is

$$(2.10) \quad \mathbf{v}'_{lj} = \begin{cases} \mathbf{0} & \sum_{i=1}^n x_{ij} + \Delta x_{lj} = 0, \\ \frac{\sum_{i=1}^n x_{ij}}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \mathbf{v}_{lj} & \sum_{i=1}^n x_{ij} + \Delta x_{lj} \neq 0. \end{cases}$$

Proof. 1) We first consider the case $\sum_{i=1}^n x_{ij} + \Delta x_{lj} = 0$. This condition means that the j th group is empty after perturbation by Δx_{lj} . Therefore

$$s_j(\mathbf{x}_j + \Delta \mathbf{x}_j) = 0, \quad \mathbf{v}_{lj} = \mathbf{0}.$$

2) Now assume $\sum_{i=1}^n x_{ij} + \Delta x_{lj} \neq 0$.

If $\sum_{i=1}^n x_{ij} = 0$, the j th group is empty before perturbation. In this case, $s_j(\mathbf{x}_j + \Delta \mathbf{x}_j) = 0$ and $\mathbf{v}'_{lj} = \mathbf{0}$, since the j th group now contains only $\Delta x_{lj} \mathbf{a}_l$.

In the remaining of the proof, we assume $\sum_{i=1}^n x_{ij} \neq 0$. After perturbation, the j th centroid is

$$\mathbf{c}'_j = \frac{\sum_{i=1}^n x_{ij} \mathbf{a}_i + \Delta x_{lj} \mathbf{a}_l}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}}.$$

By (2.7),

$$(2.11) \quad \sum_{i=1}^n x_{ij} \mathbf{a}_i = (\mathbf{a}_l - \mathbf{v}_{lj}) \left(\sum_{i=1}^n x_{ij} \right).$$

Plugging (2.11) into the expression of \mathbf{c}'_j , we get

$$(2.12) \quad \begin{aligned} \mathbf{c}'_j &= \frac{(\mathbf{a}_l - \mathbf{v}_{lj}) (\sum_{i=1}^n x_{ij}) + \Delta x_{lj} \mathbf{a}_l}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \\ &= \mathbf{a}_l - \frac{\sum_{i=1}^n x_{ij}}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \mathbf{v}_{lj}. \end{aligned}$$

Using (2.7) and (2.12), we have

$$(2.13) \quad \begin{aligned} \mathbf{c}_j - \mathbf{c}'_j &= (\mathbf{a}_l - \mathbf{v}_{lj}) - \left(\mathbf{a}_l - \frac{\sum_{i=1}^n x_{ij}}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \mathbf{v}_{lj} \right) \\ &= -\frac{\Delta x_{lj}}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \mathbf{v}_{lj}. \end{aligned}$$

Then

$$\begin{aligned} s_j(\mathbf{x}_j + \Delta \mathbf{x}_j) &= \sum_{i=1}^n x_{ij} \|\mathbf{a}_i - \mathbf{c}_j + \mathbf{c}_j - \mathbf{c}'_j\|_2^2 \\ &+ \Delta x_{lj} \|\mathbf{a}_l - \mathbf{c}'_j\|_2^2 = 2 \sum_{i=1}^n x_{ij} (\mathbf{a}_i - \mathbf{c}_j)^T (\mathbf{c}_j - \mathbf{c}'_j) \\ &+ \sum_{i=1}^n x_{ij} \|\mathbf{c}_j - \mathbf{c}'_j\|_2^2 + s_j(\mathbf{x}_j) + \Delta x_{lj} \|\mathbf{a}_l - \mathbf{c}'_j\|_2^2. \end{aligned}$$

The first term vanishes, because by the definition of \mathbf{c}_j ,

$$\sum_{i=1}^n x_{ij} (\mathbf{a}_i - \mathbf{c}_j) = \mathbf{0}.$$

After plugging (2.12) and (2.13) into the above expression of $s_j(\mathbf{x}_j + \Delta \mathbf{x}_j)$, we get

$$\begin{aligned} s_j(\mathbf{x}_j + \Delta \mathbf{x}_j) &= s_j(\mathbf{x}_j) + \frac{\sum_{i=1}^n x_{ij} (\Delta x_{lj})^2}{(\sum_{i=1}^n x_{ij} + \Delta x_{lj})^2} \|\mathbf{v}_{lj}\|_2^2 \\ &+ \frac{(\sum_{i=1}^n x_{ij})^2 \Delta x_{lj}}{(\sum_{i=1}^n x_{ij} + \Delta x_{lj})^2} \|\mathbf{v}_{lj}\|_2^2 \\ &= s_j(\mathbf{x}_j) + \frac{\sum_{i=1}^n x_{ij} \Delta x_{lj}}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \|\mathbf{v}_{lj}\|_2^2 \\ &= s_j(\mathbf{x}_j) + \Delta x_{lj} \|\mathbf{v}_{lj}\|_2^2 - \frac{(\Delta x_{lj})^2}{\sum_{i=1}^n x_{ij} + \Delta x_{lj}} \|\mathbf{v}_{lj}\|_2^2. \end{aligned}$$

Hence, (2.9) is proved. The expression (2.10) is from (2.12).

COROLLARY 2.1. *At a local optimum of (1.4) or that of its continuous relaxation (1.5) (without constraint (1.2c)),*

1. *no cluster is empty;*
2. *all the centroids \mathbf{c}_j 's are distinct.*

Proof. To prove (1), we only need to show that the total sum-of-squared error of a feasible solution to (1.4) or (1.5) with an empty cluster is strictly larger than that of some other feasible solution in its neighborhood (related to (1.4) or (1.5)). Suppose the g th cluster ($1 \leq g \leq k$) is empty. From (1.2b) and the assumption that there are at least k different points, we deduce that there exist at

least a cluster j and two different entities $\mathbf{a}_l, \mathbf{a}_m$, so that $x_{lj} > 0, x_{mj} > 0$. In addition, $\mathbf{a}_l \neq \mathbf{c}_j$. Perturbing x_{lj} and x_{lg} to $x_{lj} - \Delta x$ and $x_{lg} + \Delta x$, with $0 < \Delta x \leq x_{lj}$ ($\Delta x = 1$ for (1.4)), by (2.9), we get

$$s(X) - s(X + \Delta X) = \frac{\sum_{i=1}^n x_{ij} \Delta x}{\sum_{i=1}^n x_{ij} + \Delta x} \|\mathbf{v}_{lj}\|_2^2 > 0.$$

Therefore, X is not a local minimum. This shows that the constraint (1.2c) is redundant.

Next, we prove that no two clusters have the same centroid at a local minimum. Assume an assignment X has two centroids $\mathbf{c}_j = \mathbf{c}_g$. Merging the j th and the g th clusters will not change the total sum-of-squared error, but produce an empty cluster. By (1), X is not a local minimum.

Hence, we have proved (1) and (2).

The above results also imply that the optimum sum-of-squared error decreases with k .

LEMMA 2.1. *The matrix X is a local minimum of (1.5), iff for each $l \in \{1, \dots, n\}$ and $j \in \{1, \dots, k\}$,*

$$(2.14) \quad x_{lj} = \begin{cases} 1 & \|\mathbf{v}_{lj}\|_2 < \|\mathbf{v}_{lm}\|_2 \ (\forall m = 1, \dots, k, m \neq j), \\ 0 & \text{otherwise.} \end{cases}$$

REMARK 2.1. *The relation (2.14) equals to the following conditions:*

- (i) *all the assignment variables have values 0 or 1, i.e., no entity is assigned fractionally to two clusters;*
- (ii) *for every entity \mathbf{a}_l ($l = 1, \dots, n$), there is only one cluster whose centroid is closest to it;*
- (iii) *every entity \mathbf{a}_l ($l = 1, \dots, n$) is assigned to the cluster whose centroid is closest to it.*

Proof. We first prove the sufficiency, i.e., if (2.14) is satisfied by some $X^* \in \mathbb{R}^{n \times k}$, X^* is a local minimum of (1.5). It suffices to prove that for any feasible search direction $Y \neq 0$, the directional derivative

$$D_Y s(X^*) \stackrel{\text{def}}{=} \lim_{t \downarrow 0} \frac{s(X^* + tY) - s(X^*)}{t} > 0.$$

By (2.8),

$$(2.15) \quad D_Y s(X^*) = \sum_{i=1}^n \sum_{j=1}^k \|\mathbf{v}_{ij}\|_2^2 y_{ij}.$$

Because X^* satisfies (1.2b) and Y is a feasible search direction, we have

$$(2.16) \quad \sum_{j=1}^k y_{lj} = 0 \quad (l = 1, \dots, n).$$

In addition, from (2.14) and the constraints $0 \leq x_{lj} \leq 1$, we get for all $l \in \{1, \dots, n\}$ and $j \in \{1, \dots, k\}$,

$$(2.17) \quad y_{lj} \begin{cases} \leq 0 & \|\mathbf{v}_{lj}\|_2 < \|\mathbf{v}_{lm}\|_2 \ (\forall m = 1, \dots, k, m \neq j), \\ \geq 0 & \text{otherwise.} \end{cases}$$

Combining (2.16) and (2.17), we conclude that

$$\sum_{j=1}^k \|\mathbf{v}_{lj}\|_2^2 y_{lj} \geq 0 \quad (l = 1, \dots, n).$$

Since $Y \neq 0$, the above inequality holds strictly for at least one $l \in \{1, \dots, n\}$. Therefore,

$$D_Y(X^*) > 0.$$

Next, we use contradiction to prove the necessity.

Suppose at a local minimizer of (1.5), denoted as X , the assignment of entity \mathbf{a}_l didn't satisfy the conditions in (2.14). Assume the distance from \mathbf{a}_l to the j th cluster is one of the shortest, i.e.,

$$j \in \arg \min_y \|\mathbf{v}_{ly}\|_2.$$

Suppose (i) or (iii) was violated, i.e., $x_{lj} \neq 1$. Then there existed $m \neq j$ such that $x_{lm} > 0$ and $\|\mathbf{v}_{lm}\|_2 \geq \|\mathbf{v}_{lj}\|_2$. Note that $\|\mathbf{v}_{lm}\|_2 = \|\mathbf{v}_{lj}\|_2$ would be possible if the j th cluster was not the only closest cluster to \mathbf{a}_l , i.e., $m \in \arg \min_y \|\mathbf{v}_{ly}\|_2$.

If (ii) wasn't satisfied as well, i.e., $\|\mathbf{v}_{lm}\|_2 = \|\mathbf{v}_{lj}\|_2$. Then neither $\|\mathbf{v}_{lj}\|_2$ nor $\|\mathbf{v}_{lm}\|_2$ could be zero. Were either of them zero, \mathbf{a}_l would be the centroid of both the j th and m th clusters, which violated Corollary 2.1. This also shows that $\sum_{i=1}^n x_{im} > x_{lm}$ whether $\|\mathbf{v}_{lm}\|_2 = \|\mathbf{v}_{lj}\|_2$ or $\|\mathbf{v}_{lm}\|_2 \neq \|\mathbf{v}_{lj}\|_2$, since otherwise \mathbf{a}_l would be the unique element in the m th cluster; hence $\mathbf{v}_{lm} = 0$.

According to (2.9), perturbing x_{lj} and x_{lm} to

$$x'_{lj} = x_{lj} + \Delta x, \quad x'_{lm} = x_{lm} - \Delta x, \quad (0 < \Delta x \leq x_{lm})$$

would not violate any constraints of (1.5), but would decrease its objective value by

$$\Delta x \left(\|\mathbf{v}_{lm}\|_2^2 - \|\mathbf{v}_{lj}\|_2^2 \right) + (\Delta x)^2 \left(\frac{\|\mathbf{v}_{lj}\|_2^2}{\sum_{i=1}^n x_{ij} + \Delta x} + \frac{\|\mathbf{v}_{lm}\|_2^2}{\sum_{i=1}^n x_{im} - \Delta x} \right) > 0.$$

In other words, consider the feasible search direction Y whose only nonzero entries were $y_{lj} = 1$, $y_{lm} = -1$. By (2.1), the first order and second order directional derivative of s in the direction Y evaluated at X would be

$$(2.18) \quad \begin{aligned} D_Y s(X) &= \|\mathbf{v}_{lj}\|_2^2 - \|\mathbf{v}_{lm}\|_2^2 \leq 0, \\ D_{YY} s(X) &\stackrel{\text{def}}{=} \lim_{t \downarrow 0} \frac{D_Y s(X + tY) - D_Y s(X)}{t} \\ &= -\frac{2\|\mathbf{v}_{lj}\|_2^2}{\sum_{i=1}^n x_{ij}} - \frac{2\|\mathbf{v}_{lm}\|_2^2}{\sum_{i=1}^n x_{im}} < 0. \end{aligned}$$

Therefore, Y would be a strictly descent direction of s at X .

That means X could not be a local minimum for s .

In addition, by (2.10), $\|\mathbf{v}_{lj}\|_2$ would decrease which would make j the unique solution to

$$\arg \min_y \|\mathbf{v}_{ly}\|_2.$$

Proceeding with the above procedure, we can increase x_{lj} to 1 and obtain a better objective value.

Suppose that only (iii) was violated, i.e., $x_{lj} = 1$, and $\exists m \neq j$, $\|\mathbf{v}_{lm}\|_2 = \|\mathbf{v}_{lj}\|_2$. Then similarly as above, perturbing x_{lj} and x_{lm} to $x_{lj} - \Delta x$ and $x_{lm} + \Delta x$ would decrease the total within group sum-of-squared error and make m the unique solution to $\arg \min_y \|\mathbf{v}_{ly}\|_2$.

In other words, consider the feasible direction Y whose only nonzero entities were $y_{lj} = -1$, $y_{lm} = 1$. Similarly as (2.18), Y would be a strictly descent direction of s at X .

Thus, we have shown that a local optimum of (1.5) must satisfy (2.14).

REMARK 2.2. Note that (2.14) is not a sufficient condition for a local minimum of the integer programming model (1.4). To see this, consider the following example:

$$d = 1, \quad k = 2, \quad \mathbf{a}_1 = -2, \quad \mathbf{a}_2 = 0, \quad \mathbf{a}_3 = 3.$$

The optimal clustering is $(\{\mathbf{a}_1, \mathbf{a}_2\}, \{\mathbf{a}_3\})$. However, the clustering $(\{\mathbf{a}_1\}, \{\mathbf{a}_2, \mathbf{a}_3\})$ also satisfies (2.14).

Next we give a necessary and sufficient condition for a local minimum of (1.5).

Setting $\Delta \mathbf{v}_{lj} = -1$ and $\Delta \mathbf{v}_{lg} = 1$ in (2.2), we get the following.

• Switching \mathbf{a}_l from the j th cluster to the g th cluster will change the total sum-of-squared error by

$$(2.19) \quad \begin{cases} \frac{\|\mathbf{v}_{lg}\|_2^2 \sum_{i=1}^n x_{ig}}{\sum_{i=1}^n x_{ig} + 1} - \frac{\|\mathbf{v}_{lj}\|_2^2 \sum_{i=1}^n x_{ij}}{\sum_{i=1}^n x_{ij} - 1} & \sum_{i=1}^n x_{ij} \neq 1 \\ \frac{\|\mathbf{v}_{lg}\|_2^2 \sum_{i=1}^n x_{ig}}{\sum_{i=1}^n x_{ig} + 1} & \text{otherwise.} \end{cases}$$

In our algorithm, we use (2.19) to find the steepest descent neighbor vertex and calculate the objective value change after the reassignment.

From (2.19), we have the following conclusion.

- At a local optimum of (1.4), the entity \mathbf{a}_l is assigned to the j th cluster iff 1) \mathbf{a}_l is the unique entity of the j th cluster; or 2) for any $m \in \{1, \dots, k\}$, $m \neq j$:

$$(2.20) \quad \frac{\sum_{i=1}^n x_{im}}{\sum_{i=1}^n x_{im} + 1} \|\mathbf{v}_{lm}\|_2^2 \geq \frac{\sum_{i=1}^n x_{ij}}{\sum_{i=1}^n x_{ij} - 1} \|\mathbf{v}_{lj}\|_2^2.$$

K-means type algorithms and some other heuristic algorithms for the minimum sum-of-squared error clustering are based on assigning each entity to the cluster whose centroid is closest to it; so these methods may find a local minimum of (1.5), but cannot guarantee to find a local minimum of (1.4), let alone a global minimum.

Note that (2.20) is stronger than (2.14); so although our algorithm is designed for (1.5), we use (2.20) to search for a local minimum.

3 Concavity cuts for MSSC.

Our finitely convergent algorithm for the minimum sum-of-squared error clustering problem is based on concave optimization technique. A large number of approaches for concave minimization problems can be traced back to Tuy's cutting algorithm [20] for minimizing a concave function over a full dimensional polytope. We will briefly describe Tuy's cuts in the first part of this section for completeness. In general, without adding more expensive cuts, this procedure cannot find a global optimum in finite steps. Furthermore, Tuy's cuts can't be applied directly to (1.5), because its feasible region doesn't have full dimension. In the second part of this section, we will show how to adapt Tuy's cutting algorithm to (1.5) and prove that this method can find a global minimum of (1.5) in finite steps.

3.1 Basic ideas of Tuy's cuts. For self-completeness, we sketch Tuy's cuts (also known as convexity cuts) below (see [10] for details). We assume $\mathbf{x} \in \mathbb{R}^n$ in this subsection.

Tuy's cuts are originally designed to find a global minimum of a concave function $f(\mathbf{x})$ over a polyhedron $D \in \mathbb{R}^n$. It requires 1) D has full dimension, i.e. $\text{int } D \neq \emptyset$; 2) for any real number α , the level set $\{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \geq \alpha\}$ is bounded.

Let \mathbf{x}^0 be a local minimum and a vertex of D . Denote $\gamma = f(\mathbf{x}^0)$. Since D is full-dimensional, \mathbf{x}^0 has at least n adjacent vertices. Let $\mathbf{y}^1, \dots, \mathbf{y}^p$ denote the vertices adjacent to \mathbf{x}^0 ($p \geq n$). For $i = 1, \dots, n$, let

$$(3.21) \quad \theta_i \stackrel{\text{def}}{=} \sup\{t : t \geq 0, f(\mathbf{x}^0 + t(\mathbf{y}^i - \mathbf{x}^0)) \geq \gamma\};$$

denote

$$\mathbf{z}^i \stackrel{\text{def}}{=} \mathbf{x}^0 + \theta_i(\mathbf{y}^i - \mathbf{x}^0).$$

Because f is concave, any point in the simplex $\text{Spx} \stackrel{\text{def}}{=} \text{conv}\{\mathbf{x}^0, \mathbf{z}^1, \dots, \mathbf{z}^n\}$ has objective value no less than γ . Therefore, one can cut off Spx from further search for a global minimum. Since \mathbf{x}^0 is a vertex of D which has full dimension, one can always find n binding constraints at \mathbf{x}^0 , and \mathbf{x}^0 has n linearly independent edges. Without loss of generality, assume $\mathbf{z}^1 - \mathbf{x}^0, \dots, \mathbf{z}^n - \mathbf{x}^0$ are linearly independent. Define

$$(3.22) \quad \pi = \mathbf{e}^T \text{Diag}\left(\frac{1}{\theta_1}, \dots, \frac{1}{\theta_n}\right) U^{-1}, \quad U = [\mathbf{y}^1 - \mathbf{x}^0, \dots, \mathbf{y}^n - \mathbf{x}^0].$$

Then the inequality

$$(3.23) \quad \pi(\mathbf{x} - \mathbf{x}^0) > 1$$

provides a γ -valid cut for (f, D) , i.e., any \mathbf{x} having objective value $f(\mathbf{x}) < \gamma$ must satisfy (3.23). In other words, if (3.23) has no solution in D , \mathbf{x}^0 must be a global minimum. Note that 1) $\theta_i \geq 1$; so Spx contains \mathbf{x}^0 and all its neighbor vertices; 2) the larger the θ_i , the deeper the cuts. Following is the original pure convexity cutting algorithm based on the above idea.

Cutting Algorithm (Algorithm V.1., Chapter V, [10])

Initialization

Search for a vertex \mathbf{x}^0 which is a local minimizer of $f(\mathbf{x})$. Set $\gamma = f(\mathbf{x}^0)$, $D_0 = D$.

Iteration $i = 0, 1, \dots$

1. At \mathbf{x}^i construct a γ -valid cut π^i for (f, D_i) .
2. Solve the linear program

$$(3.24) \quad \max \pi^i(\mathbf{x} - \mathbf{x}^i) \quad \text{s.t. } \mathbf{x} \in D_i.$$

Let ω^i be a basic optimum of this LP. If $\pi^i(\omega^i - \mathbf{x}^i) \leq 1$, then stop: \mathbf{x}^0 is a global minimum. Otherwise, go to step 3.

3. Let $D_{i+1} = D_i \cap \{\mathbf{x} : \pi^i(\mathbf{x} - \mathbf{x}^i) \geq 1\}$. Starting from ω^i find a vertex \mathbf{x}^{i+1} of D_{i+1} which is a local minimum of $f(\mathbf{x})$ over D_{i+1} . If $f(\mathbf{x}^{i+1}) \geq \gamma$, then go to iteration $i + 1$. Otherwise, set $\gamma \leftarrow f(\mathbf{x}^{i+1})$, $\mathbf{x}^0 \leftarrow \mathbf{x}^{i+1}$, $D_0 \leftarrow D_{i+1}$, and go to iteration 0.

THEOREM 3.1. (Theorem V.2, [10]) *If the sequence $\{\pi^i\}$ is bounded, then the above cutting algorithm is finite.*

3.2 The adapted Tuy's cutting algorithm for MSSC. To adapt Tuy's cuts to (1.5). We include constraints (1.2c) in (1.5) to ensure that it is a concave program (Proposition 2.1), although we have proved that these constraints are redundant for finding a local solution (Corollary 2.1). In this section, we will first

show how we find a local minimum. Our algorithm searches for a local minimum of (1.4), since it is stronger than that of (1.5), as is discussed before. Then, we will describe how we construct the concavity cuts. Finally, we will prove the finite convergence of our algorithm and compare it with the K-means algorithm.

3.2.1 Finding a local minimum. To find a local minimum of (1.4), we use pivot: moving from one vertex of the feasible domain to an adjacent one that can mostly decrease the total within-group sum-of-squared error based on (2.19), until a vertex complying with (2.20).

At r th iteration,

do **Loop** until (2.20) is satisfied for all $l = 1, \dots, n$.

Loop For $l = 1, \dots, n$:

Assume \mathbf{a}_l is assigned to j th cluster. When $\sum_{i=1}^n x_{ij} > 1$, let $f_l = \min_{q \neq j} \frac{\sum_{i=1}^n x_{iq}}{\sum_{i=1}^n x_{iq} + 1} \|\mathbf{v}_{lq}\|_2^2$.

If $f_l < \frac{\sum_{i=1}^n x_{ij}}{\sum_{i=1}^n x_{ij} - 1} \|\mathbf{v}_{lj}\|_2^2$, move \mathbf{a}_l to cluster $\arg\min f_l$ and update $s(X)$ by (2.19).

Note that we only search for a local minimum of (1.4). The number of vertices of the domain of (1.4) is finite and the objective value of (1.4a) is strictly decreased after each pivot. Hence, the number of pivots for finding a local minimum is finite.

3.2.2 Construction of the cutting planes. Once we find a local minimum, we need to add some cut to the constraints set. At r th iteration, let $X^0 \in \mathbb{R}^{n \times k}$ be a local optimal solution to (1.4) in D_r and γ be the smallest total within-group sum-of-squared errors obtained from the previous iterations. Next, we will give details on how we form (3.22), including the construction of U and θ_i for (3.22), although the feasible region of (1.4) doesn't have full dimension — each vertex is adjacent to only $n \times (k-1)$ other vertices.

1) Adjacent vertices.

We give $n \times k$ adjacent vertices to X^0 below.

Let $E_{i,j}$ denote the matrix whose (i,j) entry is 1, the other entries are 0; and let $E_{(i,\cdot)}$ denote the matrix whose i th row are 1's, the remaining entries are 0. The orders of $E_{i,j}$ and $E_{(i,\cdot)}$ will be clear from the context. For $l = 1, \dots, n$, assume \mathbf{a}_l is assigned to cluster l_j . Let $Y^{l,i}$ ($i = 1, \dots, k; i \neq l_j$) denote the matrix different from X^0 only by the assignment of \mathbf{a}_l to cluster i . Choose $1 \leq l_p \leq k, l_p \neq l_j$. And let Y^{l,l_p} denote the matrix different from X^0 only in its (l, l_p) entry being 1

as well, i.e.,

$$Y^{l,i} = \begin{cases} X^0 - E_{l,l_j} + E_{l,i} & i \neq l_j \\ X^0 + E_{l,l_p} & i = l_j \end{cases}.$$

Then $Y^{l,i}$ ($l = 1, \dots, n; i = 1, \dots, k, i \neq l_j$) are $n \times (k-1)$ adjacent vertices of X^0 in the feasible domain of (1.4). We form the vector \mathbf{x}^0 by stacking all the columns of X^0 together. Similarly, we form the vectors $\mathbf{y}^{l,i}$. It is not hard to see that $U = [\mathbf{y}^{1,1} - \mathbf{x}^0, \dots, \mathbf{y}^{1,k} - \mathbf{x}^0, \dots, \mathbf{y}^{n,k} - \mathbf{x}^0]$ has full rank. Let I represent the identity matrix. It is straightforward to verify that the corresponding U^{-1} of (3.23) is a block diagonal matrix with l th block being $I + E_{(l_j, \cdot)} - E_{(l_m, \cdot)} - E_{l_j, l_j}$.

Because Y^{l,l_j} is not feasible to (1.5), part of the simplex $\text{conv}\{\mathbf{x}^0, \mathbf{y}^{1,1}, \dots, \mathbf{y}^{1,k}, \dots, \mathbf{y}^{n,k}\}$ lies outside the feasible region of (1.5); nevertheless, the concavity cut can exclude some part of the feasible region of (1.4).

2) The cutting plane.

Next, we will determine the θ_i 's of (3.22), and subsequently the cutting plane π .

For $l = 1, \dots, n$: $\theta^{l,l_j} = +\infty$; when $m \neq l_j$, by Proposition 2.2, $\theta^{l,m}$ is a solution to the problem below. (3.25)

$$\begin{aligned} & \max t \\ & \text{s.t. } 0 \leq t \leq \sum_{i=1}^n x_{i,l_j}^0, \\ & s(X^0) - \frac{\sum_{i=1}^n x_{i,l_j}^0 t}{\sum_{i=1}^n x_{i,l_j}^0 - t} \|\mathbf{v}_{l,l_j}\|_2^2 + \frac{\sum_{i=1}^n x_{i,m}^0 t}{\sum_{i=1}^n x_{i,m}^0 + t} \|\mathbf{v}_{lm}\|_2^2 \geq \gamma. \end{aligned}$$

Denote the number of points in the j th cluster as $N_j \stackrel{\text{def}}{=} \sum_{i=1}^n x_{i,j}^0$. Next, we will solve (3.25).

Solving the second inequality in (3.25), we get $t \leq t^*$, where

$$t^* = - \frac{(s(X^0) - \gamma)(N_m - N_{l_j}) + N_m N_{l_j} \|\mathbf{v}_{l,l_j}\|_2^2 - \|\mathbf{v}_{lm}\|_2^2 - \sqrt{\omega}}{2 s(X^0) - \gamma + N_{l_j} \|\mathbf{v}_{l,l_j}\|_2^2 + N_m \|\mathbf{v}_{lm}\|_2^2},$$

where $\omega = \left[(s(X^0) - \gamma)(N_m + N_{l_j}) + N_m N_{l_j} (\|\mathbf{v}_{l,l_j}\|_2^2 - \|\mathbf{v}_{lm}\|_2^2) \right]^2 + 4(s(X^0) - \gamma) N_{l_j} N_m (N_{l_j} + N_m) \|\mathbf{v}_{lm}\|_2^2$.

Since $s(X^0) \leq \gamma$ and X^0 is a local minimum of (1.4), by (2.19), $t^* \geq 1$. In view of the first constraint in (3.25), we set

$$\theta^{l,m} = \min \{N_{l_j}, t^*\}, \quad (m \neq l_j).$$

Observe that when $s(X^0) = \gamma$, since X^0 is a local minimum of (1.4), by Lemma 2.1, $\|\mathbf{v}_{l,m}\|_2 > \|\mathbf{v}_{l,l_j}\|_2$; hence

$$t^* = \frac{N_{l_j} N_m (\|\mathbf{v}_{lm}\|_2^2 - \|\mathbf{v}_{l,l_j}\|_2^2)}{N_{l_j} \|\mathbf{v}_{l,l_j}\|_2^2 + N_m \|\mathbf{v}_{lm}\|_2^2} \leq N_{l_j}.$$

Therefore, in this case, $\theta^{l,m} = t^*$.

It follows that the coefficients of (3.24) are

$$\pi^{l,i} = \begin{cases} \frac{1}{\theta^{l,i}} - \frac{1}{\theta^{l,l_p}} & i \neq l_j \\ -\frac{1}{\theta^{l,l_p}} & i = l_j \end{cases}, \quad \pi \mathbf{x}^0 = -\sum_{l=1}^n \frac{1}{\theta^{l,l_p}}.$$

3) Solving the LP subproblems.

Without the constraints (1.2c), the LP for the first cut (3.24) is a knapsack problem, whose solution is trivial and is integer. If the solution doesn't satisfy (1.2c), by splitting some clusters as is discussed in Corollary 2.1, one can find a vertex that satisfies (1.2c).

The solutions to the succeeding LPs (3.24) are not necessarily integers. From a fractional solution to (3.24), one can move to a vertex using the procedure in Lemma 2.1. If the vertex is not in the remaining feasible region D_i , we use breadth-first search to find a vertex that satisfies all the cutting plane constraints. When pivoting to a local minimum, we also skip those vertices that are not in D_i . This is different from the pivoting in K-means algorithm. This ensures that no vertices in cut-out region will be revisited.

3.3 Finite convergence of the algorithm. The simplex Spx in our algorithm is centered at a local minimum of (1.4); so each concavity cut eliminates at least one vertex of (1.4). In addition, the number of vertices of (1.4) is finite. Therefore, the number of cuts is finite. From this along with the previous argument that only finite pivots are needed to reach a local minimum of (1.4), we conclude that our method can find a global minimum of the MSSC problem in finite steps.

REMARK 3.1. *It is possible to get a good solution for the MSSC problem by multiple runs of K-means algorithm with random restart. However, it is difficult to guess the new starting points. Without trying all the vertices, a global minimum cannot be guaranteed. An advantage of Tuy's convexity cut to enumeration is that each convexity cut may eliminate a certain number of vertices.*

4 Numerical Examples

We have implemented the above cutting algorithm in C with the LP for cutting plane solved by CPLEX 8.0.

Minimizing a concave function over a polytope is NP-hard (see for example [17]). In the worst case, Tuy's cutting method needs to enumerate all the vertices in a polytope; hence the worst-case complexity of Tuy's method for a general concave program is exponential. At this moment, we don't know whether the complexity of our algorithm can be reduced based on some special properties of MSSC. In addition, although the computa-

tional cost of each iteration is very low — solving an LP and pivoting, as more and more cuts are added, the size of the LP subproblem might be very large, which may require large amount of memory. From our preliminary numerical experience, we notice that the objective value improves significantly in the first several cuts, but very slowly in the later cuts. To maintain a tradeoff between the difficulty of finding a global minimum of the MSSC problem and the practical efficiency of the algorithm, we specify that our computer program terminates iff one of the following conditions is satisfied:

1. a global optimum is found;
2. the objective cannot be improved after 5 succeeding cuts;
3. the total number of cuts exceeds 20.

To give an initial solution, we assign the first k entities to the k clusters. Each of the remaining entities are assigned to the cluster whose centroid is closest to it. The initial sum-of-squared error is calculated from this assignment. Next, we run the K-means algorithm to find a local optimum of (1.5), from which we further pivot to a local minimum of (1.4) to start the cutting algorithm.

The figures below give some numerical examples. We use x -axis to represent the number of clusters— k , and y -axis to denote the total within-group sum-of-squared errors. Each group of vertical bars represents various sum-of-squared errors for a certain k . The first bar in a group denotes the initial sum-of-squared error; the second one is that when each entity is assigned to a cluster whose centroid is closest to it; the third one reports the sum-of-squared error when (2.20) is satisfied for each entity; the fourth one is that resulting from the cutting algorithm. The fifth bar of the first two examples represents the best known objective value taken from [2].

1. *The Iris plants database.*

Our first example is from [4]. It has 150 samples each having 4 attributes.

We have tested for $k = 2, \dots, 10$. Of the 9 cases, the cutting algorithm hits the best solution in 6 cases, and outperforms the k-means algorithm as well as the initial local minima in 8 cases.

2. *The Ruspini data set.*

This data set, consisting of 75 observations on 2 variables, is from [18].

For this data set, the cutting algorithm finds the global solution in 4 out of total 9 cases. It improves the K-means results in 8 cases; and get a better solution than the initial local minimum in 7 cases.

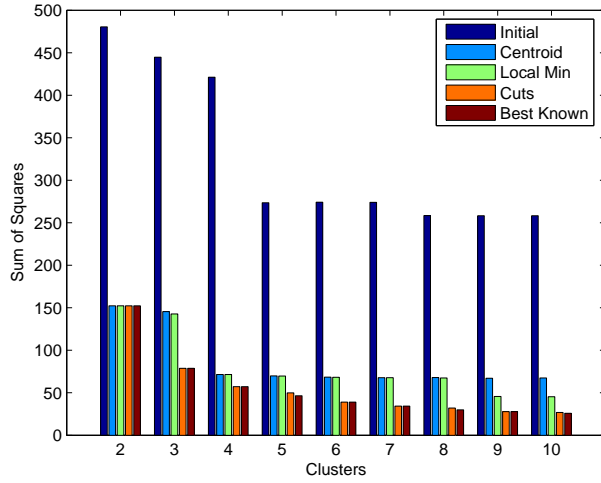


Figure 1: Iris Plants Database

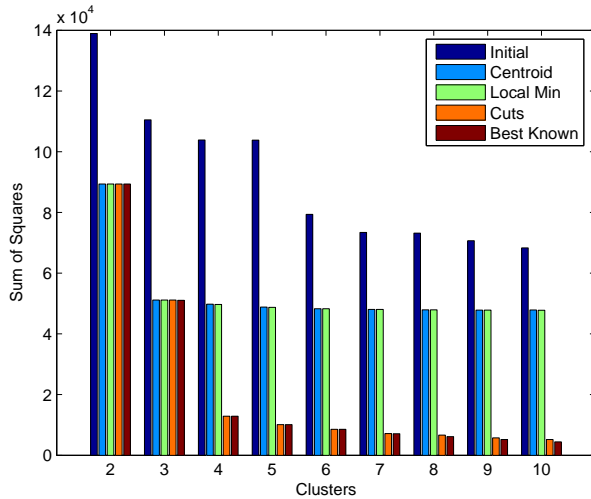


Figure 2: Ruspini data

By Corollary 2.1, the optimum within-group sum-of-squared error decreases with the number of cluster k . In the following examples, which are taken from the UCI Machine Learning Repository [16], we use the solution from the k -clustering as the starting point for the $k + 1$ clustering ($k = 2, \dots$).

1. *The Boston housing data set.*

This data set is from [8] concerning housing values in suburbs of Boston. It has 506 samples each having 13 attributes.

Among all the 9 cases $k = 2, \dots, 10$, the cutting algorithm over-performs the k-means algorithm in 7 cases and gets a better solution than the original local minimum of the integer programming formula (1.4) in 6 cases.

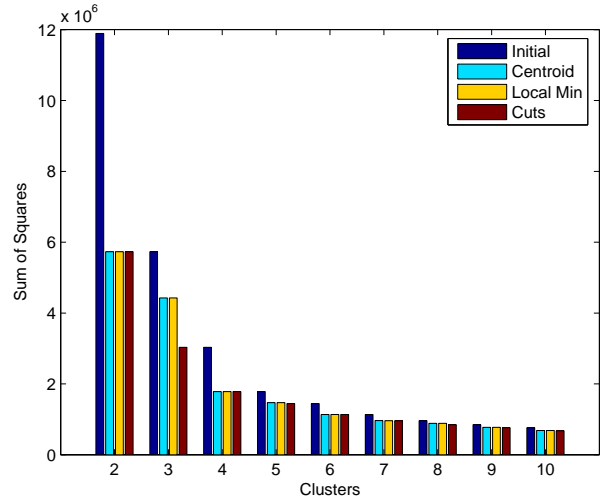


Figure 3: Boston Housing

2. *The spam E-mail database*

This dataset is created by M. Hopkins et al. at Hewlett-Packard Labs in June-July 1999 for spam filter. It has 4601 samples, 57 attributes (we remove the last class attribute which denotes whether the e-mail was considered spam or not).

Of the 6 cases $k = 2, \dots, 7$, the cutting algorithm finds a better solution than that by the K-means algorithm in 4 cases, the objective value is reduced from the initial local minimum of (1.4) in 2 instances.

For all the examples, the computer program terminated due to no improvement in objective value after 5 succeeding cuts. All of them terminated within 25 seconds. However, the results are satisfactory. Observe that in many cases, the solution obtained by the

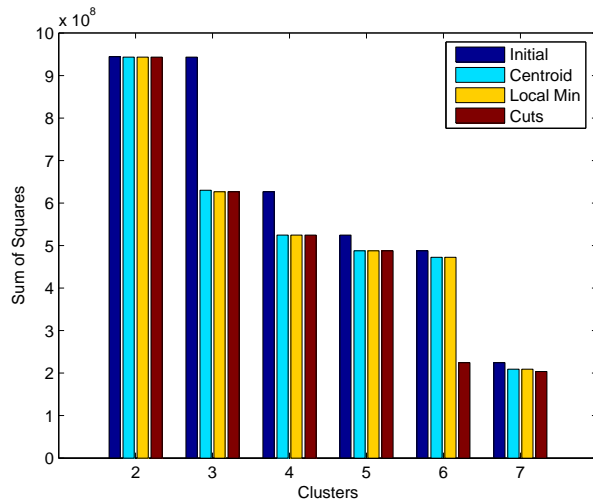


Figure 4: Spam E-mail

K-means algorithm is worse than that by the local minimum criterion (2.20).

5 Conclusion

In this paper, we have proved that the minimum sum-of-squared error clustering problem can be formulated as a concave minimization problem whose every local optimum solution is integer. We have characterized the local optimality of the continuous optimization model (1.5) and compared it with that of the discrete one (1.4). These properties can help us find a better solution. We have adapted the globally convergent Tuy's convexity cut to the concave optimization problem derived from MSSC. Preliminary numerical examples demonstrate that our method outperforms the popular K-means algorithm in the quality of the solution without a big increase in the running time.

References

- [1] Werner Dinkelbach. On nonlinear fractional programming. *Management Sci.*, 13:492–498, 1967.
- [2] O. Du Merle, P. Hansen, B. Jaumard, and N. Mladenović. An interior point algorithm for minimum sum-of-squares clustering. *SIAM J. Sci. Comput.*, 21(4):1485–1505 (electronic), 1999/00.
- [3] James E. Falk and Karla R. Hoffman. Successive underestimation method for concave minimization problem. *Math. Oper. Res.*, 1(3):251–259, Aug. 1976.
- [4] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annual of Eugenics*, 7:179–188, 1936.
- [5] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768, 1965.
- [6] A. D. Gordon and J. T. Henderson. An algorithm for euclidean sum of squares classification. *Biometrics*, 33(2):355–362, Jun. 1977.
- [7] Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Math. Programming*, 79(1-3, Ser. B):191–215, 1997.
- [8] D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *J. Environ. Economics & Management*, 5:81–102, 1978.
- [9] Reiner Horst. An algorithm for nonconvex programming problems. *Math. Programming*, 10(3):312–321, 1976.
- [10] Reiner Horst and Hoang Tuy. *Global optimization*. Springer-Verlag, Berlin, 1993.
- [11] Stephen E. Jacobsen. Convergence of a Tuy-type algorithm for concave minimization subject to linear inequality constraints. *Appl. Math. Optim.*, 7(1):1–9, 1981.
- [12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
- [13] R. C. Jancey. Multidimensional group analysis. *Australian J. Botany*, 14:127–130, 1966.
- [14] O. L. Mangasarian. Mathematical programming in data mining. *Data Min. Knowl. Discov.*, 1(2):183–201, 1997.
- [15] J. McQueen. Some methods for classification and analysis of multivariate observations. *Computer and Chemistry*, 4:257–272, 1967.
- [16] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1994. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [17] P. M. Pardalos and J. B. Rosen. Methods for global concave minimization: a bibliographic survey. *SIAM Rev.*, 28(3):367–379, 1986.
- [18] E. H. Ruspini. Numerical methods for fuzzy clustering. *Inform. Sci.*, 2(3):319–350, 1970.
- [19] Shokri Z Selim and M. A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–87, Jan. 1984.
- [20] H. Tuy. Concave programming under linear constraints. *Soviet Mathematics*, 5:1437–1440, 1964.
- [21] Hoang Tuy, A. M. Bagirov, and A. M. Rubinov. Clustering via d.c. optimization. In *Advances in convex analysis and global optimization (Pythagorion, 2000)*, volume 54 of *Nonconvex Optim. Appl.*, pages 221–234. Kluwer Acad. Publ., Dordrecht, 2001.

Dynamic Classification of Defect Structures in Molecular Dynamics Simulation Data

Sameep Mehta* Steve Barr† Tat-Sang Choy† Hui Yang *
Srinivasan Parthasarathy* Raghu Machiraju* John Wilkins†

*Department of Computer Science and Engineering, Ohio State University

†Department of Physics, Ohio State University

Contact: {mehtas,srini,raghu}@cse.ohio-state.edu

January 10, 2005

Abstract

In this application paper we explore techniques to classify anomalous structures (defects) in data generated from Molecular Dynamics (MD) simulations of Silicon (Si) atom systems. These systems are studied to understand the processes behind the formation of various defects as they have a profound impact on the electrical and mechanical properties of Silicon. In our prior work [12, 13, 14] we presented techniques for defect detection. Here, we present a two-step dynamic classifier to classify the defects. The first step uses up to third-order shape moments to provide a smaller set of candidate defect classes. The second step assigns the correct class to the defect structure by considering the actual spatial positions of the individual atoms. The dynamic classifier is robust and scalable in the size of the atom systems. Each phase is immune to noise, which is characterized after a study of the simulation data. We also validate the proposed solutions by using a physical model and properties of lattices. We demonstrate the efficacy and correctness of our approach on several large datasets. Our approach is able to recognize previously seen defects and also identify new defects in real time.

1 Introduction

Traditionally, the focus in the computational sciences has been on developing algorithms, implementations, and enabling tools to facilitate large scale realistic simulations of physical processes and phenomenon. However, as simulations become more detailed and realistic, and implementations more efficient, scientists are finding that analyzing the data produced by these simulations is a non-trivial task. Dataset size, providing reasonable response time, and modeling the underlying scientific phenomenon during the analysis are some of the critical challenges that need to be addressed.

In this paper we present a framework that addresses

these challenges for mining datasets produced by Molecular Dynamics (MD) simulations to study the evolution of defect structures in materials. As component size decreases, a defect - any deviation from the perfectly ordered crystal lattice - in a semiconductor assumes even greater significance. These defects are often created by introducing extra atom(s) in the Silicon lattice during ion implantation for device fabrication. Such defects can aggregate to form larger extended defects, which can significantly affect device performance in an undesirable fashion.

Simulating defect dynamics can potentially help scientists understand how defects evolve over time and how aggregated/extended defects are formed. Some of these defects are stable over a period of time while other are short-lived. Efficient, automated or semi-automated analysis techniques can help simplify the task of wading through a pool of data and help quickly identify important rules governing defect evolution, interactions and aggregation. The key challenges are: i) to detect defects; ii) to characterize and classify both new and previously seen defects accurately; iii) to capture the evolution and transitioning behavior of defects; and iv) to identify the rules that govern defect interactions and aggregation. Manual analysis of these simulations is a very cumbersome process. It takes a domain expert more than eight hours to manually analyze a very small simulation of 8000 time frames. Therefore, a systemic challenge is to develop an automated, scalable and incremental algorithmic framework so that the proposed techniques can support in-vivo analysis in real time.

In our previous work [12, 13, 14], we presented algorithms to address the first challenge. Here we address the second challenge coupled with the systemic challenge outlined above. The design tenets include not only accuracy and execution time but also both statistical and physical validation of the proposed models. We also present preliminary results to show that our approach can aid in handling the third

challenge.

The main contributions of our application case study paper are:

1. We develop a two-step incremental classifier that classifies both existing and new defects (generates a new class label).
2. We validate each step of our 2-step classifier theoretically, relying on both physical and statistical models.
3. We validate our approach on large (greater than 4GB) real MD simulation datasets and demonstrate both the exceptional accuracy and efficiency of the proposed framework.
4. We present initial results which show that our approach can be used to capture defect evolution and to generate labeled defect trajectories.

Our paper is structured in the following manner. Section2 discusses the basic terminology of MD and related work. An overview of our framework is provided in Section3. We present our algorithm in Section4. Results on large simulation datasets are presented in Section5. Finally, we conclude and discuss directions for future work in Section6.

2 Background and Related Work

2.1 Background: In this section, we first define basic terms that are used throughout this article. Later, we describe pertinent related work. A **lattice** is an arrangement of points or particles or objects in a regular periodic pattern in three dimensions. The elemental structure that is replicated in a lattice is known as a **unit cell**. Now, consider adding a single atom to the lattice. This extra atom disturbs the geometric structure of lattice. This disturbance, comprised of atoms which deviate from the regular geometry of lattice is referred to as the **defect** structure. Defects created by adding an extra atom are known as single-interstitial defects. Similarly one can define **di-** and **tri-interstitial** defects by adding two or three single interstitial defects in a lattice, respectively. Figure 1(a) shows a Si bulk lattice with a certain unit cell shaded differently (black). Figure 1(b) shows another lattice with a single interstitial defect. Figure 1(c) depicts two interstitials: in the lower left and upper right corners, respectively, of a 512-atom lattice. The different shades again represent separate and distinct defects.

Ab initio relaxations and MD simulations are frequently used to discover stable defect structures [2, 16, 22]. Our MD simulations are performed with the Object-Oriented High Performance Multi-scale Materials Simulator (OHMMS) [16, 17] employing a classical potential to describe the interactions between the Si atoms [11].

2.2 Related Work: Several methods have been proposed to detect defects in crystal lattice structures. The most pertinent work that is closely related to our own work employs the method of Common Neighbor Analysis (CNA) [3, 8]. CNA attempts to glean crystallization structure in lattices and uses the number of neighbors of each atom to determine the structure sought. However, it should be noted that the distribution of neighbors alone cannot characterize the defects, especially at high temperatures.

Related to our work is the large body of work in biomolecular structure analysis [1, 10, 18, 23]. In these techniques, the data is often abstracted as graphs and transactions and subsequently mined. However, such an abstraction often does not exploit and explain many of the inherent spatial and dynamical properties of data that we are interested in. Moreover, while some of these techniques [19, 23], deal with noise in the data, the noise handling capabilities are limited to smoothing out uncorrelated and small changes in spatial location of atoms. Within the context of MD data, noise can also change the number of defect atoms detected, for essentially the same defect structure at different time frames. None of the methods in the biological data mining literature deal with this uncertainty.

Matching of two structures has drawn a lot of attention in recent past. Zhang et al. [25], propose a protein matching algorithm that is rotation and translation invariant. This method relies on the shape of the point cloud and it works well for proteins given the relatively large number of atoms; the presence of a few extra atoms does not change the shape of point cloud. A potential match is not stymied by the presence of extra atoms. However, in MD simulation data, we are interested in anomalous structures which can be as small as just six atoms. Extra atoms that may be included in a defect given the thermal noise will skew a match significantly even if the two defects differ by one atom. Geometric hashing, an approach that was originally developed in the robotics and vision communities [24], has found favor in the biomolecular structure analysis community [15, 23]. Rotation and translations are well handled in this approach. The main drawback of geometric hashing is that it is very expensive because an exhaustive search for optimal basis vector is required. A more detailed discussion on various shape matching algorithms can be found in the survey paper by Veltamp and Hagedoorn [21].

We use statistical moments to represent the shape of a defect for initial pruning. The seminal work by Hu [6] described a set of seven moments which captures many of the features of a two-dimensional binary images. In recent work, Galvez [5] proposed an algorithm which uses shape moments to characterize 3D structures using moments of its two-dimensional contours. However, owing to relatively small number of atoms present in a typical defect, the contours or the corresponding implicit 3D surface are impossible

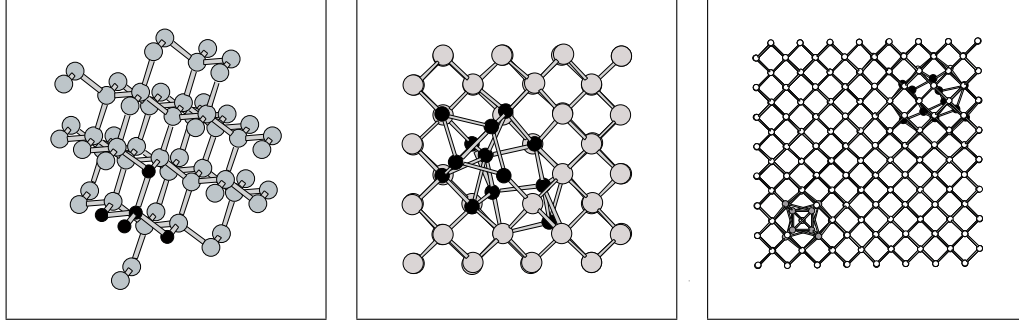


Figure 1: Defects in the crystal lattice. (a) The unit cell of the perfect crystal lattice is marked by the dark atoms. (b) A single interstitial defect distorts the lattice. (c) Two separate interstitial defects are present in the lattice.

to obtain with accuracy.

3 Dynamic Classification Framework

Figure 2 shows our framework for MD simulation data analysis. The framework is divided into three phases.

- *Phase 1 - Defection Detection*: This phase detects, spatially clusters (or segments) defects and handles periodic boundary conditions. A detailed explanation on this phase is given in our earlier work [12, 13, 14].
- *Phase 2 - Dynamic Classification*: This phase classifies each defect found in *Phase 1* and consists of three major components:
 1. A feature vector composed of weighted statistical moments is generated for each detected defect.
 2. The set of defects is pruned based on the feature vector. This step provides a subset of defect classes to which an unlabeled defect can potentially belong to.
 3. An exact matching algorithm assigns the correct class label to the defect. This step takes into account the spatial position of the atoms. The defect is assigned a class label if it matches with any of the previously seen defects, otherwise the defect is considered new.

Phase 2 maintains three databases for all detected defects. Section 4 gives detailed information regarding these three databases and their update strategies.

The framework is made robust by modeling noise in both *Phase 1* and *Phase 2*. Our noise characterization models the aggregate movement of the defect structure and the arrangement of the atoms (in terms of neighboring bonds). Our framework can be deployed to operate in a streaming fashion. This is important since it enables us to naturally handle data in a continuous fashion

while a simulation is in progress. *Phase 1* handles the entire frame and detects all the defects. Each defect is then pipelined into *Phase 2*. Thus, we are able to incrementally detect and classify defects while consistently updating the database in real time.

- *Phase 3 - Knowledge Mining*: This phase uses the databases generated by *Phase 2*. These databases store the information about all the defects in a given simulation. These databases can be used to track and generate the trajectories of the defects, which can assist us to better understand the defect evolution process. Additionally, various data mining algorithms can be applied on the databases. Mining spatial patterns within one simulation can aid in understanding the interactions among defects. Finding frequent patterns across multiple simulations can help to predict defect evolution. In this paper, we describe *Phase 2* in detail. We also show some initial results for *Phase 3*.

4 Algorithm

Our previous work [12, 13, 14] describes the defect detection phase in detail. Every atom in the lattice is labeled either as a bulk atom or as a defect atom. However, upon further evaluation we found that this binary labeling is not well suited for robust classification of defect structures. Therefore, we propose to divide the atoms into three classes based on their membership or proximity to a defect. We also validate the correctness of this taxonomy by using a physical model. Our taxonomy is:

- **Core-Bulk Atoms (CB)**: Atoms that conform to the set of rules defined by the unit cell are *bulk* or non-defect atoms. Bulk atoms which are connected exclusively to other bulk atoms are labeled as core bulk atoms.
- **Core-Defect Atoms (CD)**: Atoms that do not conform to the set of rules are *defect* atoms. All defect atoms that

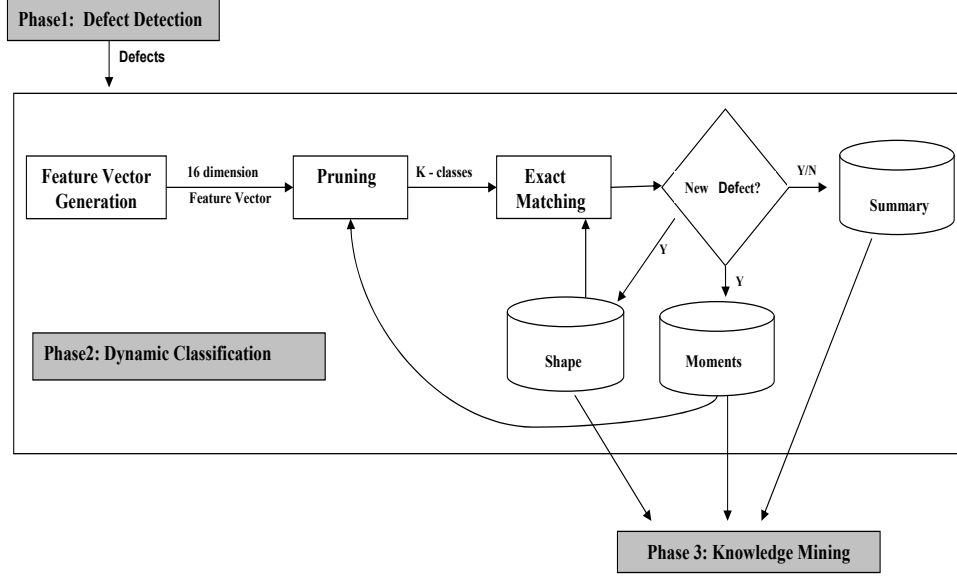


Figure 2: Defect Detection and Classification Framework

are connected to more defect atoms than bulk atoms are labeled as core defect atoms. These atoms dominate the shape and properties of a defect.

- **Interface Atoms (I):** Defect atoms that are connected to more bulk atoms than defect atoms are labeled as interface atoms. These atoms form the boundary between core bulk and core defect atoms. They fail to conform to the prescribed set of rules by a small margin (i.e. the thresholds for bond lengths and angles are violated in a marginal way) and are thus marked as defect atoms. They form a layer between core bulk and core defect atoms. Presence or absence of these atoms can considerably change the shape of a defect, which exacerbates the matching of defect structures.

Figure 3(a) illustrates all three types of atoms. The black atoms belong to core defect, light gray atoms form the core bulk and the dark gray atoms are interface atoms. This taxonomy is justified by a physical model.

The lattice system can be represented by a Mechanical Molecular Model as follows: the atoms are represented by points and the bonds by springs connecting the points. The energy of the lattice system consists of three terms:

$$E_{\text{total}} = E_{\text{length}} + E_{\text{angle}} + E_{\text{int}},$$

where E_{length} and E_{angle} denote the energies due to bond stretching and angle bending, respectively, and E_{int} denotes the remaining interaction energy not accounted for by the bond stretching and bending terms. For the following analysis the interaction term, E_{int} , is neglected.

The energy of bond stretching and bending is described by spring constants, K , and the deviation, δ , of the bonds from the ideal state:

$$E = \frac{1}{2} K \delta^2, \quad (4.1)$$

The stretching and bending energies E_{length} and E_{angle} are computed for each atom using the two spring constants, K_{length} and K_{angle} , respectively. The ideal bond length and bond angle are used for the uncompressed spring state. The values for K_{length} and K_{angle} are taken from [4] as 185 Newton/meter and 0.35 Newton/radian, respectively. For each atom we find the bond lengths and bond angles it forms with its first neighbors and then find the deviations, δ , from the ideal values. Core bulk atoms deviate little from the ideal values, whereas for core defect atoms the deviations are large. Since the energy is directly proportional to the square of the deviations, core bulk atoms have low energies while core defect atom energies are large.

To validate our taxonomy, we sampled 1400 frames from different simulations and calculated the energy for each atom in the lattice. Figure 3(b) shows the distribution of energy. It is clear from the distribution that the majority of the atoms have very low energy ($\in [0, 0.2]$). These atoms are core bulk atoms. The core defect atoms have very high energy (≥ 1.2). All the atoms which lie between low and high energy levels are interface atoms. Thus, this physical model clearly validates our taxonomy of atoms. Therefore we refine our original binary labeling [12] of individual atoms by further dividing defect atoms into core defect atoms and interface atoms.

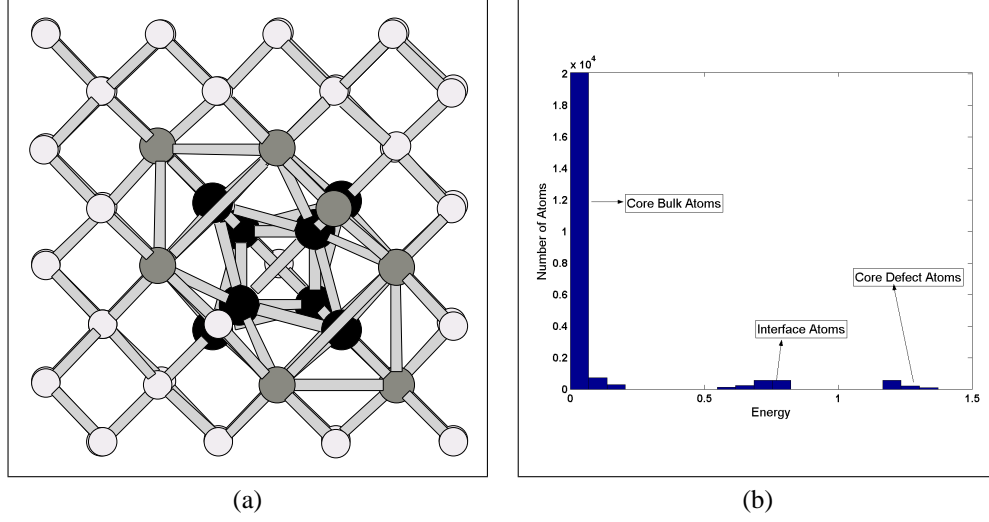


Figure 3: (a) Taxonomy of atoms (b) Energy Plot

Before describing the classification method, we discuss the challenges which need to be addressed to build a robust and efficient classifier for MD data. We list each of them and describe how they are addressed within the context of the proposed algorithm.

4.1 Challenges and Proposed Solutions

4.1.1 Thermal Noise: Thermal motion causes atoms to change their spatial positions. These changes can have two kinds of effect on the detected defect structures:

1. The precise location of the atoms and their inter-pair distances changes from frame to frame. Thus, the classification method must be tolerant to small deviations in spatial positions.
2. The change in spatial positions can also force a previously labeled bulk atom to violate the rules and be labeled as a defect atom (and vice versa) in the next time frame. Therefore the number of atoms in a defect can change, which in turn changes the overall shape of the defect structure making the classification task more difficult.

To address the first problem we consider a data driven approach to derive noise thresholds. From our study we know that the change in position of each atom between consecutive frames is influenced by the position and number of its neighbors. To model this behavior we define a random variable D_i :

$$D_i = \frac{1}{F+1} (M_i + \sum_{j=1}^F M_j),$$

where M_i is the displacement between two consecutive time steps of atom i , with F nearest neighbors within a distance of 2.6 \AA (bond length of Si), and $i \in [1, N]$, N being the total number of atoms in the lattice. We observe that D_i is described by a normal distribution with parameters, μ_{noise} and σ_{noise} (the average mean and standard deviation of all D_i 's). We find μ_{noise} to be close to zero (which is expected because a given atom cannot move very far from its original location between two consecutive time frames). The parameter σ_{noise} is used to model the effect of noise in the defect classification algorithm. From a set of randomly selected 4500 frames, we find σ_{noise} to be 0.19 \AA .

It should be pointed out here that the physical origin of the noise is the thermal motion of the atoms. The magnitude of the thermal motion is given at high temperatures by

$$\langle x^2 \rangle = \frac{3\hbar T}{mk_B \Theta^2},$$

where Θ is the Debye temperature which for Si is 645 K [9], \hbar is planck's constant, k_B is Boltzman constant, m is mass of the atom and T is temperature at which simulation is done. At $T = 1000 \text{ K}$, this yields for the average thermal displacement a value of 0.1 \AA , close to the measured noise value above.

To solve the second problem posed by thermal noise, we propose a weighting mechanism. The weighting mechanism is based on the following two observations:

1. In two consecutive time frames the core defect atoms cannot change considerably.
2. Interface atoms can make a transition from bulk to defect and vice-versa quickly.

Figure 4(a) and Figure 4(b) show the defects detected in two consecutive frames after applying local operators. The

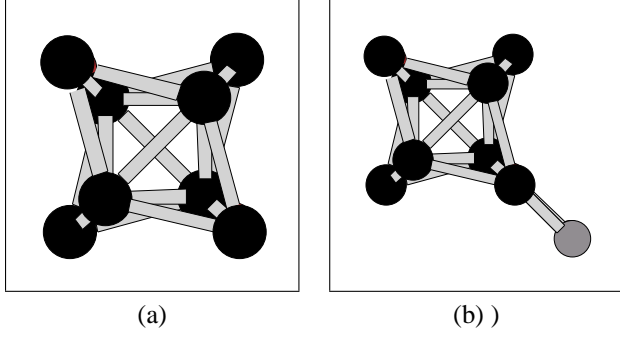


Figure 4: The thermal motion of atoms between two consecutive frames (a) and (b) does not change the core defect atoms (black) but can change the number interface atoms (dark gray). Here an extra interface atom appears in frame (b).

defect in Figure 4(b) has an extra interface atom but the core defect (black atoms) remains unchanged. Therefore, a *weighting mechanism* is proposed to reduce the influence of interface atoms relative to that of core atoms within a defect structure. Each atom in a given defect is assigned a weight given by the number of nearest neighbors within the defect structure. Thus, core defect atoms contribute more to defect the classification than interface atoms. The weights are also used for handling translations (described below) and for computing the feature vector (weighted moments).

The first observation is explained by the fact that the deviations from the perfect bond length and angle is largest for the core defect atoms. Therefore, the thermal motion has a small effect on the already large energy assigned to the core defect atoms. The second observation can be understood by the observation that interface atoms have smaller deviations from ideality than core bulk atoms and usually fail (or conform to) the set of rules by a small margin. Small variation in their locations due to thermal noise can alter their labels.

To summarize, over a period of time core defect atoms will change considerably less than interface atoms. Therefore more emphasis (weight) should be given to core defect atoms while matching two defect structures. This is precisely what our weighting mechanism does.

4.1.2 Translational and Rotational Invariance: Translations and rotations pose another problem in defect classification. The same defect can occur in different positions and orientations in the lattice. To correctly classify a defect, translations and rotations must be resolved before assigning the class.

We describe the shape of a defect by using statistical moments. We use the complete set of first, second and third

order moments. The first order moments describe the location, the second order moments the shape, and the third order moments captures the skewness of the defect. To account for the interface atoms we calculate weighted moments instead of simple moments. (Recall that the *weighting mechanism* assigns high weights to core defect atoms and low weights to interface atoms). The feature vector comprising of weighted moments of a defect is calculated as :

$$D_w^{mnp} = \frac{1}{\sum_{j=1}^N w_j} \sum_{i=1}^N w_i * R_{ix}^m * R_{iy}^n * R_{iz}^p, \quad (4.2)$$

with $m + n + p \leq 3$ and R_{ij} is the j^{th} coordinate of the i^{th} atom of the defect. There are a total of three first-order moments, six second-order moments and ten third-order moments. The three first order moments describe the center of mass of the defect. To recover translational invariance we set the three first order moments D_w^{100} , D_w^{010} and D_w^{001} to zero. The remaining second and third order moments comprise a 16-dimensional feature vector D_w .

There is a finite set of point symmetry operations that map a given lattice onto itself [9]. These symmetry operations consists of rotations, reflections, inversions, and combinations of those. The lattice of crystalline Si is invariant under 24 point symmetry operations. Defects that are related by point symmetry operations of the lattice are classified as the same defect since they have the same physical properties. Instead of working with the defects itself we apply the finite set of symmetry operations on the feature vector. As an example, if a defect is mirrored on the x -plane, all moments involving an odd power of the x -component change sign. In a similar fashion, all rotations are resolved by checking a set of permutations of the original moments.

4.1.3 Shape Based Classification: When matching two defect structures, the classifier should take into account the positions of all individual atoms in the defects. This atom-to-atom matching is relatively expensive. Furthermore because of large numbers of defect classes present in simulation datasets, it would be unrealistic to carry out such an atom-to-atom matching for all classes at each and every time step. Therefore a scheme is needed to effectively reduce the number of candidate classes on which an exhaustive atom-to-atom matching is performed.

We address this challenge by adopting a two step classification process. The first step uses the weighted moments of Eq. (4.2) to determine a subset of defect classes. The weighted moments (feature vector) are used since the describe the overall shape of an object [6]. The second step finds the closest class by taking into account the positions of the atoms and their arrangement in three dimensional space. In essence, both steps use the information about the defect shape. The first step uses the high level information of the

weighted moments and the next step refines it by matching individual atoms. We achieve the desired efficiency because the first step is computationally very cheap and reduces the search space considerably for the next step. Experimental results to corroborate this are shown in Section 5.

The classification based on shape is motivated by physical insight. Physical properties of defects are governed by their atomic structure which is reflected in their overall shape. The shape directly controls the strain field induced by defects [20]. The long-ranged interaction between defects is dominated by the strain field and hence the shape of the defects. For example, small defect clusters in silicon arrange preferentially in a compact shape and attract additional defect atoms, while larger defect clusters prefer an elongated, extended shape [16].

4.1.4 Emergence of new defect classes: The underlying motivation of our effort is to discover information which can assist scientists to better understand the physics behind defect evolution and dynamics, ideally in real time. This can result in new defect classes and defect migration mechanisms. This requires the classification process to be dynamic [7]. The classifier should be dynamic in the classical sense, as in new streaming data elements can be classified, but should also be dynamic in the sense that new classes (defects) if discovered can be added to the classifier model in real time. The new defect should be available when the next frame is processed.

We next present our two-step classification process which integrates all the proposed solutions to the above-mentioned challenges.

4.2 Two Step Classification Algorithm: *Phase 1* of our framework detects the defect(s) from the lattice and *Phase 2* classifies the defect(s). *Phase 1* has been explained in Section 3. The goal of *Phase 2* is to find the type **T** for a given defect **D**. If the type of **D** does not match the type of any of the previously seen defects in the simulation, it is labeled as a new defect and stored in the databases **ID_{shape}** and **ID_{moment}**, where **ID** is a unique simulation identification number, **ID_{shape}** stores the actual three dimensional coordinates and **ID_{moment}** stores the weighted central moments (feature vector) of the defect structure. These databases store all the unique defects detected in the current simulation.

The label of a new defect is of the form *defect.i.j*, indicating that the new defect is the j^{th} defect in the i^{th} frame of the simulation. If **D** is not new then a pointer to the defect class which closely matches **D** is stored. Besides these two databases a summary file is generated which stores names of all detected defects in the simulation along with corresponding frame numbers. We now proceed to describe the two steps of our classifier in detail.

4.2.1 Step 1 - Feature Vector based Pruning: We use a variant of the *KNN* classifier for this task. The value of K is not fixed: instead, it is determined dynamically for each defect. Given the feature vector D_w of a defect D , we compute and sort the distances between D_w and ID_{M_i} , where ID_{M_i} is the mean moment vector of the i^{th} defect in ID_{moment} . All classes with distances less than an empirically-derived threshold are chosen as candidate classes. **Step 2**, works on these K classes only. If no class can be selected, D is considered a new defect. Databases ID_{shape} and ID_{moment} are updated immediately, so that D is available when the next frame is processed.

In a similar fashion, one can use Naive Bayes and Voting based classifiers. Like the *KNN* classifier, these classifiers also provide metrics which can be used to select the top K candidate classes. More specifically, a Naive Bayes classifier provides the probabilities of a feature vector belonging to each class, and a voting based classifier gives the number of votes for each class. The top K classes can then be chosen based on probabilities and votes. We chose VFI as our voting based classifier. As for other types of classifiers, such as the decision tree-based ones, it is not trivial to pick K candidate classes, therefore they are not considered in this work.

From the three applicable classifiers, the *KNN* classifier is chosen because it gives the highest classification accuracy, as described in Section 5. Besides its high accuracy, the *KNN* classifier is incremental in nature. In other words, there is no need to re-build the classification model from scratch if a new class is discovered. In contrast, Naive Bayes and VFI will require the classification model to be re-built every time a new class is discovered.

The K candidate classes are passed to **Step 2**. The representative shapes of these K classes are matched using an exact shape matching algorithm based on the Largest Common Substructure (LCS). Next, we explain the main steps of our exact matching approach.

4.2.2 Step 2 - Largest Common Substructure based algorithm: Assume, A is a defect of unknown type and B is the median defect representing one of the candidate classes from **Step 1**. The defects are mean centered and the rotation is resolved. We next describe all the steps of the **LCS** algorithm in detail.

- **Atom Pairs Formation:** The defects are sorted w.r.t. their x -ordinate. Two atoms i and j in defect A form an atom pair A_{ij} if $\text{distance}(A_i, A_j) \leq \text{bond length}$. This step uses the information about neighbors and connectivity calculated in *Phase 1*. These atom pairs are calculated for both defects. For each atom pair A_{ij} , we store the projection onto the x , y and z -axes represented by A_{ijx} , A_{ijy} and A_{ijz} respectively.

- **Find matching Pairs:** For each pair A_{ij} we find all pairs B_{kl} such that

$$\begin{aligned} |A_{ijx} - B_{klx}| &\leq \sigma_{noise} \\ |A_{ijy} - B_{kly}| &\leq \sigma_{noise} \\ |A_{ijz} - B_{klz}| &\leq \sigma_{noise}, \end{aligned}$$

where the threshold σ_{noise} is obtained as explained in Section 4.1.1.

We represent this equality of atom pairs as $A_{ij} \leftrightarrow B_{kl}$, which implies that the length and orientation of the bond formed by atoms i and j of defect A is similar to the bond formed by atoms k and l of defect B .

By comparing each projection separately, we intrinsically take care of both: bond length and orientation.

- **Find Largest Common Substructure (LCS):** The rules generated in the previous step are used to find the largest common substructure between two defects. We use a region growing based approach to find LCS.

The pseudo code for finding LCS is shown in Figure 5. Before explaining each step in detail, we define the notion of **compatible substructures**:

Two substructures U and V are considered compatible w.r.t. the rule $A_{ij} \leftrightarrow B_{kl}$, if the last atom added to U is the i^{th} atom of defect A and the last atom added to V is the k^{th} atom of defect B .

Being compatible implies that the two substructures have the same number of atoms and the orientation of atoms (which defines shape) is approximately same (within noise thresholds).

The algorithm starts by finding all **compatible substructures** U and V w.r.t to the rule $A_{ij} \leftrightarrow B_{kl}$ (Line 4). The length of U (and V) is increased by one and atoms j and l are added. Lines 5-10 of Figure 5 show this process. However, if no **compatible substructure** is found a new substructure U (and V) is initialized with atoms i and j (k and l). Lines 11-16 in Figure 5 refer to this case. The same process is then repeated for all the rules.

This method also provides the correspondence between atoms. Atoms in U and V have a one-to-one relationship between them.

- **Similarity Metric Computation:** The **Largest Structure (LS)** is then chosen from the common substructures. We use the following metric to determine the similarity between A and B :

$$Sim(A, B) = \frac{2 * \|LS\|}{\|A\| + \|B\|}$$

```

1 Input : All rules
2 For each rule :  $A_{ij} \leftrightarrow B_{kl}$ 
3 {
4   Find Compatible substructures  $U$  and  $V$ 
5   If  $U$  and  $V$  found
6   {
7      $Length = Length + 1$ ;
8      $U[Length] = j$ ;
9      $V[Length] = l$ ;
10  }
11 else
12 {
13   Create new  $U$  and  $V$ ;
14   Store  $i$  and  $j$  in  $U$ ;
15   Store  $k$  and  $l$  in  $V$ ;
16 }
17 }
```

Figure 5: Pseudo code for finding Largest Common Substructure

This similarity is calculated between A and all the K candidate defect classes. The class which gives the maximum similarity greater than a user defined threshold is chosen as the target class. If the maximum similarity is less than the user defined threshold the defect is considered new and both the databases, ID_{shape} and ID_{moment} are updated. The summary database is updated for each defect (previously seen or new).

5 Experiments and Results

In this section we present the results of our framework. As noted earlier we use OHMMS (see Section 2) to generate the datasets. We first, show the advantage of weighted moments over unweighted moments by comparing the accuracies of various classifiers. Next, we demonstrate the accuracy of the LCS algorithm bootstrapped with different classifiers: *KNN*, *Naive Bayes* and *VFI*. Later, we show the scalable aspects of our framework by deploying it on very large datasets (in the giga-byte range). Finally, we present preliminary results demonstrating how our two-step classifier can help us gain a better understanding of defect evolution.

5.1 Robust Classification: To illustrate the importance of using weighted moments as opposed to unweighted moments, we performed the following experiment: a total of 1,400 defects were randomly sampled across multiple simulations conducted at different temperatures. The noise in the simulation depends on the temperature at which the lattice is simulated. Therefore two defects belonging to the same class can have different number of atoms and/or different positions

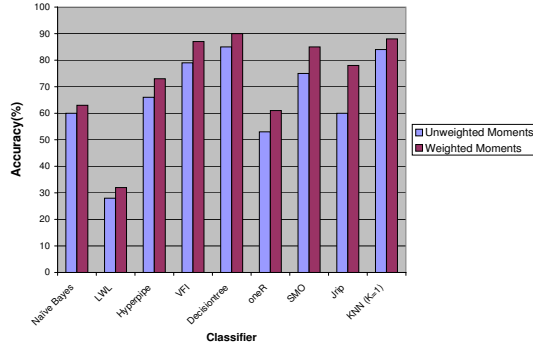


Figure 6: Accuracies of various classifiers

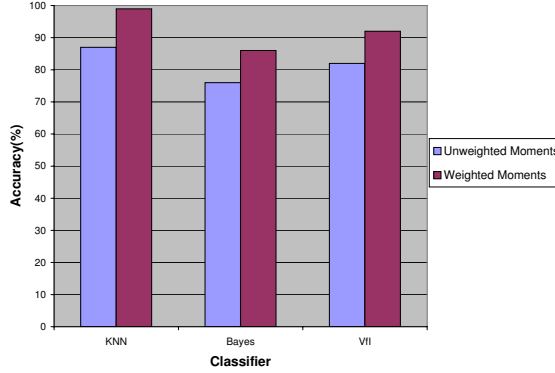


Figure 7: The accuracy of different classifiers for weighted and unweighted moments. The weighted moments on average increase the accuracy by about 8%.

of atoms depending on the temperature, even though their core defect shape remains approximately the same. This sampling strategy ensures that no two defects of the same class are exactly the same. Each defect, in this experiment, belongs to one of the fourteen classes of single interstitial defects that are known to arise in Si.

For comparison purposes, we tried nine different classifiers. Figure 6 clearly demonstrates that all classifiers perform better when weighted moments were used. Classification accuracies of *VFI*, *KNN* ($K=1$) and *Decision tree* based classifiers are comparable (close to 90%). *SMO* (SVM based classifier), also provided good accuracy (85%) but it was quite slow; classifying 1,400 defects took over 25 minutes. On average the classification accuracy increased by 8% when weighted moments were used.

Next, we present the classification accuracies of *Naive Bayes*, *KNN* and *VFI*. These classifiers are modified to pick the K most important classes dynamically (as explained

in Section 4). Figure 7 shows the results for this experiment. *KNN* with weighted moments outperforms all other classifiers by achieving an accuracy of 99% whereas *Naive Bayes* is the least accurate with an accuracy of 86%. Again, weighted moments outperform unweighted moments.

An important point to note is that all the 1,400 defects used for this experiment were labeled manually by a domain expert. However, in actual simulation data there are no predetermined labels since new classes can be created as the simulation progresses. Also there is no training data to build the initial model for *Decision tree* and *Naive Bayes* classifiers. For the purpose of this experiment, we artificially divided the dataset into training (90%) and testing data (10%) for all the classifiers that require training data to build model. Classifier accuracies are averaged over 10 runs of the classifiers.

Only *KNN* and *VFI* can discover new classes in real time. Both classifiers calculate a similarity metric for classification: *distance in the case of KNN and votes for VFI*. If this similarity metric is less than a user defined threshold, a new class label can be assigned to the defect. However, *VFI* will have to build the whole classification model from start whenever a new defect class is discovered. Since large number of defect classes can be created in a simulation, rebuilding the classification model repeatedly will degrade the performance considerably.

Thus the *LCS* algorithm bootstrapped with the *KNN* classifier using weighted central moments is the best choice in terms of accuracy and efficiency.

5.2 Scalable Classification - Large Simulations: We use three large datasets, namely **Two Interstitials**, **Three Interstitials**, and **Four Interstitials** for these experiments. Table 1 summarizes the number of frames, size of the dataset, total number of defects present in the simulation and number of unique defect classes identified by our framework. For all three datasets, our framework was able to correctly identify all the defect structures. However, given the paucity of space we only present an in-depth analysis of the **Three Interstitials** dataset. Similar results were also obtained for other datasets.

Dataset	Number of Frames	Size (in GB)	Total Defects	Unique Defects Detected
Two Interstitials	512,000	4	350,000	2,841
Three Interstitials	200,200	6	320,000	1,543
Four Interstitials	297,000	10	410,000	3,261

Table 1: Datasets Used in Evaluation

Figure 8 shows the defect evolution of a simulation starting with three separated single interstitial defects. The defects move around in the lattice during the first 19,000

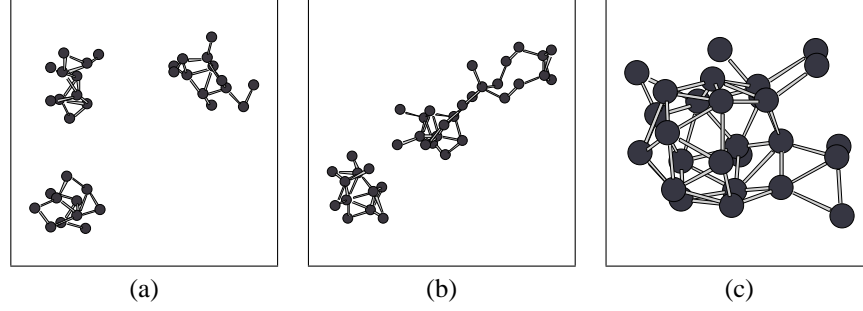


Figure 8: Transitions for tri-interstitials defects. The simulation starts with three separate single interstitials in the first frame (a). In $20,000^{th}$ frame (b) two of the defect form new defect. After $130,000^{th}$ frames the third defect combines with the other defect and forms a large compact defect.

time frames. However, at the $20,000^{th}$ time frame two of the defects join and form a *new* larger defect. This larger defect does not change for a long period of time. However, at the $130,000^{th}$ time frame the third defect joins the new defect and forms a single large defect which remains unchanged until the end of the simulation. For the remainder of this paper we refer to changes in defect shape or type as "transitions".

Though, transitions occur over a large period (thousands of time steps), atoms do not stay at the exact same position in two consecutive frames due to thermal noise. Thermal motion can cause bulk atoms to be labeled as defect atoms (and vice versa). As a result, there exist marginal fluctuations in the shape of a defect from frame to frame. However, the effect of these changes on weighted central moments is small. For example, in the **Three Interstitials** dataset, the total number of defect instantiations in the simulation was around 320,000. However, our classifier detected only 1,543 unique defect classes. These 1,543 defects capture the actual transitions as verified by a domain expert. To reiterate, the use of weighted moments minimizes false positives and ensures robust classification.

The use of weighted moments and pruning in **Step 1** also allows our approach to achieve good scalability. Finding the *LCS* is a relatively expensive algorithm, therefore we want to use it as infrequently as possible. In most cases the number of candidate classes K from **Step 1** (*KNN* classifier) of our dynamic classifier is less than three. For example, in the **Two interstitials** dataset 2,841 unique defects were found however, the *LCS* algorithm only evaluates less than three closest matches. This underlines the usefulness of the pruning step of our classifier. The discovery of all the unique defect classes demonstrates that the correct defect classes are not pruned away. To summarize, pruning based on weighted moments provides scalability to the framework without affecting the accuracy.

Many of these defects are not stable, i.e., they may

exist for as few as 100 time frames; however these unstable defect structures are extremely important since they allow one to understand the physics behind the creation of, and transitioning to, stable structures. We can easily eliminate these unstable structures from our repositories by either maintaining simple counts or by time averaging the frames. However, using both these techniques will result in loss of transition information. To illustrate this point we took the same **Three Interstitials** dataset and averaged it over every 128 frames. In this averaged data, we found only 18 unique defects. It turns out that we found all the possible stable structures, but the actual transitioning behavior was lost.

5.2.1 Timing Results: Figure 9 shows the time taken by OHHMS to complete the simulation and time taken by our framework to analyze the data. The figure also shows the individual time taken by *Phase1* (defect detection) and *Phase2* (classification). *Phase1* takes around 45% of the time and *Phase2* requires the rest of the time. All the experiments are carried out on Pentium 4 2.8 GHz dual processor machine with 1 GB of main memory.

Our classifier can analyze the data almost 1.5 times faster than the data generation rate. This allows us to analyze the data and build the defect databases in real time without dropping/losing any frames. Another advantage is that we are not required to store the large simulation file (of the order of 15GB) on disk. All the needed information about defect type(s), number(s) and transitions, can be obtained from the simulation databases and the summary file.

Next we show how the results produced by the framework can be used for tracking and understanding the movement of defects in the simulation.

5.3 Meta-stable Transitions: Transitions between meta-stable defect structures are important for the description of the defect evolution and dynamics. We present experimental results of a simulation that depicts the transition from one

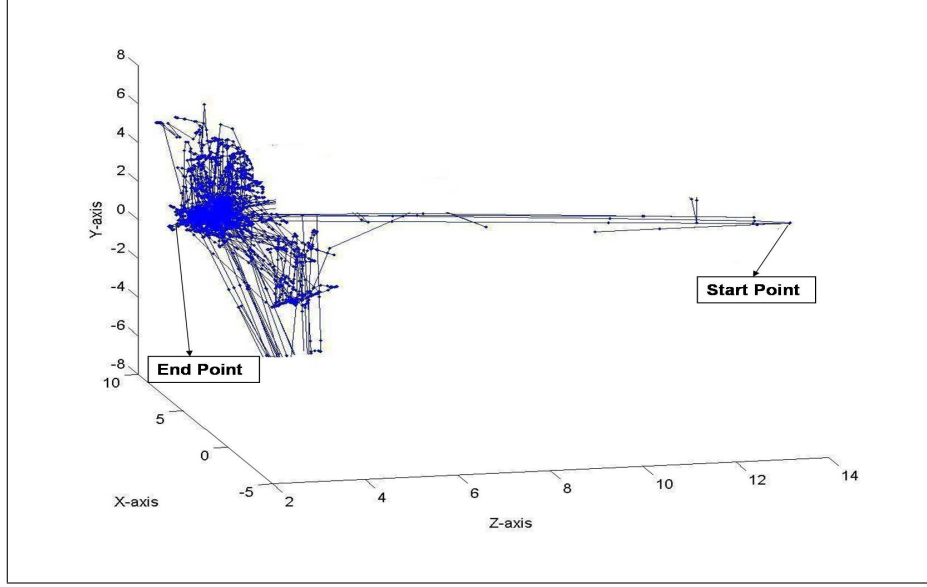


Figure 10: Capturing the movement of defect

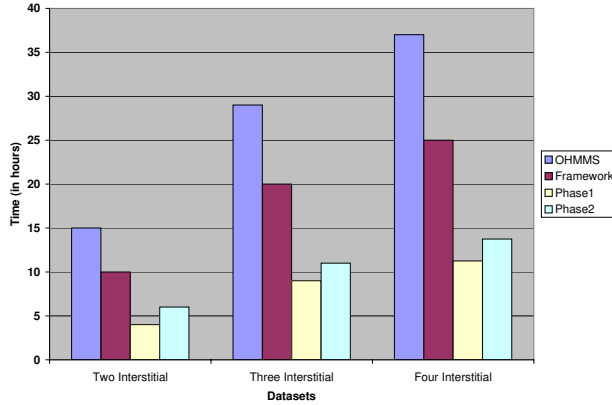


Figure 9: Timing Results. The classification of the defects is about 40% faster than the MD simulation allowing for real-time analysis of the defect evolution.

defect to another.

The simulation is fairly small but provides important insight into the transition between two relevant defect structures in Si [16]. The simulation has 1,300 frames with 67 atoms in each frame and one tri-interstitial defect. We detect 50 unique defects that capture the transitions from the defect of type *I3us-01* to *I3us-03*. (These labels are provided by domain experts). The defect does not separate into multiple parts; therefore, we do not have to deal with the correspon-

dence problem in this case¹. Again, all these results have been verified by our domain expert by manually checking every frame of the simulation.

5.4 Generating defect trajectories: From the summary database produced at the end of simulation analysis, we can glean important information about the movement of a defect in lattices. The summary database provides information to construct a defect's motion trajectory over a period of time. We use a 10,000 frame simulation to show this. In this simulation the defect moves in the $-z$ direction through the lattice, reaches the end of the lattice and then stays in the xy plane. We found 70 unique defects in this simulation. All the detected defects are labeled as one of these 70 classes. Most of these defects were highly unstable. We plot the (x, y, z) coordinates of all the detected defect's weighted centroid at each time stamp. Figure 10 clearly shows the movement of the defect in the $-z$ direction. This idea can be extended to a multiple defect simulation. Since the defects in the summary database are labeled therefore, it should be fairly easy to construct multiple trajectories for multiple defects. By studying these labeled trajectories, one can gain more insight on how a defect evolves and interacts with other defects over time.

6 Conclusions

In this application case study, we propose a two-step classifier to classify the defects in large scale MD simulation

¹Correspondence allows the labeling of two defects with the same class label at two different time epochs.

datasets. The classifier is scalable and incremental in nature. New classes of defects can be discovered and added to classifier model in real time. The approach is also robust to noise (inherent to MD simulations). We present various noise handling schemes and validate these schemes using a physical model and properties of the lattice systems. We demonstrate the capabilities of our approach by deploying it on very large datasets (≥ 4 GB). We were able to find a very small number of unique defect classes from these large datasets. These unique classes capture the defect transitions very well.

We are currently working on solving the correspondence problem in the context of multiple defects. This will enable us to build an automated system to capture important events such as *defect disintegration* and *defect amalgamation*. Another future goal is to understand the interactions among defects in a simulation. Towards this goal, we plan to model the movement of defect as trajectories, tagged by defect class labels, and analyze these trajectories. We also plan to apply other data mining techniques including frequent itemset mining and spatial patterns mining to gain more insight in the actual defect evolution process.

7 Acknowledgments

This work is supported by the National Science Foundation under the auspices of grants NGS-0326386, ACI- 0234273 and NSF Career Award IIS-0347662. The authors would also like to thank Kaden Hazzard and Thomas Lenosky for useful discussions and comments. We would also like to thank Richard Henning for his useful comments while preparing this manuscript.

References

- [1] C. Borgelt and M. Berthold. Mining Molecular fragments: Finding relevant substructures of molecules. In *ICDM*, 2002.
- [2] S.J. Clark and G.J. Ackland. Ab initio calculation of the self interstitial in silicon. *Physical Review Letters* vol. 56, 1997.
- [3] A. S. Clarke and H. Jansson. Structural changes accompanying densification of random hard-sphere packings. *Physical Review E* vol.47, pages 3975–3984, 1993.
- [4] K. Eric Drexler. *Nanosystems: molecular machinery, manufacturing, and computation*. Wiley Publishers, 1992.
- [5] J.M. Galvez and M. Canton. Normalization and shape recognition of three-dimensional objects by 3d moments. *PR*, 26:667–681, 1993.
- [6] M. Hu. Visual Pattern Recognition by Moment Invariants. In *IRE Trans Information Theory*, pages 179–187.
- [7] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Knowledge and Data Discovery (SIGKDD)*, 2001.
- [8] H. Jonsson and H. C. Andersen. Icosahedral ordering in the lennard-jones liquid and glass. *Physical Review Letters* vol. 60, pages 2295–2298, 1988.
- [9] Charles Kittel. *Introduction to Solid State Physics*. John Wiley and Sons, 1971.
- [10] L. Dehapse, H. Toivonen and R. King. Finding Frequent substructures in chemical compounds. In *Knowledge Discovery and Data Mining*, 1998.
- [11] T. J. Lenosky, B. Sadigh, E. Alonso, V. V. Bulatov, T. Diaz de la Rubia, J. Kim, A. F. Voter, and J. D. Kress. Highly optimized empirical potential model of silicon. *Modeling and Simulation in Materials Science and Engineering*, 8:825–841, 2000.
- [12] M. Jiang, T.-S. Choy, S. Mehta, M. Coatney, S. Barr, K. Hazzard, D. Richie, S. Parthasarathy, R. Machiraju, D. Thompson, J. Wilkins, and B. Gatlin. Feature Mining Algorithms for Scientific Data. In *SIAM*, 2003.
- [13] Sameep Mehta, Kaden Hazzard, Raghu Machiraju, Srinivasan Parthasarathy, and John Wilkins. Detection and visualization of anomalous structures in molecular dynamics simulation data. In *IEEE Conference on Visualization*, 2004.
- [14] R. Machiraju, S. Parthasarathy, J. Wilkins, D. Thompson, B. Gatlin, D. Richie, T. Choy, M. Jiang, S. Mehta, M. Coatney, and S. Barr. Mining of Complex Evolutionary Phenomena, Next Generation Data Mining. In *NGDM*, 2003.
- [15] R. Nussinov and H. Wolfson. Efficient Detection of three dimensional Structural Motifs in Biological Macromolecules by Computer Vision Techniques. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 88, Dec 1, 1991.
- [16] D. A. Richie, J. Kim, S. A. Barr, K. R. A. Hazzard, R. G. Hennig, and J. W. Wilkins. Complexity of small silicon self-interstitial defects. *Physical Review Letters*, 92(4):045501, 2004.
- [17] D. A. Richie, J. Kim, and J.W. Wilkins. Applications of real-time multiresolution analysis for molecular dynamics simulations of infrequent events. In *Materials Research Society Symposia Proceedings*, volume 677, pages AA5.1–7. MRS Press, 2001, 2001.
- [18] S. Djoko, D. Cook and L. Holder. Analyzing the benefits of domain knowledge in substructure discovery. In *Knowledge Discovery and Data Mining*, 1995.
- [19] S. Parthasarathy and M. Coatney. Efficient Discovery of Common Substructures in Macromolecules. In *ICDM*, 2002.
- [20] R. Thomson, S. J. Zhou, A. E. Carlsson, and V. K. Tewary. Lattice imperfections studied by use of lattice green’s functions. *Physical Review B* 46, pages 10613–10622, 1992.
- [21] R. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.
- [22] R.J. Needs W.K. Leung and G. Rajagopal. Calculation of silicon self interstitial defects. *Physical Review Letters* vol. 83, 1999.
- [23] X. Wang, J. Wang, D. Shasha, B. Shapiro, S. Dikshitulu, I. Rigoutsos and K. Zhang. Automated discovery of active motifs in three dimensional molecules. In *Knowledge Discovery and Data Mining*, 1997.
- [24] Y. Lamdan and H. Wolfson. Geometric Hashing : a general and efficient model-based recognition scheme. In *Proceedings of the second ICCV*, pages 238–289, 1988.
- [25] C. Zhang and T. Chen. Efficient Feature Extraction for 2D/3D Objects in mesh representation. In *ICIP*, 2001.

Striking Two Birds With One Stone: Simultaneous Mining of Positive and Negative Spatial Patterns

Bavani Arunasalam *

Sanjay Chawla[†]

Pei Sun[‡]

Abstract

We propose an efficient algorithm to mine positive and negative patterns in large spatial databases. The algorithm is based on exploiting a complementarity property for a certain support-like measure. This property guarantees that if a positive k -pattern is "frequent" then $O(k)$ related negative patterns will be infrequent. For the traditional support measure this complementarity property holds true only when the minimum support is over fifty percent. We also confirm the correctness of our approach using Ripley's K-Function, a standard tool in spatial statistics for analyzing point patterns. Extensive experimentation on data extracted from the Sloan Digital Sky Survey (SDSS) database demonstrates the utility of our approach to large scale data exploration.

1 Introduction.

Spatial data mining is the task of discovering, extracting and learning patterns (relationships) in spatial databases. Spatial data is fundamentally different from transactional data in its fundamental nature and distributional tendencies. The objects in a spatial database are characterized by a spatial location and several non-spatial attributes. For example, a galaxy database may contain the x , y and z co-ordinates of the galaxies, the types of galaxies and several other attributes. An example of a spatial data mining task here would be to determine the confidence of finding a *spiral* galaxy in the neighbourhood of an *elliptic galaxy*.

In terms of spatial statistics this problem could be stated as follows: Given a set of points $S = \{x_i\}$, is there a way to quantify that the points in S exhibit a *random*, *correlated* or *negatively correlated* behavior? Scientists working in diverse fields including ecology, geology and astronomy are interested in inferring such information from their data sets as it often provides insights about the underlying mechanisms at play.

The standard tool to test for such patterns is the two-point correlation function ζ . Intuitively ζ calculates the likelihood for finding a point near the vicinity of another given point and can be expressed as

$$\zeta(\delta A) = \frac{N_{obs}(\delta A)}{N_{background}(\delta A)} - 1$$

where $N_{obs}(\delta A)$ is the number of points observed in a small area δA and $N_{background}(\delta A)$ is the expected number of points that will be observed assuming that the data is sampled from a given background distribution. Now the test to characterize the data set can be summarized as

$$\zeta = \begin{cases} < 0 & \text{then } S \text{ is negatively correlated} \\ \approx 0 & \text{then } S \text{ agrees with background} \\ > 0 & \text{then } S \text{ is positively correlated} \end{cases}$$

However choosing a background model is a non-trivial exercise and is largely domain-dependent. Furthermore is the zeta function independent (homogeneous) or dependent upon the coordinates (inhomogeneous) or is it only dependent on the distance between points (isotropic) or does the direction matter (an-isotropic)? Depending upon the assumption, various ζ functions have been proposed in the literature.

Our objective is to use methods from spatial association rule mining to discover complex spatial relationships and then use statistical techniques to determine if these relationships are indeed substantive.

Complex relationships include both positive and negative association rules. Positive rules are the traditional association rules of the form $A \rightarrow B$, which indicates the presence of B in the neighbourhood of A . Negative rules are of the form $A \rightarrow -B$, which indicates the absence of B in the neighbourhood of A .

While much research has been done on positive association rules mining, very little progress has been achieved in mining negative patterns. The reason for this is while generating candidate itemsets in association rules mining, it would be necessary to consider not only the positive patterns, but also the negative patterns and each positive pattern of length k gives rise to $O(k)$ negative patterns making the search space exponentially

*University of Sydney, School of Information Technologies Sydney, NSW, Australia.

[†]University of Sydney, School of Information Technologies Sydney, NSW, Australia.

[‡]University of Sydney, School of Information Technologies Sydney, NSW, Australia.

larger than the space for positive patterns. In this paper we present an efficient algorithm to generate complex relationships in spatial databases.

1.1 Related work Association rules are considered one of the major success stories of data mining research [1]. Association Rules are traditionally described in the framework of market basket analysis. Given a set of items I and a set of transactions T consisting of subsets of I , an Association Rule is a relationship of the form $A \rightarrow^{s,c} B$, where A and B are subsets of I while s and c are the minimum support and confidence of the rule. A is called the antecedent and B the consequent of the rule. The support $\sigma(A)$ of a subset A of I is defined as the percentage of transactions which contain A and the confidence of a rule $A \rightarrow B$ is $\frac{\sigma(A \cup B)}{\sigma(A)}$. Most algorithms for association rule discovery take advantage of the anti-monotonicity property exhibited by the support level: If $A \subset B$ then $\sigma(A) \geq \sigma(B)$.

Our focus is to apply the principle of Apriori in spatial data. Koperski and Han [6] proposed the first extension of the Apriori paradigm to spatial data. However in their method they materialized all the possible spatial relationships that they intended to mine. This is equivalent to determining the universe of candidate interesting relationships. Thus in some ways their technique was hypothesis driven rather than hypothesis generating.

An efficient algorithm to mine a kind of spatial co-locations was proposed by Shekhar and Huang [9]. The concepts of neighbourhood, participation ratio, participation index were defined. Instead of support, the participation index was used as a pruning measure in the conventional Apriori-like technique. The drawback of their method is that some confident co-location rules with low support are also pruned. In order to solve this problem, Huang et al. [5] proposed the concept of maximal participation index and it was used as pruning measure to replace participation index. We will discuss these measures in detail in section 2.2, as they are central to our approach.

An algorithm to mine both positive and negative association rules was proposed by Wu et al. [11]. In their algorithm negative rules are generated from infrequent itemsets and interest is used as a further pruning measure. By considering the negative rules only in the infrequent itemsets, some potential confident negative rules could be lost. For example consider $\text{supp}(A)=0.5$, $\text{supp}(A,B)=0.25$ and $\text{minsup}=0.2$. Here $\{A,B\}$ is a frequent pattern since $\text{supp}(A,B) > \text{minsup}$. Therefore the negative pattern $\{A,-B\}$ will not be considered at all. However, $\text{supp}(A,-B) = \text{supp}(A) - \text{supp}(A,B) = 0.25$. This shows that even though $\{A,-B\}$ is a frequent pattern,

it is pruned in this approach. Therefore this method will not be able to mine the complete set of frequent negative patterns in the dataset.

Another approach for mining positive and negative relationships was proposed by Antonie and Zaiane [2]. In addition to the *minsup* and *minconf* measures they introduced the correlation threshold as the third parameter. Their approach generates only a subset of negative rules, which they refer to as *confined negative association rules*, where the entire antecedent or consequent must be a conjunction of negated attributes or a conjunction of non-negated attributes.

1.2 Key Insight and Main Contributions

1. Our objective is to simultaneously mine complex spatial relationships which include positive and negative patterns. We can achieve this objective using the following observation:

Suppose $\{A, B\}$ is a positive 2-itemset and $\sigma(A, B)$ its support. Then the following relationship can easily be derived

$$(1.1) \quad \sigma(A, -B) = \sigma(A) - \sigma(A, B)$$

Here $-B$ represents the absence of B . Now, because of the anti-monotonicity of the support measure, $\sigma(A) \geq \sigma(A, B)$. Therefore $\sigma(A, B) > 50\%$ implies that $\sigma(A, -B) < 50\%$. Note, a support greater than 50% is required for this to hold uniformly. This will enable the pruning of negative patterns (those that contain negative items) based on frequent positive patterns.

However, choosing a support level of greater than fifty percent will not lead to *interesting* results because either there will be very few itemsets which have such high support or the itemsets which do have such high support will probably be well known. Thus we need a measure M such that

- (a) A relationship similar to Equation 1.1 holds.
- (b) It is natural to set high threshold values (like confidence).
- (c) A monotonic-like property holds which will enable aggressive pruning of candidate patterns.

We will use the Maximal Participation Index (MaxPI) introduced by Huang et al. [5] and show that it satisfies all the three properties enumerated above.

To the best of our knowledge this is the first efficient algorithm which can simultaneously mine positive and negative patterns.

2. In our earlier work [7], we had introduced a taxonomy of complex spatial relationships that are worthy of being mined. Table 1 gives the basic definitions of these relationships and we will show how to efficiently mine these relationships using our approach.
3. We have carried out detailed experiments on a large extract of the Sloan Digital Sky Survey database to show how our approach can be used in practice. We discover patterns, which are known to be genuine and others which may turn out to be "interesting" thus confirming the role of data mining as a tool for credible hypothesis generation.
4. One of the weaknesses of association rule mining is that the number of rules that are generated far exceeds the "genuine" number of patterns that are interesting. We follow an approach analogous to the filter-refine strategy popular in spatial databases. We will use the NP_MaxPI algorithm to generate candidate patterns and use the Ripley's K-function to test and filter out rules, which are not substantive. However, our approach is more general compared to Ripley's K-function as Ripley's K-function can only find relationships between two features, whereas our approach can be applied to any number of features. Also Ripley's K-function finds relationships globally from the entire set of spatial objects whereas, our approach generates relationships locally within the neighbourhood of spatial objects. We will discuss Ripley's K-function in detail in section 4.3.

The remainder of the paper is organized as follows: Section 2 gives the definitions and the basic concepts involved in spatial data mining. In this section we also describe the Maximal Participation Index with examples, which is central to our approach. In Section 3 we introduce our approach to mining complex patterns including positive and negative rules. In this section we present a lemma, by using which our algorithm prunes a large number of negative patterns and hence effectively reduces the number of candidate itemsets. Section 4 presents our experimental results, which show the scalability, applicability and correctness of our algorithm. Section 5 concludes the paper with a summary of the research.

2 Basic Definitions and Concepts

In this section we present the basic concepts and definitions used in this paper.

Relationship	Notation	Description	Example
Positive	$A \rightarrow B$	Presence of B in the neighbourhood of A	Sa type Spiral Galaxies \rightarrow Sb type Spiral Galaxies
Negative	$A \rightarrow -B$	Absence of B in the neighbourhood of A	Elliptic galaxies tend to exclude spiral galaxies. $E \rightarrow -S$.
Self-Co-location	$A \rightarrow A+$	Presence of many instances of the same feature in a given neighbourhood	Elliptic galaxies tend to cluster more strongly. $E \rightarrow E+$.
Complex	$A+ \rightarrow -C, B$	combination of two or more of the above relationships	Clusters of elliptic galaxies tend to exclude other types of galaxies. $E+ \rightarrow -S$.

Table 1: Types of Relationships

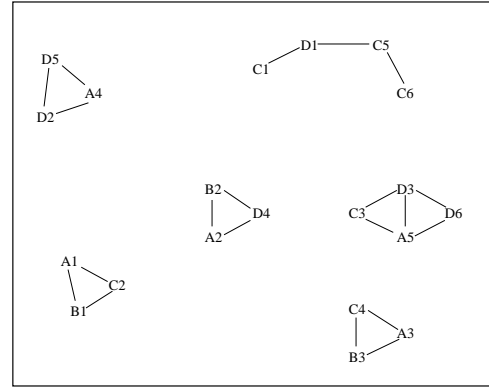


Figure 1: An Example of spatial co-location patterns

2.1 Co-location An important task in spatial data mining is extracting relationships between features that co-locate.

DEFINITION 2.1. (Co-location) Two spatial objects are said to co-locate if the Euclidean distance between the objects is less than or equal to the user-specified neighbourhood distance d .

Figure 1 gives an example of a set of spatial objects. In Figure 1, A_1 represents the ID of a spatial object of feature A, B_1 represents the ID of a spatial object of feature B and so on. The line joining two points indicates that the distance between them is less than or equal to the neighbourhood distance 'd' and hence they co-locate.

DEFINITION 2.2. (Clique) A set of spatial objects S is said to be a clique if every object in S co-locate with every other object in S and no super set of S has this property.

No	Clique
i	C_1, D_1
ii	C_5, D_1
iii	C_5, C_6
iv	A_4, D_2, D_5
v	A_5, C_3, D_3
vi	A_5, D_3, D_6
vii	A_1, B_1, C_2
viii	A_2, B_2, D_4
ix	A_3, B_3, C_4

Table 2: Cliques in Figure 1

Table 2 gives the cliques found in Figure 1.

Co-locational patterns are the relationships between the features in cliques of spatial databases. For example, in Figure 1 and Table 2, the co-locational pattern $\{A, C\}$ occurs 3 times, in v, vii and ix. Spatial relationships are confident co-locational patterns found in spatial databases.

2.2 Maximal Participation Ratio In this section we will briefly describe the notion of Maximal Participation Index (maxPI) as described by Huang et al. in [5] where more details can be found.

DEFINITION 2.3. (Participation ratio) Given a co-location pattern L and a feature $f \in L$, the participation ratio of f , $pr(L, f)$, can be defined as the support of L divided by the support of f . For example, in Figure 1, the support of $\{A, B, C\}$ is 2 and the support of C is 6, so $pr(\{A, B, C\}, C) = 2/6$.

DEFINITION 2.4. (Maximal Participation Index) Given a co-location pattern L , the maximal participation index of L , $maxPI(L)$ can be defined as the maximal participation ratio of all the features in L , i.e. $maxPI(L) = \max_{f \in L} \{pr(L, f)\}$. For example, in Figure 1, $maxPI(\{A, B, C\}) = \max(\frac{2}{5}, \frac{2}{3}, \frac{2}{6}) = \frac{2}{3}$.

A high maximal participation index indicates that at least one spatial feature (which we call the *maxfeature*) strongly implies the pattern. By using maxPI, rules with low frequency but high confidence can be found, which would otherwise be pruned by a support threshold [5].

2.2.1 Mining rules with low support and high confidence using maxPI As discussed above the maxPI measure could be used to generate rules with low support and high confidence. For example if A is an infrequent feature the support of $\{A, B\}$ will be very low and hence will be pruned by the support threshold.

However the confidence of the rule $A \rightarrow B$, defined as, $conf(A \rightarrow B) = \frac{support(A, B)}{support(A)}$ could be high.

This problem can be directly addressed by using the *maxPI* measure. *maxPI*(A, B) is given as,

$$\begin{aligned} maxPI\{A, B\} &= \max(pr(\{A, B\}, A), pr(\{A, B\}, B)) \\ &= \max(conf(A \rightarrow B), conf(B \rightarrow A)) \end{aligned}$$

This shows that a high confidence for the rule $A \rightarrow B$ will lead to high maxPI, which will prevent rules with low support and high confidence from being pruned¹.

2.2.2 The weak monotonic property of maxPI

Maximal participation index is not monotonic with respect to the pattern containment relations. For example, in Figure 1, $(maxPI(\{A, C\}) = 3/5 < (maxPI(\{A, B, C\}) = 2/3$. Interestingly, as pointed out in [5] the maximal participation index does have the following weak monotonic property:

If P is a k -co-location pattern, then there exists at most one $(k-1)$ subpatterns P' of P such that $maxPI(P') < maxPI(P)$.

Relying on this weak monotonic property, the Apriori-like algorithm can be modified to mine confident patterns by using a maxPI threshold.

3 The NP_MaxPI Algorithm

The *maxPI* measure was introduced to discover *positive* co-location patterns with high confidence and low support. However our goal is to mine complex relationships which include both positive and negative patterns. We now show that *maxPI* is a good candidate to help achieve our goal, i.e., simultaneously discover both positive and negative patterns.

The main challenge in mining complex relationships is the high processing cost due the large number of candidate itemsets which include positive and negative features. Each positive candidate k -pattern will give rise to $O(k)$ k -patterns which contain a negative feature.

Let $F = \{A, B, C, D\}$ be the set of all features in the spatial database. Then for mining complex relationships, the candidate 1- itemsets would be $\{A, B, C, D, -A, -B, -C, -D\}$. Hence as the number of features in the spatial database increases, the candidate 1-itemsets is doubled. This would result in an exponential growth in the candidate space for larger itemsets. In this paper we propose an approach, which effectively reduces the number of candidate itemsets when mining for positive and

¹This is a strength and weakness of the maxPI measure. While patterns with low support and high confidence are likely to be "interesting", it is possible that they are an artifact of the data and may not be statistically significant.

negative patterns. Our approach is based on Lemma 1 which shows how a large number of negative patterns can be pruned when a positive pattern is greater than the threshold t where $t \geq 0.5$. Since maxPI is based on the confidence, such a high threshold is not unusual for maxPI.

For example, let $F = \{A, B, C, D\}$ be a 4-itemset. Then, if $\maxPI(\{A, B, C, D\}) = \text{pr}(\{A, B, C, D\}, A) > t > 0.5$, then we prove that $\maxPI(\{A, B, C, D\})$, $\maxPI(\{A, B, -C, D\})$ and $\maxPI(\{A, B, C, -D\})$ are all less than t and hence could be pruned.

Notations used:

1. Let $F_k = \{f_1, f_2, \dots, f_k\}$ be a k -pattern and $F_{k-l} = \{f_1, \dots, f_{l-1}, -f_l, f_{l+1}, \dots, f_k\}$.
For example, if $F_3 = \{A, B, C\}$ where $f_1 = A$, $f_2 = B$ and $f_3 = C$, then $F_{3-2} = \{A, -B, C\}$.
2. Let $\sigma(F_k) = \text{support}(F_k)$ and $\sigma(F_{k-l}) = \text{support}(F_{k-l})$.

LEMMA 3.1. *Let F_k be a k -pattern and $t \geq 0.5$. Furthermore, assume that $\sigma(-f) > \sigma(f) \forall f$. If $\maxPI(F_k) \geq t$ and $\maxPI(F_k) = \text{pr}(F_k, f_j)$ then $\maxPI(F_{k-l}) < t$ for every $f_l \in F_k, f_l \neq f_j$.*

Proof. By definition, $\maxPI(F_{k-l})$

$$= \text{Max} \left\{ \text{Max}_{i=1, i \neq l}^k \left\{ \frac{\sigma(F_{k-l})}{\sigma(f_i)} \right\}, \frac{\sigma(F_{k-l})}{\sigma(-f_l)} \right\}$$

In spatial data, the number of instances of absence of a feature f_l , i.e., $\sigma(-f_l)$, will be equal to the number of points in the area of observation which do not contain that feature. In real spatial datasets it is appropriate to assume that the support of absence of features would be greater than the support of presence of features.² Therefore,

$$\maxPI(F_{k-l}) = \text{Max}_{i=1, i \neq l}^k \left\{ \frac{\sigma(F_{k-l})}{\sigma(f_i)} \right\}$$

But from the assumption,

$$\maxPI(F_k) = \text{pr}(F_k, f_j) = \frac{\sigma(F_k)}{\sigma(f_j)}$$

This implies that $\sigma(f_j) \leq \sigma(f_i)$ for every $f_i \in F_k, i \neq j$. Hence,

$$\maxPI(F_{k-l}) = \frac{\sigma(F_{k-l})}{\sigma(f_j)}$$

²This assumption implies that patterns of the form $(-f_i, -f_j)$ are likely to be below the threshold value. We therefore disregard them during the mining process.

$$= \frac{\sigma(f_1, \dots, f_{l-1}, f_{l+1}, \dots, f_k)}{\sigma(f_j)} - \frac{\sigma(F_k)}{\sigma(f_j)} \quad (1)$$

We know that

$$\{f_1, \dots, f_{l-1}, f_{l+1}, \dots, f_k\} \subset F_k$$

and

$$\frac{\sigma(F_k)}{\sigma(f_j)} = \text{MaxPI}(F_k) > t$$

Therefore from the anti-monotonic property of the support measure, it follows that

$$\frac{\sigma(f_1, f_2, \dots, f_{l-1}, f_{l+1}, \dots, f_k)}{\sigma(f_j)} > t > 0.5$$

Hence it follows from (1) that $\maxPI(F_{k-l}) < t$.

COROLLARY 3.1. *Let $F = \{f_1, f_2, \dots, f_m\}$ be the set of all features in a spatial database. Then, $0 \leq P \leq m^2 - m$ where P is the size of the candidate 2-itemsets pruned because of Lemma 3.1.*

When mining for positive and negative patterns, for every $f_i \in F$, we also have to consider $-f_i$. This increases the feature set size to $2m$.

Hence the number of candidate itemsets of size 2 generated from this feature set is $\binom{2m}{2}$ (because of maxPI no pruning is done at the first level).

From this set we remove candidates of the form $(-f_i, -f_j)$ and $(f_i, -f_i)$. This reduce the size of the candidate set to $\binom{2m}{2} - \binom{m}{2} - m$.

Case 1 : All positive candidate item sets of size 2 are greater than the threshold t . Then by Lemma 3.1 all the candidate itemsets of size 2 with negative features would be pruned. Hence the additional number of pruning will be

$$\binom{2m}{2} - \binom{m}{2} - m - \binom{m}{2} = m^2 - m$$

Case 2 : All positive candidate item sets of size 2 are less than the threshold t .

Then all negative candidate item sets will be checked and hence there will be no additional pruning because of Lemma 3.1 .

3.1 NP_MaxPI Algorithm: Example With the help of an example we describe the details of the *Negative Pruning MaxPI* (NP_MaxPI) algorithm for mining complex spatial relationships.

Algorithm: NP_MaxPI

Input: Transaction table, t

Output: Confident complex patterns

```

k=1
 $F_k \leftarrow \{f_1, f_2, \dots, f_n\} \cup \{-f_1, -f_2, \dots, -f_n\}$ 
 $\cup \{f_1^+, f_2^+, \dots, f_n^+\}$ 
While  $F_k \neq \phi$ 
    k=k+1
    //generate candidate itemsets
     $C_k = \text{maxPIgen}(F_{k-1})$ 
    for each  $c \in C_k$ 
        if every feature in  $c$  is positive
            OR  $\text{check\_negative}(c) = \text{true}$  then
                generate  $F_k$  from  $C_k$  using MaxPI
            end if
        end for
    end While
End While

```

procedure: check_negative

Input: candidate itemset, c

Output: Boolean variable, *check*

```

check=true
for each negative feature  $-f_i \in c$ 
     $cp = c - (-f_i) + f_i$ 
    if  $cp \in F_k$  and  $f_i \notin \text{maxFeature}(cp)$  then
        check = false
    end if
end for
return check

```

Figure 2: *AlgorithmNP_MaxPI*. We generate candidate itemsets in the same way as *apriori-gen*(F_{k-1}), except that we use the weak anti-monotonic property instead of anti-monotonic property.

No	Clique	Transaction
i	A_1, B_1, C_1	A, B, C
ii	A_2, B_2, C_2	A, B, C
iii	C_2, D_1	C, D
iv	A_3, C_3	A, C
v	A_3, B_6, D_2	A, B, D
vi	A_4, B_3, D_3	A, B, D
vii	A_5, B_5, D_4	A, B, D
viii	B_4, C_4, D_5	B, C, D
ix	A_6, D_6	A, D

Table 3: Clique Set and Transaction Set

1. Given a spatial database, create a clique set such that every row represents a set of point forming a clique in the dataset. Table 3 gives an example of a clique set.
2. Create the transaction set from the clique set, in which every row represents the features of points in the corresponding row of the clique set. For example, row (i) in Table 3 gives the set of point A_1, B_1, C_1 forming a clique. Here A_1 is a spatial point with feature 'A'. Similarly B_1 is a spatial point with feature 'B' and C_1 is a spatial point with feature 'C'. Hence the transaction corresponding to this clique is $\{A, B, C\}$.
3. Generate candidate itemsets of size one with positive and negative features. For the given example, the candidate 1-itemsets are $\{A, -A, B, -B, C, -C, D, -D\}$. For simplicity (in this example), we ignore the self_co-locations such as $A+$. Since maxPI for all the 1-itemsets are 1, we do not prune any candidates at this level.
4. Generate candidate 2-itemsets from 1-itemsets, automatically pruning patterns such as $\{A, -A\}$ and patterns with all negative features such as $\{-A, -B\}$.
5. Check maxPI of the candidate 2-itemsets against the user-defined threshold, and generate frequent 2-itemsets. While checking for negative candidate itemsets use Lemma 1 for additional pruning. Tables 4 and 5 show the process of checking candidate itemsets of sizes 2 and 3 with a threshold t of 60%. The column 'checked' shows whether the candidate itemset was checked against the threshold or not. When using the MaxPI algorithm all the given candidate itemsets would have been checked. However NP_MaxPI avoids these checks because of Lemma 1. This increases the efficiency of mining complex patterns. The column *MaxFeature* de-

candidate itemset	checked	Pr	MaxPI	MaxFeature	Pruned
AB	Y	5/6,5/6	5/6	A,B	N
A-B	N				
AC	Y	3/6,3/4	3/4	C	N
A-C	Y	3/6	3/6	-	Y
AD	Y	4/6,4/6	4/6	A,D	N
A-D	N				
-AB	N				
-AC	N				
-AD	N				
BC	Y	3/6,3/4	3/4	C	N
B-C	Y	3/6	3/6	-	Y
BD	Y	4/6,4/6	4/6	B,D	N
B-D	N				
-BC	N				
-BD	N				
CD	Y	2/4,2/6	2/4	-	Y
C-D	Y	2/4	2/4	-	Y
-CD	Y	4/6	4/6	D	N

Table 4: Mining candidate 2- itemset

candidate 3-itemset	checked	pr	MaxPI	MaxFeature	Pruned
ABC	Y	2/6,2/6,2/4	2/4	-	Y
ABD	Y	3/6,3/6,3/6	3/6	-	Y
ACD	Y	0/6,0/4,0/6	0	-	Y
A-CD	Y	4/6,4/6	4/6	A,D	N
BCD	Y	1/6,1/4,1/6	1/4	-	Y

Table 5: Mining candidate 3- itemset

notes the feature for which the participation ratio of the candidate itemset was greater than the threshold. For example in Table 4, for candidate item AB, $\text{pr}(\{A,B\},A)=5/6 > t$ and $\text{pr}(\{A,B\},B)=5/6 > t$. Therefore by Lemma 1, we do not have to check $\text{maxPI}(\{-A,B\})$ and $\text{maxPI}(\{A,-B\})$ as we know that they will be less than t . If we consider AC, $\text{pr}(\{A,C\},A)=3/6 < t$ and $\text{pr}(\{A,C\},C)=3/4 > t$. Hence by Lemma 1, we do not need to check $\text{maxPI}(\{-A,C\})$. However we need to check $\text{maxPI}(\{A,-C\})$.

- Repeat steps 4 and 5 for the consequent levels until there are no more frequent itemsets.

4 Experiments, Results and Analysis

We have carried out three sets of experiments to demonstrate the scalability, applicability and the statistical correctness of NP_MaxPI.

Scalability We created a large synthetic data set in order to compare the MaxPI and the NP_MaxPI

algorithms. In particular, we wanted to test how much additional pruning was being achieved by the use of Lemma 1.

Applicability We extracted a large sample from the Sloan Digital Sky Survey Database and applied the NP_MaxPI algorithm to test the efficacy of our approach on a real data set.

Correctness We applied the NP_MaxPI algorithm on a small data set to obtain an output O which contains patterns of the form $\{A,B\}$ and $\{C,-D\}$. We then applied the Ripley's K-function test to check whether A and B are positively correlated and C and D are negatively correlated.

4.1 Performance evaluation of the algorithm

We generated synthetic datasets of sizes ranging from 170K to 1 million transactions with ten different features. These datasets were generated using the following method. With each of the twenty features(ten simple features eg. 'A' and ten self-co-locations eg. 'A+')

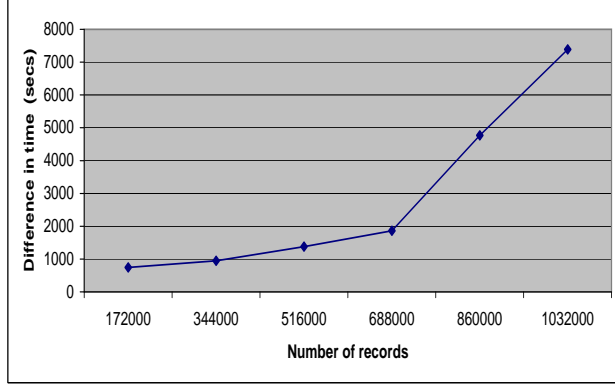


Figure 3: Comparison of efficiencies of MaxPI and *NP_MaxPI* algorithms

as the first feature, a random number of transactions were generated. Each transaction was generated with a random size from 1 to 10. Depending on the size of the transaction the consequent features were chosen randomly from the feature set.

We applied the MaxPI algorithm and *NP_MaxPI* algorithm on the datasets to generate confident complex patterns. Figure 3 shows the difference in time taken by both these methods to mine complex patterns from datasets of different sizes. It can be seen that the time difference increases rapidly with the size of the dataset, which indicates the extensive pruning of the negative patterns done by *NP_MaxPI* algorithm and hence the increase in efficiency. We confirmed that the rule sets generated by the two method are identical by using the *diff* function.

We carried out a second experiment to compare the performance of MaxPI algorithm and *NP_MaxPI* with increasing number of features in the dataset. We generated synthetic datasets with number of features ranging from 10 to 25. Figure 4 shows the time taken by the two algorithms to mine confident complex patterns from these datasets. From the figure, it can be seen that there is a rapid increase in time with the MaxPI algorithm, while with *NP_MaxPI* the time taken is comparatively less.

4.2 Application of NP_MaxPI algorithm on SDSS database

4.2.1 Data Preparation The data for this experiment was obtained from the SDSS Data Release 1 online catalogue service. The online data contained information about over 150000 celestial objects. Among the hundreds of attributes stored in the database for each object we extracted the following attributes for our ex-

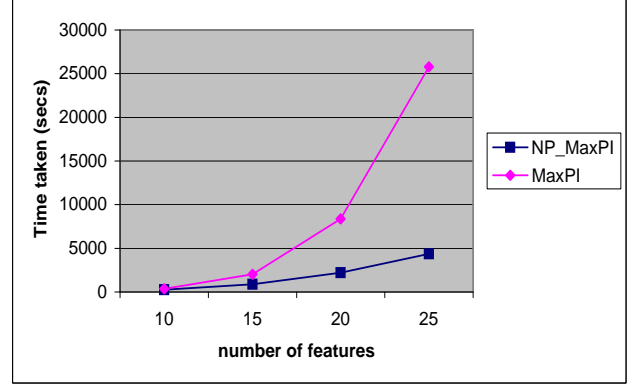


Figure 4: Comparison of efficiency with increasing number of features

periment

- The object's ID
- coordinates (as a unit vector)
- object type
- primary target flags
- red shift
- the difference between the u & r light magnitudes (used to separate galaxy types)

The distances between objects were calculated using Hubble's Law given by $D = c * z / H_0$, where c is the speed of light, z is the red shift and H_0 is the Hubble's constant and is $71 km sec^{-1} Mpc^{-1}$ [4].

To ensure that the results for measuring the distance to each object would be as accurate as possible, only objects with a $zConf$ value > 0.95 (i.e. the object's red shift is $> 95\%$ certain) and $zWarning = 0$ (i.e. there's no problem with the red shift) are used. This filtering cuts down the number of objects to around 117000.

The 'object type' attribute classifies the objects into 25 categories. However the current online data has objects only in 17 of these categories. Among the various object types, we extracted only the galaxies since 90% of the objects were classified as 'Galaxies'. We further classified the galaxies into 'main' galaxies and Luminous Red Galaxies (LRG). The main galaxies are closer (to the Earth) and brighter than the LRG. In the SDSS database the LRG were flagged with a particular value. We classified the 'main' and LRG galaxies further into Early and Late galaxies using the UV & red light magnitudes. A $u-r$ value greater than or equal to 2.22 indicate Early galaxy and less than 2.22 indicate Late galaxies. From Hubbles Tuning Fork model [10] it

Objects	Symbol
GALAXY-LRG-EARLY	A
GALAXY-LRG-LATE	B
GALAXY-MAIN-EARLY	C
GALAXY-MAIN-LATE	D

Table 6: List of object types and their symbols

Neighbourhood Distance :1 megaparsec
min confidence : 70%
$B+ \Rightarrow -C$
$D+ \Rightarrow -C$
$A+ \Rightarrow -B - D$
$B \Rightarrow -A - C$
$C \Rightarrow -B - D$
$C+ \Rightarrow -B - D$
$A \Rightarrow -B - C - D$

Table 7: Rules from SDSS database

follows that early galaxies are elliptical in shape and late are spiral/irregular. Table 6 lists the different types of galaxies and the corresponding symbols used in this paper.

4.2.2 Rules Generated from SDSS Database

We applied the NP_MaxPI algorithm to the data set extracted from the SDSS database and some of the interesting rules generated are shown in Table 7. The entire result set could be obtained from <http://www.cs.usyd.edu.au/~chawla/sdss.html>. From Table 6 we see that features A and C are early galaxies and hence they are elliptical in shape. Features B and D are late galaxies and hence they are spiral in shape.

Among the rules in Table 7, the rules to be noted are $A+ \Rightarrow -B - D$ and $C+ \Rightarrow -B - D$. These rules show that A+ is negatively correlated with B and D but not C. Similarly C+ is negatively correlated with B and D and not A. These rules conform to the well know fact that when elliptical galaxies co-locate the spiral galaxies are excluded.

4.3 NP_MaxPI and Ripley's K-Function Ripley's K function is a method in spatial statistics which is used to characterize the spatial pattern of point data [3]. The K function is given by

$$K(t) = \lambda^{-1} E$$

where E is the number of spatial points within distance t of a randomly chosen point and λ is the density (number per unit area) of events. If A is the area of the study region and N is the observed number

of points then, $\lambda = N/A$.

Ripley's K function could be used to test complete spatial randomness i.e. to test whether the observed events are consistent with a homogeneous Poisson process. If so, $K(t) = \pi t^2$ for all t. i.e. under complete randomness $t = (K(t)/\pi)^{1/2}$ for all t. Let

$$L(t) = (K(t)/\pi)^{1/2}$$

Depending on the value of L(t), the distribution of the points could be characterized as follows:

$$L(t) = \begin{cases} < t & \text{points are negatively correlated} \\ \approx t & \text{points are randomly distributed} \\ > t & \text{points are positively correlated} \end{cases}$$

Dixon [8] has suggested the following generalization of K(t) function for multivariate spatial point process:

$$K_{ij}(t) = \lambda_j^{-1} E$$

Where E is the number of type j events within distance t of a randomly chosen type i event.

Under complete spatial randomness $K_{ij}(t) = \pi t^2$. Hence

$$L_{ij}(t) = (K_{ij}(t)/\pi)^{1/2} = t$$

This shows that values of $L_{ij}(t) > t$ indicate attraction between the i type and j type points and values $< t$ indicate repulsion.

We applied the NP_MaxPI algorithm to a very small synthetic data set with 20 points and generated a set of complex rules with *minconf* 70%. The rules are given in Table 8.

We applied Ripley's K function to the synthetic data set and calculated the L_{ij} values for different values of t where i, and j were the types of features in each of the rules in Table 8. The different values of L_{ij} are given in Table 9.

Table 9 shows that for type A and D, $L_{AD} \geq t$ which indicates that these two types of objects are positively correlated. When comparing types C and D, we find that $L_{CD} \leq t$ which shows negative correlation between the types. Similarly types C and B show negative correlation. These confirm the rules in Table 8. However it should be noted that Ripley's K-function finds relationships globally from the entire set of spatial objects whereas, our approach generates relationships locally within the neighbourhood of spatial objects.

5 Conclusion

In this paper we demonstrated the problem of generating complex patterns in spatial databases. We then presented an efficient approach, NP_MaxPI, which

Neighbourhood Distance =4
$A \rightarrow D$ conf=83.33%
$D+ \rightarrow A$ conf=100.00%
$D+ \rightarrow -C$ conf=75.00%
$C+ \rightarrow -B$ conf=100.00%

Table 8: Rules from the synthetic data set

t	L_{AD}	L_{DC}	L_{CB}
0.00	0.00	0.00	0.00
0.50	0.00	0.00	0.00
1.00	0.00	0.00	0.00
1.50	0.00	0.00	0.00
2.00	2.02	0.00	0.00
2.50	3.50	1.81	2.33
3.00	4.52	2.52	2.33
3.50	4.52	3.11	3.37
4.00	4.52	3.11	3.37

Table 9: Results from Ripley’s function

uses the maxPI measure instead of the traditional support measure for mining complex patterns. We showed that our approach effectively reduces the candidate itemsets by exploiting a complementarity property of maxPI measure which extensively prunes candidate negative patterns. We then presented and analysed various experimental results. The results of our experiments show (i) the significant performance improvement over MaxPI algorithm, (ii) the efficacy of our approach on real dataset by generating confident complex pattern in SDSS spatial database and (iii) the statistical correctness of our algorithm using Ripley’s K function.

6 Acknowledgements

Thanks to Chris Bowman for helping us create the data set. The second author is partially supported by an ARC Discovery Research Grant.

References

- [1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of 20th International Conference on Very Large Data Bases VLDB*, pages 487–499. Morgan Kaufmann, 1994.
- [2] Maria-Luiza Antonie and Osmar R. Zaiane. Mining positive and negative association rules: An approach for confined rules. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD04*, 2004.
- [3] Noel A.C. Cressie. *Statistics for spatial Data*. John Wiley and Sons, 1993.
- [4] D.N.Spergel, M.Bolte, and W.Freedman. The age of the universe. *Proceedings of the National Academy of Science*, 94:6579–6584, 1997.
- [5] Yan Huang, Hui Xiong, Shashi Shekhar, and Jian Pei. Mining confident co-location rules without a support threshold. In *Proceedings of the 18th ACM Symposium on Applied Computing ACM SAC*, 2003.
- [6] Krzysztof Koperski and Jiawei Han. Discovery of spatial association rules in geographic information databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases*, pages 47–66. Springer-Verlag, 1995.
- [7] Rob Munro, Sanjay Chawla, and Pei Sun. Complex spatial relationships. In *Proceedings of the 3rd IEEE International Conference on Data Mining, ICDM 2003*, pages 227–234. IEEE Computer Society, 2003.
- [8] P.Dixon. Ripley’s k function. department of statistics, iowa state university, 2001.
- [9] Shashi Shekhar and Yan Huang. Discovering spatial co-location patterns: a summary of results. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases SSTD01*, 2001.
- [10] V.J.Martin and E.Saar. *Statistics of the Galaxy Distribution*. Chapman and Hall/CRC, 2002.
- [11] Xindong Wu, Chengqi Zhang, and Shichao Zhang. Mining both positive and negative association rules. In *Proceedings of 19th International Conference on Machine Learning, ICML2002*, 2002.

Finding Young Stellar Populations in Elliptical Galaxies from Independent Components of Optical Spectra

Ata Kabán

Louisa A. Nolan[†]

Somak Raychaudhury[†]

Abstract

Elliptical galaxies are believed to consist of a single population of old stars formed together at an early epoch in the Universe, yet recent analyses of galaxy spectra seem to indicate the presence of significant younger populations of stars in them. The detailed physical modelling of such populations is computationally expensive, inhibiting the detailed analysis of the several million galaxy spectra becoming available over the next few years. Here we present a data mining application aimed at decomposing the spectra of galaxies into several coeval stellar populations, without the use of detailed physical models. This is achieved by performing a linear independent basis transformation that essentially decouples the initial problem of joint processing of a set of correlated spectral measurements into that of the independent processing of a small set of prototypical spectra. Two methods are investigated: (1) A fast projection approach is derived by exploiting the correlation structure of neighboring wavelength bins within the spectral data. (2) A factorisation method that takes advantage of the positivity of the spectra is also investigated. The preliminary results show that typical features observed in stellar population spectra of different evolutionary histories can be convincingly disentangled by these methods, despite the absence of input physics. The success of this basis transformation analysis in recovering physically interpretable representations indicates that this technique is a potentially powerful tool for astronomical data mining.

1 Introduction

The optical spectrum of a galaxy is a linear superposition of the spectra of its billions of constituent stars. Yet, since large populations of stars form in galaxies at definite periods of its lifetime, and the atomic and nuclear physics of the evolution of stellar populations,

though complex, are well understood, the detailed modelling of composite spectra of stellar populations can be used to yield a wealth of information about the history of a galaxy from its spectrum.

The spectrum of a star can be modelled as a function of three parameters—its mass, its age and its composition (since it is made mostly of hydrogen and helium, the last parameter is characterised by the relative abundance of other elements, and is known as the “chemical abundance”). Elliptical galaxies, which account for about 20% for all galaxies in the Universe, are believed to consist predominantly of a single coeval stellar population (e.g. [3, 12, 22]), all formed at an early epoch in the Universe. This implies that an elliptical galaxy can be modelled as a system of stars, all of the same age and chemical abundance, evolving together, if validated assumptions can be made about the distribution of stellar masses. However, as a result of detailed spectral studies conducted in the last decade (e.g. [10]), it now transpires that elliptical galaxies are more complex objects, at least some of which have undergone more recent bursts of substantial star formation, and consequently are likely to contain more than one stellar population component.

The determination of the star formation history of a galaxy has important implications for the still-controversial issue of the formation and evolution of galaxies. Until recently, the analysis of a large statistical sample of stellar populations of galaxies would not have been possible since only small ensembles of galaxy spectra were available. However, the development of data mining tools for automating parts of the analysis is becoming more and more essential in the light of the rapid increase in the availability of data that is approaching.

Recent and ongoing galaxy spectral surveys (2dFGRS, www.mso.anu.edu.au/2dFGRS/ (completed in 2003) and SDSS, www.sdss.org/) will produce more than two million galaxy spectra in the next few years, which is to be integrated into a more ambitious database of publicly-available astronomical data, incorporating Grid technology (the *Virtual Observatory*, www.ivoa.net). Since the detailed physical modelling of stel-

[—]School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK. Email: A.Kaban@cs.bham.ac.uk.

[†]School of Physics and Astronomy, The University of Birmingham, Birmingham B15 2TT, UK. Email: {lan, somak}@star.sr.bham.ac.uk

lar populations is numerically expensive, even a simple question like “what fraction of elliptical galaxies contain a significantly younger stellar population?” will take years to address by conventional modelling techniques using stellar population synthesis. The timely extraction of useful knowledge, such as the characteristics of the star formation history (ages, chemical abundances and masses of the component stellar populations) of galaxies, from these data will largely depend on developing appropriate data analysis tools that are able to complement more specialised astrophysical analyses.

The astrophysical questions motivating this study are:

1. Can we disentangle major stellar population components of elliptical galaxies without the use of detailed physical models?
2. How do the results from a data-driven analysis of observed galaxy spectra correlate with the parameters of star formation history determined via a completely independent modelling technique used in astrophysics?

These questions have not been addressed before in a data driven manner — that is, based on the data characteristics only, without any specialised physical input.

1.1 Roadmap In this paper, we discuss and investigate statistical methods that attempt to solve the described inverse modelling problem by relating multivariate observations to lower-dimensional vectors of statistically independent unobserved variables through the use of a linear model. The required statistical assumptions will be derived from general characteristics of the data, in order to employ these methods in an appropriate manner.

The preliminary results presented in the next sections are based on the data described in Section 2. A projection approach that exploits the correlation structure of the spectra is presented in Section 3. In this approach, the required assumption for solving the inverse modelling is derived from exploiting the correlation structure between neighboring wavelength bins, which comes naturally with spectral data. The independent spectral components obtained turn out to be also physically interpretable and exhibit typical features of spectra of the young and mature stellar populations. We then compare the results with a positivity-based single stage approach, presented in Section 4, that has been often employed for analysing spectral data in different domains [9, 18]. We provide a simple probabilistic reformulation of this method that highlights its implicit assumptions, links it to the methods developed in [13] and

also allows us to potentially incorporate measurement errors (if known from domain knowledge) into the algorithm. In Section 5, the results are presented, discussed and comparatively assessed, first in a data-driven manner and then, more importantly, from the astrophysical perspective. Finally, our conclusions are summarised in the last section.

2 The data and model setting

The data we use here represent the *observed* optical spectra of 21 nearby elliptical galaxies, compiled by blending together 5855 measurements over the range 2000-8000 Å from various observatories on ground and in space. These represent the following galaxies: NGC 0205, NGC 0224, NGC 1052, NGC 1400, NGC 1407, IC 1459, NGC 1553, NGC 3115, NGC 3379, NGC 3557, NGC 3605, NGC 3904, NGC 3923, NGC 4374, NGC 4472, NGC 4621, NGC 4697, NGC 5018, NGC 5102, NGC 7144 and NGC 7252. The spectra have been corrected for redshift (i.e. converted from their observed wavelengths to their emitted wavelengths), and the fluxes are normalised to unity in the region 5020-5500 Å.

Since these spectra are compiled from sources with varying spectral coverage, the resulting data matrix has 1453 missing values, which are first imputed using a KNN imputation [21] from synthetic data. We preferred this non-parametric procedure here, as the missing data mechanism may not be random — an assumption made by most of other imputation schemes. The validity of the ‘missing at random’ assumption in the case of the analysed data set will need further study, simply because in some wavelength regions it is consistently hard or impossible to take a measurement.

In addition, for each measurement, an error value is also provided from known instrumental characteristics and uncertainty in calibration.

2.1 How many stellar populations? According to existing domain knowledge, it is likely that there are (at least) two components of interest [10]. However, the first eigenvector explains more than 95% of the data. Therefore, prior to deciding that a 2D representation space is justified (i.e. that at the given noise levels there is enough useful information in the data and we are not attempting to model the noise in a second component), we perform some simple, data-driven rank tests. We use the error matrix to derive thresholds for these tests. As shown on Figure 1, both a 2-norm test and an F-norm test [19] suggest that at the given error levels, the ‘clean’ matrix of spectra has rank 2. Although it is known that these perturbation bounds often tend to underestimate the rank [19], there are no records of over-estimation, so

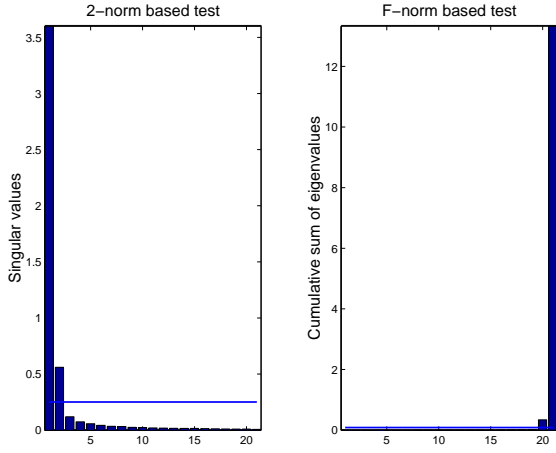


Figure 1: Rank-tests for the matrix of stellar population spectra. The threshold values represent the 2-norm and the F-norm respectively of the matrix of known measurement errors.

we proceed to searching for a suitable two-dimensional latent representation space.

2.2 The model Each stellar population spectrum is essentially a vector over a binned wavelength range, that represents the flux (in arbitrary units) per unit wavelength in the bin centered on the wavelength. The data matrix, having T spectra in rows will be denoted by $\mathbf{Y} \in \mathcal{R}^{T \times N}$. Single elements of this matrix will be referred to as y_{tn} , the t -th row is denoted by \mathbf{y}_t or more explicitly $\mathbf{y}_{t,1:N}$ and likewise, the n -th column is denoted by \mathbf{y}_n or $\mathbf{y}_{1:T,n}$. The whole matrix \mathbf{Y} will also be referred to as $\mathbf{y}_{1:N}$. Similar notational convention will also apply to other variables in the model.

To account for the generation mechanism described in the previous section, namely that the observed optical spectrum of a galaxy is a linear superposition of the stars in the galaxy, a linear factor model will be assumed in this study. That is, we hypothesise that the T observations can be explained as a superposition of $K < T$ latent underlying component spectra $\mathbf{s}_k \in \mathcal{R}^K$ (sometimes termed also as factors or hidden causes [17, 13, 6]) that are not observable directly but only through an unknown linear mapping $\mathbf{A} \in \mathcal{R}^{T \times N}$. Formally, this can be written as the following.

$$(2.1) \quad \mathbf{y}_n = \mathbf{A}\mathbf{s}_n + \boldsymbol{\epsilon}$$

In eq. (2.1), the first term of the r.h.s. is the so called systematic component and $\boldsymbol{\epsilon}$ is the noise term or the stochastic component of this model. The noise term $\boldsymbol{\epsilon}$ is assumed to be zero-mean i.i.d. Gaussian, where the

diagonal structure of the covariance accounts for the notion that all dependencies that exist in \mathbf{Y} should be explained by the underlying hidden components.

The K components will be assumed statistically independent, this being a standard assumption of independent factor models [6, 13]. In the present application, this assumption is also cosmologically plausible, as there is little (no) interaction between stellar populations at different ages in a galaxy. The linearity of the mixture is physically justified as the fluxes of the hypothesised different subpopulations mix in an additive way.

3 Independent projections of stellar population spectra

A projection based approach is presented in this section. This will be accomplished in stages. A linear dimensionality reduction will first be performed. We then proceed at identifying independent directions in the low-dimensional projection space. This multi-stage projection approach is well-suited as a first attempt. It allows us to formulate sub-tasks in statistical terms, and given the 2D nature of the problem, it also allows us to benefit from a visual control over the data representation obtained at various stages.

3.1 Dimensionality reduction using SVD Dimensionality reduction is a useful preprocessing stage for both computational convenience and de-noising. It is well known from linear algebra [4, 19] that the best rank- K approximation of a matrix under any unitarily invariant norm is its rank- K SVD (Singular Value Decomposition) approximation. This is given by $\mathbf{Y} \approx \mathbf{U}\mathbf{D}\mathbf{V}^T$, where \mathbf{U} is the $T \times K$ matrix of left singular vectors, \mathbf{D} is the $K \times K$ diagonal matrix of singular values, \mathbf{V} is the $N \times K$ matrix of right singular vectors, and $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}_K$, where \mathbf{I}_K is the K -dimensional identity matrix. The projection is then simply obtained as $\mathbf{X} = \mathbf{U}^T\mathbf{Y}$.

It should be pointed out that the scope of an SVD-based projection is to identify an optimal (in the sense of any unitarily invariant norm) subspace of the data space. However, generally, individual singular vectors or eigenvectors are not interpretable separately, as they are not independent from each other. The same is true for PCA [17], for much the same reasons.

3.2 Finding non-orthogonal informative directions using contextual ICA We now turn to the key part of our analysis, where the directions of independent projection need to be found. Approaches with this aim are known under the name of Independent Component Analysis (ICA) [7, 6, 14]. A vast number of ICA

algorithms have been developed over the last decade, each having different built-in assumptions. In general terms, we can write the data likelihood of the desired basis transformation as the following

$$(3.2) \quad p(\mathbf{x}_{1:N}|\mathbf{B}) = \int d\mathbf{s}_{1:N} p(\mathbf{x}_{1:N}|\mathbf{s}_{1:N}, \mathbf{B}) p(\mathbf{s}_{1:N}).$$

where \mathbf{B} is the $K \times K$ unknown linear mapping (squared mixing matrix) that transforms the latent components \mathbf{S} into \mathbf{X} . That is, as standard in ICA, instead of inferring \mathbf{S} from \mathbf{Y} , it is easier to infer them from \mathbf{X} .

Assuming that the SVD projection performed as described in the previous section has removed the noise, then the noise term is a delta function

$$(3.3) \quad p(\mathbf{x}_{1:N}|\mathbf{s}_{1:N}, \mathbf{B}) = \delta(\mathbf{x}_{1:N} - \mathbf{B}\mathbf{s}_{1:N})$$

where \mathbf{B} is a squared $K \times K$ parameter matrix (mixing matrix) that contains the desired new bases in its columns and $\mathbf{s}_{1:N}$ are the independent representations in the new basis — both having to be estimated from the data. Thus, (3.2) reduces to the simple form below

$$(3.4) \quad p(\mathbf{x}_{1:N}) = |\det \mathbf{B}|^{-N} \prod_{k=1}^K p(s_{k,1:N})$$

$$(3.5) \quad = |\det \mathbf{B}^{-1}|^N \prod_{k=1}^K p((\mathbf{B}^{-1})_k \mathbf{x}_{1:N}).$$

Standard in squared ICA problems, it is easier to optimise for the inverse of \mathbf{B} . That is, instead of the ‘top-down’, or ‘generative’ transform \mathbf{B} , we estimate the ‘bottom-up’ or ‘projection’ transform \mathbf{B}^{-1} . However, without knowing $p(\mathbf{s}_{1:N})$, this is still an ill-posed problem. Clearly, a mechanical application of any ICA algorithm, out of the hundreds of existing ones, would produce different results, although any of these would be somewhat arbitrary. What we need is a well motivated prior distribution $p(\mathbf{s}_{1:N})$. However, as in most data mining applications of ICA, there is no such information explicitly available.

3.2.1 Exploiting correlations within the spectral data Let us observe, however, that in spectral data, there is a natural correlation structure between flux values in neighbouring wavelength bins. This is what we exploit here, by capturing it in a form of a

contextual (predictive) model.

$$\begin{aligned} p(s_{k,1:N}) &= \prod_{n=1}^N p(s_{kn}|s_{k,1:n-1}) \\ &= \prod_{n=1}^N p(s_{kn} - E[s_{kn}|s_{k,1:n-1}]) \\ &= \prod_{n=1}^N p((\mathbf{B}^{-1})_k(\mathbf{x}_n - E[\mathbf{x}_n|\mathbf{x}_{1:n-1}])) \end{aligned}$$

$\forall k = 1 : K$. The advantage of doing so is that now we only need to specify the form of density of the residual projections. Assuming a good enough predictor, then the residual is likely to have a heavy tailed (termed also super-Gaussian [7] or kurtotic) form of density. Indeed, using just the simplest first order predictor, which is an identity function

$$(3.6) \quad E[s_{kn}|s_{k,1:n-1}] \equiv s_{k,n-1}, \forall k = 1 : K$$

the difference process $\mathbf{x}_n - \mathbf{x}_{n-1}$ of the data already becomes highly kurtotic, as shown on Figure 2.

A similar approach has been previously taken and successfully demonstrated in the context of face image separation [5], where neighbouring pixel values of an image do also exhibit significant correlations.

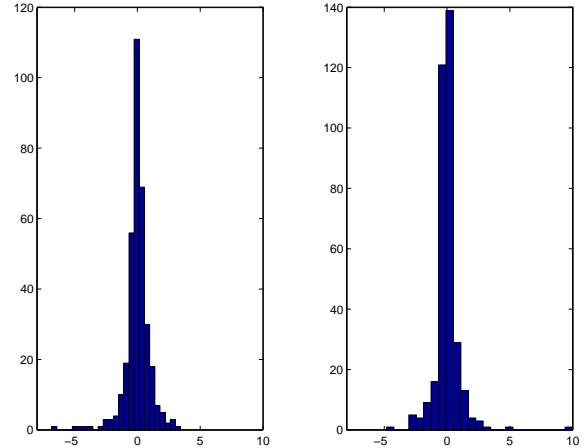


Figure 2: Histograms of the difference process. Kurtosis values are 11.0503 and 33.9364 respectively.

Let us denote $\mathbf{r}_n = \mathbf{x}_n - E[\mathbf{x}_n|\mathbf{x}_{1:n-1}]$ and $u_{kn} = s_{kn} - E[s_{kn}|s_{k,1:n-1}]$. The data likelihood is then the following.

$$(3.7) \quad p(\mathbf{x}_{1:N}) = |\det \mathbf{B}^{-1}|^N \prod_n \prod_k p((\mathbf{B}^{-1})_k \mathbf{r}_n)$$

where now we know that $p(u_{kn})$ is a super-Gaussian density. Maximisation of this likelihood can now be

accomplished by employing any standard ICA algorithm — over $\mathbf{r}_{1:N}$ rather than $\mathbf{x}_{1:N}$. As the predictor may not be very accurate (we just used an identity predictor in our experiments), it is preferable to chose a robust approximation of the generalised exponential density, that grows relatively slowly in $|u_{kn}|$. Following the arguments in [7], in our experiments we have used the following:

$$(3.8) \quad \log p(u_k) \propto \exp(-|u_k|^2/2),$$

and the optimisation has been performed using the Newton method implemented in the FastICA routines [6], employing the faster stationary approach. This has a cubic convergence [7]. Indeed, highly kurtotic independent projections have been found (kurtosis: 33.8796 and 12.7484 respectively) on the data investigated, in about ten iterations only.

A geometric illustration of the procedure just described is shown on Fig. 3. The SVD-compressed data are shown as dots and indeed, informative directions would be difficult to determine directly from the data. The scatter-plot of the difference process is shown as crosses. A star-like structure is apparent, with two main, non-orthogonal linear directions of high data density. These are the new bases (columns of \mathbf{B}) that are determined by the ICA procedure. Indeed, the two directions defined by the new bases found by the algorithm are highlighted on the plot as dark lines. The PCA axes of the data are also shown on the same plot for comparison. Interestingly, one of the axes is almost identical to one of the independent directions. The second is, however, just orthogonal to the first, while the ICA axes are not orthogonal to each other but do follow the two main directions of high density in the data.

To obtain the component spectra from the ICA procedure described, we simply compute the projections of the individual flux values of all galaxies at all wavelength bins onto the new bases (which is the composition of the two linear transforms performed during the analysis process described):

$$(3.9) \quad \mathbf{s}_{1:N} = \mathbf{B}\mathbf{x}_{1:N} = \mathbf{A}\mathbf{y}_{1:N}$$

where \mathbf{B} and \mathbf{A} now denote the recovered mixing parameters. Specifically, after \mathbf{B} is found, \mathbf{A} is computed as the matrix product $\mathbf{U}\mathbf{B}$.

The component-wise reconstruction of the 21 individual stellar population spectra from their independent components are shown on Figure 4. The physical interpretability of these components will be assessed in Section 5, however, as a data-driven observation, it is interesting to note that the recovered spectral components turned out to be positive valued — although positivity

has not been artificially imposed at this stage during the analysis process. We note that indeed negative values of the flux would be difficult to interpret, therefore in the next section we discuss a different approach, where the required latent density $p(\mathbf{s}_{1:N})$ is derived from a positivity constraint.

4 A Positivity-based approach

The use of positive factorisation of positive matrices to replace PCA for analysing positive data, such as spectral data dates back to work reported in [9]. Positive (more exactly non-negative) factor models have been further developed in [18]. However, in the absence of either a density-based or a geometric interpretation, the implicit assumptions are not clear and therefore the interpretation of the results may not be straightforward. Somewhat related, in [13], a fully Bayesian formulation of a positive factorisation model is given and variants with sparse positive priors are applied to synthetic stellar population spectra in a different context of investigations than ours. Retaining the probabilistic framework, that allows us to make all assumptions explicit, we present a simpler version of their algorithm, based on maximum a posteriori (MAP) / maximum likelihood (ML) estimation, which will highlight the link with positive factorisation algorithms [18]. A MAP or ML estimation is sufficient for our purposes as we are concerned with a data explanation task for a fixed data set only. Once the most appropriate model is found, the full Bayesian machinery remains available to derive fully generative models that are able to better generalise on new data.

Retaining the positivity of the representation, and the fact that the overall transform in the approach adopted in the last section has been linear, the following linear model can be formulated.

$$(4.10) \quad p(\mathbf{y}_n|\mathbf{A}, \mathbf{E}_n) = \int d\mathbf{s} p(\mathbf{y}_n|\mathbf{A}\mathbf{s}, \mathbf{E}_n)p(\mathbf{s})$$

A Gaussian measurement noise will be assumed. Furthermore, according to prior knowledge about the levels of imprecision of the physical instruments, which vary independently for each stellar population and each wavelength bin, we will have individual diagonal variances \mathbf{E}_n^2 at each wavelength bin n , $p(\mathbf{y}_n|\mathbf{s}, \mathbf{A}, \mathbf{E}_n) \sim \mathcal{N}(\mathbf{y}_n|\mathbf{A}\mathbf{s}_n, \mathbf{E}_n^2)$. In rest, we use the same notation as before, \mathbf{y}_n refers to the relative flux values observed at the n -th wavelength bin, \mathbf{A} is the unknown mixing matrix parameter and $p(\mathbf{s})$ is the distribution of the latent components.

Now the latent prior needs to be specified. Apart from its positive support we don't have much information in this respect. Therefore we formulate a vague

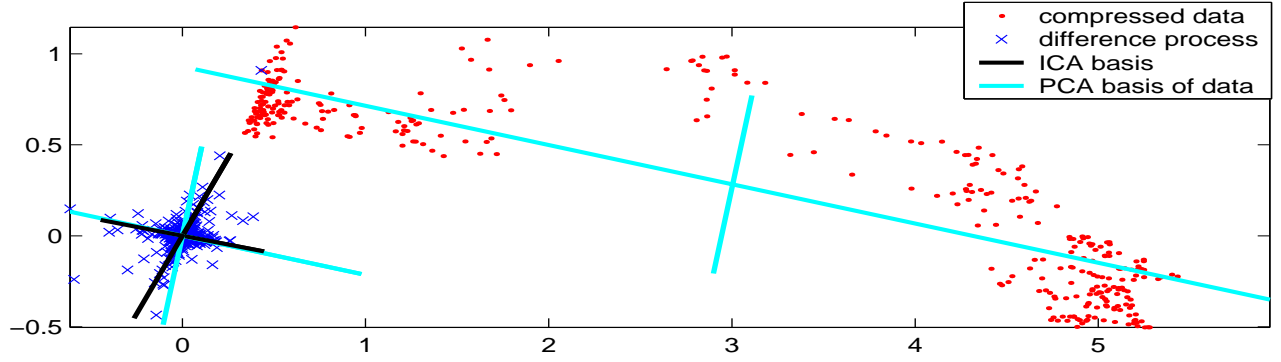


Figure 3: Geometric illustration of the described ICA procedure. The number of points shown as dots equals the number of different wavelength bins in the set of measurements, each point being the 2D compressed representation of the fluxes at one of these bins. The differences between these 2D vectors at consecutive wavelength bins are marked with 'x'. The PCA basis of the data is superimposed (in light color) and also translated to the origin for comparison with the ICA basis found on the difference process. The ICA basis is shown in dark color.

exponential prior, that is

$$(4.11) \quad p(\mathbf{s}) \propto \prod_k \exp(-\gamma_{kn} |s_k|)$$

where $\gamma_{kn} = \gamma, \forall k, n$ is a small positive constant. If $\gamma \rightarrow 0$, then the prior becomes non-informative but also improper and in this case the MAP estimation procedure given below becomes ML.

To obtain a MAP estimate, the posterior needs to be maximised $p(\mathbf{s}_n | \mathbf{y}_n, \mathbf{A}, \mathbf{E}_n) \propto p(\mathbf{y}_n | \mathbf{s}_n, \mathbf{A}, \mathbf{E}_n) p(\mathbf{s}_n)$ which is proportional to the complete data likelihood.

Positivity of the elements of both \mathbf{A} and \mathbf{S} are then imposed by adding Lagrangian terms [20] to the complete data log likelihood.

$$\begin{aligned} \mathcal{L} = & \sum_n \{ \log p(\mathbf{y}_n | \mathbf{s}_n, \mathbf{A}, \mathbf{E}_n) + \log p(\mathbf{s}_n) \} \\ & + \text{Tr} \mathbf{L}_1^T \mathbf{A} + \text{Tr} \mathbf{L}_2^T \mathbf{S} \end{aligned}$$

where \mathbf{L}_1 and \mathbf{L}_2 are a set of non-negative Lagrange multipliers and Tr denotes the trace of a matrix.

From the stationary equations w.r.t. \mathbf{A} and \mathbf{S} the Lagrange multipliers are obtained.

$$\begin{aligned} \mathbf{L}_1 &= \sum_n \mathbf{E}_n^{-2} \mathbf{A} \mathbf{s}_n \mathbf{s}_n^T - \sum_n \mathbf{E}_n^{-2} \mathbf{y}_n \mathbf{s}_n^T \\ \mathbf{L}_2 &= \sum_n \mathbf{A}^T \mathbf{E}_n^{-2} \mathbf{A} \mathbf{s}_n - \sum_n \mathbf{A}^T \mathbf{E}_n^{-2} \mathbf{y}_n + \end{aligned}$$

Now from the Karush-Kuhn-Tucker conditions [20] $L_{tk} A_{tk} = 0$ and $L_{kn} s_{kn} = 0, \forall t, k, n$, we have two fixed point equations which provide the convergent alternating iterative algorithm of the multiplicative form below.

$$\begin{aligned} \mathbf{A} &= \mathbf{A} \odot (\mathbf{E}^{-2} \odot \mathbf{Y}) \mathbf{S}^T \oslash [\mathbf{E}^{-2} \odot \mathbf{A} \mathbf{S}] \mathbf{S}^T \\ \mathbf{S} &= \mathbf{S} \odot \mathbf{A}^T (\mathbf{E}^{-2} \odot \mathbf{Y}) \oslash \left\{ \mathbf{A}^T [\mathbf{E}^{-2} \odot \mathbf{A} \mathbf{S}] + \right\} \end{aligned}$$

where \odot denotes element-wise multiplication and \oslash denotes element-wise division. If $\gamma = 0$, the iterative algorithm above is identical to the least-squares based non-negative factorisation algorithm proposed in [18] from a non-probabilistic starting point, with the only difference that now we also have the \mathbf{E} terms to account for known measurement errors.

5 Evaluation

5.1 Data driven evaluation Here we assess the effectiveness of the methods presented above according to two indicators: (1) data reconstruction and (2) the mutual information (MI) between the components of the representation created.

Method	Min	Median	Max
SVD, cICA	0	3.36×10^{-4}	0.294
NMF	10^{-10}	3.64×10^{-4}	0.317
NMFe	0	4.08×10^{-4}	0.351

Table 1: Data reconstruction errors under the L2 norm. cICA = contextual ICA, NMF = Non-negative matrix factorisation with $\gamma = 0$, NMFe = NMF with exponential prior having $\gamma = 0.1$

Table 1 shows the data reconstruction results across all $N \times T$ measurements for the various methods. These

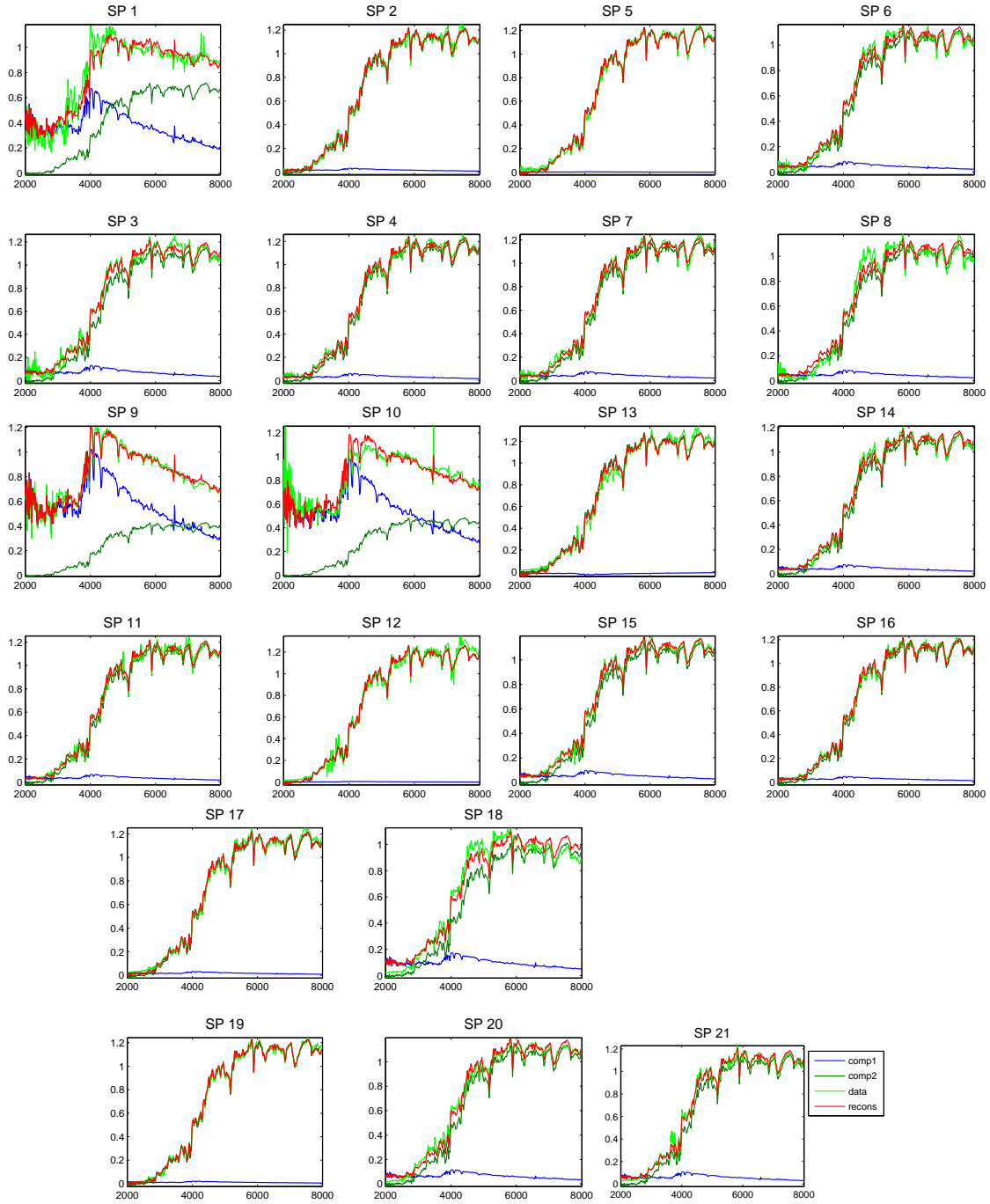


Figure 4: The reconstruction of stellar population spectra using the projection based contextual ICA. Here, our algorithm has decomposed the observed spectrum of each galaxy into two inferred populations (blue and dark green — the two darkest colors, in black-and-white printing), whose sum is given in red, superimposed with the data in light green (intermediate gray and light grey respectively, in black-and-white printing). In many cases (e.g. #1,9,10,18), a significant younger stellar population is found, which is confirmed by our detailed physical modelling. The numbering of the spectra corresponds to the enumeration order of the corresponding galaxies as given in Section 2.

are measured as the squared distances between the data and the reconstruction, which is in accordance with the Gaussian noise assumption. The reconstruction error of cICA are identical to that of the SVD, since the ICA transform does not reduce the dimensionality further. For NMF and NMFe, 15 randomly initialised runs were performed and the one with the highest likelihood value was selected in this evaluation, in order to avoid the possibility of getting trapped in a local optimum. Indeed, note that the positivity-based single stage approach involves a non-convex optimisation whereas the SVD-based preprocessing is a convex problem. The median of the error appears to be the smallest in the case of SVD+cICA, however, a pairwise application of the non-parametric Wilcoxon rank sum test to the whole sample distribution of the individual $T \times N$ reconstruction errors returned that the difference of the medians for cICA and NMF is not statistically significant at the 0.05% level. In the case of all other pairs, the differences between medians were found statistically significant — as expected, the additional term that enforces the latent prior distribution in NMFe causes a slight increase in data reconstruction error. However, this small difference on its own would not be a practically concerning issue here, as we deliberately formulated constraints in our models in order to obtain representations that are statistically independent to a higher degree — in the hope that these may then be interpretable and capable of being independently further processed. More interesting is therefore to evaluate to what extent do these methods achieve statistically independent representations.

The information theoretic quantity that measures the degree of statistical dependence is the Mutual Information (MI) [1]. This is a non-negative value that vanishes if the component densities are perfectly independent (the smaller the MI the better). It can be shown [14] that maximising the ICA log likelihood in the noise-free case is equivalent to minimising MI between the representation components. Here we compute the sample-based MI of the components, as estimated according to the procedure described in [2]. The comparative values obtained by various methods for this data are shown in Table 2. For the contextual ICA method, that works on predictive residuals, two values are given: the first value is computed from the projections of the residuals whereas the second value is computed from the projections of the data, taken as it was iid. (just for obtaining a value to compare with those obtained with the rest of the methods). It is apparent from the table that in both cases the contextual ICA model achieves lower MI (greater independence of the components). The positivity-based method with non-

Method	MI
NMF	0.5923
NMFe	0.6019
cICA	(u) 0.00010394 (s) 0.5583
PCA	1.0931

Table 2: Sample-based mutual information estimates of the two components obtained with the various methods for the set of stellar population spectra investigated. Smaller values signify higher independence achieved after the estimated basis transform.

informative improper prior is the next best performing method, whereas employing sparse priors (greater values) does not lead to components that are more independent, for this data. PCA is used as a baseline, as we know that it doesn’t produce an independent representation. Visually, the positivity based components do not look much different from the ones obtained from cICA (Fig. 4), although they are slightly more noisy. The PCA-based decomposition, however, in general, are theoretically not guaranteed to be interpretable, as already discussed in Section 3.1 (see [17] for details). We now turn to evaluate the interpretability of the obtained results from the astrophysical perspective.

5.2 Astrophysical evaluation Here, we compare the results from the linear independent basis transformation analysis with an entirely independent determination of the star formation history, based on detailed astrophysical models of the evolution of stellar populations.

The 21 observed spectra have been analysed by matching them with synthetic stellar population spectra. For each of the observed spectra, a two-stellar population component model [8] was fitted. The age, chemical abundance and relative mass fraction of each component were allowed to vary freely. The best fit in each case was determined by a minimum χ^2 test [15, 11].

The principle for creating synthetic stellar population spectra is simple, although the input physics is complex. The spectral energy distribution of a star evolves according to its initial mass and chemical abundance. If the initial mass distribution and the chemical abundance of a stellar population is known, and the spectral evolution of each individual star in this initial population may be modelled, the stellar spectra may be summed over the mass distribution at any point in time to give the integrated spectrum of the population at that age. The ingredients for a stellar population model are therefore: stellar evolutionary tracks; a library of stellar

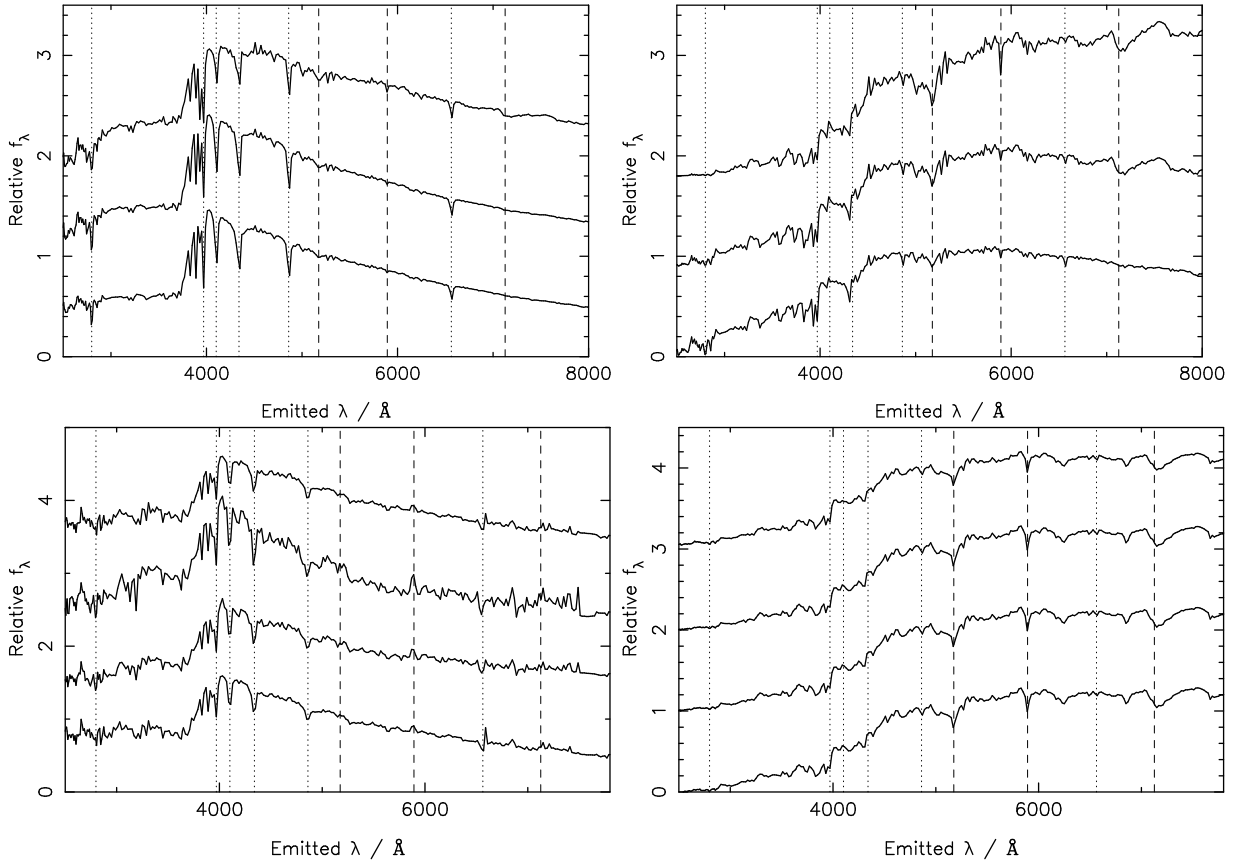


Figure 5: Comparison of the derived components with physical models of the stellar population spectra. *Top two plots:* Synthetic stellar population spectra according to the physical models of [8]. *Right:* Spectra of a population of age 10 Gyr, where chemical abundance, from bottom to top, 0.2, 1.0 and 2.5 times solar; *Left:* age = 0.7 Gyr, same chemical abundances. The dotted lines mark some of the absorption features in the spectrum which are typically strong in young stellar populations, and the dashed lines mark some of the absorption features which are typically strong in old stellar populations. From left to right, the absorption line species are: MgII (2799 Å), H ϵ (3970 Å), H δ (4102 Å), H γ (4340 Å), H β (4861 Å), Mgb (5175 Å), NaD (5893 Å), H α (6563 Å), TiO (7126 Å). *Bottom two plots:* the 2 components found from the various different linear independent basis transformation analyses, from bottom to top: cICA, NMF, NMFe, PCA. (The spectra are shifted along the vertical axis for the sake of clarity.) The recovered spectra are convincingly disentangled into one component with young stellar population features (MgII, H ϵ , H δ , H γ , H β , H α : dotted lines) and shape, and a second with the features (Mgb, NaD, TiO: dashed lines) and shape of an old, high chemical abundance stellar population.

spectra; a method of calibrating the theoretical luminosity and effective temperature, determined from the evolutionary sequence, so that the appropriate atmosphere may be assigned to each star at each time-step in its evolution. In contrast, the linear independent basis transformation method employs no knowledge of the underlying physics in the observed spectra.

Model spectra [8] for a young (0.7 Gyr) and an old (10 Gyr) stellar population, with three different chemical abundances (0.2, 1.0 and 2.5 times solar abundance) are shown in Fig. 5, together with the spectra recov-

ered from the linear independent basis transformation analyses. The similarity is most apparent. The data-driven linear analyses, whilst not reproducing any of the physical model spectra precisely (which would not be expected anyway), extract many of the important identifying characteristics of these two categories of model spectra, which are indeed quite different from each other both in their overall shape and details. Some of the most important features, the so-called absorption-lines, are marked with dashed (Mgb, NaD and TiO, typically strong features in old, high chemical abundance stel-

lar population spectra) and dotted (MgII, H ϵ , H δ , H γ , H β and H α , typically strong in the spectra of young ($\lesssim 1$ Gyr) stellar populations) vertical lines on the figure (Fig. 5), which demonstrate the physical interpretability of the representations created by the basis transformation analyses.

We correlate the star formation history parameters derived from fitting the two-component model spectra to the observed spectra (i.e. a physical analysis approach) with the weight of the contributions from the linear basis transformation analyses, by defining these weights as $c_k = a_{tk} / \sum_{k'} a_{tk'}$ for any given spectrum t . Here, a_{tk} is the (t, k) -th element of the matrix \mathbf{A} of the new basis and $k = 1 : 2$. Fig. 6 shows the results of the correlations, and Fig. 7 graphically shows some of these correlations.

From Figs. 6 and 7 we can conclude that, for the ICA and NMF analyses, c_1 (and hence also c_2 , as $c_1 + c_2 = 1$) correlates with the proportion of young ($\lesssim 1$ Gyr) stellar population component present in the observed spectrum, regardless of their chemical abundance.

6 Conclusions

We have presented a scientific data mining application that searches for linear independent basis transformations of galaxy spectra to find the spectra of individual stellar populations characterised by age and chemical abundance. We have shown that characteristic stellar population components of elliptical galaxies can be disentangled from the observed spectra of these galaxies, without the use of detailed physical models. The components returned by the linear basis transformation analyses are clearly physically interpretable, with one component displaying the shape and many of the absorption-line features typical of a young stellar population, and the second component having the over-all shape and typical absorption features of an old, high chemical abundance stellar population. The weights of the contributions from the linear basis transformation analyses correlate well with both the ages of the younger stellar populations and the mass fractions of the smaller stellar populations determined from the (completely independent) detailed physical modelling of the observed galaxy spectra.

The computational demand of the projection approach presented is essentially that of the SVD computation, so it is expected that the method is easily applicable to large sets of measurements as they become available. The positivity based approach, per iteration, has a comparable scaling, however, in all our experiments the number of iterations to convergence was of an order of magnitude larger for the positivity based single

stage approach. Further study is necessary to investigate models with other types of positively supported priors as well as refining the best performing models and algorithms to be able to deal with previously unseen data.

The use of the data analysis presented in this paper, integrated with the more complex process of astrophysical analysis will be detailed elsewhere [16]. We intend to investigate the effectiveness of these data-driven methods on larger sets of UV-optical spectra as they become available, where more comprehensive statistical evaluation will be possible. From the analysis of large archives of galaxy spectra using this technique, we hope to address some of the fundamental questions in astrophysics, that of when and how galaxies form and evolve.

Acknowledgement

This research was partly supported by a Paul & Yuanbi Ramsay research award from the School of Computer Science of the University of Birmingham.

References

- [1] T. Cover and J. Thomas, Elements of Information Theory. John Wiley and Sons, Inc., 1991.
- [2] G.A. Darbellay and I. Vajda, Estimation of the information by an adaptive partitioning of the observation space. IEEE Trans. Information Theory, Vol. 45, no. 4, pp. 1315-1321, May 1999.
- [3] O.J. Eggen, D. Lynden-Bell, A.R. Sandage, Evidence from the motions of old stars that the Galaxy collapsed, Astrophys. J., 136, 748, 1962
- [4] G.H. Golub and C.F. Van Loan, Matrix Computations. Johns Hopkins University Press, 1989.
- [5] A. Hyvarinen, Independent Component Analysis for Time-dependent Stochastic Processes, Proc. Int. Conf. on Artificial Neural Networks (ICANN'98) 1998, pp. 541-546.
- [6] A. Hyvarinen, Fast and Robust Fixed Point Algorithms for Independent Component Analysis. IEEE Transactions on Neural Networks 10(3): 626-634, 1999.
- [7] A. Hyvarinen, One-unit Contrast Functions for Independent Component Analysis: A Statistical Analysis. Proc IEEE Neural Networks for Signal Processing VII, 1997, pp. 388-397.
- [8] R. Jimenez, P. Padoan, F. Matteucci, A. Heavens, Galaxy formation and evolution: low-surface-brightness galaxies Mon. Not. R. Astron. Soc. 299, 123, 1998
- [9] M. Juvela, K. Lehtinen, P. Paatero, The Use of Positive Matrix Factorisation in the analysis of Molecular Line Spectra. Mon. Not. R. Astron. Soc., 280 pp.616-626, 1996.

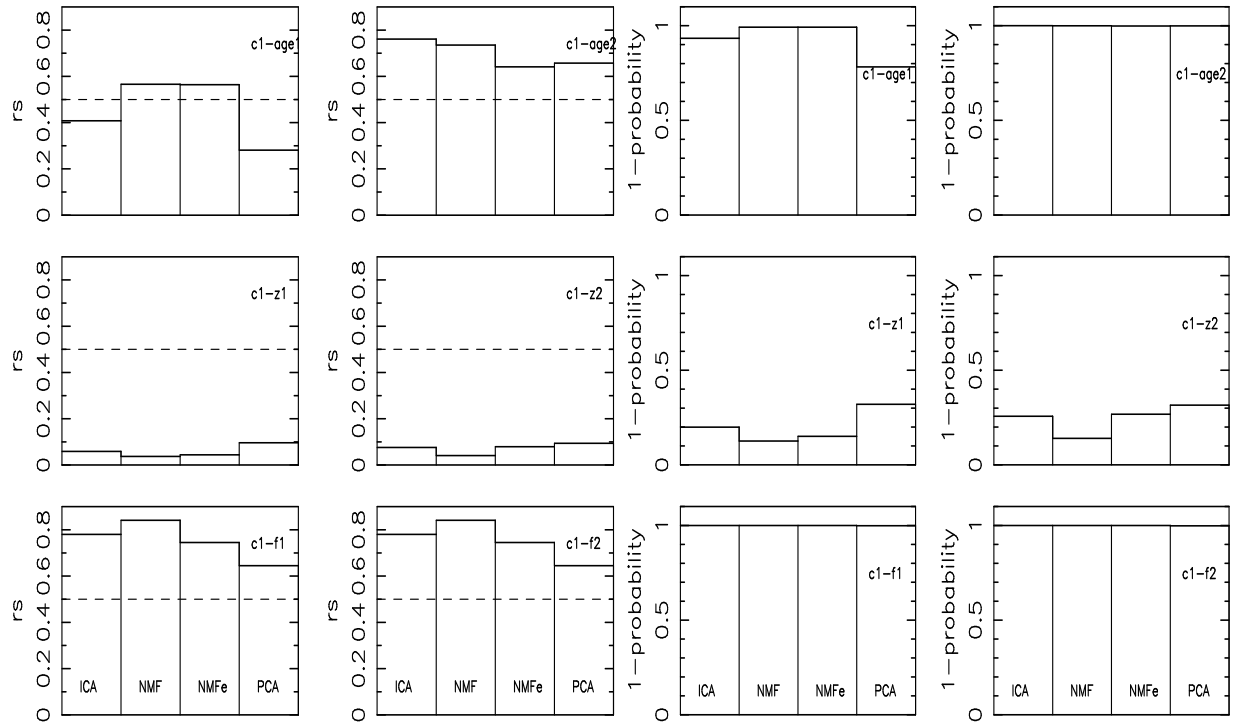


Figure 6: *Left:* The modulus of Spearman’s rank order correlation coefficient rs for the weight of the first component of the various linear basis transformation analyses, $c1$, correlated with the parameters recovered from the physical analysis i.e. a two-component spectral model fitting to the observed galaxy spectra. The parameters investigated in this correlation analysis are the age, chemical abundance z and relative mass fraction f for each component. Values greater than 0.5 indicate a strong correlation. *Right:* the significance of rs . High values of (1-probability) indicate a high level of significance.

- [10] G. Kauffmann, S. White, B. Guideroni, The Formation and Evolution of Galaxies Within Merging Dark Matter Halos, *Mon. Not. R. Astron. Soc.* 264, 201, 1993
- [11] R. P. Kraft, L. A. Nolan, T. J. Ponman, C. Jones, S. Raychaudhury, A Chandra observation of the nearby lenticular galaxy NGC 5102: where are the x-ray binaries? *Astrophys. J.*, submitted, 2004
- [12] R. B. Larson, Models for the formation of elliptical galaxies, *Mon. Not. R. Astron. Soc.* 173, 671, 1975
- [13] J. Miskin, Ensemble Learning for Independent Component Analysis. PhD Thesis. University of Cambridge, 2000.
- [14] T.-W. Lee, M. Girolami, A.J. Bell and T.J. Sejnowski, A Unifying Framework for Independent Component Analysis. *Computers and Math. with Applications*, vol. 39, no. 11, pp.1–21, 2000.
- [15] L.A. Nolan et al., The star-formation history of galaxies in the GEMS groups, in preparation.
- [16] L.A. Nolan, M.O. Harva, A. Kabán, S. Raychaudhury, Finding Young Stellar Population in Elliptical Galaxies from Independent Components of their UV-optical spectra, in preparation for submission to *Mon. Not. R. Astron. Soc.*
- [17] M. Tipping and C. Bishop, Probabilistic Principal Component Analysis, *Journal of the Royal Statistical Society, Series B*, 61, Part 3, pp. 611–622, 1999.
- [18] D. Lee and S. Seung, Algorithms for Non-Negative Matrix Factorisation, *Advances in Neural Information Processing Systems* 13, 556–562, MIT Press, 2001.
- [19] G. W. Stewart, Perturbation Theory for the Singular Value Decomposition. Appeared in *SVD and Signal Processing, II*, R. J. Vacarro ed., Elsevier, 1991.
- [20] H.A. Taha, *Operations Research – An Introduction*. 1997, Prentice-Hall.
- [21] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R.B. Altman, Missing Value Estimation Methods for DNA Microarrays. *Bioinformatics* Vol. 17 no 6 2001.
- [22] S. D. M. White & M. J. Rees, Core condensation in heavy halos - A two-stage theory for galaxy formation and clustering, *Mon. Not. R. Astron. Soc.* 183, 341, 1978

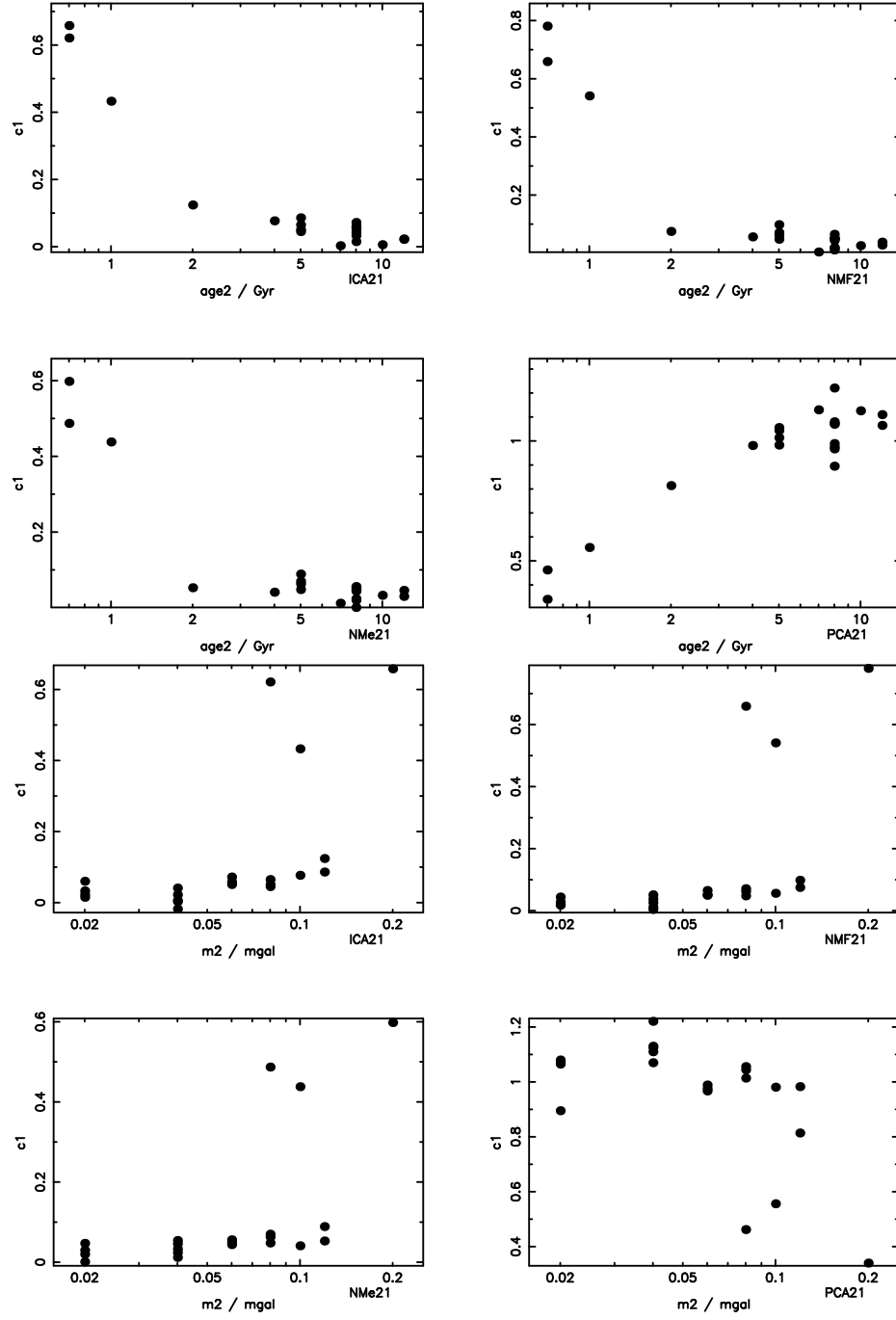


Figure 7: Scatter-plots showing the correlation of (*top*) the age of the younger stellar population and (*bottom*) the mass fraction of the smaller stellar population determined from the model spectra fitting with the weight of the first component of the various linear basis transformation analyses (c1). A high value (low for PCA) of c1 clearly corresponds to a substantial young ($\lesssim 1$ Gyr) stellar population.

Hybrid Attribute Reduction for Classification Based on A Fuzzy Rough Set Technique

Qinghua Hu*

Daren Yu[†]

Zongxia Xie[‡]

Abstract

Data usually exists with hybrid formats in real-world applications, and a unified data reduction for hybrid data is desirable. In this paper a unified information measure is proposed to computing discernibility power of a crisp equivalence relation and a fuzzy one, which is the key concept in classical rough set model and fuzzy rough set model. Based on the information measure, a general definition of significance of nominal, numeric and fuzzy attributes is presented. We redefine the independence of hybrid attribute subset, reduct, and relative reduct. Then two greedy reduction algorithms for unsupervised and supervised data dimensionality reduction based on the proposed information measure are constructed. Experiments show the reducts found by the proposed algorithms get a better performance compared with traditional rough set approaches.

1 Introduction.

In recent years, data has become increasingly larger not only in rows (i.e. number of instances) but also in columns (i.e. number of features) in many applications, such as gene selection from microarray data and text automatic categorization, where the number of features in the raw data ranges from hundreds to tens of thousands[1]. Such high dimensionality brings great difficulty to pattern recognition, machine learning and data mining [2, 3]. Data reduction is a well-known data mining problem which is usually considered as an enhancement preprocessing technique for subsequent machining [4]. It will bring many potential benefits: reducing the measurement, storage and transmission, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance in terms of speed, accuracy and simplicity, facilitating data visualization and data understanding [5, 6]. A lot of data reduction techniques are proposed to deal with these challenging tasks. Due to the complexity of data and classification in real-world applications, it seems not an easy task to build a general data reduction technique, so researches on data reduction have been conducted for

last several decades and are still extracting much attention from pattern recognition and data mining society. Data reduction can begin with two aspects: reducing the number of samples or reducing the number of features. The first one will be implemented by resample techniques and the second is done with dimensionality reduction techniques [7, 8]. This paper will be focused on the second problem.

An extensive amount of researches have been conducted over last decades to get reliable approaches for dimensionality reduction, which roughly falls into two types of paradigms: feature extraction and feature subset selection [9]. Feature extraction refers to constructing new features by a linear or nonlinear transformation from the original input space to a feature space, while feature subset selection is to find some informative features from the original input space and delete the others. Principal component analysis (PCA) [10, 11, 12], Independent component analysis (ICA)[13, 14], Linear discriminant analysis (LDA) are to find a linear transformation and Projection pursuit regression constructs a nonlinear mapping from input space to feature space. A main drawback of these methods is that the constructed features do not have true meaning, and complex computation may be required [4].

In last decade, much attention has been paid to feature subset selection. Two extensive reviews were published [7, 15] in artificial intelligence and a special issue of machine learning research was present in 2003 [1]. Generally speaking, there are four basic components in all feature subset selections: an evaluation function of feature subset, a search strategy to find the best feature subset as defined by the corresponding evaluation function, a stopping criterion to decide when to stop and a validation procedure to check whether the subset is valid [16]. According to evaluation methods the feature subset selection can be classified into two kinds: filtering and wrapper. Distance measures [17, 18], information measures [19, 20, 21], correlation coefficient [22] and consistency measures [6] are used for filtering methods. Wrapper refers to using a classifier as evaluation function in selection. KNN, neural network, SVM all can be employed. Isabelle Guyon [1] pointed that se-

*Harbin Institute of Technology, China.

[†]Harbin Institute of Technology, China.

[‡]Harbin Institute of Technology, China.

lecting the most relevant features is usually suboptimal for building a good predictor in filtering because the performance of the trained predictor depends on not only feature subset, but also the learner used. In other words, a best feature subset in terms of an evaluation function doesn't mean a best prediction performance. An optimal feature subset selection should be conducted by the corresponding classifier employed, which leads to wrapper methods. However Wrapper methods will take high time-complexity which is may be infeasible in real-world applications. Filtering as an efficient feature selection is widely used in practice. In filtering methods, information measures and consistency measures work effectively when data are nominal. Compared with these measures, distance measures and correlation coefficient are proposed for numeric data in nature because there is no distance measure in the nominal domain. Data usually comes with a hybrid form in applications. For example, nominal attributes: sex, color, numeric attributes: age, temperature are coexist in hospital data. The above selection methods are suitable for a single format of features in nature. A feature subset selection for hybrid data is desirable in applications.

Rough set theory has proved to be a powerful tool for uncertainty and has been applied to data reduction, rule extraction, data mining and granularity computation. Reduct is a minimal attribute subset of the original data which is independent and has the same discernibility power as all of the attributes in rough set framework. Obviously reduction is a feature subset selection process, where the selected feature subset not only retains the representational power, but also has minimal redundancy. So rough set methodology based dimensionality reduction will get a good feature subset. Some rough set based reduction and feature selection algorithms have been proposed. Consistency of data [24, 25], dependency of attributes [26], mutual information [27], discernibility matrix [28] and genetic algorithm are employed to find reducts of an information system [29]. And these techniques are applied to text classification [30], face recognition [3], texture analysis [31] and process monitoring [32]. An extensive review about rough set based feature selection was given in [33].

As we know, Pawlak's rough set model [26] works in case that only nominal attributes exist in information systems. However, data usually comes with a hybrid form. Nominal attributes, fuzzy attributes and numeric features coexist in real-world databases. Some generalizations of the model were proposed to deal with the problem. Rough set theory and fuzzy set theory were putted together and rough fuzzy sets and fuzzy rough sets were defined in [34]. The properties and axiomati-

zation of fuzzy rough set theory [35, 36] were analyzed in detail. And the fuzzy rough set methods were applied to mining stock price [37], vocabulary for information retrieval [38] and fuzzy decision rules [39].

Just as reduction plays an important role in classical rough set theory, a reduction algorithm for fuzzy information systems is desirable. In traditional processing, discretization is performed on numeric data as a preprocessing for machine learning [40]. Qiang Shen etc pointed that this processing may lead to some information loss in the original data. A fuzzy-rough attribute reduction, called fuzzy-rough QUICKREDUCT algorithm, was given in [42] based on fuzzy dependency function. Fuzzy dependency function has the power to measure the discernibility power of nominal attributes and fuzzy attributes.

In this paper, we will introduce an information measure for fuzzy equivalence relations. Then we will redefine the dependency of a hybrid attribute set and give unsupervised and supervised reduction algorithms for hybrid data based on the measure. The rest of the paper is organized as follows: some preliminary knowledge about rough set and fuzzy-rough set theory is present in §2. A novel information measure and its properties will be presented in §3. §4 gives another definition of dependency of attribute set and reduction algorithms for hybrid data. An extensive experimental analysis is described in §5. §6 concludes the paper.

2 Some primary definitions on fuzzy rough set model.

Pawlak's rough set model can only deal with data containing nominal values. As we know the real-world applications usually contain real-valued or fuzzy attributes. A fuzzy equivalence relation would be generated by a real-valued attribute or a fuzzy attribute, instead of crisp equivalence relation. The fuzzy-rough set model is fitted for the case where both the relation and the object subset to be approximated are fuzzy.

DEFINITION 2.1. Given a non-empty finite set X , R is a relation defined on X , denoted by a relation matrix $M(R)$:

$$M(R) = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & \dots & r_{nn} \end{pmatrix}$$

where $r_{ij} \in [0, 1]$ is the relation value of x_i and x_j .

R is a fuzzy equivalence relation, if $\forall x, y, z \in X$, R satisfies:

- 1) Reflexivity: $R(x, x) = 1, \forall x \in U$;
- 2) Symmetry: $R(x, y) = R(y, x), \forall x, y \in U$;
- 3) Transitivity: $R(x, z) \geq \min_y \{R(x, y), R(y, z)\}$.

Given arbitrary set X , R is a fuzzy equivalence relation defined on X . $\forall x, y \in X$, some operations on relation matrices are defined as

- 1) $R_1 = R_2 \Leftrightarrow R_1(x, y) = R_2(x, y), \forall x, y \in X$;
- 2) $R = R_1 \cup R_2 \Leftrightarrow R(x, y) = \max\{R_1(x, y), R_2(x, y)\}$;
- 3) $R = R_1 \cap R_2 \Leftrightarrow R(x, y) = \min\{R_1(x, y), R_2(x, y)\}$;
- 4) $R_1 \subseteq R_2 \Leftrightarrow R_1(x, y) \leq R_2(x, y)$.

A crisp equivalence relation will generate a crisp partition and a fuzzy equivalence relation generates a fuzzy partition.

DEFINITION 2.2. The fuzzy equivalence classes generated by a fuzzy equivalence relation R is defined as

$$U/R = \{[x_i]_R\}_{i=1}^n, \\ \text{where } [x_i]_R = \left\{ \frac{r_{i1}}{x_1} + \frac{r_{i2}}{x_2} + \cdots + \frac{r_{in}}{x_n} \right\}.$$

THEOREM 2.1. Given arbitrary set X , R is a fuzzy equivalence relation defined on X . The fuzzy quotient set of X by relation R is denoted by X/R . $\forall x, y \in X$, we have

- 1) $R(x, y) = 0 \Leftrightarrow [x]_R \cap [y]_R = \emptyset$;
- 2) $\bigvee_{x \in X} [x]_R = 1$;
- 3) $R(x, y) = 1 \Leftrightarrow [x]_R = [y]_R$;

DEFINITION 2.3. Given a fuzzy approximation space $\langle U, R \rangle$, X is a fuzzy subset of U . The lower approximation and upper approximation, denoted by $\underline{R}X$ and $\overline{R}X$, are defined as

$$\begin{cases} \mu_{\underline{R}X}(x) = \bigwedge \{ \mu_X(y) \vee (1 - R(x, y)) : y \in U \}, x \in U \\ \mu_{\overline{R}X}(x) = \bigvee \{ \mu_X(y) \wedge (1 - R(x, y)) : y \in U \}, x \in U \end{cases}$$

The membership of an object $x \in U$, belonging to the fuzzy positive region is defined as

$$\mu_{POS_B(d)} = \sup_{X \subseteq U/d} \mu_{\underline{R}X}(x).$$

DEFINITION 2.4. Given a fuzzy information system $\langle U, A \rangle$, B and d are two subset of attribute set A , the dependency degree of d to B is defined as

$$\gamma_B(d) = \sum_{x \in U} \mu_{POS_B(d)}(x).$$

DEFINITION 2.5. Given a fuzzy information system $\langle U, A, V, f \rangle$, $B \subseteq A, a \in B$, if $U/B = U/(B - a)$, we say knowledge a is redundant or superfluous in B . Otherwise, we say knowledge a is indispensable. If any a belonging to B is indispensable, we say B is independent. If attribute subset $B \subseteq A$ is independent and $U/B = U/A$, we say B is a reduct of A .

DEFINITION 2.6. Given a fuzzy information system $\langle U, A, V, f \rangle$, $A = C \cup d$. B is a subset of C . $\forall a \in B$, a is redundant in B relative to d if $\gamma_{B-a}(d) = \gamma_B(d)$, otherwise a is indispensable. B is independent if $a \in B$ is indispensable, otherwise B is dependent. B is a subset of C . B is a reduct of C if B satisfies:

- 1) $\gamma_B(d) = \gamma_C(d)$;
- 2) $\forall a \in B : \gamma_{B-a}(d) < \gamma_C(d)$.

The fuzzy rough set model is the generalization of classical rough set model and rough-fuzzy set model. When the relations between objects are crisp equivalence relations and the object subset to be approximated is a fuzzy set then the model will degrade to rough-fuzzy set model. Furthermore, if object subset to be approximated is crisp, the model is the classical one.

3 Information measure for fuzzy-rough set model.

In this section we will propose a new entropy to measure the discernibility power of a fuzzy equivalence relation.

Given a finite set U , A is a fuzzy or real-valued attribute set, which generates a fuzzy equivalence relation R_A on U . The fuzzy relation matrix $M(R_A)$ is denoted by

$$M(R_A) = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{pmatrix}$$

where $r_{ij} \in [0, 1]$ is the relation value of x_i and x_j . In fact, the nominal attribute is a special case, where $r_{ij} \in \{0, 1\}$, which will generate a crisp equivalence relation.

DEFINITION 3.1. The quotient set generated by an equivalence relation is defined as

$$U/R = \{[x_i]_R\}_{i=1}^n \\ \text{where } [x_i]_R = \left\{ \frac{r_{i1}}{x_1} + \frac{r_{i2}}{x_2} + \cdots + \frac{r_{in}}{x_n} \right\}.$$

DEFINITION 3.2. The cardinality $|[x_i]_R|$ of $[x_i]_R$ is defined as

$$|[x_i]_R| = \sum_{j=1}^n r_{ij}.$$

DEFINITION 3.3. Information quantity of the fuzzy attribute set or the fuzzy equivalence relation is defined as

$$H(R_A) = -\frac{1}{n} \sum_{i=1}^n \log \lambda_i.$$

where $\lambda_i = \frac{|[x_i]_R|}{n}$.

Property 1. If A is a nominal attribute, $M(R_A)$ is the relation matrix generated by A , $H(A)$ denotes the Shannon information quantity, and $H(R_A)$ is the information value computed according to definition 3.3, and then we have

$$H(A) = H(R_A)$$

According to property 1, if the relation R is a crisp equivalence relation, the proposed information measure is identical to Shannon's one. The following definitions of joint entropy and conditional entropy have the same

property. In the follows we will denote two information measures indiscriminatingly.

The formula of information measure forms a map: $H : R \rightarrow R^+$, where R is a equivalence relation matrix, R^+ is the non-negative real-number set. This map builds a foundation on that we can compare the discernibility power, partition power or approximating power of multiple fuzzy equivalence relations. Entropy value increases monotonously with the discernibility power or the knowledge's fineness. So the finer partition is, the greater entropy is, and the more significant attribute set is.

DEFINITION 3.4. Given a fuzzy information system $\langle U, A, V, f \rangle$, A is the fuzzy or numeric attribute set. B and E are two subsets of A . $[x_i]_B$ and $[x_i]_E$ are fuzzy equivalence classes containing x_i generated by B and E , respectively. The joint entropy of B and E is defined as

$$H(BE) = H(R_E R_B) = -\frac{1}{n} \sum_{i=1}^n \log \frac{|[x_i]_B \cap [x_i]_E|}{n}.$$

DEFINITION 3.5. Given a fuzzy information system $\langle U, A, V, f \rangle$, A is the attribute set. B and E are two subsets of A . $[x_i]_B$ and $[x_i]_E$ are fuzzy equivalence classes containing x_i generated by B and E , respectively. The conditional entropy of E conditioned to B is defined as

$$H(E|B) = -\frac{1}{n} \sum_{i=1}^n \log \frac{|[x_i]_E \cap [x_i]_B|}{|[x_i]_B|}.$$

THEOREM 3.1. $H(E|B) = H(BE) - H(B)$

THEOREM 3.2. Given a fuzzy information system $\langle U, A, V, f \rangle$, A is the fuzzy attribute set. B and E are two subsets of A . $[x_i]_B$ and $[x_i]_E$ are fuzzy equivalence classes containing x_i generated by B and E , respectively. The fuzzy equivalence relations induced by B and E are denoted by R and S , respectively. Then we have:

- 1) $\forall B \subseteq A : H(B) \geq 0$;
- 2) $H(BE) \geq \max\{H(B), H(E)\}$;
- 3) $B \supseteq E$ or $R_B \subseteq R_E : H(BE) = H(B)$;
- 4) $B \supseteq E$ or $R_B \subseteq R_E : H(E|B) = 0$;

The first item of theorem 3.1 shows the information introduced by any attribute subset is non-negative, the second shows the discernibility power of the union of two attribute subset will be no less than that of any single subset, which means introducing a new attribute or attribute subset at least will not decrease the discernibility power. The last two items show attribute subset won't introduce information relative B if E is contained by B. the properties of the information

measure has a same observation of classification as the Boolean logic methodology, which is a class of paradigm of classifier, such as ID3, CART, C4.5 and rough set theory.

THEOREM 3.3. Given a fuzzy information system $\langle U, A, V, f \rangle$, $B \subseteq A$, $a \in B$, $H(B) = H(B-a)$ if a is redundant, $H(B) > H(B-a)$ if B is independent. B is a reduct if B satisfies:

- 1) $H(B) = H(A)$;
- 2) $\forall a \in B : H(B) > H(B-a)$.

THEOREM 3.4. Given a fuzzy information system $\langle U, A, V, f \rangle$, $A = C \cup d$. B is a subset of C . $\forall a \in B$, $H(d|B-a) = H(d|B)$ if a is redundant in B relative to d ; $H(d|B-a) > H(d|B)$ if B is independent. B is a reduct of C relative to d if B satisfies:

- 1) $H(d|B) = H(d|C)$;
- 2) $\forall a \in B : H(d|B-a) > H(d|B)$.

Theorems 3.3 and 3.4 give the definitions of dependency, reduct and relative reduct in terms of information theory, while definitions 2.5 and 2.6 are defined in terms of algebra. In fact two classes of definitions are equivalent. The proof was given in [50].

4 Reduction algorithms for unsupervised and supervised hybrid data.

Reduct is an important concept in rough set theory and data reduction is a main application of rough set theory in pattern recognition and data mining. As it has been proven that finding the minimal reduct of an information system is a NP hard problem. Some heuristic algorithms have been invented based on significance measures of attributes. These algorithms get a suboptimal result but relatively low time-consuming [1, 25, 27]. Shannon's entropy was used as a significance measure in some classical machine learning algorithm, such as the famous ID3 algorithm series, and proven to be a good measure. In the above section, we propose a novel information measure for fuzzy indiscernibility or equivalence relation and show that the entropy can be degraded to Shannon's one when the relation measured is a crisp equivalence one. It shows that the proposed measure can be used as a measure of discernibility power of a crisp equivalence relation and a fuzzy one. So unified reduction algorithms for hybrid data are feasible.

Data dimensionality reduction will be divided into three steps: relation computation, reduction and reduct validation. Relation computation is to generate relation matrices using a relation function with attributes. Then reduction algorithms are performed on the matrices and find some reduct of the original data. Finally employing a validation function, which may be a classifier or

a discriminability criterion, we test the reduct and find a best one. The procedure is shown as follows. No matter cases $\{x_i\}_{i=1}^n$ are described by nominal attributes or numeric features or fuzzy variables, the relations between the cases can all be denoted by a relation matrix : $M(R) = (r_{ij})_{n \times n}$.

If A is a nominal attribute set,

$$r_{ij} = \begin{cases} 1, & f(x_i, a) = f(x_j, a), \forall a \in A \\ 0, & \text{otherwise} \end{cases};$$

If attribute a is a numeric attribute, the value the relation can mapped by a symmetric function:

$$r_{ij} = f(\|x_i - x_j\|),$$

where function f should satisfy:

- 1) $f(0) = 1, f(\infty) = 0$ and $f(\bullet) \in [0, 1]$;
- 2) $r_{ij} = r_{ji}$ and $r_{ii} = 1$

According to 2), Relation R will satisfies reflexivity and symmetry. So a similarity relation matrix will be produced by the functions.

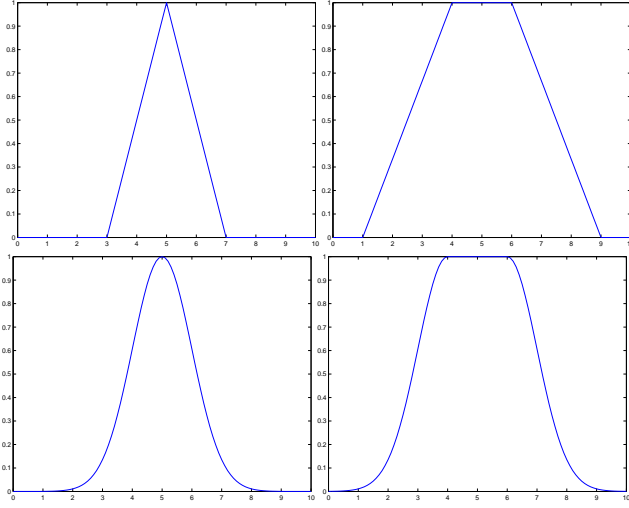


Figure 1: some similarity relation functions for numeric data.

As to fuzzy attributes, there are a great many candidate similarity measures [47]. For example:

- 1) Hamming similarity measure:

$$S(x_i, x_j) = \frac{1}{m} \sum_{k=1}^m (1 - |\mu_{A_k}(x_i) - \mu_{A_k}(x_j)|);$$

- 2) Max-Min similarity measure:

$$S(x_i, x_j) = \frac{1}{m} \left\{ \sum_{k=1}^m \frac{\min(\mu_{A_k}(x_i), \mu_{A_k}(x_j))}{\max(\mu_{A_k}(x_i), \mu_{A_k}(x_j))} \right\}.$$

Employing a max-min closure operation, we can get a fuzzy equivalence relation [48].

As has pointed in §3, the proposed entropy can be used as measure of the discernibility power of a relation or an attribute. The greater the entropy value is, the stronger the discernibility is and the more significant the attribute is. According to the properties of proposed

entropy, adding a novel condition attribute into the information system, the entropy value will increase monotonously, which reflexes that adding information will lead to enhancement of the discernibility power. The increment of information by an attribute reflexes the increment of discernibility of the system. So the significance of an attribute can be defined as follows.

DEFINITION 4.1. Given a fuzzy information system $\langle U, A, V, f \rangle$, $B \subseteq A, a \in B$, the significance of attribute a in attribute set B is defined as

$$SIG(a, B) = H(B) - H(B - a)$$

The above definition works in unsupervised feature selection. $SIG(a, B)$, called Significance of attribute a in B , measures the increment of discernibility power introduced by attribute a .

DEFINITION 4.2. Given a fuzzy information system $\langle U, A, V, f \rangle$, $A = C \cup d$, where C is the condition attribute set and d is the decision attribute. $B \subseteq C, \forall a \in B$, the significance of attribute a in attribute set B relative to d is defined as

$$SIG(a, B, d) = H(d|B - a) - H(d|B)$$

This definition computes the increment of discernibility power relative to the decision introducing by attribute a . So it may be used as a supervised measure for feature selection.

Based on the above measures, two greedy algorithms for computing reduct and relative reduct can be constructed, respectively.

Algorithm 1: Algorithm for calculating reduct

Input: Information system $IS \langle U, A, V, f \rangle$.

Output: One reduct of IS

Step 1: $\forall a \in A$: compute the equivalence relation;

Step 2: $\phi \rightarrow red$;

Step 3: For each $a_i \in A - red$ Compute $H_i = H(a_i, red)$

End

Step 4: Choose attribute which satisfies:

$$H(a|red) = \max_i (SIG(a_i, red))$$

Step 5: If $H(a|red) > 0$, then $red \cup a \rightarrow red$ goto step3, Else return, End

Algorithm 2: Algorithm for calculating relative reduct.

Input: Information system $IS \langle U, A = C \cup d, V, f \rangle$.

Output: One relative reduct D_red of IS

Step 1: $\forall a \in A$ compute the equivalence;

Step 2: $\phi \rightarrow D_red$;

Step 3: For each $a_i \in C - D_red$, Compute $H_i = SIG(a_i, D_red, d)$ End

Step 4: Choose attribute which satisfies:

$$SIG(a, red, d) = \max_i (H_i)$$

Step 5: If $SIG(a, red, d) > 0$, then $D_red \cup a \rightarrow D_red$ goto step3, Else return D_red End

R. Jensen [42] proposed that a problem may arise when this approach is compared to the crisp attribute reduction. In classical rough set attribute reduction, a reduct is defined as a subset of attributes which has the same information quantity as the full attribute set, which means that the value $H(B)H(d|B)$ should be identical to $H(A)H(d|A)$. However, in the fuzzy-rough approaches, it is not necessarily the case. We can specify the degree threshold λ . So that the algorithms will stop if the condition $SIG(a, red) \leq \lambda(SIG(a, red, d) \leq \lambda)$ is satisfied.

5 Experiments and analysis.

A series of experiments have been conducted to test the proposed significance measure of attributes and feature selection based on UCI data. In this section we will show some experimental results and analysis. All experiments have been performed on data set shown in the following table. We find the attributes of data BC and BCW are nominal, and others are hybrid.

Experiment 1: ranking based feature selection vs. the proposed dimensionality reduction. In feature subset selection, many algorithms include ranking as a principal or auxiliary selection mechanism because of its simplicity, scalability and good empirical success [1]. Ranking methods employ an evaluation function, such as inter-class distance, correlation criteria, mutual information and accuracy of a classifier to sort the candidate features. Some top features are selected. The main drawback of ranking is it can not detect the redundancy or correlation among condition set. So although they are the greatest discernible feature individually, their combination may have weak discernible power. Only under certain independence or orthogonality, ranking may be optimal with respect to a given classifier [1].

In the follows, an experiment is shown based on data wine. the order of significance of attribute set is 7, 13, 12, 10, 1, 11, 6, 2, 8, 4, 9, 5, 3. With reduction algorithm 2, attribute subset 7, 1, 11, 6, 3, 12 are selected one by one as a reduct, called subset 1.

In order to compare two feature subset selection, top six attributes 7, 13, 12, 10, 1, 11 are selected in ranking, called subset 2. Figures 2 and 3 show the distribution of data in 2-D feature space. Figure 2 is the distribution with attribute 7, 1, 11, 11, 6, 6, 3, 3, 12, respectively. And Figure 3 is the distribution with attribute 7, 13, 13, 12, 12, 10, 10, 1, 1, 11. From the two-dimension feature space, we find that the attributes by ranking have even better discernibility power than the attributes selected by the fuzzy-rough reduction algorithm. Here we choose SVM as a validation function for feature selection. 2/3 samples are randomly selected as training set, and the others are test set.

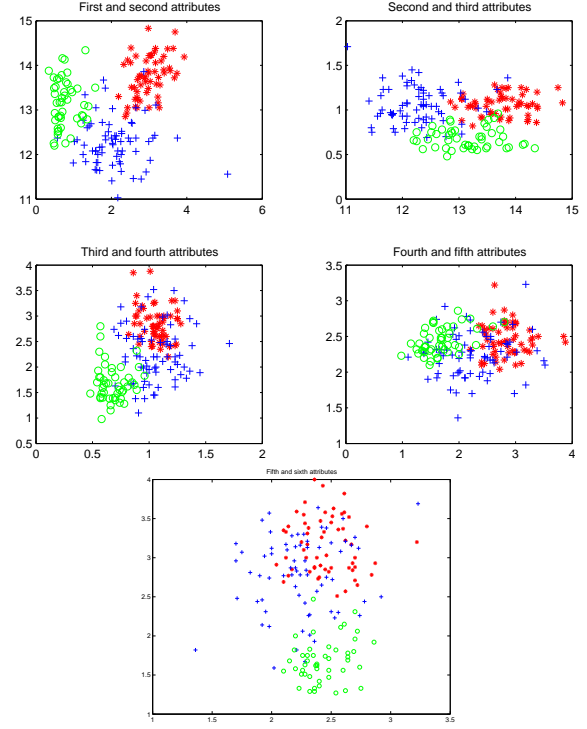


Figure 2: Distribution of wine samples with attributes 7, 1, 11, 6, 3, 12, Accuracy: 94.87%.

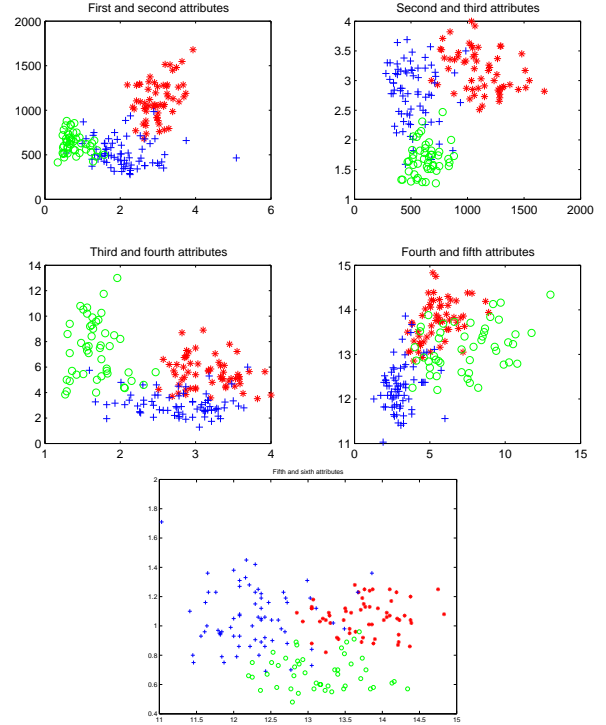


Figure 3: Distribution of wine samples with attributes 7, 13, 12, 10, 1, 11, Accuracy: 93.33%.

Table 1: Summary of the experiment data sets.

Data set		Size	Class Number	Attribute number		
Abr.	Original name			Total	Numeric	Nominal
BC	Breast cancer	286	2	10	0	10
BCW	Breast-cancer-wisconsin1	699	2	10	0	10
WDBC	Breast-cancer-wisconsin2	569	2	31	30	1
WPBC	Breast-cancer-wisconsin3	198	2	33	32	1
Cre	Credit Approval	690	2	16	6	10
Cle	Cleve Database	303	5	14	5	9
Der	Dermatology	366	6	34	33	1
Eco	Protein Localization	336	8	8	7	1
Gls	Glass Identification	214	6	9	8	1
Heart	Heart Disease	270	2	14	6	8
Ion	Ionosphere	351	2	35	34	1
Son	Sonar mines	1389	3	61	60	1
Win	Wine Recognition	178	3	14	13	1
Vow	Vowel Database	990	11	11	10	1

Table 2: Correlation coefficient matrix of attribute set 7, 1, 11, 6, 3, 4 with correlation entropy 0.8110.

	A1	A2	A3	A4	A5	A6
A1	1.0000	0.2368	0.5435	0.8646	0.1151	-0.3514
A2	0.2368	1.0000	-0.0717	0.2891	0.2115	-0.3102
A3	0.5435	-0.0717	1.0000	0.4337	-0.0747	-0.2740
A4	0.8646	0.2891	0.4337	1.0000	0.1290	-0.3211
A5	0.1151	0.2115	-0.0747	0.1290	1.0000	0.4434
A6	-0.3514	-0.3102	-0.2740	-0.3211	0.4434	1.0000

Table 3: Correlation coefficient matrix of attributes 7, 13, 12, 10, 1, 11 with correlation entropy 0.7364.

	A1	A2	A3	A4	A5	A6
A1	1.0000	0.4942	0.7872	-0.1724	0.2368	0.5435
A2	0.4942	1.0000	0.3128	0.3161	0.6437	0.2362
A3	0.7872	0.3128	1.0000	-0.4288	0.0723	0.5655
A4	-0.1724	0.3161	-0.4288	1.0000	0.5464	-0.5218
A5	0.2368	0.6437	0.0723	0.5464	1.0000	-0.0717
A6	0.5435	0.2362	0.5655	-0.5218	-0.0717	1.0000

We choose support vector machine (SVM) as a validation function for feature subsets. 2/3 samples are randomly selected as training set, and the others are test set. The accuracy with attribute subset 1 is 94.87%, while the accuracy with attribute subset 2 is 93.33%.

Why the attributes with better discriminability in two-dimensional space get an even worse classification performance? As we have pointed, selecting the most relevant features is usually suboptimal for building a classifier if the features are redundant or dependent. Generally speaking, ranking method only computes the dependency between condition attributes and decision attribute, while neglect the dependency among condition attributes. Let's analyze the correlation between the selected condition attributes. Correlation coeffi-

cients are showed in table 2 and 3. Wang [49] introduced correlation entropy to measure the correlation of a variable set. The entropy is defined as

$$H_R = - \sum_{i=1}^N \frac{\lambda_i}{N} \log_N \frac{\lambda_i}{N}$$

where λ_i is i th eigenvalue of correlation coefficient matrix. the greater the entropy value is, the weaker the correlation of attribute set is. If all attributes are linear correlation, the correlation entropy is 0, and if all the correlation coefficient are zero, then the entropy is 1. Wang called the dependency of attributes overlap information. We employ the measure to compute the correlation degree of the selected attributes. The correlation entropy of subset 1 is 0.8110, while entropy of subset 2 is 0.7364, which shows the correlation degree of subset 1 is lower than that of subset 2.

Experiment 2: Comparison of reduction methods. In order to test the performance of the proposed reduction algorithm, some contrastive experiments are conducted based on UCI data set. We compare the classical rough set reduction with the proposed one and employ SVM classifier as the validation function. The experiment data is shown in table 4.

The classical rough set theory works in nominal domain. We perform discretization on numeric data. The numeric attributes are discretized into three intervals by equal-width, equal-frequency and fuzzy c-means clustering. As to fuzzy-rough reduction algorithm, the relation matrices are computed with a triangle function. The numbers of selected attributes and accuracy of classification with SVM are shown in table 4. There is no numeric attribute in data sets BC and BCW. From the table we find the results of reduction and classification

Table 4: Comparisons of Fuzzy-rough technique vs. discretization with SVM classifiers.

Data	Original data		Reduct(Equi-width)		Reduct(Equi-Frequency)		Reduct(FCM)		Reduct(Fuzzy-rough)	
	n	Accuracy	n	Accuracy	n	Accuracy	n	Accuracy	n	Accuracy
BC	10	71.58%	8	72.63%	8	72.63%	8	72.63%	8	72.63%
BCW	10	98.28%	4	98.71%	4	98.71%	4	98.71%	4	98.71%
WDBC	31	93.16%	8	94.21%	12	93.68%	6	95.26%	17	95.26%
WPBC	33	74.24%	8	71.21%	6	75.76%	6	68.18%	17	81.82%
Cre	16	82.17%	11	81.74%	9	83.04%	11	81.74%	12	81.74%
Cle	14	59.41%	10	57.43%	8	60.4%	9	59.41%	12	56.44%
Der	34	90.91%	12	93.39%	11	99.17%	11	99.17%	11	94.21%
Eco	8	70.18%	7	70.18%	7	70.18%	7	70.18%	7	70.18%
Gls	9	61.97%	7	64.79%	6	54.93%	8	63.38%	8	63.38%
Heart	14	83.33%	9	83.33%	8	82.22%	8	84.44%	9	83.33%
Ion	35	92.31%	7	85.47%	7	85.47%	8	87.18%	12	88.03%
Son	61	78.57%	6	71.43%	6	52.86%	8	74.29%	9	74.29%
Win	14	96.67%	4	91.67%	4	91.67%	4	91.67%	6	94.87%
Vow	11	59.09%	10	63.94%	10	63.94%	10	63.94%	10	63.94%
Average		79.42%		78.58%		77.46%		79.30%		79.92%

with classical rough set method and the fuzzy one are identical, respectively, which shows that the method we proposed can degenerate to the classical case.

6 Conclusions.

Rough set theory has proven a powerful tool for feature subset selection and rule extraction. The classical rough set model just works in nominal domain. In this paper we propose a novel information measure, which can measure the discernibility power of a crisp equivalence relation and fuzzy one. And it is proven that when the relation matrix is a crisp equivalence one, the proposed entropy will be degraded to Shannon's entropy. Based on the proposed entropy, some basic definitions in fuzzy rough set model are presented. Two reduction algorithms for unsupervised and supervised dimensionality reduction are given. Experiments show the algorithms get the same results as that of the classical rough set approaches when the attributes of data are all nominal. However, the performance of the proposed reduction is better than the classical methods with respect to hybrid data.

References

- [1] Isabelle Guyon and Andre Elisseeff, *An introduction to variable and feature selection*, Journal of machine learning research, 3 (2003), pp. 1157–1182.
- [2] David Hand, Heikki Mannila and Padhraic Smyth, *Principles of data mining*, MIT publisher, 2001.
- [3] H. Liu and R. Setiono, *Some issues on scalable feature selection*, Expert systems with applications, 15 (1998), pp. 333–339.
- [4] E.C.C. Tsang, D.S. Yeung and X. Z. Wang, *OFFSS: Optimal fuzzy-valued feature subset selection*, IEEE transactions on fuzzy systems, 2(2003), pp. 202–213.
- [5] Kari Torkkola, *Feature extraction by non-parametric mutual information maximization*, Journal of machine learning research, 3 (2003), pp. 1415–1438.
- [6] M. Dash and H. Liu, *Consistency-based search in feature selection*, AI 151(2003), pp. 155–176.
- [7] Avrim L. Blum and Pat Langley, *Selection of relevant features and examples in machine learning*, Artificial intelligence 97(1997), pp. 245–271.
- [8] H. Liu, H. Motoda and L. Yu, *Feature Selection with Selective Sampling*, Proceedings of the 19th ICML, July 8–12, 2002, Sydney, pp. 395–402.
- [9] H. X. Li and L. D. Xu, *Feature space theory—a mathematical foundation for data mining*, Knowledge-based systems 14 (2001), pp. 253–257.
- [10] Hwang, Kuo-Feng, Chang and Chin-Chen, *A fast pixel mapping algorithm using principal component analysis*, Pattern Recognition Letters Volume: 23, Issue: 14, December, 2002, pp. 1747–1753.
- [11] Gilmour, Justin and Wang Liuping, *Detection of process abnormality in food extruder using principle component analysis*, Chemical Engineering Science Volume: 57, Issue: 7, April, 2002, pp. 1091–1098.
- [12] Chen Songcan and Zhu Yulian, *Subpattern-based principle component analysis*, Pattern Recognition Volume: 37, Issue: 5, May, 2004, pp. 1081–1083.
- [13] Cheung, Y. and Xu, L., *Independent component ordering in ICA time series analysis*, Neurocomputing Volume: 41, Issue: 1–4, October, 2001, pp. 145–152.
- [14] Wakako H., *Separation of independent components from data mixed by several mixing matrices*, Signal processing. Vol.82, No.12, 2002, pp. 1949–1961.
- [15] Ron Kohavi and George H. John, *Wrappers for feature subset selection*, AI 97 (1997), pp. 73–324.
- [16] Selwyn Piramuthu, *Evaluating feature selection meth-*

- ods for learning in data mining applications, European journal of operational research, 156 (2004), pp. 483–494.
- [17] K. Kira and L.A. Rendell, *The feature selection problem: Traditional methods and a new algorithm*, Proceedings of AAAI-92, 1992, pp. 129–134.
 - [18] Kwak, N. Chong-Ho Choi, *Input feature selection for classification problems*, IEEE transaction on neural networks, Vol.13, No.1, 2002, pp. 143–159.
 - [19] Lei Yu, Huan Liu, *Efficiently handling feature redundancy in high dimensional data*, In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 24 - 27(2003), pp. 685–690.
 - [20] W. Duch, et al, *Feature selection based on information theory, consistency and separability indices*, Proceeding on 9th neural information processing, vol.4(2002), pp. 1951–1955.
 - [21] L. Yu and H. Liu, *Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution*, In Proceedings of The Twentieth International Conference on Machine Learning (ICML-03), August 21-24(2003), pp. 856–863.
 - [22] P. Mitra. C.A. Murthy, S. K. Pal, *Unsupervised feature selection using feature similarity*, IEEE transactions on pattern analysis and machine intelligence, Vol. 24, No. 3(2002), pp. 301–312.
 - [23] Beynon, Malcolm, *Reducts within the variable precision rough sets model: A further investigation*, European Journal of Operational Research Volume: 134, Issue: 3, November 1(2001), pp. 592–605.
 - [24] Mi, Ju-Sheng; Wu, Wei-Zhi and Zhang, Wen-Xiu, *Approaches to knowledge reduction based on variable precision rough set model*, Information Sciences. Vol. 159, Issue: 3-4, 15(2004), pp. 255–272.
 - [25] Pawlak Z, *rough sets-theoretical aspects of reasoning about data*. Kluwer academic publishers, 1991.
 - [26] Wang G., Hu H., Yang D., *Decision table reduction based on conditional information entropy*, Chinese journal of computers. Vol. 25, No. 7, 1-8(2002).
 - [27] Skowron A. Rauszer C., *the discernibility matrices and functions in information systems*, Intelligent decision support: handbook of applications and advances of rough set theory, 1992, pp. 331–362.
 - [28] Wang Jue, Miao Duo-Qian, *Analysis on attribute reduction strategies of rough set*, Journal of computer science and technology. Vol. 13, No.2, 1998, pp. 189–193.
 - [29] Moradi, Hamid; Grzymala-Busse, Jerzy W.; Roberts, James A. , *Entropy of English Text: Experiments with Humans and a Machine Learning System Based on Rough Sets*, Information Sciences Volume: 104, Issue: 1-2, January, 1998, pp. 31–47.
 - [30] Swiniarski, Roman W. Larry Hargis, *Rough sets as a front end of neural networks texture classifier*, Neurocomputing., 36(2001) pp. 85–102.
 - [31] Swiniarski, Roman W.; Skowron, Andrzej, *Rough set methods in feature selection and recognition*, Pattern Recognition Letters Volume: 24, Issue: 6, March, 2003, pp. 833–849.
 - [32] D. Dubois, H. Prade, *Putting fuzzy sets and rough sets together*, R. Slowinski (Ed.), Intelligent Decision support, Kluwer Academic, Dordrecht, 1992, pp. 203–232.
 - [33] Morsi, Nehad N.; Yakout, M.M., *Axiomatics for fuzzy rough sets*, Fuzzy Sets and Systems Volume: 100, Issue: 1-3, November 16, 1998, pp. 327–342.
 - [34] Radzikowska, Anna Maria; Kerre, Etienne E., *A comparative study of fuzzy rough sets*, Fuzzy Sets and Systems Vol.126, No.2, 2002, pp. 137–155.
 - [35] Wu, Wei-Zhi; Mi, Ju-Sheng; Zhang, Wen-Xiu, *Generalized fuzzy rough sets*. Information Sciences Volume: 151, May, 2003, pp. 263–282.
 - [36] Wu, Wei-Zhi; Zhang, Wen-Xiu, *Constructive and axiomatic approaches of fuzzy approximation operators*, Information Sciences Volume: 159, Issue: 3-4, February 15, 2004, pp. 233–254.
 - [37] Wang Yi-Fan, *Mining stock price using fuzzy rough set system*, Expert Systems with Applications Volume: 24, Issue: 1, January, 2003, pp. 13–23.
 - [38] Srinivasan, Padmini; Ruiz, Miguel E.; Kraft, Donald H.; Chen, Jianhua, *Vocabulary mining for information retrieval: rough sets and fuzzy sets*, Information Processing and Management Volume: 37, Issue: 1, January 1, 2001, pp. 15–38.
 - [39] Q. Shen and A. Chouchoulas, *A rough-fuzzy approach for generating classification rules*, Pattern Recognition, 35(11)(2002) pp. 2425–2438.
 - [40] Chmielewski, Michal R.; Grzymala-Busse, Jerzy W., *Global Discretization of Continuous Attributes as Preprocessing for Machine Learning*, International Journal of Approximate Reasoning Volume: 15, Issue: 4, November, 1996, pp. 319–331.
 - [41] Roy, Amitava; Pal, Sankar K., *Fuzzy discretization of feature space for a rough set classifier*, Pattern Recognition Letters V. 24, No.6(2003), pp. 895–902.
 - [42] R. Jensen, Q. Shen, *Fuzzy-rough attribute reduction with application to web categorization*, Fuzzy sets and systems, 141 (2004), pp. 469–485.
 - [43] L. Zadeh, *Probability measures of fuzzy events*, J. Math. Anal. Appl. 23(1965), pp. 421–427.
 - [44] Yager, Ronald R., *Measures of Entropy and Fuzziness Related to Aggregation Operators*, Information Sciences Volume: 82, Issue: 3-4, January, 1995, pp. 147–166.
 - [45] Bertoluzza, Carlo; Doldi, Viviana; Naval, Gloria., *Uncertainty measure on fuzzy partitions*, Fuzzy Sets and Systems Vol.142, No.1, 2004, pp. 105–116.
 - [46] Guo, Caimei and Zhang, Deli, *On set-valued fuzzy measures*, Information Sciences Volume: 160, Issue: 1-4, March 22, 2004, pp. 13–25.
 - [47] Dengfeng, Li; Chuntian, Cheng., *New similarity measures of intuitionistic fuzzy sets and application to pattern recognitions*, Pattern Recognition Letters Volume: 23, Issue: 1-3, January, 2002, pp. 221–225.
 - [48] Lee, Hsuan-Shih. *An optimal algorithm for computing the max-min transitive closure of a fuzzy similarity matrix*, Fuzzy Sets and Systems Vol.123, No.1(2001), pp. 129–136.

- [49] Qiang Wang, Yi Shen, Ye Zhang, *A fast method to evaluate the performance of image fusion techniques and its error analysis*, Instrumentation and measurement technology conference, 2003.
- [50] Qinghua Hu and Daren Yu, *Entropies of fuzzy indiscernibility relation and its operations*, International Journal of uncertainty, fuzziness and knowledge-based systems. Vol. 12, No. 5, pp. 575–589.
- [51] Qinghua Hu, Daren Yu and Zongxia Xie, *Reduction algorithms for hybrid data based on fuzzy rough set approaches*, International Conference on Machine Learning and Cybernetics(2004), pp. 1469–1474.

HARMONY: Efficiently Mining the Best Rules for Classification *

Jianyong Wang[†]

George Karypis[‡]

Abstract

Many studies have shown that rule-based classifiers perform well in classifying categorical and sparse high-dimensional databases. However, a fundamental limitation with many rule-based classifiers is that they find the rules by employing various heuristic methods to prune the search space, and select the rules based on the sequential database covering paradigm. As a result, the final set of rules that they use may not be the globally best rules for some instances in the training database. To make matters worse, these algorithms fail to fully exploit some more effective search space pruning methods in order to scale to large databases.

In this paper we present a new classifier, HARMONY, which directly mines the final set of classification rules. HARMONY uses an instance-centric rule-generation approach and it can assure for each training instance, one of the highest-confidence rules covering this instance is included in the final rule set, which helps in improving the overall accuracy of the classifier. By introducing several novel search strategies and pruning methods into the rule discovery process, HARMONY also has high efficiency and good scalability. Our thorough performance study with some large text and categorical databases has shown that HARMONY outperforms many well-known classifiers in terms of both accuracy and computational efficiency, and scales well w.r.t. the database size.

1 Introduction

As one of the most fundamental data mining tasks, classification has been extensively studied and various

types of classification algorithms have been proposed. Among which, one category is the rule-based classifiers [26, 27, 13, 30]. They build a model from the training database as a set of high-quality rules, which can be used to predict the class labels of unlabeled instances. Many studies have shown that rule-based classification algorithms perform very well in classifying both categorical databases [27, 25, 24, 30] and sparse high-dimensional databases such as those arising in the context of document classification [6, 5].

Some traditional rule-based algorithms like FOIL [27], RIPPER [13], and CPAR [30] discover a set of classification rules one-rule-at-a-time and employ a sequential covering methodology to eliminate from the training set the positive instances that are covered by each newly discovered rule. This *rule induction* process is done in a greedy fashion as it employs various heuristics (e.g., information gain) to determine how each rule would be extended. Due to this heuristic rule-induction process and the sequential covering framework, the final set of discovered rules are not guaranteed to be the best possible. For example, due to the removal of some training instances, the information gain is computed based on the incomplete information; thus, the variable (or literal) chosen by these algorithms to extend the current rule will be no longer the globally optimal one. Moreover, for multi-class problems, these algorithms need to be applied multiple times, each time mining the rules for one class. If the training database is large and contains many classes, the algorithms will be inefficient.

Since the introduction of association rule mining [2], many association-based (or related) classifiers have been proposed [17, 22, 7, 4, 8, 23, 15, 5, 31, 14]. Some typical examples like CBA [25] and CMAR [24] adopt efficient association rule mining algorithms (e.g., Apriori [3] and FP-growth [20]) to first mine a large number of high-confidence rules satisfying a user-specified minimum support and confidence thresholds, then use various sequential-covering schemes to select from them a set of high-quality rules to be used for classification. Since these schemes defer the selection step only after a large intermediate set of high-confidence rules have been identified, they tend to achieve somewhat better accuracy than the traditional heuristic rule induc-

*This work was supported in part by NSF CCR-9972519, EIA-9986042, ACI-9982274, ACI-0133464, and ACI-0312828; the Digital Technology Center at the University of Minnesota; and by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

[†]Work was done while at University of Minnesota and currently is with Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P.R. China. Email: jianyong@cs.umn.edu.

[‡]Department of Computer Science and Engineering/Digital Technology Center/Army HPC Research Center, University of Minnesota. Email: karypis@cs.umn.edu.

tion schemes [30]. However, the drawback of these approaches is that the number of initial rules is usually extremely large, significantly increasing the rule discovery and selection time.

In this paper we propose a new classification algorithm, HARMONY¹, which can overcome the problems of both the rule-induction-based and the association-rule-based algorithms. HARMONY directly mines for each training instance one of the highest confidence frequent classification rules that it supports, and builds the classification model from the union of these rules over the entire set of instances. Thus HARMONY employs an *instance-centric* rule generation framework and is guaranteed to find and include the best possible rule for each training instance. Moreover, since each training instance usually supports many of the discovered rules, the overall classifier can better generalize to new instances and thus achieve better classification performance.

To achieve high computational efficiency, HARMONY mines the classification rules for all the classes simultaneously and directly mines the final set of classification rules by pushing deeply some effective pruning methods into the projection-based frequent itemset mining framework. All these pruning methods preserve the completeness of the resulting rule-set in the sense that they only remove from consideration rules that are guaranteed not to be of high quality. We have performed numerous performance studies with various databases and shown that HARMONY can achieve better accuracy while maintaining high efficiency.

The rest of the paper is organized as follows. Section 2 introduces some basic definitions and notations. Section 3 discusses in detail the HARMONY algorithm. Section 4 describes some of the related work in this area. The thorough performance study is presented in Section 5. Finally, the paper concludes with Section 6.

2 Notations and Definitions

A *training database* $TrDB$ is a set of training instances², where each training instance, denoted as a triple $\langle tid, X, cid \rangle$, contains a set of items (i.e., X) and is associated with a unique training instance identifier tid , and a class identifier $cid \in \{c_1, c_2, \dots, c_k\}$ (a class identifier is also called a class label, and we assume there are totally k distinct class labels in $TrDB$). Table 1 illustrates an example training database, which contains totally eight instances and two classes. Let

$I = \{i_1, i_2, \dots, i_n\}$ be the complete set of distinct items appearing in $TrDB$. An *itemset* Y is a non-empty subset of I and is called an *l -itemset* if it contains l items. An itemset $\{x_1, \dots, x_l\}$ is also denoted by $x_1 \cdots x_l$. A training instance $\langle tid, X, cid \rangle$ is said to *contain* itemset Y if $Y \subseteq X$. The number of instances in $TrDB$ containing itemset Y is called the (absolute) *support* of itemset Y , denoted by sup_Y . The number of instances containing itemset Y and associated with a class label c_i (where $i \in \{1, 2, \dots, k\}$) is called the support of $Y \cup \{c_i\}$, denoted by $sup_Y^{c_i}$. A classification rule has the form: ' $Y \rightarrow c_i : sup_Y^{c_i}, conf_Y^{c_i}$ ', where Y is called the body, c_i the head, $sup_Y^{c_i}$ the support, and $conf_Y^{c_i} = \frac{sup_Y^{c_i}}{sup_Y}$ the confidence of the rule, respectively. In addition, we use $|TrDB|$ to denote the number of instances in database $TrDB$, and for brevity, we sometimes use the instance identifier tid to denote an instance $\langle tid, X, cid \rangle$.

Table 1: An example training database $TrDB$.

Instance identifier	Set of items	Class identifier
01	a, c, e, g	1
02	b, d, e, f	0
03	d, e, f	0
04	a, b, c, e	1
05	a, c, e	1
06	b, d, e	0
07	a, b, e	1
08	a, b, d, e	0

Given a minimum support threshold, min_sup , an itemset Y is *frequent* if $sup_Y \geq min_sup$. A frequent itemset Y supported by any training instance $\langle t_j, X_j, c_i \rangle$ ($1 \leq j \leq |TrDB|$ and $1 \leq i \leq k$) is also called a frequent covering itemset of instance t_j , and ' $Y \rightarrow c_i : sup_Y^{c_i}, conf_Y^{c_i}$ ' is called a frequent covering rule of instance t_j . Among the frequent covering rules of any instance t_j , those with the highest confidence are called the *Highest Confidence Covering Rules* w.r.t. instance t_j . We denote a Highest Confidence Covering Rule w.r.t. instance t_j by $HCCR_{t_j}$, and use $HCCR_{t_j}^{sup}$ and $HCCR_{t_j}^{conf}$ to denote its support and confidence.

3 HARMONY: An Instance-Centric Classifier

In this paper we present HARMONY, an accurate and efficient rule-based classifier with good scalability that is designed to overcome the problems of both the traditional rule-based and the recently proposed association-based classifiers. The key idea behind HARMONY is to build a classifier that instead of using various heuristic methods to discover and/or select rules, it uses the best possible rules for each training instance. As such, it takes an *instance-centric* view and directly mines the database of training instances to find at least one of the highest confidence frequent covering rules (if there

¹ HARMONY stands for the **H**ighest confidence **c**lassification **R**ule Mining **f**or **i**nstance-centric **c**lassif**Y**ing.

² Note there may exist a test database, which is in the same form as the training database and is used to evaluate the performance of a classifier. We denote it by $TeDB$.

is any) and include it in the final set of classification rules. Moreover, HARMONY employs some effective search strategies and pruning methods to speed up the model learning.

The HARMONY algorithm consists of three different modules, referred to as RULEMINER, BUILDMODEL, and NEWINSTANCECLASSIFICATION. The RULEMINER module, takes as input the training database $TrDB$ and the minimum support min_sup , and outputs the set of the highest confidence covering rules (abbreviated as $HCCR$). The BUILDMODEL module, takes $HCCR$ as input and outputs a classification model (abbreviated as CM), which is used by the NEWINSTANCECLASSIFICATION module to classify a new test instance ti . The algorithmic details behind these modules are presented in the rest of this section.

3.1 Mining the Classification Rules

The rule-discovery problem that HARMONY needs to solve in order to generate the sets of rules needed by its classification methodology can be formally defined as follows. Given a training database $TrDB$ and a minimum support threshold min_sup , the problem is to find one of the highest confidence frequent covering rules for each of the training instances in $TrDB$ ³.

A naïve way of solving this problem is to use an existing frequent closed itemset discovery algorithm to first generate all frequent closed itemsets, and then extract from them the highest confidence rule for each training database instance. However, this approach is not very computationally efficient because the number of frequent closed itemsets is usually huge, and both the itemset generation and rule selection are very expensive. For this reason, HARMONY adopts another more efficient way. It directly mines the final set of highest confidence classification rules. By maintaining the highest confidence among the covering rules mined so far for each instance, HARMONY can employ some efficient pruning methods to accelerate the rule discovery.

Note that although we mainly focus on mining any one of the highest confidence frequent covering rules for each training instance, it is straightforward to revise HARMONY to mine the complete set of the highest confidence frequent covering rules or the one with the highest support for each training instance.

3.1.1 Rule Enumeration

The projection-based itemset enumeration framework has been widely used in many frequent itemset mining algorithms [20, 1, 18], and will be used by HARMONY as the basis in enumerating the classification rules. Given a training database $TrDB$ and a minimum support min_sup , HARMONY first computes the frequent items by scanning $TrDB$ once, and sorts them to get a list of frequent items (denoted by f_list) according to a certain ordering scheme. Assume the min_sup is 3 and the lexicographical ordering is the default ordering scheme, the f_list computed from Table 1 is $\{a, b, c, d, e\}$. HARMONY applies the *divide-and-conquer* method plus the *depth-first search strategy*. In our example, HARMONY first mines the rules whose body contains item ‘a’, then mines the rules whose body contains ‘b’ but no ‘a’, ..., and finally mines the rules whose body contains only ‘e’. In mining the rules with item ‘a’, item ‘a’ is treated as the current prefix, and its conditional database (denoted by $TrDB|_a$) is built and the *divide-and-conquer* method is applied recursively with the depth-first search strategy. To build conditional database $TrDB|_a$, HARMONY first identifies the instances in $TrDB$ containing ‘a’ and removes the infrequent items, then sorts the left items in each instance according to the f_list order, finally $TrDB|_a$ is built as $\{\langle 01, ce, 1 \rangle, \langle 04, bce, 1 \rangle, \langle 05, ce, 1 \rangle, \langle 07, be, 1 \rangle, \langle 08, be, 0 \rangle\}$ (infrequent items ‘d’ and ‘g’ are removed). Following the *divide-and-conquer* method, HARMONY first mines the rules with prefix ‘ab’, then mines rules with prefix ‘ac’ but no ‘b’, and finally mines rules with prefix ‘ae’ but no ‘b’ nor ‘c’.

During the mining process, when HARMONY gets a new prefix, it will generate a set of classification rules w.r.t. the training instances covered by the prefix. For each training instance, it always maintains one of its currently highest confidence rules mined so far. Assume the current prefix P is ‘a’ (i.e., $P=a$). As shown in the above example, P covers five instances with *tids* 01, 04, 05, 07, and 08. HARMONY computes the covering rules according to the class distribution w.r.t. the prefix P . In this example, $sup_P=5$, $sup_P^0=1$, $sup_P^1=4$, and HARMONY generates two classification rules:

Rule 1: $a \rightarrow 0 : 1, \frac{1}{5}$

Rule 2: $a \rightarrow 1 : 4, \frac{4}{5}$

Rule 1 covers the instance with *tid* 08, while Rule 2 covers the instances with *tids* 01, 04, 05 and 07. Up to this point, we have $HCCR_{01} = HCCR_{04} = HCCR_{05} = HCCR_{07} = \text{Rule 2}$, and $HCCR_{08} = \text{Rule 1}$.

³Note the input training database must be in the form that is consistent with the corresponding definition in Section 2, otherwise, the training database should be first converted to that form. For example, a numerical database should be first discretized into a categorical one in order to use HARMONY to build the model.

3.1.2 Ordering of the Local Items

In the above rule enumeration process, we used the lexicographical ordering as an illustration to sort the set of local frequent items in order to get the *f_list*. Many projection-based frequent itemset mining algorithms use the item support to order the local frequent items (e.g., the support descending order was adopted in [20] as the ordering scheme). However, because we are interested in the highest confidence rules w.r.t. the training instances, the support-based ordering schemes may not be the most efficient and effective ways. As a result, we propose the following three new ordering schemes as the alternatives.

Let the current prefix be P , its support be sup_P , the support and confidence of the classification rule w.r.t. prefix P and class label c_i , ' $P \rightarrow c_i$ ', be $sup_P^{c_i}$ and $conf_P^{c_i}$, respectively, the set of local frequent items be $\{x_1, x_2, \dots, x_m\}$, the number of prefix P 's conditional instances containing item x_j ($1 \leq j \leq m$) and associated with class label c_i ($1 \leq i \leq k$) be $sup_{P \cup \{x_j\}}^{c_i}$, and the support of $P \cup \{x_j\}$ be $sup_{P \cup \{x_j\}} = \sum_{i=1}^k sup_{P \cup \{x_j\}}^{c_i}$.

Maximum confidence descending order. Given a local item x_j ($1 \leq j \leq m$) w.r.t. P , we can compute k rules with body $P \cup \{x_j\}$, among which, the i -th rule with rule head c_i is:

$$P \cup \{x_j\} \rightarrow c_i : sup_{P \cup \{x_j\}}^{c_i}, \frac{sup_{P \cup \{x_j\}}^{c_i}}{sup_{P \cup \{x_j\}}}$$

The highest confidence among the k rules with body $P \cup \{x_j\}$ is called the maximum confidence of local item x_j , and is defined as the following:

$$(3.1) \quad \frac{\max_{\forall i, 1 \leq i \leq k} sup_{P \cup \{x_j\}}^{c_i}}{sup_{P \cup \{x_j\}}}$$

To mine the highest confidence covering rules as quickly as possible, a good heuristic is to sort the local frequent items in their maximum confidence descending order.

Entropy ascending order. The widely used entropy to some extent measures the purity of a cluster of instances. If the entropy of the set of instances containing $P \cup \{x_j\}$ ($1 \leq j \leq m$) is small, it is highly possible to generate some high confidence rules with body $P \cup \{x_j\}$. Thus another good ordering heuristic is to rank the set of local frequent items in their entropy ascending order, and the entropy w.r.t. item x_j is defined as follows:

$$(3.2) \quad -\frac{1}{\log k} \sum_{i=1}^k \left(\frac{sup_{P \cup \{x_j\}}^{c_i}}{sup_{P \cup \{x_j\}}} \right) \log \left(\frac{sup_{P \cup \{x_j\}}^{c_i}}{sup_{P \cup \{x_j\}}} \right)$$

Correlation coefficient ascending order. Both the maximum confidence descending order and entropy ascending order do not consider the class distribution of the conditional database w.r.t. prefix P , which may cause some problems in some cases. Let us see an example. Assume the number of class labels $k=2$, $sup_P^{c_1} = 12$, and $sup_P^{c_2} = 6$, then we can get two rules with body P as follows:

$$\begin{aligned} \text{Rule 3: } P \rightarrow c_1 : 12, \frac{12}{18} \\ \text{Rule 4: } P \rightarrow c_2 : 6, \frac{6}{18} \end{aligned}$$

Suppose there are two local items, x_1 and x_2 , and $sup_{P \cup \{x_1\}}^{c_1}=2$, $sup_{P \cup \{x_1\}}^{c_2}=1$, $sup_{P \cup \{x_2\}}^{c_1}=1$, and $sup_{P \cup \{x_2\}}^{c_2}=2$. According to Equation 3.1 and Equation 3.2, the maximum confidence and entropy w.r.t. item x_1 are equal to the corresponding maximum confidence and entropy w.r.t. x_2 . Thus we cannot determine which one of x_1 and x_2 should be ranked higher. However, because the conditional database $TrDB|_{P \cup \{x_1\}}$ has the same class distribution as conditional database $TrDB|_P$, we cannot generate rules with body $P \cup \{x_1\}$ and a confidence higher than those with body P (i.e., Rule 3 and Rule 4). The two rules with body $P \cup \{x_1\}$ are shown as the following.

$$\begin{aligned} \text{Rule 5: } P \cup \{x_1\} \rightarrow c_1 : 2, \frac{2}{3} \\ \text{Rule 6: } P \cup \{x_1\} \rightarrow c_2 : 1, \frac{1}{3} \end{aligned}$$

If we examine the rules generated from prefix itemset $P \cup \{x_2\}$ as shown in Rule 7 and Rule 8, we can see Rule 8 has higher confidence than Rule 4, and can be used to replace Rule 4 for the instances covered by Rule 8. In this case, item x_2 should be ranked before item x_1 .

$$\begin{aligned} \text{Rule 7: } P \cup \{x_2\} \rightarrow c_1 : 1, \frac{1}{3} \\ \text{Rule 8: } P \cup \{x_2\} \rightarrow c_2 : 2, \frac{2}{3} \end{aligned}$$

This example suggests that the more similar the class distribution between conditional databases $TrDB|_P$ and $TrDB|_{P \cup \{x_j\}}$ ($1 \leq j \leq m$), the lower is the possibility to generate higher confidence rules from $TrDB|_{P \cup \{x_j\}}$. Because the correlation coefficient is a good metric in measuring the similarity between two vectors (the larger the coefficient, the more similar the two vectors), it can be used to rank the local items. In HARMONY, the correlation coefficient ascending order is by default adopted to sort the local items.

Let \overline{sup}_P be $\frac{1}{k} \sum_{i=1}^k sup_P^{c_i}$, $\overline{sup}_{P \cup \{x_j\}}$ be $\frac{1}{k} \sum_{i=1}^k sup_{P \cup \{x_j\}}^{c_i}$, σ_P be $\sqrt{\frac{1}{k} \sum_{i=1}^k (sup_P^{c_i})^2 - \overline{sup}_P^2}$, $\sigma_{P \cup \{x_j\}}$ be $\sqrt{\frac{1}{k} \sum_{i=1}^k (sup_{P \cup \{x_j\}}^{c_i})^2 - \overline{sup}_{P \cup \{x_j\}}^2}$, the correlation coefficient between prefix P and $P \cup \{x_j\}$ ($1 \leq j \leq m$) is defined as follows.

$$(3.3) \quad \frac{\frac{1}{k} \sum_{i=1}^k (sup_P^{c_i} \quad sup_{P \cup \{x_j\}}^{c_i} - \overline{sup_P} \quad \overline{sup_{P \cup \{x_j\}}})}{\sigma_P \quad \sigma_{P \cup \{x_j\}}}$$

3.1.3 Search Space Pruning

Unlike the association-based algorithms, HARMONY directly mines the final set of classification rules. By maintaining the current highest confidence among the covering rules for each training instance during the mining process, some effective pruning methods can be proposed to improve the algorithm efficiency.

Support equivalence item elimination. Given the current prefix P , among its set of local frequent items $\{x_1, x_2, \dots, x_m\}$, some may have the same support as P . We call them support equivalence items and can be safely pruned according to the following Lemma 3.1.

LEMMA 3.1. (*Support equivalence item pruning*) Any local item x_j w.r.t. prefix P can be safely pruned if it satisfies $sup_{P \cup \{x_j\}} = sup_P$.

Proof. Because $sup_{P \cup \{x_j\}} = sup_P$ holds, $TrDB|_P$ and $TrDB|_{P \cup \{x_j\}}$ contain the same set of conditional instances; thus, their class distributions are also the same and the following equation must hold:

$$\forall i, 1 \leq i \leq k, sup_{P \cup \{x_j\}}^{c_i} = sup_P^{c_i}$$

Given any itemset, Y , which can be used to extend P (Y can be empty), can also be used to extend $P \cup \{x_j\}$, and the following must hold:

$$\forall i, 1 \leq i \leq k, sup_{P \cup \{x_j\} \cup Y}^{c_i} = sup_{P \cup Y}^{c_i}$$

We can further have the following equation:

$$\forall i, 1 \leq i \leq k, \frac{sup_{P \cup \{x_j\} \cup Y}^{c_i}}{sup_{P \cup \{x_j\} \cup Y}} = \frac{sup_{P \cup Y}^{c_i}}{sup_{P \cup Y}}$$

This means the confidence of the rule ' $P \cup \{x_j\} \cup Y \rightarrow c_i$ ' is equal to the confidence of the rule ' $P \cup Y \rightarrow c_i$ ', and we cannot generate higher confidence rules from prefix $P \cup \{x_j\} \cup Y$ in comparison with the rules with body $P \cup Y$. Thus item x_j can be safely pruned. \square

Note $P \cup Y$ is a subset of $P \cup \{x_j\} \cup Y$, by pruning item x_j , we prefer the more generic classification rules. A similar strategy was adopted in [7, 14].

Unpromising item elimination. Given the current prefix P , any one of its local frequent items, x_j ($1 \leq j \leq m$), any itemset Y that can be used to extend $P \cup \{x_j\}$ (where Y can be empty and $P \cup \{x_j\} \cup Y$ is frequent), and any class label c_i ($1 \leq i \leq k$), the following equation must hold:

$$conf_{P \cup \{x_j\} \cup Y}^{c_i} = \frac{sup_{P \cup \{x_j\} \cup Y}^{c_i}}{sup_{P \cup \{x_j\} \cup Y}} \leq \frac{sup_{P \cup \{x_j\} \cup Y}^{c_i}}{min_sup}$$

$$\leq \frac{sup_{P \cup \{x_j\}}^{c_i}}{min_sup}$$

Because $conf_{P \cup \{x_j\} \cup Y}^{c_i} \leq 1$ also holds, we have the following equation:

$$(3.4) \quad conf_{P \cup \{x_j\} \cup Y}^{c_i} \leq \min\{1, \frac{sup_{P \cup \{x_j\}}^{c_i}}{min_sup}\}$$

LEMMA 3.2. (*Unpromising item pruning*) For any conditional instance $\langle t_l, X_l, c_i \rangle \in TrDB|_{P \cup \{x_j\}}$ ($\forall l, 1 \leq l \leq |TrDB|_{P \cup \{x_j\}}|$, and $1 \leq i \leq k$), if the following always holds, item x_j is called an unpromising item and can be safely pruned.

$$(3.5) \quad HCCR_{t_l}^{conf} \geq \min\{1, \frac{sup_{P \cup \{x_j\}}^{c_i}}{min_sup}\}$$

Proof. By combining Equation 3.4 and Equation 3.5 we get that for any itemset Y (Y can be empty) the following must hold:

$$conf_{P \cup \{x_j\} \cup Y}^{c_i} \leq HCCR_{t_l}^{conf}$$

This means that any rule mined by growing prefix $P \cup \{x_j\}$ will have a confidence that is no greater than the current highest confidence covering rules (with the same rule head) of any conditional instance in $TrDB|_{P \cup \{x_j\}}$; thus, item x_j can be safely pruned. \square

Unpromising conditional database elimination. Given the current prefix P , any itemset Y (where Y can be empty and $P \cup Y$ is frequent), any class label c_i ($1 \leq i \leq k$), the confidence of rule ' $P \cup Y \rightarrow c_i$ ', $conf_{P \cup Y}^{c_i}$, must satisfy the following equation:

$$conf_{P \cup Y}^{c_i} = \frac{sup_{P \cup Y}^{c_i}}{sup_{P \cup Y}} \leq \frac{sup_{P \cup Y}^{c_i}}{min_sup} \leq \frac{sup_P^{c_i}}{min_sup}$$

In addition, because $conf_{P \cup Y}^{c_i} \leq 1$ also holds, we have the following equation:

$$(3.6) \quad conf_{P \cup Y}^{c_i} \leq \min\{1, \frac{sup_P^{c_i}}{min_sup}\}$$

LEMMA 3.3. (*Unpromising conditional database pruning*) For any conditional instance $\langle t_l, X_l, c_i \rangle \in TrDB|_P$ ($\forall l, 1 \leq l \leq |TrDB|_P|$, and $1 \leq i \leq k$), if the following always holds, the conditional database $TrDB|_P$ can be safely pruned.

$$(3.7) \quad HCCR_{t_l}^{conf} \geq \min\{1, \frac{sup_P^{c_i}}{min_sup}\}$$

Proof. By combining Equation 3.6 and Equation 3.7 we can get that for any itemset Y (Y can be empty) and $\forall l, 1 \leq l \leq |TrDB|_P$, $\langle t_l, X_l, c_i \rangle \in TrDB|_P$ ($1 \leq i \leq k$), the following must hold:

$$conf_{P \cup Y}^{c_i} \leq HCCR_{t_l}^{conf}$$

This means that any rule mined by growing prefix P will have a confidence that is no greater than the current highest confidence rules (with the same rule head) of any conditional instance in $TrDB|_P$; thus, the whole conditional database $TrDB|_P$ can be safely pruned. \square

ALGORITHM 1.1: RULEMINER($TrDB$, min_sup)

INPUT: (1) $TrDB$: a training database, and (2) min_sup : a minimum support threshold.

OUTPUT: (1) $HCCR$: the set of the highest confidence frequent covering rules w.r.t. each instance in $TrDB$.

01. for all $t_i \in TrDB$
 02. $HCCR_{t_i} \leftarrow \emptyset$;
 03. call **ruleminer**(\emptyset , $TrDB$).

SUBROUTINE 1.1 : **ruleminer**(pi , cdb)

INPUT: (1) pi : a prefix itemset, and (2) cdb : the conditional database w.r.t. prefix pi .

04. if($pi \neq \emptyset$)
 05. for all $\langle t_l, X_l, c_j \rangle \in cdb$
 06. if($HCCR_{t_l}^{conf} < \frac{sup_{pi}^{c_j}}{sup_{pi}}$)
 07. $HCCR_{t_l} \leftarrow \text{rule } 'pi \rightarrow c_j'$;
 08. $I \leftarrow \text{find_frequent_items}(cdb, min_sup)$;
 09. $S \leftarrow \text{support_equivalence_item_pruning}(I)$; $I \leftarrow I - S$;
 10. $S \leftarrow \text{unpromising_item_pruning}(I, cdb)$; $I \leftarrow I - S$;
 11. if($I \neq \emptyset$)
 12. if($\text{unpromising_conditional_database_pruning}(I, pi, cdb)$)
 13. return;
 14. $\text{correlation_coefficient_ascending_ordering}(I)$;
 15. for all $x \in I$ do
 16. $pi' \leftarrow pi \cup \{x\}$;
 17. $cdb' \leftarrow \text{build_cond_database}(pi', cdb)$;
 18. call **ruleminer**(pi' , cdb');

3.1.4 The Integrated Rule Mining Algorithm

The overall structure of the RULEMINER algorithm is shown in ALGORITHM 1.1. First, it initializes the highest confidence classification rules w.r.t. each training instance to empty (lines 01-02), then enumerates the classification rules by calling subroutine **ruleminer**(\emptyset , $TrDB$) (line 03). Subroutine **ruleminer**() takes as input a prefix itemset pi and its corresponding conditional database cdb . For each conditional instance, it checks if a classification rule with higher confidence can be computed from the current prefix pi , if so, it replaces the corresponding instance's current highest confidence rule with the new rule (lines 04-07). It then finds the frequent local items by scanning cdb (line 08), prunes invalid items based on the *support equivalence item pruning* method and the *unpromising item pruning* method (lines 09-10). If the set of valid local items

is empty or the whole conditional database cdb can be pruned based on the *unpromising conditional database pruning* method, it returns directly (lines 11-13). Otherwise, it sorts the left frequent local items according to the correlation coefficient ascending order (line 14), and grows the current prefix (line 16), builds the conditional database for the new prefix (line 17), and recursively calls itself to mine the highest confidence rules from the new prefix (line 18).

Discussion. The above RULEMINER() algorithm takes as input a uniform support threshold for all classes; however, it can be easily revised to take class-specific support thresholds as input. That is, the user can specify a support threshold for each class. This is sometimes beneficial for some imbalanced databases. However, due to limited space, we will do not elaborate the details and leave it to the interested readers.

ALGORITHM 1.2: BUILDMODEL($HCCR$)

INPUT: (1) $HCCR$: the set of the highest confidence covering rules.
 OUTPUT: (1) CM : the classification model (i.e., k groups of ranked rules).

01. $\text{Cluster_rules_into_k_groups}(HCCR)$; //according to class label
 02. for each group of rules
 03. $\text{Sort_rules}()$; //in confidence and support descending order

ALGORITHM 1.3: NEWINSTANCECLASSIFICATION(CM , ti)

INPUT: (1) CM : the classification model, (2) ti : a test instance.
 OUTPUT: (1) PCL : a predicted class label (or a set of class labels).

01. for $j=1$ to k // CM_j : the j -th group of rules in CM
 // SCR_j : the score for ti computed from CM_j
 02. $SCR_j \leftarrow \text{ComputeScore}(CM_j, ti)$;
 03. $PCL \leftarrow \text{PredictClassLabel}(SCR)$.

3.2 Building the Classification Model

After the set of the highest confidence covering rules have been mined, it will be straightforward to build the classification model. HARMONY first groups the set of the highest confidence covering rules into k groups according to their rule heads (i.e., class labels), where k is the total number of distinct class labels in the training database. Within the same group of rules, HARMONY sorts the rules in their confidence descending order, and for the rules with the same confidence, sorts them in support descending order. In this way, HARMONY prefers the rules with higher confidence, and the rules with higher support if the confidence is the same. The BUILDMODEL algorithm is shown in ALGORITHM 1.2.

3.3 Classifying a New Instance

After the classification model, CM , has been built, it can be used to classify a new test instance, ti , using the NEWINSTANCECLASSIFICATION algorithm shown in

ALGORITHM 1.3. HARMONY first computes a score w.r.t. ti for each group of rules in CM (lines 01-02), and predicts for ti a class label or a set of class labels if the underlying classification is a multi-class multi-label problem (i.e., each instance can be associated with several class labels).

Scoring function. In HARMONY, the score for a certain group of rules is defined in three different ways. The first scoring function is called *HIGHEST*, which computes the score as the highest confidence among the covering rules w.r.t. test instance ti (by a ‘covering rule’, we mean its rule body is a subset of ti). The second method is based on the *ALL* function. It is the default scoring function in HARMONY and computes the score as the sum of the confidences of all the covering rules w.r.t. ti . The third function is called *TOP-K*, where K is a user-specified parameter. It computes the score for a group of rules as the sum of the top K highest confidences of the covering rules w.r.t. ti . The *HIGHEST* and *ALL* functions can be thought of as two special cases of the *TOP-K* function when K is set at 1 and $+\infty$. For a multi-class single-label classification problem, HARMONY simply chooses the class label with the highest score as the predicted class label. While for a multi-class multi-label classification problem, the prediction is a little complicated.

Multi-class multi-label classification. In [5], the *dominant factor*-based method was proposed to predict the class labels for a multi-class multi-label classification problem and works as follows. Given a user-specified *dominant factor*, let the class label with the highest score be c_{max} and the corresponding highest score w.r.t. test instance ti be $SCORE_{ti}^{c_{max}}$, then any class label whose corresponding score is no smaller than $SCORE_{ti}^{c_{max}}$ is a predicted class label for ti . This method has been verified to be effective in practice [5]. However, in many imbalanced classification problems, the average confidence of each group of classification rules may be quite different from each other, this uniform *dominant factor*-based method will not work well. A large *dominant factor* may lead to low recalls (i.e., the percentage of the total test instances for the given class label that are correctly classified) for the classes with low average rule confidences, while a small *dominant factor* can lead to low precisions (i.e., the percentage of predicted instances for the given class label that are correctly classified) for the classes with high average rule confidences. To overcome this problem, HARMONY adopts a *weighted dominant factor*-based method. Let the average confidence of the group of classification rules w.r.t. class label c_k be $conf_{c_k}^{avg}$, the score w.r.t. instance ti and class label c_k

be $SCORE_{ti}^{c_k}$. Instance ti is predicted to belong to class c_k if it satisfies the equation:

$$SCORE_{ti}^{c_k} \geq SCORE_{ti}^{c_{max}} \left(\frac{conf_{c_k}^{avg}}{conf_{c_{max}}^{avg}} \right)^\delta$$

Here, δ ($\delta \geq 0$) is called the *score differential factor* and the larger the δ , the more the difference of the *weighted dominant factors* (i.e., $\left(\frac{conf_{c_k}^{avg}}{conf_{c_{max}}^{avg}} \right)^\delta$) among different class labels. It is set to 1 by default in HARMONY.

4 Related Work

There are two classes of algorithms that are directly related to this work. One is the traditional rule-induction-based methods and the other is the recently proposed association-rule-based methods. Both of these classes share the same idea of trying to find a set of classification rules to build the model. The rule-induction-based classifiers like C4.5 [26], FOIL [27], RIPPER [13], and CPAR [30] use various heuristics such as information gain (including Foil gain) and gini index to identify the best variable (or literal) by which to grow the current rule, and many of them follow a sequential database covering paradigm to speed up rule induction. The association-based classifiers adopt another approach to find the set of classification rules. They first use some efficient association rule mining algorithms to discover the complete (or a large intermediate) set of association rules, from which the final set of classification rules can be chosen based on various types of sequential database covering techniques. Some typical examples of association-based methods include CBA [25], CMAR [24], and ARC-BC [5].

In contrast to the rule-induction-based algorithms, HARMONY does not apply any heuristic pruning methods and the sequential database covering approach. Instead, it follows an instance-centric framework and mines the covering rules with the highest confidence for each instance, which can achieve better accuracy. At the same time, by maintaining one of the currently best rules for each training instance and pushing deeply several effective pruning methods into the projection-based frequent itemset mining framework [20, 1, 18], HARMONY directly mines the final set of classification rules, which avoids the time consuming rule generation and selection process used in several association-based classifiers [25, 24, 5].

The idea of directly mining a set of high confidence classification rules is similar to those in [7, 14]. Unlike these methods, HARMONY does not need the user to specify the minimum confidence and/or chi-square. Instead, it mines for each training instance one of the highest confidence frequent rules that it covers. In addition,

by maintaining one of the currently best classification rules for each instance, HARMONY is able to incorporate some new pruning methods under the unpromising item (or conditional database) pruning framework, which has been proven very effective in pushing deeply the length-decreasing support constraint or tough block constraints into closed itemset mining [28, 18].

Table 2: UCI database characteristics.

Database	# instances	# items	# classes
adult	48842	131	2
chess	28056	66	18
connect	67557	66	3
led7	3200	24	10
letRecog	20000	106	26
mushroom	8124	127	2
nursery	12960	32	5
pageBlocks	5473	55	5
penDigits	10992	90	10
waveform	5000	108	3

5 Empirical Results

5.1 Test Environment and Databases

We implemented HARMONY in C and performed the experiments on a 1.8GHz Linux machine with 1GB memory. We first evaluated HARMONY as a frequent itemset mining algorithm to show the effectiveness of the pruning methods, the algorithm efficiency and scalability. Then we compared HARMONY with some well-known classifiers on both categorical and text databases.

The UCI Databases. Many previous studies used some small databases to evaluate both the accuracy and efficiency of a classifier. For example, most of the 26 databases used in [25, 24, 30] only contain several hundred instances, which means the test databases contain too few test instances (i.e., only a few tens) if the 10-fold cross validation is adopted to evaluate the classification accuracy. In this paper, we mainly focus on relatively large databases. By large, we mean the database should contain no fewer than 1000 instances.

In [12], the author used 23 UCI databases to compare FOIL and CPAR algorithms. Among these 23 databases, 10 of them are large databases and will be used to compare HARMONY with FOIL, CPAR, and SVM⁴. The characteristics of these databases are summarized in Table 2. All the 10 databases were obtained from the author of [12] and the 10-fold cross validation is used for comparison with FOIL, CPAR, and SVM. Among these databases, *connect* is a too

⁴The numerical attributes in these databases have been discretized by the author of [12], and the discretization technique is different from those used in [25, 24, 30]; thus, the performance reported here may be different from the previous studies even for the same algorithm and the same database.

dense database, during the 10-fold cross validation in our experiments HARMONY only used the items whose supports are no greater than 20,000 to generate rules for this database.

Table 3: Top 10 topics in reuters-21578.

Category Name	# train labels	# test labels
acq	1650	719
corn	181	56
crude	389	189
earn	2877	1087
grain	433	149
interest	347	131
money-fx	538	179
ship	197	89
trade	369	118
wheat	212	71
total	7193	2787

Table 4: Class distribution in *sports* database.

Class Name	Number of labels
baseball	3412
basketball	1410
football	2346
hockey	809
boxing	122
bicycle	145
golf	336
total	8580

Text Databases. We also used two text databases in our empirical evaluation. The first database is the popularly used ‘ModeApte’ split version of the reuters-21578 collection, which was preprocessed and provided by the authors of [11], and both the database and its description are available at [10]. After preprocessing, it contains totally 8575 distinct terms, 9603 training documents, and 3299 test documents. Like many other studies [21, 16, 5, 11], we are more interested in the top 10 most common categories (i.e., topics). These ten largest categories form 6488 training documents and 2545 test documents. A small portion of the training and test documents are associated with multiple category labels (that is, *reuter-21578* is a multi-class multi-label database). In our experiments, we treated each one of the training documents with multiple labels as multiple documents, each one with a distinct label. The top 10 categories and their corresponding number of labels in the training and test databases are described in Table 3. The second text database is *sports*, which was obtained from San Jose Mercury (TREC). In our experiments, we removed some highly frequent terms, and finally it contains totally 8580 documents, 7 classes, and about 1748 distinct terms. The seven classes and their corresponding number of documents are shown in Table 4.

5.2 Experimental Results

5.2.1 Evaluate HARMONY as a Frequent Itemset Mining Algorithm

To mine the highest confidence covering rule(s) for each instance, a naïve method is like the association-based classifiers: first use an efficient association rule mining algorithm to compute the complete set of classification rules, from which the set of the highest confidence covering rules w.r.t. each instance can be selected. Our empirical results show that this method is usually inefficient if the database is large and a more efficient way is to push some effective pruning methods into the frequent itemset mining framework and to directly mine the final set of classification rules.

Ordering of the local items. In HARMONY, we provide three options for item ordering, that is, CoRelation coefficient Ascending order (denoted by CRA), Entropy Ascending order (denoted by EA), and Maximum Confidence Descending order (denoted by MCD). We first evaluated the effectiveness of these item ordering schemes against Support Descending order (denoted by SD) that is popularly used in frequent itemset mining. Our experiments have shown that the three newly proposed item ordering schemes are always more efficient than the support descending ordering scheme. In addition, these schemes also lead to slightly different classification accuracy. This is partly because different item ordering schemes may mine a different highest confidence covering rule w.r.t. a certain training instance, which may have different supports, although their confidence is the same. The experimental results also show that although the correlation coefficient ascending ordering scheme is not always the winner, on average it is more efficient and has better accuracy than all the other schemes. As a result, in HARMONY, it is chosen as the default option for item ordering. Table 5 shows the comparison result for the *sports* database at absolute support of 200. We can see that the correlation coefficient ascending ordering scheme is more efficient and also has slightly higher classification accuracy which was measured using 10-fold cross validation.

Table 5: Item ordering test on *sports* database.

Ordering scheme	CRA	EA	MCD	SD
Runtime(in seconds)	55.7	88.6	110.8	156.3
Accuracy(in %)	85.57	85.51	85.53	85.52

Effectiveness of the pruning methods. We also evaluated the effectiveness of the pruning methods. Figure 1a shows the results for database *penDigits* with absolute support threshold varying from 512 to

8. At first glance of Equation 3.5 and Equation 3.7, the *unpromising item* and *conditional database* pruning methods seem to be less effective at lower support, however this is not the case when considering more covering rules with higher confidence can be found at lower support and can be used to more quickly raise the currently maintained highest confidences. As we can see from Figure 1a, if we turn off the pruning methods used in HARMONY (denoted by ‘without pruning’), it can become over an order of magnitude slower at low support.

Scalability test. Figure 1b shows the scalability test result for databases *letRecog*, *waveform*, and *mushroom* with relative support set at 0.5%. In the experiments, we replicated the instances from 2 to 16 times. We can see that HARMONY has linear scalability in the runtime with increasing number of instances.

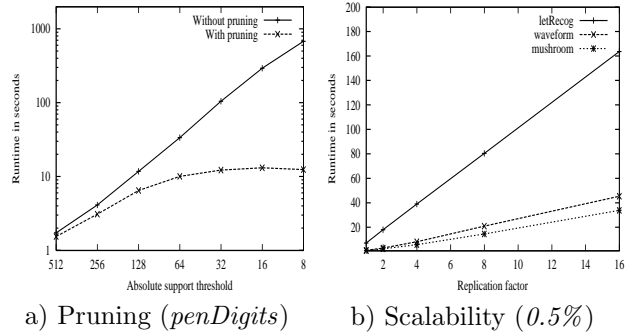


Figure 1: Pruning and scalability test.

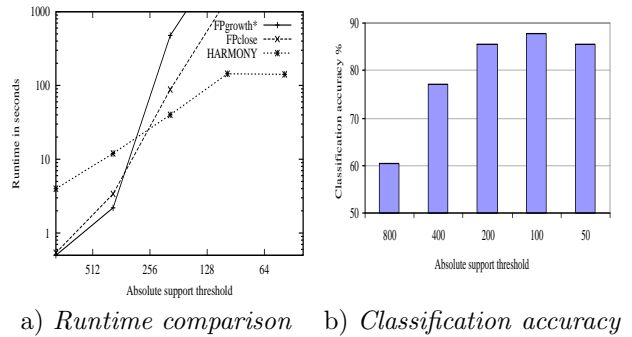


Figure 2: Efficiency test (*sports*).

Efficiency test. As we mentioned above, the traditional frequent (closed) itemset mining algorithms can be revised to mine the complete set of high confidence classification rules, from which a subset of high quality rules can be further identified. Our efficiency tests for HARMONY in comparison with FPgrowth* and FPclose, two recently developed efficient frequent/closed

Table 6: Breakeven performance on the *Reuters-21578* database with some well-known classifiers.

Categories	HARMONY <i>min_sup=60</i>	HARMONY <i>min_sup=70</i>	HARMONY <i>min_sup=80</i>	Find Similar	Naïve Bayes	Bayes Nets	Decision Trees	SVM (linear)
acq	95.3	95.3	95.3	64.7	87.8	88.3	89.7	93.6
corn	78.2	78.6	75.2	48.2	65.3	76.4	91.8	90.3
crude	85.7	85.0	88.0	70.1	79.5	79.6	85.0	88.9
earn	98.1	98.2	97.6	92.9	95.9	95.8	97.8	98.0
grain	91.8	90.4	90.1	67.5	78.8	81.4	85.0	94.6
interest	77.3	76.6	75.1	63.4	64.9	71.3	67.1	77.7
money-fx	80.5	81.9	82.1	46.7	56.6	58.8	66.2	74.5
ship	86.9	82.9	82.8	49.2	85.4	84.4	74.2	85.6
trade	88.4	88.0	86.1	65.1	63.9	69.0	72.5	75.9
wheat	62.8	60.6	58.7	68.9	69.7	82.7	92.5	91.8
micro-avg	92.0	91.7	91.4	64.6	81.5	85.0	88.4	92.0

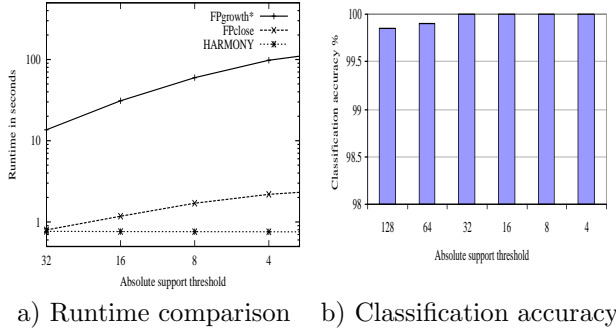


Figure 3: Efficiency test (*mushroom*).

itemset mining algorithms [19], show that such an approach is not realistic at low support, while our experiments demonstrate that the classification accuracy is usually higher at low support.

Figure 2 shows the comparison results for database *sports*. As we can see, although at high support, both FPgrowth* and FPClose are faster than HARMONY, once we continue to lower the support, they will be much slower. For example, at absolute support of 100, HARMONY is several orders of magnitude faster than FPgrowth* and FPClose. Figure 2b shows the classification accuracy at different support thresholds using the 10-fold cross validation. We can see that HARMONY can achieve higher accuracy at lower support like 100. It is also interesting to see that the accuracy at a too low support 50 is worse than that at support 100 for this database, due to the ‘overfitting’ problem.

Figure 3a shows similar comparison results for categorical database *mushroom*. HARMONY is faster than both FPgrowth* and FPClose at absolute support lower than 32. Figure 3b shows that HARMONY has better accuracy at low support threshold.

5.2.2 Classification Evaluation

The reuters-21578 (*ModApte*) text database. For a multi-class multi-label database like *reuters-*

21578, most previous studies used the breakeven point of precision and recall to measure the classifier performance [6, 21, 16, 29, 9, 5, 11], which is defined as the point at which precision is equal to the recall. To our best knowledge, the best breakeven performance for the *reuters-21578* database is the linear SVM [16]. For comparison with earlier results, we first found the overall breakeven point in terms of all top 10 categories by adjusting the dominant factor , then reported the average of precision and recall for each category as their corresponding breakeven performance [16].

Table 6 shows the comparison results with some previous results. The results for Find-Similar, Naïve-Bayes, Bayes-Nets, Decision-Trees, and Linear-SVM were obtained from [16]. The micro-avg is the overall breakeven performance over all 10 categories. For HARMONY, we used three different absolute support thresholds, 60, 70, and 80, respectively. From Table 6 we can see that both HARMONY and Linear-SVM have similar breakeven performance and perform much better than all the other classifiers. Among the 10 categories, HARMONY achieves the best performance at support of 60 for five categories, *acq*, *earn*, *money-fx*, *ship*, and *trade*. While Linear-SVM performs best for another three categories, *crude*, *grain*, and *interest*. Decision-Trees also performs good and has the best performance for two small categories, *corn* and *wheat*. SVM is very well known for classifying high dimensional text databases. Our results show that HARMONY can achieve similar performance to SVM.

The UCI databases. We evaluated HARMONY on the UCI databases in comparison with FOIL, CPAR, and SVM. FOIL and CPAR are two well-known algorithms for classifying categorical data. The results in [30] show that CPAR has better accuracy than c4.5 [26] and ripper [13], and has comparable accuracy to the association-based algorithms CMAR [24] and CBA [25], but is orders of magnitude faster; thus, we will do not compare HARMONY with c4.5, ripper, and the association-based algorithms. The results for FOIL and CPAR were provided by Frans Coenen and

are available at [12]. Because most databases we used contain more than two class labels, when comparing with SVM, we used $SVM^{multiclass}$ (Version: 1.01), which is an implementation of the multi-class Support Vector Machine and is available at http://www.cs.cornell.edu/People/tj/svm_light/svm_multiclass.html. In the experiments, we ran SVM with its default setting⁵. All the results including the accuracy and runtime are computed using the 10-fold cross validation. The reported accuracy is the corresponding average value of the 10-fold cross validation results, while the runtime is the total runtime of the 10-fold cross validation, including both training and testing time. In the experiments, we fixed the absolute support threshold at 50 for HARMONY with all 10 UCI databases.

Table 7: Accuracy comparison on 10 large UCI databases ($min_sup=50$ for HARMONY).

Database	FOIL	CPAR	SVM	HARMONY
adult	82.5	76.7	84.16	81.9
chess	42.6	32.8	29.83	44.87
connect	65.7	54.3	72.5	68.05
led7	62.3	71.2	73.78	74.56
letRecog	57.5	59.9	67.76	76.81
mushroom	99.5	98.8	99.67	99.94
nursery	91.3	78.5	91.35	92.83
pageBlocks	91.6	76.2	91.21	91.6
penDigits	88.0	83.0	93.2	96.23
waveform	75.6	75.4	83.16	80.46
average	75.66	70.68	78.663	80.725

Table 7 shows the accuracy comparison results, which reveal that HARMONY has much better overall accuracy than FOIL and CPAR, and has comparable accuracy with SVM. The average accuracy of HARMONY over all 10 UCI databases is about 5% higher than FOIL, 10% higher than CPAR, and 2% higher than SVM. SVM performs very well for the databases with few class labels, like *adult*, *connect*, and *waveform*, but has much worse accuracy than HARMONY for the databases with many class labels, like *chess* and *letRecog*. Compared with SVM, HARMONY has reasonably stable and good performance over all 10 UCI databases. Note in the experiments we fixed the minimum support at 50 for all 10 UCI databases. If we choose some tuned supports, HARMONY can achieve better performance than what we reported here for some databases. For example, if we choose the minimum support at 5 for the *chess* database, HARMONY has an accuracy of 58.43%, which is over 13% higher than the accuracy at support 50, while it only becomes about two times slower.

Table 8 compares the runtime (in seconds) of the four algorithms. Note that FOIL and CPAR were

Table 8: Runtime comparison on 10 large UCI databases ($min_sup=50$ for HARMONY).

Database	FOIL	CPAR	SVM	HARMONY
adult	10251.0	809.0	2493.1	1395.5
chess	10122.8	1736.0	13289.4	11.34
connect	35572.5	24047.1	74541.1	85.44
led7	11.5	5.7	17.12	1.29
letRecog	4365.6	764.0	17825.2	778.91
mushroom	38.3	15.4	16.6	8.78
nursery	73.1	51.7	322.4	6.21
pageBlocks	43.1	15.5	11.2	2.5
penDigits	821.1	101.9	512.7	82.6
waveform	295.3	38.1	36.2	130.0
total	61594.3	27584.4	109065.02	2502.57

implemented in java and were tested on a different machine from that of HARMONY and SVM. As a result, their runtime cannot be directly compared to those reported for HARMONY and SVM but they only provide an overall idea on the relative computational requirements of the various schemes. Table 8 shows that on average the runtime of HARMONY is over an order of magnitude smaller than those of FOIL, CPAR, and SVM. For some large databases like *chess*, the runtime of HARMONY can be over two orders of magnitude smaller than those of FOIL and CPAR, and over three orders of magnitude smaller than that of SVM.

Table 9: Test of scoring functions ($min_sup=50$)

Database	HIGHEST	TOP 3	TOP 5	ALL
adult	82.52	82.59	82.61	81.9
chess	43.06	40.44	37.81	44.87
connect	67.7	67.35	67.16	68.05
led7	72.98	73.34	71.2	74.56
letRecog	73.69	72.99	71.79	76.81
mushroom	99.95	99.95	99.95	99.94
nursery	94.62	93.98	93.72	92.83
pageBlocks	91.34	91.34	91.34	91.6
penDigits	94.49	94.24	93.93	96.23
waveform	78.82	78.82	79.52	80.46
average	79.917	79.504	79.513	80.725

Scoring function test. In the above classification evaluation, HARMONY adopted its default scoring function, *ALL*, to compute the score for a certain group of rules. In our experiments, we also evaluated the effectiveness of different scoring functions in HARMONY, including *ALL*, *HIGHEST*, and *TOP-K* (K was set at 3 and 5 in the experiments). The results w.r.t. the UCI databases are shown in Table 9. We see that the *ALL* function can achieve overall better accuracy than the other functions, while other functions can also have better accuracy for some databases. For example, the *HIGHEST* function achieves better accuracy for database *nursery*, and the *TOP-K* function can achieve better performance for database *adult*.

⁵We used its default linear kernel function.

6 Conclusion

Designing accurate, efficient, and scalable classifiers is an important research topic in data mining, and the rule-based classifiers have been proven very effective in classifying the categorical or high-dimensional sparse data. However, to achieve high accuracy, a good rule-based classifier needs to find a sufficient number of high quality classification rules and use them to build the model. In this paper, we proposed an instance-centric classification rule mining paradigm and designed an accurate classifier, HARMONY. Several effective search space pruning methods and search strategies have also been proposed, which can be pushed deeply into the rule discovery process. Our performance study shows that HARMONY has high accuracy and efficiency in comparison with many well known classifiers for both the categorical and high dimensional text data. It also has good scalability in terms of the database size.

Acknowledgements

We are grateful to Frans Coenen at the University of Liverpool and Shane Bergsma at the University of Alberta for providing us the discretized UCI databases and the reuters-21578 database, respectively, and promptly answering our various questions. We also thank Osmar R. Zaiane and Maria-Luiza Antonie at the University of Alberta for answering our questions related to the ARC-BC algorithm.

References

- [1] R. Agarwal, C. Aggarwal, V. Prasad. *A Tree Projection Algorithm for Generation of Frequent Item Sets*, Journal of Parallel and Distributed Computing. 61(3), 2001.
- [2] R. Agrawal, T. Imielinski, A. Swami. *Mining Association Rules between Sets of Items in Large Databases*, SIGMOD'93.
- [3] R. Agrawal, R. Srikant. *Fast Algorithms for Mining Association Rules*, VLDB'94.
- [4] K. Ali, S. Manganaris, R. Srikant. *Partial Classification Using Association Rules*, KDD'97.
- [5] M. Antonie, O. Zaiane. *Text Document Categorization by Term Association*, ICDM'02.
- [6] C. Apte, F. Damerau, S.M. Weiss. *Towards Language Independent Automated Learning of Text Categorization Models*, SIGIR'94.
- [7] R.J. Bayardo. *Brute-force Mining of High-confidence Classification rules*, KDD'97.
- [8] R.J. Bayardo, R. Agrawal. *Mining the most interesting rules*, KDD'99.
- [9] R. Bekkerman, R. El-Yaniv, N. Tishby, Y. Winter. *On Feature Distribution Clustering for Text Categorization*, SIGIR'01.
- [10] S. Bergsma. *The Reuters-21578 (ModApte) dataset*, Department of Computer Science, University of Alberta. Available at <http://www.cs.ualberta.ca/~bergsma/650/>.
- [11] S. Bergsma, D. Lin. *Title Similarity-Based Feature Weighting for Text Categorization*, CMPUT 650 Research Project Report, Department of Computer Science, University of Alberta.
- [12] F. Coenen. (2004) The LUCS-KDD Implementations of the FOIL, PRM, and CPAR algorithms, http://www.csc.liv.ac.uk/~frans/KDD/Software/FOIL_PRM_CPAR/foilPrmCpar.html, Computer Science Department, University of Liverpool, UK.
- [13] W. Cohen. *Fast effective rule induction*, ICML'95.
- [14] G. Cong, X. Xu, F. Pan, A. Tung, J. Yang. *FARMER: Finding Interesting Rule Groups in Microarray Datasets*, SIGMOD'04.
- [15] M. Deshpande, G. Karypis. *Using Conjunction of Attribute Values for Classification*, CIKM'02.
- [16] S. Dumais, J. Platt, D. Heckerman, M. Sahami. *Inductive Learning Algorithms and Representations for Text Categorization*, CIKM'98.
- [17] T. Fukuda, Y. Morimoto, S. Motishita. *Constructing Efficient Decision Trees by Using Optimized Numeric Association Rules*, VLDB'96.
- [18] K. Gade, J. Wang, G. Karypis. *Efficient Closed Pattern Mining in the Presence of Tough Block Constraints*, to appear in KDD'04.
- [19] G. Grahne, J. Zhu. *Efficiently Using Prefix-trees in Mining Frequent Itemsets*, ICDM-FIMI'03.
- [20] J. Han, J. Pei, Y. Yin. *Mining Frequent Patterns without Candidate Generation*, SIGMOD'00.
- [21] T. Joachims. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*, ECML'98.
- [22] B. Lent, A. Swami, J. Widom. *Clustering Association Rules*, ICDE'97.
- [23] N. Lesh, M. Zaki, M. Ogihara. *Mining Features for Sequence Classification*, KDD'99.
- [24] W. Li, J. Han, J. Pei. *CMAR: Accurate and Efficient Classification based on multiple class-association rules*, ICDM'01.
- [25] B. Liu, W. Hsu, Y. Ma. *Integrating Classification and Association Rule Mining*, KDD'98.
- [26] J. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1993.
- [27] J. Quinlan, R. Cameron-Jones. *FOIL: A Midterm Report*, ECML'93.
- [28] J. Wang, G. Karypis. *BAMBOO: Accelerating Closed Itemset Mining by Deeply Pushing the Length-Decreasing Support Constraint*, SDM'04.
- [29] Y. Yang. *An Evaluation of Statistical Approaches to Text Categorization*, Information Retrieval, Vol. 1, No. 1-2, 1999.
- [30] X. Yin, J. Han. *CPAR: Classification based on Predictive Association Rules*, SDM'03.
- [31] M. Zaki, C. Aggarwal. *XRULES: An Effective Structural Classifier for XML Data*, KDD'03.

On Error Correlation and Accuracy of Nearest Neighbor Ensemble Classifiers

Carlotta Domeniconi and Bojun Yan

Information and Software Engineering Department

George Mason University

carlotta@ise.gmu.edu byan@gmu.edu

Abstract

Recent empirical work has shown that combining predictors can lead to significant reduction in generalization error. Unfortunately, many combining methods do not improve nearest neighbor (NN) classifiers at all. This is because NN methods are very robust with respect to variations of a data set. In contrast, they are sensitive to input features. We exploit the instability of NN classifiers with respect to different choices of features to generate an effective and diverse set of NN classifiers. Interestingly, the approach takes advantage of the high dimensionality of the data. We investigate techniques to decorrelate errors while keeping the individual classifiers accurate. We analyze the results both in terms of error rates and error correlations. The experimental results show that our technique can offer significant performance improvements with respect to competitive methods.

1 Introduction

An ensemble of classifiers succeeds in improving the accuracy of the whole when the component classifiers are both diverse and accurate. Diversity is required to ensure that the classifiers make uncorrelated errors. If each classifier makes the same error, the voting carries that error into the decision of the ensemble, thereby gaining no improvement. In addition, accuracy is required to avoid poor classifiers to obtain the majority of votes. These requirements have been quantified. Under simple voting and error independency conditions, if all classifiers have the same probability of error, and such probability is less than 50%, then the error of the ensemble decreases monotonically with an increasing number of classifiers [15, 2].

One way to generate an ensemble with the required properties is to train the classifiers on different sets of data, obtained by sampling from the original training set [6, 18, 13, 8]. Breiman's *bagging* [6] and Freund and Schapire's *boosting* [13] are well known examples of successful iterative methods for improving the predictive power of classifier learning systems. Bagging uses sampling with replacement. It generates multiple classifiers by producing replicated samples of the data. To classify

an instance, a vote for each class j is recorded by every classifier that chooses it, and the class with the most votes is chosen by the aggregating scheme. Boosting uses adaptive sampling. It uses all instances at each repetition, but maintains a weight for each instance in the training set that reflects its importance as a function of the errors made by previously generated hypotheses. As for bagging, boosting combines the multiple classifiers by voting, but unlike bagging boosting assigns different voting strengths to component classifiers on the basis of their accuracy.

Experimental evidence [21] proved that both bagging and boosting are quite effective in reducing generalization error, with boosting providing in general higher improvements. Dramatic error reductions have been observed with decision trees such as CART and C4.5 [6, 21, 13]. This behavior can be explained in terms of the bias-variance components of the generalization error [7]. The variance component measures the scatter in the predictions obtained from using different training sets, each one drawn from the same distribution. The effect of combination is to reduce the variance, that is what both bagging and boosting achieve. In addition, boosting does something more. By concentrating the attention of the weak learner on the harder examples, it challenges the weak learner algorithm to perform well on these harder parts of the sample space, thereby reducing the bias of the learning algorithm.

It turns out that sampling the training set is not effective with NN classifiers [6]. To gain some insights as to why this is the case, let us analyse the conditions under which the bagging procedure is effective. As observed above, bagging reduces the variance component of the generalization error. When the weak learner is unstable with respect to variations in the training set, perturbing the training data can cause significant variability in the resulting predictor. Thus, bagging the ensemble improves accuracy in this case. Suppose the weak learner is the NN classifier. It has been shown that

the probability that any given training point is included in a data set bootstrapped by bagging is approximately 63.2% [6]. It follows that the nearest neighbor will be the same in 63.2% of the nearest neighbor classifiers. Thus, errors are highly correlated, and bagging becomes ineffective.

The fact that NN methods are very robust with respect to variations of the data set makes ensemble methods ineffective. In contrast, NN methods are sensitive to input features. In this paper we exploit such instability of NN classifiers with respect to different choices of features, to generate an effective and diverse set of NN classifiers. We explore the challenge of producing NN classifiers with decorrelated errors which are, at the same time, accurate. In general, the balance where the gains due to decreased correlations outweigh the losses due to reduced information available to individual classifiers must be found to provide the best results.

2 Ensemble of Nearest Neighbors in Weight-Driven Subspaces

As discussed above, kNN methods are very robust with respect to variations of the data set. The stability of nearest neighbor classifiers to variations in the training set makes ensemble methods obtained by bootstrapping the data ineffective. In contrast, kNN techniques are sensitive to features (i.e., intolerant to irrelevant features) [19], and to the chosen distance function [14, 16, 12, 11]. As such, in order to achieve diversity and accuracy with nearest neighbor classifiers, we ought to sample the feature space, to which the kNN method is highly sensitive. The idea is then to exploit the instability of NN classifiers with respect to different choices of features to generate a diverse set of NN classifiers with (possibly) uncorrelated errors.

In [3], the outputs of multiple nearest neighbor classifiers, each having access only to a random subset of features, are combined using simple voting. In [17], instead, the class membership decision is delayed until the aggregation phase. It is shown [3] that random feature selection can increase the diversity without increasing the error rates. This fact results in accuracy improvements on a variety of data sets. However, as also pointed out in [3], the technique has some major drawbacks that cause the degradation in performance observed in some cases. While the *random* selection of features is likely to increase diversity among the classifiers, it gives no guarantee that the selected features carry the necessary *discriminant* information. If they don't, poor classifiers will be generated, and the voting will increase the generalization error.

To reduce the risk of discarding discriminant infor-

mation, while preserving a reasonable degree of diversity, we propose to perform *adaptive* sampling over the feature space. In particular, in order to keep the bias of individual classifiers low, we use feature relevance to guide the sampling mechanism. This process has the potential of producing accurate classifiers in disagreement with each other. While it is expected that the level of diversity obtained by this adaptive mechanism may be lower than the diversity given by random sampling, the higher accuracy of the individual classifiers should allow the ensemble to improve performance. It is interesting to observe that, since the method uses subsets of features, it will be effective for problems with a large number of dimensions, which is often the case for many applications. Although it defies common sense, sampling in feature space takes advantage of the high dimensionality of the data. The experimental results we present support this conjecture.

2.1 Learning Feature Weights. In this work we use the ADAMENN algorithm to estimate feature relevance, and therefore the corresponding weight vector [12], at any given test point. Other techniques can be considered as well [14, 16, 11]. For completeness, we provide here a brief description of the ADAMENN algorithm. ADAMENN performs a *Chi-squared* distance analysis to compute a flexible metric for producing neighborhoods that are highly adaptive to query locations. Let \mathbf{x} be the nearest neighbor of a query \mathbf{x}_0 computed according to a distance metric $D(\mathbf{x}, \mathbf{x}_0)$. The goal is to find a metric $D(\mathbf{x}, \mathbf{x}_0)$ that minimizes $E[r(\mathbf{x}_0, \mathbf{x})]$, where

$$(2.1) \quad r(\mathbf{x}_0, \mathbf{x}) = \sum_{j=1}^J P(j|\mathbf{x}_0)(1 - P(j|\mathbf{x})).$$

Here $P(j|\mathbf{x})$ is the class conditional probability at \mathbf{x} . That is, $r(\mathbf{x}_0, \mathbf{x})$ is the finite sample error risk given that the nearest neighbor to \mathbf{x}_0 by the chosen metric is \mathbf{x} . It can be shown [12] that the weighted *Chi-squared* distance

$$(2.2) \quad D(\mathbf{x}, \mathbf{x}_0) = \sum_{j=1}^J \frac{[P(j|\mathbf{x}) - P(j|\mathbf{x}_0)]^2}{P(j|\mathbf{x}_0)}$$

approximates the desired metric, thus providing the foundation upon which the ADAMENN algorithm computes a measure of local feature relevance, as shown below.

We first notice that $P(j|\mathbf{x})$ is a function of \mathbf{x} . Therefore, we can compute the conditional expectation of $P(j|\mathbf{x})$, denoted by $\bar{P}(j|x_i = z)$, given that x_i assumes value z , where x_i represents the i th component

of \mathbf{x} . That is,

$$\begin{aligned}\overline{P}(j|x_i = z) &= E[P(j|\mathbf{x})|x_i = z] \\ &= \int P(j|\mathbf{x})p(\mathbf{x}|x_i = z)d\mathbf{x}.\end{aligned}$$

Here $p(\mathbf{x}|x_i = z)$ is the conditional density of the other input variables defined as

$$p(\mathbf{x}|x_i = z) = p(\mathbf{x})\delta(x_i - z) / \int p(\mathbf{x})\delta(x_i - z)d\mathbf{x},$$

here $\delta(x - z)$ is the Dirac delta function having the properties $\delta(x - z) = 0$ if $x \neq z$ and $\int_{-\infty}^{\infty} \delta(x - z)dx = 1$. Let

$$(2.3) \quad r_i(\mathbf{z}) = \sum_{j=1}^J \frac{[P(j|\mathbf{z}) - \overline{P}(j|x_i = z_i)]^2}{\overline{P}(j|x_i = z_i)}.$$

$r_i(\mathbf{z})$ represents the ability of feature i to predict the $P(j|\mathbf{z})$ s at $x_i = z_i$. The closer $\overline{P}(j|x_i = z_i)$ is to $P(j|\mathbf{z})$, the more information feature i carries for predicting the class posterior probabilities locally at \mathbf{z} .

We can now define a measure of feature relevance for \mathbf{x}_0 as

$$(2.4) \quad \bar{r}_i(\mathbf{x}_0) = \frac{1}{K_0} \sum_{\mathbf{z} \in N(\mathbf{x}_0)} r_i(\mathbf{z}),$$

where $N(\mathbf{x}_0)$ denotes the neighborhood of \mathbf{x}_0 containing the K_0 nearest training points, according to a given metric. \bar{r}_i measures how well on average the class posterior probabilities can be approximated along input feature i within a local neighborhood of \mathbf{x}_0 . Small \bar{r}_i implies that the class posterior probabilities will be well captured along dimension i in the vicinity of \mathbf{x}_0 . Note that $\bar{r}_i(\mathbf{x}_0)$ is a function of both the test point \mathbf{x}_0 and the dimension i , thereby making $\bar{r}_i(\mathbf{x}_0)$ a local relevance measure.

To formulate the measure of feature relevance as a weighting scheme, we first define

$$R_i(\mathbf{x}_0) = \max_j \{\bar{r}_j(\mathbf{x}_0)\} - \bar{r}_i(\mathbf{x}_0),$$

i.e., the more relevant dimension i is, the larger R_i becomes. An exponential weighting scheme is then given by

$$w_i(\mathbf{x}_0) = \exp(R_i(\mathbf{x}_0)) / \sum_{l=1}^q \exp(R_l(\mathbf{x}_0)).$$

Such weights are real values between 0 and 1, and their sum equals 1. Therefore, they define a probability distribution over the feature space that can be employed in our adaptive sampling mechanism. In addition, the

exponential weighting scheme avoids zero values. As such, for each test point and each classifier of the ensemble, any given feature has a non zero probability to be selected. This property guarantees a certain level of diversity among the classifiers of our ensemble. For the details on how to estimate the unknown quantities involved in the feature relevance measure, see [12]. We point out that ADAMENN outperforms decision trees, and other well known locally adaptive classifiers on a variety of data sets [12]. In addition, it has shown accuracy results similar to support vector machines in a variety of cases. Thus, being able to improve upon its performance is a significant objective to achieve.

2.2 Ensemble Algorithm. The general formulation of our approach is as follows:

Input: Number-Of-Classifiers (*NoC*), Number-Of-Features (*NoF*), k , test point \mathbf{x}_0 ;

Compute the weight vector \mathbf{w}_0 reflecting feature relevance at \mathbf{x}_0 (e.g., using the ADAMENN algorithm);

- For 1 to *NoC*:

1. Sample *NoF* features *with or without replacement*, according to the probability distribution given by the weight vector \mathbf{w}_0 ;
2. Use selected features (*Self*) only (and their weights) to compute the k closest neighbors, according to the weighted Euclidean distance: $D(\mathbf{x}_0, \mathbf{y}) = \sqrt{\sum_{i \in \text{Self}} w_{0i}(x_{0i} - y_i)^2}$;
3. Classify test point using kNN rule;

- Apply the voting scheme in use among the *NoC* classifiers.

Output: Decision of the ensemble.

The algorithm has three input parameters: The Number-Of-Classifiers to combine, the Number-Of-features to be selected, and the size k of the neighborhoods. The values of these parameters can be determined based on cross-validation accuracy estimated on the training set for the whole ensemble. When sampling with replacement is used, if a feature is selected more than once, say t times, its weight is multiplied by a factor t for distance computation.

3 Voting Methods

The classifiers can be combined using a simple majority voting. We also investigate an alternative mechanism to combine the classifiers. Instead of computing the most frequent class label within the neighborhood of the test

point, we keep all estimated class posterior probabilities. That is, for each classifier, all class labels of the k nearest neighbors are recorded. After NoC iterations, the test point is assigned to the class that has the most frequent occurrence. This voting scheme selects the class with the largest expected posterior probability in the ensemble. As such, it takes into account not only the “winner” of each classifier, but also the margin of the win. The class with the largest overall margin will be selected by the ensemble.

As a simple example, suppose we have three classifiers and two classes (positive and negative). For a given test point \mathbf{x}_0 , the recorded labels of its five nearest neighbors ($k = 5$) are as follows: Classifier 1: 2 positives and 3 negatives; Classifier 2: 2 positives and 3 negatives; Classifier 3: 4 positives and 1 negative. The expected class posterior probabilities, estimated by the ensemble, are: $E[P(+|\mathbf{x}_0)] = 8/15$ and $E[P(-|\mathbf{x}_0)] = 7/15$. Thus, the ensemble chooses the positive class. Note that, although the negative class is the winner for the majority of classifiers, its overall margin of win is two. The positive class wins only once but with a (larger) margin of three. Simple voting predicts a negative label in this example.

In addition, we consider the Borda Count method [10]. It is a positional-scoring technique: each candidate class gets 0 points for each last place vote received, 1 point for each next-to-last point vote, and so on up to $C - 1$ points for each first place vote (where C is the number of classes). The candidate class with the largest point total wins the election. When $C = 2$, the Borda Count method reduces to a simple majority voting technique. It is often used for polls which rank sport teams or academic institutions.

4 Experimental Results

We have conducted experiments to compare the accuracy and diversity of Random and Weight-Driven feature subspace methods. Both sampling *with* and *without replacement* have been used. The three voting schemes described above (*Simple*, *Counting*, and *Borda*) were used to compute the decision of the ensemble ($NoC = 200$ classifiers). Tables 1-2 show the error rates and standard deviations obtained on five data sets [5]. We also report the error rates of ADAMENN and kNN using Euclidean distance. The characteristics of each data set (number of dimensions, number of data (N), and number of classes (C)) are given in parenthesis. Leave-one-out cross-validation was used to generate training and test data in each classifier. We have tested values of k between 1 and 5; for NoF we considered values from 1 (or higher, for data with a larger dimensionality) to the total number of dimensions. For each

combination of parameter values, the experiment was repeated 10 times, and the average error rate was computed. For all methods compared, validation of parameters was performed over the training data. Tables 3-4 specify the parameter values for the error rates given in Tables 1-2. For each data set and each technique, Tables 3-4 show: the value of k , if sampling with (1) or without (0) replacement was performed, and the number of selected features (NoF).

The results show that our Weight-driven approach offers significant accuracy improvements (over both ADAMENN and the Random approach) for the three data sets with a larger number of dimensions (*spectf-test*, *lung*, *sonar*). For *liver* and *ionosphere* the Random and Weight approaches give similar performances. This result provides evidence that bootstrapping features using an “intelligent” distance metric (Weight-Driven method) takes advantage of the high dimensionality of the data. Thus, it provides an effective method to dodge the curse-of-dimensionality phenomenon. Figures 1-2-3-4-5 plot the error rate as a function of the number of selected features (NoF) for all five data sets considered here. For the Weight-driven technique, and for *ionosphere* and *lung* data, the largest values of NoF are 23 and 50, respectively (see Figures 2-4). This is because eleven and four features, respectively for the first and second data set, received very small weights which were approximated to zero (thus, were never selected). The plots show the robustness of the Weight-driven approach as the number of selected features increase. On the contrary, the error rate of the Random approach can be quite sensitive to the value of the NoF parameter. In particular, the results for the *ionosphere*, *spectf-test*, and *sonar* data clearly demonstrate the drawback of the Random approach: as the fraction of selected features *not* carrying discriminant information increases, poor classifiers are generated, and the voting increases the generalization error.

In some cases the Counting voting method improves performance (with respect to Simple). The Borda technique on *lung* data gives the best result for both Random and Weight-driven algorithms. (Note that we test the Borda voting method only on *lung* data since the other data sets all involve two classes. In such case, the Borda count method reduces to simple voting.) In most cases, sampling without replacement outperformed sampling with replacement (see parameter values in Tables 3-4).

4.1 Measure of Diversity and Accuracy To measure both the accuracy and the diversity of the classifiers, we make use of the Kappa statistic, κ [9, 20]. In particular, a *Kappa-Error* diagram [20] allows us to visualize the diversity and the accuracy of an ensemble

of classifiers. A Kappa-Error diagram is a scatterplot where each point corresponds to a pair of classifiers. The x coordinate is the value of κ for the two classifiers. Smaller κ values indicate a larger diversity: $\kappa = 0$ when the agreement of the two classifiers equals that expected by chance, and $\kappa = 1$ when the two classifiers agree on every example. The y coordinate is the average error rate of the two classifiers.

We report the Kappa-Error diagrams for the five data sets in Figures 6-7-8-9-10. As expected, the level of diversity obtained with the Weight-driven technique is in general lower than the diversity given by the Random approach. However, the “intelligent” metric employed by the Weight-driven technique allows to reduce bias, and thus achieve a better error rate.

4.2 Reduction of Error Correlations In light of the results provided by the Kappa statistic, we explore the possibility of decorrelating errors by introducing new elements of diversification among the NN classifiers. The benefits and pitfalls of reducing the correlation among classifiers, especially when the training data are limited, are well discussed in [22]. Here we face the challenge of reaching a trade-off between error decorrelation and accuracy in the context of NN classifiers.

We first allow each classifier to customize the number of selected features at each query point \mathbf{x}_0 . This is achieved as follows. We sort the weight components of \mathbf{w}_0 in non increasing order: w_{01}, \dots, w_{0q} (q is the number of dimensions). The classifier chooses NoF_0 (number of selected features at \mathbf{x}_0) to be such that

$$\sum_{i=1}^{NoF_0} w_{0i} \leq f, \text{ and } \sum_{i=1}^{NoF_0+1} w_{0i} > f,$$

where $f \in (0, 1)$ is an input parameter. Basically, f is the fraction of the total weight (which is always one) captured by NoF_0 . The fewer the relevant features, the smaller NoF_0 . The actual features are then selected as before, by sampling according to the probability distribution given by the weight vector \mathbf{w}_0 . We have experimented with the following values of f : 0.2, 0.4, 0.6, 0.8, and 0.9. The error rates corresponding to the cross-validated values of f are shown in Tables 5-6. We observe that the error rates get worst (considerably for lung data), except for *liver* and Weight (Counting). We noticed that these error rates all correspond to $f = 0.9$ (interestingly, except for *liver* and Weight (Counting) where $f = 0.6$). This suggests that the classifiers are highly correlated (as confirmed by the correlation measurements given below). For *liver* data, the value $f = 0.6$ offers the best trade-off between diversity and accuracy. For *lung* data, we observed that 13 points

(out of 32) used less than five features. This may have affected the results, since previously the combination of seven or five features gave the best error rates.

In an effort to decorrelate the errors, we have also generated ensembles of a mixture of Random and Weight-driven classifiers. (Two percentage combinations were tested: 50% of each kind; 60% Weight-driven and 40% Random.) The resulting error rates are given in Tables 7-8. Simple voting was used to combine the classification results. As values of the parameters for each kind of classifier, we used the cross-validated ones given in Tables 3-4. Only for *sonar* data a better error rate with respect to both previous error rates of Random and Weight was achieved.

To analyze these results, we have measured the correlation of errors of two classifiers (indexed by 1 and 2) on each class i :

$$\delta_{1,2}^i = \frac{\text{cov}(\eta_1^i(\mathbf{x}), \eta_2^i(\mathbf{x}))}{\sigma_{\eta_1^i} \sigma_{\eta_2^i}},$$

where $\eta_j^i(\mathbf{x})$ is the error value (0 or 1) on $\mathbf{x} \in C_i$ of classifier j , and $\sigma_{\eta_j^i}$ is the standard deviation of $\eta_j^i(\mathbf{x})$, computed over all $\mathbf{x} \in C_i$. To account for all classes:

$$\delta_{1,2} = \sum_{i=1}^C \delta_{1,2}^i P(i),$$

where $P(i)$ is the prior probability of class i . In our experiments we consider equal priors, and thus

$$\delta_{1,2} = 1/C \sum_{i=1}^C \delta_{1,2}^i$$

gives the total error correlation between classifiers 1 and 2.

Sample results for *liver* and *sonar* are given in Tables 9-10. For each data set we summarize the average error correlation values computed between five classifiers. We also report the corresponding error rates of the ensembles. In each case simple voting is used. Weight-C is for Weight-driven with customized number of features. As expected, the Random approach gives very low correlation values. Decreasing the value of f for the Weight-C method, clearly decreases the correlation of errors. Though, in most cases, the gains due to the decreased correlations is outweighed by the loss of information due to the reduced number of features retained. The mixture method is effective in decreasing correlations, though it didn't offer significant accuracy improvements with respect to both the Random and Weight approach alone. Nevertheless, this analysis suggests that combining mixture of different classifiers

Table 1: Average error rates.

	<i>liver</i>	<i>ionosphere</i>	<i>spectf-test</i>
(<i>dim-N-C</i>)	(6-345-2)	(34-351-2)	(44-267-2)
kNN	32.5	13.7	23.6
ADAMENN	30.7	7.1	19.1
Random (Simple)	29.4 (0.5)	5.8 (0.2)	20.2 (0.4)
Random (Counting)	28.6 (0.5)	5.7 (0.2)	19.9 (0.4)
Weight (Simple)	29.3 (0.5)	6.3 (0.2)	17.6 (0.4)
Weight (Counting)	29.9 (0.5)	6.3 (0.2)	17.7 (0.4)

Table 2: Average error rates.

	<i>lung</i>	<i>sonar</i>
(<i>dim-N-C</i>)	(54-32-3)	(60-208-2)
kNN	50.0	12.5
ADAMENN	37.5	9.1
Random (Simple)	45.0 (0.5)	10.5 (0.3)
Random (Counting)	45.3 (0.5)	10.3 (0.3)
Random (Borda)	44.7 (0.5)	-
Weight (Simple)	35.0 (0.5)	8.3 (0.3)
Weight (Counting)	32.5 (0.5)	8.3 (0.3)
Weight (Borda)	30.9 (0.5)	-

(e.g., each kind performing a different form of adaptive sampling) may offer error rate improvements as well. In our future work we plan to consider other adaptive feature sampling mechanisms, and random projection as well.

5 Conclusions

We have introduced a mechanism to generate an effective and diverse ensemble of NN classifiers. Our analysis shows the difficulty of reaching a good balance between error decorrelation and accuracy in the context of NN classifiers. In an effort to further reduce correlations without increasing error rates, we will consider multiple adaptive mechanisms to sampling in feature space.

Table 3: Parameter values for error rates in Table 1: k - with (1) or without (0) replacement - NoF.

	<i>liver</i>	<i>ionosphere</i>	<i>spectf-test</i>
Random (Simple)	5-1-3	1-0-5	1-0-5
Random (Counting)	3-1-4	1-0-5	1-0-5
Weight (Simple)	5-1-5	1-0-8	5-0-4
Weight (Counting)	3-1-6	1-0-7	1-0-7

Table 4: Parameter values for error rates in Table 2: k - with (1) or without (0) replacement - NoF.

	<i>lung</i>	<i>sonar</i>
Random (Simple)	5-1-54	1-0-23
Random (Counting)	3-0-49	1-1-29
Random (Borda)	3-0-49	-
Weight (Simple)	3-0-7	1-0-45
Weight (Counting)	3-0-5	1-0-45
Weight (Borda)	3-0-5	-

Table 5: Weight-driven approach with customized number of features: Average error rates.

	<i>liver</i>	<i>ionosphere</i>	<i>spectf-test</i>
Weight (Simple)	30.3 (0.5)	8.3 (0.3)	18.7 (0.4)
Weight (Counting)	29.1 (0.5)	8.3 (0.3)	18.9 (0.4)

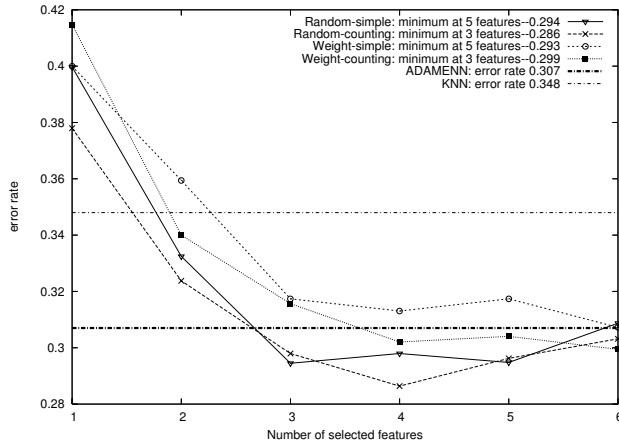


Figure 1: Liver data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

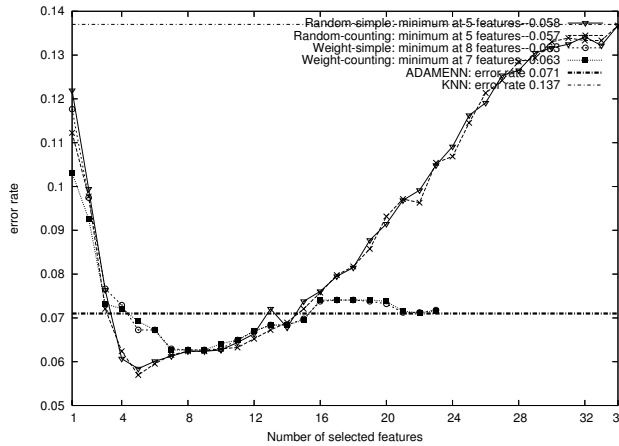


Figure 2: Ionosphere data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

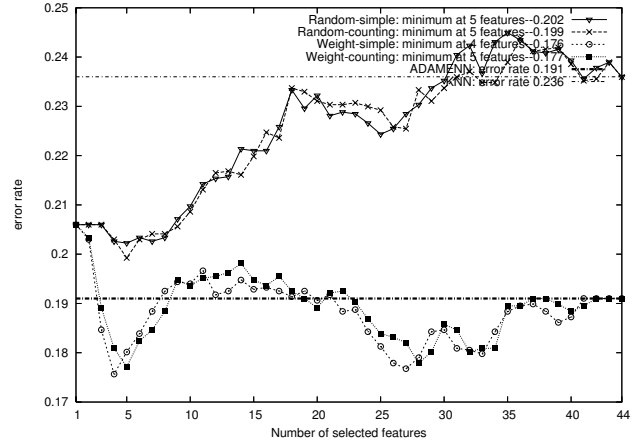


Figure 3: Spectf data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

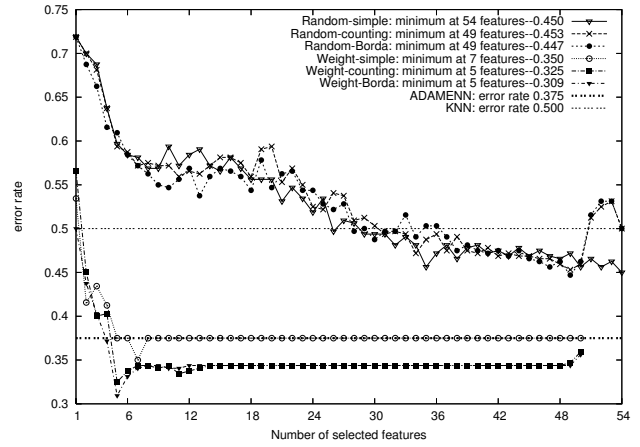


Figure 4: Lung data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

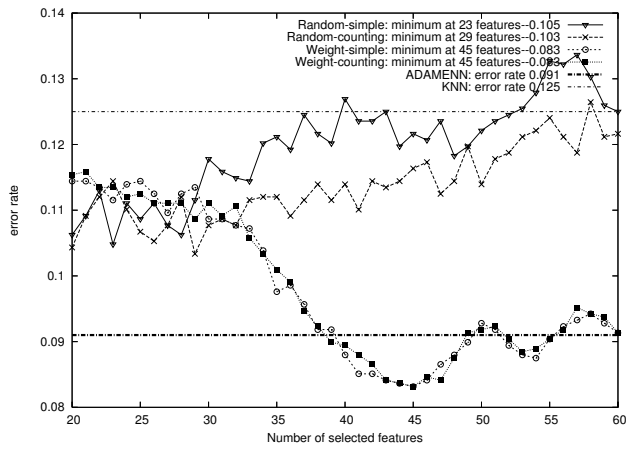


Figure 5: Sonar data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

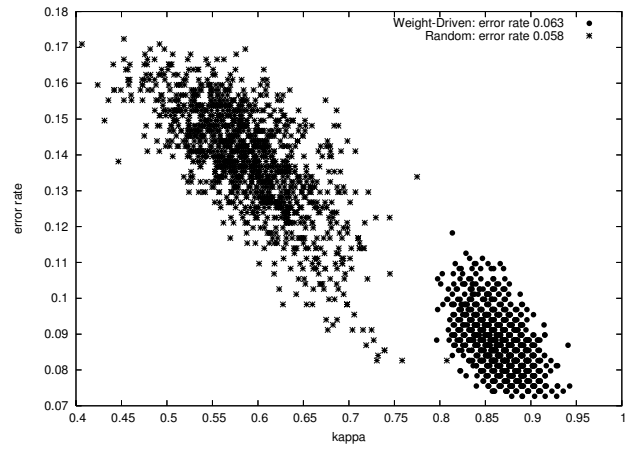


Figure 8: Ionosphere data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

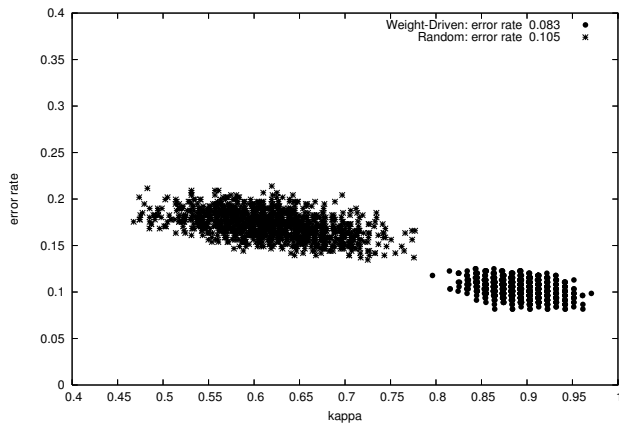


Figure 6: Sonar data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

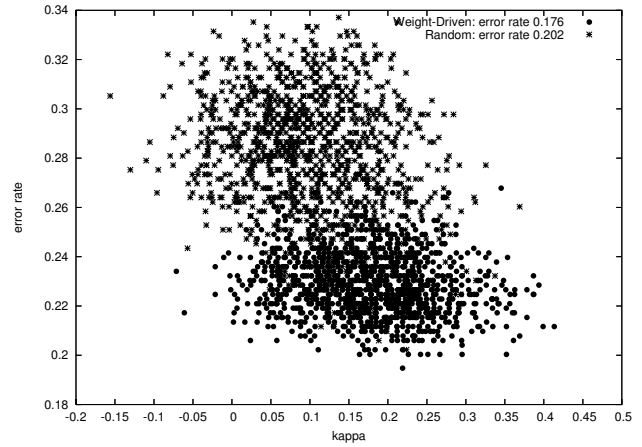


Figure 9: Spectf data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

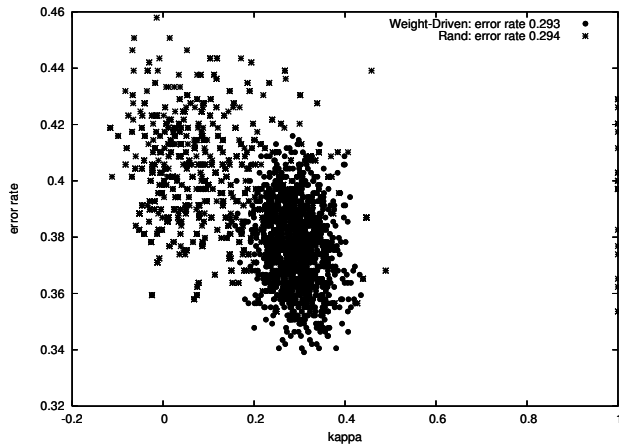


Figure 7: Liver data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

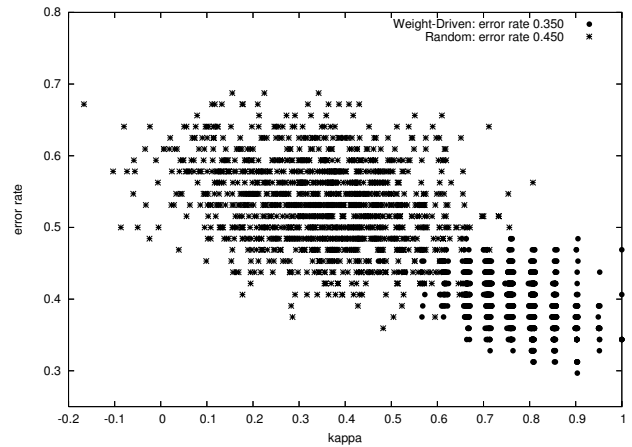


Figure 10: Lung data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

Table 6: Weight-driven approach with customized number of features: Average error rates.

	<i>lung</i>	<i>sonar</i>
Weight (Simple)	46.9 (0.5)	8.7 (0.3)
Weight (Counting)	43.4 (0.5)	8.6 (0.3)
Weight (Borda)	43.4 (0.5)	-

Table 7: Mixture of Weight-driven and random classifiers: Average error rates.

	<i>liver</i>	<i>ionosphere</i>	<i>spectf-test</i>
Simple voting	30.8 (0.5)	6.0 (0.3)	17.8 (0.4)

References

- [1] Agresti, A. (1990). *Categorical data analysis*. John Wiley & Sons, New York.
- [2] Ali, K. M., & Pazzani, M. J. (1996). Error reduction through learning multiple descriptions. *Machine Learning*, **24**:173-202.
- [3] Bay, S. D. (1999). Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, **3**(3):191-209.
- [4] Bishop, Y. M. M., Fienberg, S. E., & Holland, P. W. (1975). *Discrete multivariate analysis: theory and practice*. MIT Press, Cambridge.
- [5] Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases University of California, Department of Information and Computer Science.
- [6] Breiman, L. (1996). Bagging predictors. *Machine Learning* **24**:123-140.
- [7] Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation* **11**:1493-1517.
- [8] P. Chan, P., & Stolfo, S. (1995). A comparative evaluation of voting and meta-learning on partitioned data. *Twelfth International Conference on Machine Learning*.
- [9] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**(1):37-46.
- [10] de Borda, J. (1781). Memoire sur les elections au scrutin, historie de l'academie royale des sciences. Paris.
- [11] Domeniconi, C. & Gunopulos, D. (2002). Adaptive nearest neighbor classification using support vector

Table 8: Mixture of Weight-driven and random classifiers: Average error rates.

	<i>lung</i>	<i>sonar</i>
Simple voting	37.5 (0.5)	8.1 (0.3)

Table 9: Average error correlation values and Average error rates: *Liver* data.

	<i>Error Correlation</i>	<i>Error rate</i>
Random	0.12	29.4
Weight	0.23	29.3
Weight-C ($f = 0.9$)	0.74	30.3
Weight-C ($f = 0.8$)	0.41	31.4
Weight-C ($f = 0.6$)	0.21	31.6
Mixture	0.11	30.8

Table 10: Average error correlation values and Average error rates: *Sonar* data.

	<i>Error Correlation</i>	<i>Error rate</i>
Random	0.34	10.5
Weight	0.69	8.3
Weight-C ($f = 0.9$)	0.72	8.7
Weight-C ($f = 0.8$)	0.66	10.2
Weight-C ($f = 0.6$)	0.42	11.4
Mixture	0.43	8.1

- machines. *Advances in Neural Information Processing Systems 14*, MIT Press.
- [12] Domeniconi, C., Peng, J., & Gunopulos, D. (2002). Locally adaptive metric nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(9):1281-1285.
- [13] Y. Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. *Thirteenth International Conference on Machine Learning*.
- [14] Friedman, J. H. (1994). Flexible metric nearest neighbor classification. *Technical Report*, Department of Statistics, Stanford University.
- [15] Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**:993-1001.
- [16] Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(6):607-615.
- [17] Ho, T. K. (1998). Nearest Neighbor in random subspaces. *Joint IAPR International Workshop on Advances in Pattern Recognition*.
- [18] Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. *Thirteen International Conference on Machine Learning*.
- [19] Langley, P., & Iba, W. (1997). Average-case analysis of a nearest neighbor algorithm. *Thirteenth International Conference on Machine Learning*.
- [20] Margineantu, D. D., & Dietterich, T. (1997). Pruning adaptive boosting. *Fourteenth International Confer-*

- [21] J. R. Quinlan, J. R. (1996). Bagging, boosting and C4.5. *Fourteenth National Conference on Artificial Intelligence*.
- [22] Tumer, K., & Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3-4), pp 385-404.

Lazy Learning for Classification Based on Query Projections *

Yiqiu Han[†]

Wai Lam[‡]

Abstract

We propose a novel lazy learning method called QPAL. QPAL does not simply utilize a kind of distance measure between the query instance and training instances as many lazy learning methods do. It attempts to discover useful patterns known as *query projections*, which are customized to the query instance. The discovery for useful QPs is conducted in an innovative way. QPAL can guarantee to discover high-quality QPs in the learning process. We use some benchmark data sets and a spam email filtering problem to evaluate QPAL and demonstrate that QPAL achieves good performance and high reliability.

1 Introduction

The idea of lazy learning [1] has been proposed as the contrary of common eager learning algorithms. Common eager learning methods eagerly compile the training data into some concept descriptions (e.g., rule sets, decision trees, networks, graphical models). They attempt to seek a particular general hypothesis, which covers the entire instance space. In contrast to eager learning, lazy learning models do not involve any model construction before they encounter the unseen instance to be classified, implying that they do not conduct any processing until they are requested. The customized model and all the intermediate results are discarded when the learning process for this unseen instance completes. Therefore lazy learning algorithms need much less training costs but more storage and computational resources than eager algorithms during classification. Nevertheless lazy learning algorithms can make use of the characteristics of the unseen instance to explore a richer hypothesis space during classification.

In fact lazy learning methods sometimes significantly outperform some eager algorithms for particular learning tasks.

Lazy learning exhibits advantages in many learning scenarios. For example, some round-the-clock 24-hour online services such as spam email filtering may require to update the training data frequently, without interrupting the service. Common eager learning methods need to learn a new global classifier every time the training data is updated. When the training data is large and complex, it is not economical for the service provider to conduct eager learning frequently. Lazy learning methods have no such problems. Generally, the updating of training data is the only operation required by lazy learning methods.

Another learning scenario for which lazy learning is competitive is that the learning target class is not fixed and the attribute set is large. Under such a complex circumstance, the attribute set is usually not oriented to a specific learning task. There may be many irrelevant attributes as well as incomplete data. For example, suppose a global enterprise possesses a huge business data base with an extremely large number of attributes. The query instances to be classified are delivered from time to time, where the intended class attribute may vary from time to time too. One query might be “What will be the profit level given a specific district and a specific time period?”. Another query might be “Whether a new product of an existing category will be well accepted among a specific group of customers?”. Under these circumstances, there can be numerous tuples of a attribute set and a target class. It is not only expensive but sometimes also infeasible to conduct eager learning every time a query of different target class is given. In contrast, lazy learning has advantages in handling such problems in an efficient way. The reason is that lazy learning handles each classification as an independent learning process, and hence it can be customized to the unseen instance and focuses only on the local data patterns.

Many lazy learning algorithms are also described as “instance-based” or “memory-based”. Suppose we need to predict the class label of an unseen instance, called the *query*. Many lazy learning methods collect the training instances similar to the query instance for learning.

*The work described in this paper was substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CUHK 4187/01E, CUHK 4179/03E, and CUHK 4193/04E) and CUHK Strategic Grant (No: 4410001).

[†]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

[‡]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

Generally a distance metric will be utilized to quantify the impact of each training instance. In this paper, we propose a novel lazy learning method, which takes a subset of the query attributes as a learning unit. Our learning method is called *Query Projection Analytical Learning (QPAL)* which explores the projections of the query instance for learning. A *query projection (QP)* is represented by a set of attribute values shared by the query and, potentially, some training instances [9]. The utilization of QPs for learning helps achieve a balance between precision and robustness with a richer hypothesis space. QPAL explores and analyzes QPs, attempting to generate an appropriate set of QPs. The final prediction is made by combining some statistics of the selected QPs.

The learning process of QPAL is customized to the query instance. There is no global model construction. Nothing is done until the query comes. Moreover, after answering a query, the customized model and all the intermediate results are discarded. Hence QPAL can be regarded as a lazy learning method. QPAL starts with the query, which can be regarded as the most specific QP. By gradually removing some attribute values, QPAL obtains more general QPs. Note that these QPs should be supported by training data. Otherwise they will not be considered. Then QPAL will iteratively combine these QPs to produce more general QPs until the stopping criteria is met. At each step, QPAL will investigate the empirical class distribution of the current QP, then decide whether it should be selected or discarded or reserved as seeds. In essence, QPAL has several distinct characteristics as follows.

First, QPAL focuses on the local QPs rather than learning a set of rules which form a general classifier. The learning process is tailored to the query rather than partitioning the whole attribute hyperspace to obtain a global classifier such as a decision tree. Particularly, for real-world problems where the training data needs to be frequently updated such as spam email filtering problem, QPAL has an advantage of reducing the cost of maintenance and operation.

Second, from the perspective of concept learning, the discovered QPs can be regarded as a kind of classification knowledge. A number of classical learning algorithms such as decision tree [15] or rule-based learning [12, 13] have a common characteristic. The data space is recursively partitioned in a greedy manner, which usually leads to the horizon effect [2]. The reason is that the heuristic used to guide the partition commonly looks no further than the next attribute to select. Several extensions [17, 18, 14] have been proposed to cope with the horizon effect, but the optimal result still cannot be guaranteed. However, QPAL does not

intend to construct a global model. Its learning process is customized to the query instance. Hence QPAL does not suffer from the horizon effect as many classical eager learning models do. We will demonstrate that QPAL can guarantee to discover high-quality QPs efficiently. It can guarantee a sufficiently low probability to reject any useful QPs with an expected accuracy higher than a specified value.

Third, unlike many existing lazy learning methods [5, 6, 10, 11], QPAL does not employ ordinary Euclidean distance as the weighting scheme of training instances. Instead, QPAL considers the weighting of a QP, or a group of training instances. It makes the best of the “more-general-than” relationship which is simple but reliable.

In order to evaluate the effectiveness of QPAL, we conducted extensive experiments with benchmark data sets and a spam email filtering problem. QPAL achieves good classification performance and it also exhibits higher reliability and scalability with attribute dimension and the number of training instances.

In Section 3, we will introduce the concept of query projections and their utility in learning. Then we discuss the learning framework based on QPs. A QP selection metric and a set of rules are proposed to facilitate the discovery of useful QPs. In Section 4, we will show how undecided QPs are analyzed and present the learning algorithm. Section 5 empirically investigates the learning process of QPAL. In Section 6, we will discuss how to extend our QPAL to handle continuous attribute values. The experimental results and discussions are given in Section 7. Section 8 gives the conclusion and future work.

2 Related Work

A simple but effective way to conduct lazy learning is the classical k nearest neighbor (kNN) model and its variants [8, 5, 6, 10]. Intuitively, kNN model can be viewed as locating a fixed number of training instances, namely nearest neighbors, for the unseen instance. It uses their class labels to predict the unseen instance. The main difference between QPAL and kNN is that QPAL considers QPs closest to the unseen instance rather than the nearest training instances. Therefore, QPAL is more like a model-selection method rather than a case-based learning method. Second, QPAL does not require ordinary distance metric between instances as kNN does. kNN either assigns the same weight to every nearest neighbor or defines a weighting scheme based on the distance metric. The complicated weighting scheme might incur more computational cost and lose the advantage of pure lazy learning. QPAL uses a different scheme to perform both feature selection and

feature weighting. Third, kNN is lack of interpretability and might fail to uncover some useful but lower-order attribute patterns. QPAL can output QPs as discovered knowledge or explanations of the prediction.

Recently, Li et al. [11] proposed a learning framework, called DeEPs, using emerging patterns. It makes use of the frequency of an instance's subsets of attribute values and the frequency-change rate of the subsets among training classes to perform learning. Both DeEPs and QPAL have the idea of using subsets of attribute values rather than using distance. QPAL is different from DeEPs with respect to the characteristics of QP-based patterns and the mechanism for obtaining good QPs. Moreover, DeEPs tends to favor those frequent patterns having an infinite rate, i.e., all associated instances having the same label. QPAL considers a tradeoff between the frequency and the pattern of class distribution. QPAL also utilizes an exploration algorithm rather than the set operations in DeEPs. These characteristics enable QPAL to select subsets of attribute values in a more principled manner.

Chang and Li has recently proposed a maximizing expected generalization algorithm for learning complex query concepts (MEGA) [4]. It learns an online query concept with the minimum number of labeled instances through active learning, and it can sustain learning under working conditions when no relevant examples are provided at the beginning of the active learning process. A divide-and-conquer method is used to divide high-dimensional features into a number of groups to speed up the learning. Our method resembles MEGA in the idea of decomposing query into groups of attributes, but the objective of learning is different. QPAL focuses on handling common learning problems such as classification. MEGA mainly addresses active online query answering problems.

3 Learning with Query Projections

3.1 The Concept of Query Projection Suppose the learning problem is defined on a class variable C and a finite set $\mathbf{F} = (F_1, \dots, F_n)$ of discrete random variables, i.e., attributes. Each attribute F_i can take on values from respective domains, denoted by $V(F_i)$. To simplify the discussion without loss of generality, we assume that the class variable C is a Boolean variable since a multi-class variable can be broken into a set of binary variables. A query instance \mathbf{t} is denoted by a full set of attribute values $\{t_1, t_2, \dots, t_n\}$ where $t_i \in V(F_i)$. Its QPs can be viewed as subsets of the set $\{t_1, t_2, \dots, t_n\}$. We denote the cardinality of a QP \mathbf{A} as $|\mathbf{A}|$ where $\mathbf{A} \subseteq \mathbf{t}$.

For example, suppose the query is to classify whether the sale is good, or poor, or aver-

age. The attributes are $\{Season = Fall, Region = Asia, Product = Laptop, Clients = Students\}$. The manager cannot achieve the goal with incomplete or insufficient data. However, the data in his hand might deliver a convincing summary of the sale at a more general level, i.e., $\{Fall, Asia, ?, Students\}$ or $\{Fall, ?, laptop, ?\}$. These subsets of the query are QPs. This paper will show that some of them can contribute to learning, or at least bring users closer to the answer for the query. The QPs of a particular query constitute a lattice. The links between QPs reflect the "more-general-than" relationship [16], which can be utilized to explore QPs for learning. A QP is valid only when it is associated with some instances belonging to its concept.

Each valid QP can be regarded as a sub-classifier whose prediction is the majority class among its associated training instances. Suppose a QP has x associated training instances, which is called *frequency*. Among the x associated training instances, y instances sharing the same majority class label is called *majority count*. Then y/x is called the *majority rate* of that QP. It can also be viewed as the empirical accuracy of applying that particular QP for classification.

The ideal case is to have a QP whose associated instances 100% belonging to the same class. Since the query is a specialization of the given QP, the query should also belong to that class. However, in practice, there are several issues to be considered. First, sometimes the highest majority rate among all valid QPs is far less than the ideal case. Second, the empirical majority rate may be unreliable when there are insufficient associated instances. Third, there might be more than one QPs that can serve as good sub-classifiers.

3.2 Exploring Valid QPs Since some QPs can be regarded as sub-classifiers, we use a set of approximately optimal QPs to help learn the query. Suppose a set of qualified QPs are discovered by the method which we are going to discuss below. These discovered QPs are then combined to classify the given query. The final prediction is made by summarizing the frequency of each class on all selected QPs. The class with the maximal frequency serves as the predictor for the class label of the given query. In this step, since a particular single training instance may appear in different QPs, we can observe that it may be sampled multiple times proportional to its contribution to the learning. This usually happens when a training instance exhibits relatively high similarities to the query.

As there are 2^n subsets of a query $\{t_1, t_2, \dots, t_n\}$, It is a computational challenge to search for a set of QPs with unknown size. In fact there are 2^{2^n} combinations.

We develop techniques to cope with this problem. First, the number of valid QPs is usually much smaller than 2^n , because a valid QP must have some associated training instances. The QPs without sufficient data support cannot provide any help in learning, and hence can be ignored. Second, we design an method to explore and discover useful QPs. It can systematically examine valid QPs.

In our method, QPs are explored in sequence based on their cardinalities. Suppose the largest cardinality among all QPs is k . The exploration starts with QPs with the largest cardinality k . They can be enumerated and explored directly by investigating all qualified training instances, i.e., sharing exactly the same k attribute values with the query.

For a valid QP \mathbf{S} whose $|\mathbf{S}| < k$, it must satisfy one of the following conditions.

Cond1: \mathbf{S} has associated training instances which share exactly l identical attribute values with the query.

Cond2: \mathbf{S} is a common subset of two or more explored QPs whose cardinalities are all larger than l .

Cond3: \mathbf{S} is a pure subset of an explored QP whose cardinality is larger than l . All \mathbf{S} 's associated training instances are also inherited from that explored QP. QPs belonging to this condition need not to be considered in the exploration discussed below.

Thus all valid QPs with cardinality $l < k$ can be exhaustively enumerated in two ways. The first way is to explore all training instances which share exactly l identical attribute values with the query. The second way is to check all common subsets of explored valid QPs with cardinalities larger than l .

Consequently, we can explore all valid QPs systematically, from the largest cardinality to the smallest cardinality, via scanning the training data. The whole process is conducted by examining all available training instances. Then the number of valid QPs is constrained by both 2^n and the number of training instances that share at least one attribute value with the query \mathbf{t} . They are ranked according to their number of identical attribute values with the query. Thus at each step only a small fraction of training data needs to be read into memory. With this connection between QPs and instances, operations on QPs are transformed into operations on training instances.

The learning process is usually completed after only examining those training data which is closely related to the query. For problems with large attribute dimensions

but relatively sparse training data, our method has obvious advantage in the capability of exploring all valid QPs. Even for problems with both large attribute dimensions and large number of data instances, our method can still efficiently reduce the search space and discover useful QPs with the aid of a set of rules, as discussed below.

3.3 The Metric for Selecting Useful QPs The purpose of exploring QPs is to help learn the query. Hence we propose a selection metric to find the most useful QPs. As we have stated, the learning for the query prefers QPs with a reliably high majority rate.

To select the most useful QPs, we set two thresholds which can be adjusted to meet the demand in practice. One is the minimal tolerance accuracy for a QP, denoted by q . The other is the minimal expected accuracy for a QP, denoted by p . p is always selected to be greater than q .

Suppose a QP has the expected accuracy equal to p . We denote the probability of observing its y majority class instances among its all x associated training instances by $P(y/x)$, expressed as follows:

$$(3.1) \quad P(y/x) = \frac{x!}{y!(x-y)!} p^y (1-p)^{x-y}$$

For an arbitrary QP whose $y/x \leq q$, its actual accuracy might be higher than the minimal tolerance accuracy q . However, through Equation 3.1, we can control the possibility of incorrectly rejecting such kind of QPs. If $P(y/x)$ of this QP is less than a threshold, we will reject to consider it further. Equation 3.2 illustrates the condition of rejecting a QP whose majority rate is lower than q .

$$(3.2) \quad \left(\frac{y}{x} \leq q \right) \wedge \left(\frac{x!}{y!(x-y)!} q^y (1-q)^{x-y} \leq \right)$$

Consequently, the QPs with a low majority rate y/x and relatively large frequency x will be filtered out. We only have less than (e.g., 5%) chance to incorrectly reject an useful QP with an accuracy higher than q .

On the contrary, if $y/x > p$, we also consider the possibility to accept this QP as with an accuracy higher than p . Equation 3.3 depicts the fact that the probability of a QP having an expected classification accuracy less than p is no more than .

$$(3.3) \quad \left(\frac{y}{x} \geq p \right) \wedge \left(\frac{x!}{y!(x-y)!} p^y (1-p)^{x-y} \leq \right)$$

Since p is selected to be greater than q , QPs with large frequency x and high majority rate y/x are preferred and accepted as sub-classifiers. We only have less than

(e.g., 5%) chance to select an unreliable QP with expected accuracy less than p (e.g., the majority rate of the whole training data set).

If a QP has neither been accepted nor been rejected, it is then stored for further analysis, which we will discuss in Section 4. In the analysis, the common subsets of stored QPs are used to generate new useful QPs as stated by the condition *Cond2* mentioned in Section 3.2.

3.4 Discovering Useful QPs Figure 1 depicts the complete process of exploring QPs. First all valid QPs are ranked and examined by scanning the training data. Then we use a selection metric to find useful valid QPs. If a QP is accepted as a good predictive sub-model, it will be selected as discovered knowledge. If a QP is rejected, it will be discarded. If a QP is neither selected nor discarded, it is then stored. The common subsets of stored QPs are used to generate new QPs which could not be explored directly from the training data. Finally the discovered QPs are combined to predict the query as stated in Section 3.5. In this exploration, a set of straightforward but effective rules are utilized to improve the efficiency. These rules are designed on a basis of the subset relationship between QPs.

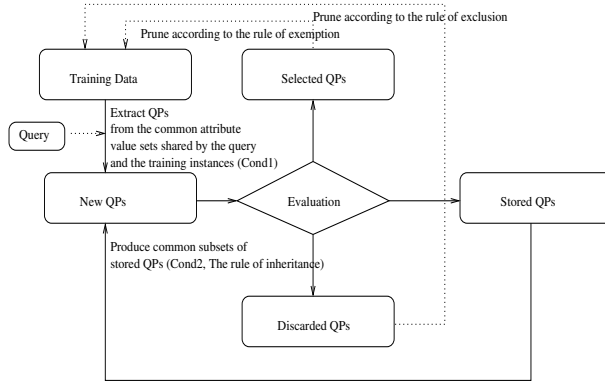


Figure 1: Exploring QPs for a given query.

In QPAL, two QPs can only be compared with each other when they have the subset relationship. This can significantly improve the learning efficiency. This approach also avoids introducing any apriori assumptions such as dependence or independence of attributes, a global Euclidean distance metric, or a weighting scheme for different attributes. This makes the learning from QPs more flexible and more robust. We introduce a property called *Posterior Property*, which can be viewed as the inverse of the common *Apriori Property* used in classical association rule learning. Apriori property can be expressed as “nonempty subsets of a frequent item

set must also be frequent.” Posterior Property can be expressed as “A specialization (superset) of a good QP is also a good QP”. For instance, if S_1 has a high majority rate and $S_1 \subset S_2$, then S_2 has a greater chance to have a high majority rate. As an extreme case, if S_1 has a 100% majority rate, so does S_2 . If S_2 is observed to have low majority rate, it implies that S_1 also cannot have 100% majority rate.

Based on the above arguments and having $S_1 \subset S_2$, we design the following rules:

- *Rule of exemption* : If S_2 has been selected, which means that it has sufficiently high majority rate and sufficient associated training instances, then S_1 should not be considered any more because S_1 is a generalization of S_2 and contains less information from the query. Returning to the previous example, $\{Fall, Asia, ?, Students\}$ can suppress $\{Fall, Asia, ?, ?\}$ if the former has sufficient data support. Meanwhile, $\{Fall, Asia, ?, Students\}$ cannot suppress $\{Fall, ?, laptop, ?\}$ although the former has larger cardinality.
- *Rule of exclusion* : If S_2 is discarded, then S_1 should not be considered any more due to the same reason as the Rule of exemption. This rule can also be regarded as an extension of the Rule of exemption.
- *Rule of inheritance* : If S_2 has the same associated training instance set as S_1 , S_2 need not to be considered as a valid QP since S_1 can replace it without any cost.

The above rules show that QPs can be exploited by analyzing their relationship and observing their class distributions. The decision of selecting or discarding a particular QP will trigger a family of QPs to be pruned. These operations can significantly accelerate the searching for an appropriate set of useful QPs. Figure 2 describes pseudo-code of the discovery process of useful QPs. Note that most operations have been transformed into operations on training instances, as shown in Steps 1-2. Steps 6 and 9 use Equations 3.2 and 3.3 to decide whether to accept or reject a QP. Steps 7 and 11 employ the rules discussed above to accelerate the discovering process. The processing of undecided QPs at Step 14 is discussed in Section 4. These components constitute our QPAL learning method.

3.5 Predict the Query with Discovered QPs

After QPAL discovers an appropriate set r of useful QPs, a majority voting is conducted among the training instances associated with selected QPs. Each discovered QP can be viewed as a sub-classifier and the final

```

1  Sort training instances according to the highest cardinality
   of associated QPs.
2  Put all training instances associated with no-empty QPs into
3  Initialize the active set u and the final set r to be empty.
4  FOR  $j = n$  to 1
5    FOR every QP whose cardinality is  $j$  and owns
      associated instances in
6      IF Equation 3.2 holds for this QP.
7        Remove all instances whose associated QPs are all
          subsets of this specific QP.
8      ELSE
9        IF Equation 3.3 holds for this QP.
10       Insert this QP into u.
11       Remove all instances whose associated QPs are all
          subsets of this specific QP.
12     ELSE
13       Insert this QP into u.
14       Analyzing the QPs in u as stated in Section 4.
15   IF  $|\mathbf{u}| = 0$ 
16     BREAK
17 IF  $|\mathbf{r}| = 0$ 
18   Move all QPs with the largest cardinality into r.
19 Return the discovered QP set r.

```

Figure 2: The pseudo-code of the discovery process of useful QPs

prediction is a combination of those sub-classifiers. The class with the maximum sum of frequencies in all discovered QPs is the classification result of the query instance.

It should be noted that associated training instances are sampled for multiple times if they are associated with more than one QP. Consequently, the training instances closer to the query tend to have a larger impact on the final prediction, because they have larger chances to associate with more discovered QPs. Moreover, the sampling scheme leads to a weighting scheme of the discovered QPs. A QP with more support from training instances plays a more important role in the final decision. The weight for a discovered QP is proportional to its observed frequency, which can be viewed as the confidence on its associated sub-classifier.

4 Generate Useful QPs from Undecided QPs

As we have discussed, the QPs are extracted from the training data and then considered in a systematic manner. In this process, there might be some QPs that can be neither rejected nor accepted, as shown in Step 14 in Figure 2. For example, a QP of the largest cardinality has the frequency $x = 5$ and the majority count $y = 4$. With the high majority rate, i.e., 80%, it will not be rejected. But its low frequency makes it

also unacceptable. Generally these QPs are reserved in a temporary QP set **u**. If the whole process does not produce any interesting QPs with Equation 3.3, **u** is analyzed to generate some useful QPs for learning.

The generation of useful QPs is conducted in an iterative manner. First, QPs are divided into groups according to their majority class. In each group, the mating of any pair of QPs will reproduce a new QP, which is the common subset of its parents. These newly generated QPs will be considered to be accepted or rejected. If there are still no interesting QPs satisfying Equation 3.3, the remaining QPs, i.e., QPs not satisfying Equation 3.2, will become the updated QP set **u**. This will iterate until some useful QPs are found. If there is no new generation of QPs, i.e., **u** becoming empty, this generation process will automatically terminate. The details of the generation is described in Figure 3.

```

1  Divide all QPs in u into groups of different majority class.
2  Initialize an empty set v.
3  FOR each group of QPs
4    FOR any pair of QPs in the current group
5      Produce the children QP  $m$ .
6      IF Equation 3.2 holds for  $m$ .
7        Discard  $m$ .
8      ELSE
9        IF Equation 3.3 holds for  $m$ .
10       Insert  $m$  into r.
11      ELSE
12        Insert  $m$  into v.
13 IF  $|\mathbf{r}| = 0$ 
14   u = v
15 IF  $|\mathbf{u}| \neq 0$ 
16   GOTO 2
17 ELSE
18   END

```

Figure 3: Generate useful QPs from undecided QPs

From another perspective, the analyzing of undecided QPs in **u** is to work through valid QPs belonging to the condition *Cond2* mentioned in Section 3.2. For those QPs cannot be explored by using the training data directly, we need to use existing QPs, i.e., **r**, to access them. The QPs rejected by Equation 3.2 and the QPs accepted by Equation 3.3 are excluded according to the rules in Section 3.4. Therefore, all valid QPs can be scanned by QPAL efficiently. This property helps QPAL to avoid the horizon effect. Particularly, suppose there is a valid QP with 100% accuracy. QPAL can guarantee the discovery of this QP, which cannot be done by common eager learning algorithms such as decision tree

or association rule-based learning. The proof is given below.

Proof. Suppose there exists a QP \mathbf{S}_i with 100% expected accuracy. Then each QP \mathbf{S}_j satisfying $\mathbf{S}_i \subseteq \mathbf{S}_j \subseteq \mathbf{t}$ should also be with 100% expected accuracy. Hence they will not be discarded in the learning process and at least one of them will be employed according to Equation 3.3. Common eager classification rule learning methods cannot achieve this, because all QP \mathbf{S}_k satisfying $\mathbf{S}_k \subseteq \mathbf{S}_i \subseteq \mathbf{t}$ may not have distinguishable information gain or observed accuracy although \mathbf{S}_i can. This observation can be found in the two synthetic problems in Section 5.

Although the generation process adds some computational cost. We can show that the time complexity of QPAL is still acceptable, and even lower than many common eager learning algorithms. First of all, The time complexity of QPAL is bounded by the number of valid QPs rather than theoretical 2^{2^n} . The number of valid QPs are actually bounded by the size of training data. Since QPs must be associated with training instances sharing at least one identical attribute value with the query, the computational cost can be significantly reduced. Second, the rules we introduce in Section 3.4 can greatly accelerate the discovery of useful QPs. Once a QP is accepted or rejected, a number of QPs will be removed from further consideration.

Furthermore, the computational cost of the generation process can be greatly reduced by constraining the size of \mathbf{u} , or by tightening the thresholds in Equation 3.2 and 3.3, or by imposing a limit on the minimum cardinality of QPs to be analyzed. In addition, the generation process is only employed when no useful QPs are found via direct exploration.

5 Empirical Investigation of QPAL

To investigate the learning process of QPAL, we synthesized two learning problems. They have the same attribute set $\mathbf{F} = \{F_1, F_2, F_3, F_4\}$ where each F_i is defined on the value domain $\{1, 2, 3, 4\}$. The class label is binary taking on values of 0 or 1. We defined the class concept to be learned for the first synthetic problem as $((F_1 = F_2) \vee (F_3 = F_4))$. The class concept to be learned for the second synthetic problem is $((F_1 = F_2) \wedge (F_3 = F_4))$. We generated two synthetic data sets, one for each synthetic problem. Each data set consists of 256 instances, which covers all possible instantiations of \mathbf{F} , from (1,1,1,1) to (4,4,4,4). In the data set of the first synthetic problem, 112 instances, i.e., 7/16 in ratio, are labeled as “positive” and 144 “negative” instances, i.e., 9/16 in ratio. For example, (1,1,2,3) and (2,2,3,3) are labeled as positive while (4,3,1,2) and (1,2,2,3) are labeled as negative. In the data set of

the second synthetic problem, there are 16 positive instances, i.e., 1/16 in ratio, and 240 negative ones, i.e., 15/16 in ratio. For example, (1,1,4,4) and (2,2,3,3) are labeled as positive while (3,3,1,4) and (1,2,2,3) are labeled as negative.

One characteristic of the synthetic learning problems is that the class labels are evenly distributed over different values for different attributes, as depicted in Figure 4. For instance, $P(F_i|C)$ remains constant for any value of any attribute. There is no difference between the information gain of any attribute values. Hence most classical learning models cannot handle this problem effectively. we have tested QPAL

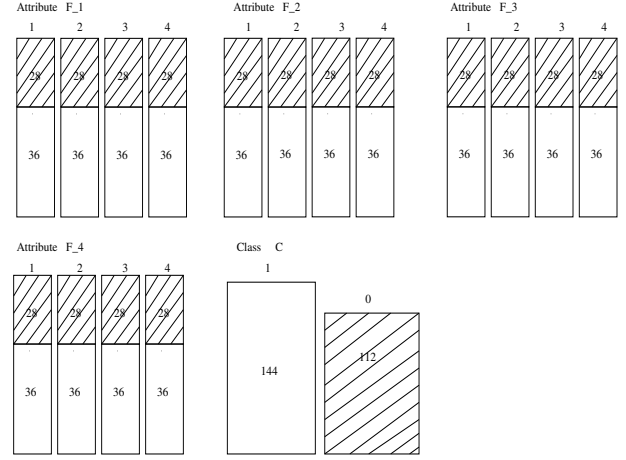


Figure 4: The class distribution of each attribute for the data set of the first synthetic problem

as well as different classical learning models, namely, Bayesian Network, Naive Bayes, kNN, Support Vector Machine (SVM), Lazy Bayesian Rule (LBR) [20], and Decision Tree (J48). For these classical learning models, we employed Weka-3-2-6 machine learning software package [19] freely available on the Web (<http://www.cs.waikato.ac.nz/ml/weka/>). Default parameter setting was used for each classical learning model. The leave-one-out method was used for measuring the classification performance.

The results in Table 1 show that most classical learning models we tested cannot handle the first problem effectively. Only QPAL and Bayesian Network can achieve perfect (100%) accuracy. For other models, Decision Tree (J48) performs better. However, it can be observed that the Decision Tree algorithm can only output an empty decision tree. The reason is that Decision Tree uses a greedy manner to chose an attribute as a node once at a time. When there are no difference between the information gain of different attributes, the partitioning process of the attribute space cannot con-

Problem	QPAL	Bayesian Network	J48	kNN	LBR	Naive Bayes	SVM
1st	100	100	81.22	62.5	62.5	56.25	44.75
2nd	100	93.75	93.75	93.75	93.75	93.75	93.75

Table 1: Classification performance, measured by accuracy percentage, of different learning algorithms for synthetic learning problems.

tinue or cannot produce a reliable model. After investigating the output of Naive Bayes, we found that Naive Bayes can only conduct majority voting for this problem. As for kNN and Lazy Bayesian Rule (LBR), only the instances satisfying $((F_1 = F_2) \wedge (F_3 = F_4))$ or $((F_1 \neq F_2) \wedge (F_3 \neq F_4))$ can be classified correctly. These two cases occupy 1/16 and 9/16 of the whole instances respectively and they precisely sum up to 62.5%. It can be observed that the Euclidean distance between two instances cannot be treated as a reliable measure of their similarity in terms of the target class concept.

Consider the scenario of determining the class label of the query instance (2,2,3,2) in the first learning problem. The QPs are organized in a lattice as shown in Figure 5. The associated training instances are also shown. The non-shaded dot denotes a negative instance whereas the shaded dot denotes a positive instance. It can be observed that the useful pattern (2,2,?,?) is in the middle of the lattice, whose associated 15 training instances have the same class label. QPAL first investigates valid QPs with the largest cardinality 3, i.e., $\{(2,2,3,?), (2,2,?,2), (2,?,3,2), (?,?,3,2)\}$. They are divided into two groups with different majority classes. After the generation, the group of positive class produces a useful new QP (2,2,?,?) whereas the group of negative class produces none. Therefore, QPAL outputs the QP (2,2,?,?) as the explanation of predicting the (2,2,3,2) as “positive”. For kNN algorithm, the nearest neighbors on the top of the lattice cannot offer reliable prediction because of the existence of “irrelevant nearest neighbors”. Although they are among the nearest training instances to the query instance, their associated QPs may not be useful for the prediction. For Decision Tree, the bottom QPs in Figure 5 represent single attribute values. They have no difference in terms of information gain. So the construction of a decision tree is not feasible. Figure 5 illustrates that QPAL benefits from the effective exploration for QPs of different attribute dimensions, which may not be fully utilized by other common learning models.

The output QPs demonstrates its good interpretability of the classification decision. Returning to the example above, If the instance to be classified is (2,2,3,2), the discovered QPs \mathbf{r} becomes $\{(2,2,?,?), (?,?,3,?), (?,?,?,2)\}$. The final class prediction is “positive”. If the instance to be classified is (1,1,2,2),

then \mathbf{r} is $\{(1,1,?,?), (?,?,2,2)\}$. The final class prediction is positive. It can be observed that QPAL appropriately captures useful QPs rather than handling each attribute separately or considering the complete attribute set as a whole. These selected QPs can provide a more comprehensible explanation for the classification decision.

For the second synthetic problem, the difference is that the positive class in the second synthetic problem is only a tiny fraction of the whole data set, which introduces challenges for the learning task. The results in Table 1 reveal that all classical learning models cannot achieve perfect performance. The reason is that the number of training instances is too small for the positive class. Hence all instances are predicted to be the majority class which occupies 93.75% of the data set. However, QPAL can perform the learning task in a perfect manner. The generation process successfully filters out the irrelevant QPs and selects the useful ones. It shows that QPAL has a good tolerance for the data sparseness problem.

Since QPAL focuses on the local QPs of the query instance, it can effectively discover useful patterns for prediction, as shown in this investigation. This empirical investigation suggests that QPAL is good at discovering some useful knowledge.

6 Handling Continuous Attribute Values and Missing Values

QPAL can be extended to handle continuous attribute values easily. A simple technique is to employ discretization methods [7] before QPAL processes the data. However, this will make the whole learning framework not a pure lazy learning because the discretization needs to preprocess the data.

An alternative method is to handle the continuous attribute values in the QPAL model. In the implementation of QPAL, each training instance is transformed into a binary string where each bit is associated with an attribute value. If a training instance shares the same value as the query on a particular attribute, then the corresponding bit is set to “1”, otherwise it is set to “0”. Thus the query can be represented by a binary string of all “1”. Such a binary string is in fact another format of QP. Two or more training instances may have the same binary string. Hence the task of discovering

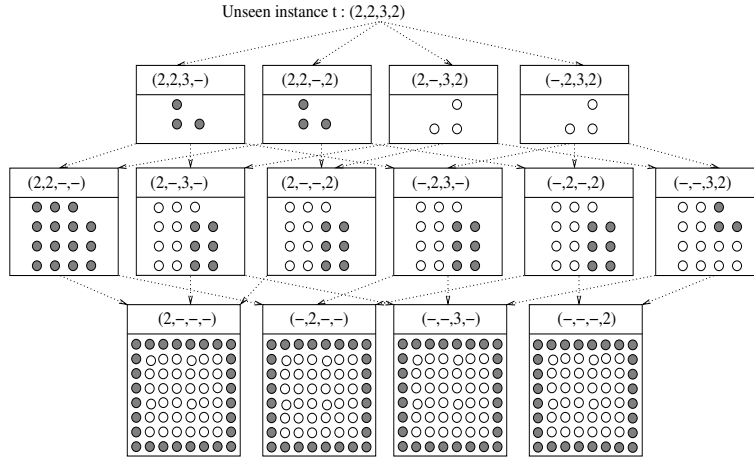


Figure 5: The lattice of QPs and associated training instances given a query instance (2,2,3,2)

QPs can be accomplished by a series of Boolean operations which can be computed efficiently.

We can integrate the handling of continuous attributes values into this transformation module. The neighborhood concept is used to replace the concept of equality. In other words, if a training instance falls into the neighborhood of the query at a given continuous attribute. Then the corresponding bit is set to 1, otherwise 0. The neighborhood is defined as x training instances with the closest values at the given attribute. The value x is usually specified as one half of the minimal class frequency among the training data. For computation, the training data only needs to be scanned only one time. A priority queue with the size x will remember the closest x training instances for each attribute. The transformation to binary strings is then guided by these priority queues.

In real-world problems, the instances usually contain missing attribute values. QPAL can handle these missing values very easily without any additional processing. The reason is that lazy learning makes each query instance has its independent learning process. Hence the attributes with missing values are just removed from the learning process for the query as if the problem is defined only on the attributes with known values.

Data Set	Number of Attributes	Number of Classes	Number of Instances
Annealing	38	5	898
Breast(W)	9	9	699
Contact(L)	4	3	24
Credit(A)	15	2	690
Glass	9	7	214
Heart(C)	13	5	303
Hepatitis	19	2	155
Ionosphere	34	2	351
Iris	4	3	150
Kr-vs-Kp	36	2	3,196
Labor	16	2	57
Letter	16	26	20,000
Lymph	18	4	148
Mushroom	22	2	8,124
Sonar	60	2	208
Soybean	35	19	683
Vowel	13	11	990
Zoo	17	7	101

Table 2: Description of 18 benchmark data sets

7 Experiments and Discussions

7.1 Benchmark Data Sets We have conducted extensive experiments on 18 benchmark data sets from the UCI repository of machine learning database [3] to evaluate our QPAL framework. These data sets, collected from different real-world problems in various domains, are shown in Table 2.

We partitioned each data set into 10 even portions

and then conducted 10-fold cross-validation. The performance is measured by the accuracy which is defined as the percentage of the number of correctly classified instances over the total number of testing instances. In these experiments, we have also investigated the performance of Naive Bayes, SVM, kNN, Lazy Bayesian Rule (LBR) and Decision Tree (J48) provided by Weka-3-2-6 machine learning software package. All these models except for kNN use default settings during the entire evaluation process. For kNN, we have conducted runs for $k = 1, 5$ and 10. We reported the results for $k = 5$ because it achieves the best average classification accuracy among different k .

The results of lazy learning methods on different data sets are depicted in Table 3. kNN achieves good performance on data sets such as Annealing, Credit, Letter, Vowel, Sonar, and Zoo. This observation suggests that kNN has some advantages for large data sets or data sets with large attribute dimensions. Our QPAL shares these advantages. Compared with existing lazy learning approaches, on most of the data sets, QPAL achieves good performance. For some data sets such as Contact, Glass, Hepatitis, Labor, and Sonar, QPAL outperforms all other classical lazy classifiers. The results also show that QPAL excels at handling data sets with incomplete data. The reason is due to the flexible framework of QPAL. It should be noted that despite its simplicity, lazy learning method kNN can achieve outstanding performances on some data sets when $k = 1$. For example, on data sets of Annealing, Letter, Sonar and Vowel, 1NN obtains 99.11%, 96.06%, 86.57% and 99.29% respectively. It significantly outperforms existing learning methods. However, 1NN suffers on other data sets due to its simplicity. QPAL can obtain comparable performance of 1NN on the above data sets, with less negative effect on other data sets. On average, the classification accuracy of QPAL on these 18 benchmark data sets is 90.6%, which shows improvement over other lazy learning models.

We also compared the performance of QPAL and some existing eager learning methods, i.e., SVM and J48. The average accuracy of each classifier is shown in Table 4. According to our experiment results, lazy learning methods have advantages over eager learning methods on some learning problems. On these 18 benchmark data sets, QPAL achieves the best average score 90.6%.

7.2 Spam Email Filtering Problem Spam emails have become an increasingly important concern. The demand for spam email filtering rises rapidly. Black list or white list are usually used to filter out spam emails. Various intelligent methods have also been proposed to

Data Set	QPAL	kNN	NaiveBayes	LBR
Annealing	97.1	97.1	86.5	97.2
Breast(W)	96.7	96.4	96.0	97.4
Contact(L)	87.5	66.7	75.0	70.83
Credit(A)	84.9	85.8	77.7	85.6
Glass	75.0	67.8	48.5	65.9
Heart(C)	82.6	81.8	84.5	82.4
Hepatitis	90.0	85.8	83.8	85.3
Ionosphere	90.3	84.9	82.4	90.9
Iris	96.0	95.3	96.0	94.7
Kr-vs-Kp	96.5	96.0	87.6	97.3
Labor	96.5	86.0	94.7	89.3
Letter	86.8	95.5	64.2	60.3
Lymph	83.8	83.8	83.8	81.5
Mushroom	99.8	100.0	95.8	99.9
Sonar	87.0	84.6	65.9	74.6
Soybean	92.1	90.2	92.8	93.0
Vowel	91.5	93.7	61.4	86.5
Zoo	96.2	95.1	95.2	95.8
Average	90.6	88.1	81.8	86.0

Table 3: Classification performance of QPAL and other lazy learning methods.

	QPAL	J48	SVM
Average	90.6	86.3	87.4

Table 4: Classification performance comparison of QPAL and eager classifiers.

tackle this problem. We investigate applying our QPAL learning framework to spam email filtering.

We used the data set from the Data Mining Cup 2003 (DMC2003) contest, whose task is to identify spam emails by means of data mining techniques. The training data set has 8,000 instances and the testing data set has 11,177 instances. Among those testing instances there are 4,374 spam emails and 6,803 non-spam emails. Each instance representing an email is recorded by 834 attributes. The attributes represent different tactics for identifying spam emails, for example, whether the email contains unsafe java script, or contains some keywords, or has a particular characteristic style. A complete description of all the attributes can be found in Open Source Project SpamAssassin (see <http://spamassassin.org>). The evaluation metric is measured by the filter-out rate which is defined as the number of emails being filtered out as spam emails divided by the number of emails being processed.

QPAL is employed in a pure lazy learning manner without any customization to this data set. The empirical results show that QPAL can obtain prominent performance. In particular, QPAL performs very well in preserving non-spam emails while keeping a high filter-out rate for spam emails. This characteristic is very important in business applications because many users

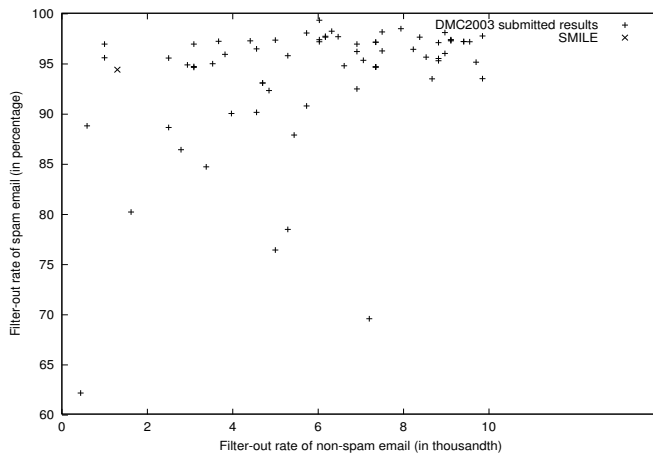


Figure 6: Performance of filtering spam emails

cannot bear a high risk of losing useful emails. The result is shown in Figure 6 where the submitted results of DMC2003 are also shown for comparison. The X-axis in the graph represents the ratio of non-spam emails being incorrectly filtered out. The Y-axis represents the ratio of spam emails being correctly filtered out. The performance of each submitted group is expressed by the symbol “+”. The performance of QPAL is indicated by the symbol “x”. The closer is the symbol to the top-left corner, the better is the performance.

It can be observed that QPAL is able to obtain a favorable tradeoff between the filtering of spam emails and the protection of normal emails. QPAL filters out more than 94% spam emails, which is far better than the filtering performance by many ISP which ranges from 60% to 80%. Even when compared with the submitted results of DMC2003, the performance of QPAL is comparable in terms of spam email filter-out rate. Note that most of the submitted results are based on customized non-lazy learning models while QPAL conducts filtering in a lazy learning manner. In a lazy learning manner, the user need not to train the learning model frequently as spam emails change their patterns from time to time to avoid being filtered out. Specifically, the required task for QPAL only includes adding some typical spam emails into the training set. This property can help to cope with those intelligent spam email creators and rapidly changing spam emails.

As for the filter-out rate of non-spam email, QPAL shows a superior performance where only 0.13% of the non-spam emails are incorrectly filtered out. Meanwhile, most of the submitted results for DMC2003 cannot reach that level. People typically favor the filters which can protect more useful emails rather than the filters which can delete more spam emails. It is also

common that there are a lot of spam emails disguised as a non-spam email, and they cannot be filtered out without raising the risk of losing non-spam emails. As a result, QPAL is indeed very competitive for spam email filtering.

8 Conclusions and Future Work

This paper proposes a novel query-driven lazy algorithm called QPAL, which attempts to discover local QPs for classifying the query instance. By customizing the learning process to the query and searching in an innovative way, it can avoid the horizon effect which commonly exists in many eager learning algorithms. We show that QPAL can guarantee the discovery of all QPs with perfect (100%) expected accuracy in polynomial time. To improve the efficiency of lazy learning on large data sets, QPAL can guarantee a low probability of rejecting any QPs with an expected accuracy higher than a specified value. The experimental results on a real world problem and benchmark data sets also demonstrate that our learning algorithm achieves prominent learning performance.

A promising future direction is to explore more sophisticated methods for assessing QPs. The class distributions on parent and children QPs can also be considered in making decisions on a particular QP as reinforcements. Moreover, some non-lazy learning techniques can be introduced into the learning framework, for example, the useful QPs selected for a given query can be stored for future usage if they meet some requirements. Another direction is to apply QPAL in online active learning. The rationale is that QPAL can organize the relevant QPs in a lattice structure and hence provide potentially useful instances in a hierarchical way to accelerate the learning with little user intervention.

References

- [1] Aha, D.: Editorial. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:7–10, February 1997.
- [2] Aha, D.: Relating relational learning algorithms. In S. Muggleton, editor, *Inductive Logic Programming*, pages 233–260. Academic Press, 1992.
- [3] Blake, C., Keogh, E., and Merz, C.: UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [4] Chang, E., and Li, B.: MEGA—the maximizing expected generalization algorithm for learning complex query concepts. *ACM Transactions on Information Systems (TOIS)*, 21(4):347–382, 2003.
- [5] Dasarthy, B.: *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, 1991.

- [6] Dasarathy, B.: Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, March 1994.
- [7] Fayyad, U., and Irani, K.: Multi-interval discretization of continuous-valued attributes as preprocessing for machine learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.
- [8] Friedman, J.: Flexible metric nearest neighbor classification. Technical report, Stanford University, November 1994.
- [9] Han, Y., and Lam, W.: Lazy learning by scanning memory image lattice. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 447–451, 2004.
- [10] Lam, W., and Han, Y.: Automatic textual document categorization based on generalized instance sets and a metamodel. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):628–633, 2003.
- [11] Li, J., Dong, G., Ramamohanarao, K. and Wong, L.: DeEPs: A new instance-based discovery and classification system. *Machine Learning*, 54:99–124, 2004.
- [12] Li, W., Han, J., and Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 369–376, 2001.
- [13] Liu, B., Hsu, W., and Ma, Y.: Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 80–86, 1998.
- [14] Matheus, J.: Adding Domain Knowledge to SBL Through Feature Construction. In *Proceedings of the National Conference on Artificial Intelligence*, pages 803–808, 1990.
- [15] Mehta, M., Rissanen, J., and Agrawal, R.: MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 216–221, 1995.
- [16] Mitchell, T.: *Machine Learning* McGraw Hill, pages 24–25, 1997.
- [17] Seshu, R.: Solving the parity problem. In *Proceedings of the Fourth European Working Session on Learning*, pages 263–271, 1989.
- [18] Utgoff, P. and Brodley, C.: An incremental method for finding multivariate splits for decision trees. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 58–65. University of Texas, Austin, Texas, 1990.
- [19] Witten, I. and Frank, E.: *Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.
- [20] Zheng, Z., Geoffrey, I., and Kai M.: Lazy Bayesian rules: a lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In *Proceedings of 16th International Conf. on Machine Learning*, pages 493–502, Morgan Kaufmann, San Francisco, CA, 1999.

Mining Non-Derivable Association Rules

Bart Goethals* Juho Muhonen Hannu Toivonen
Helsinki Institute for Information Technology
Department of Computer Science
University of Helsinki
Finland

Abstract

Association rule mining typically results in large amounts of redundant rules. We introduce efficient methods for deriving tight bounds for confidences of association rules, given their subrules. If the lower and upper bounds of a rule coincide, the confidence is uniquely determined by the subrules and the rule can be pruned as redundant, or *derivable*, without any loss of information. Experiments on real, dense benchmark data sets show that, depending on the case, up to 99–99.99% of rules are derivable. A lossy pruning strategy, where those rules are removed for which the width of the bounded confidence interval is 1 percentage point, reduced the number of rules by a further order of magnitude. The novelty of our work is twofold. First, it gives absolute bounds for the confidence instead of relying on point estimates or heuristics. Second, no specific inference system is assumed for computing the bounds; instead, the bounds follow from the definition of association rules. Our experimental results demonstrate that the bounds are usually narrow and the approach has great practical significance, also in comparison to recent related approaches.

1 Introduction

Association rule mining often results in a huge amount of rules. Attempts to reduce the size of the result for easier inspection can be roughly divided to two categories. (1) In the subjective approaches, the user is offered some tools to specify which rules are potentially interesting and which are not, such as templates [KMR⁺94] and constraints [NLHP98, GVdB00]. (2) In the objective approaches, user-independent quality measures are applied on association rules. While interestingness is user-dependent to a large extent, objective measures are needed to reduce the redundancy inherent in a collection of rules.

The objective approaches can be further categorized by whether they measure each rule independently of other rules (e.g., using support, confidence, or lift) or address rule redundancy in the presence of other rules (e.g., being a rule with the most general condition and the most specific con-

sequent among those having certain support and confidence values). Obviously only approaches of the latter type can potentially address redundancy between rules. Our work will be in this category.

We show how the confidence of a rule can be bounded given only its subrules (the condition and consequent of a subrule are subsets of the condition and consequent of the superrule, respectively). It turns out, in practice, that the lower and upper bounds coincide often, and thus the confidence can be derived exactly. We call these rules *derivable*: they can be considered redundant and pruned without loss of information. We also consider lossy pruning strategies: a rule is pruned if the confidence can be derived with a high accuracy, i.e., if the bounded interval is narrow.

Unlike practically all previous work on pruning association rules by their redundancy, our method for testing the redundancy of a rule is based on deriving absolute bounds on its confidence rather than using an ad hoc estimate. Given an error bound, we can thus guarantee that the confidence of the pruned rules can be estimated (derived) within the bounds. No (arbitrary) selection of a derivation method is involved: the bounds follow directly from the definitions of support and confidence. (A pragmatic choice we will make is that only subrules are used to derive the bounds; see below.)

In a sense, the proposed method is a generalization of the idea of only outputting the free or closed sets [PBTL99, BBR00]. Using free sets and closed sets corresponds, however, to only pruning out rules for which we know the confidence is one. In the method we propose, the confidence can have any value, and the rule is pruned if we can derive that value. Closed sets and related pruning techniques actually work on sets, not on association rules. There are other, more powerful pruning methods for sets. In particular, our work is an extension of the work on non-derivable sets [CG02] to non-derivable association rules. The method is simple, yet it has been overlooked by previous work on the topic.

Optimally, the final collection of rules should be understandable to the user. The minimal collection of rules from which all (pruned) rules can be derived would have a small

*Current affiliation: Dept. of Math and Computer Science, University of Antwerp, Belgium

size, but it would most likely be difficult for the user to see why the rest of the rules were pruned and what their confidences must be. We consider different alternatives, including the relatively popular compromise of grouping rules by their consequents and ordering them by the size of the condition. Then, each rule is checked for redundancy given only its subrules having the exactly same consequent, and only non-derivable rules are output.

As a summary, our contributions are the following. We give theoretically sound methods for bounding the confidence of an association rule given its subrules. We then propose to prune as redundant those association rules for which the confidence can be derived exactly or within a guaranteed, user-specified error bound. Experiments with several real data sets (chess, connect, mushroom, pumsb) demonstrate great practical significance: 99–99.99% of rules had (exactly) derivable confidences. Further significant pruning is obtained by removing rules derivable within just ± 0.5 percentage points: the remaining number of rules was only 0.005%–0.04%.

The rest of this article is organized as follows. Section 2 reviews the basic concepts and related work. In Section 3 we define non-derivable association rules and give methods for deriving absolute and tight upper and lower bounds for rule confidences. In Section 4 we give experimental results on a number of real data sets. Section 5 contains our conclusions.

2 Problem Definition and Related Work

The association rule mining problem can be described as follows [AIS93]. We are given a set of items \mathcal{I} and a database \mathcal{D} of subsets of \mathcal{I} called transactions. An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are sets of items, X is called the condition, and Y the consequent. The *support* of a set I is the number of transactions that include I . A set is called *frequent* if its support is no less than a given minimal support threshold. An association rule is called frequent if $X \cup Y$ is frequent and it is called *confident* if the support of $X \cup Y$ divided by the support of X exceeds a given minimal confidence threshold. The goal is now to find all association rules over \mathcal{D} that are frequent and confident.

Typically, for reasonable thresholds, the number of association rules can reach impractical amounts, such that analyzing the rules themselves becomes a challenging task. Moreover, many of these rules have no value to the user since they can be considered redundant. Removing these redundant rules is an important task which we tackle in this paper.

Previous work on pruning redundant association rules is typically based on a decision rule that compares the confidence or support of an association rule to similar rules. For instance, rule $X \Rightarrow Y$ is a “minimal non-redundant association rule” [BPT⁺00] if there is no rule $X' \Rightarrow Y'$ with $X' \subset X$, $Y' \supset Y$ such that $\text{supp}(XY) = \text{supp}(X'Y')$

and $\text{conf}(X \Rightarrow Y) = \text{conf}(X' \Rightarrow Y')$. A similar but not identical definition is given for “closed rules” in [Zak00] or “minimal rules” in [ZP03]. A recent proposal is that rule $X \Rightarrow Y$ is not a “basic association rule” [LH04] if there exists $X' \subset X$ such that for all $X'', X' \subseteq X'' \subseteq X$, $\text{conf}(X \Rightarrow Y) = \text{conf}(X'' \Rightarrow Y)$. Our proposal differs from these techniques in two significant aspects. First, it has a wider applicability: the above-mentioned concepts only apply for rules with exactly the same confidence. Second, these techniques use specific inference systems to decide when a rule is pruned, and in order to know the confidence or support of a pruned rule, the user must use the exact same inference system. In our proposed technique, the bounds follow from the definition of association rules.

Another approach is to estimate rule confidence from a collection of other rules. For example, the maximum entropy technique declares a rule to be redundant if its true confidence is close to the estimate [MPS99, JS02]. In theory, the maximum entropy principle yields consistent estimates in the sense that the value is possible, i.e., it is within the bounds implied by the constraints used. There are some critical issues in its application to rule pruning, however. First, the principle does not give any guarantees for the error bounds. Second, a pruning strategy based on removing rules for which the error is below a given upper bound alleviates the first issue, but at the cost of assuming maximum entropy principle as the inference system. Finally, it is computationally demanding to compute the maximum entropy solution. Practical alternatives rely on approximations, and then lose the benefit of producing consistent estimates.

For a good and quite recent, yet brief overview of attempts to find non-redundant association rules, see reference [LH04].

Some of the approaches mentioned above [BPT⁺00, Zak00] utilize the concept of *closed sets*. A set is called closed if it has no proper superset with the same support; from this, it follows that a non-closed set X implies the rest of its closure with 100% certainty, i.e., the confidence of rule $X \Rightarrow Y$ equals 1 when Y is a subset of X ’s closure. Given a non-closed set X , any set Y in its closure, and a rule $X \Rightarrow Z$, it has been proposed to prune rules of the form $XY \Rightarrow Z$ and $X \Rightarrow YZ$ as redundant since their frequencies and confidences are identical with the rule $X \Rightarrow Z$. As mentioned above, this approach makes assumptions, and without knowing them the user cannot know why rule $XY \Rightarrow Z$ was pruned.

A good amount of work has focused on finding condensed representations for frequent sets by pruning redundant sets. Obviously, the number of association rules is even much larger and hence the problem is even more important to solve. In the case of frequent sets, the most successful condensed representation is the notion of closed sets: all frequent sets can be derived from the closed frequent sets (or

frequent generators). δ -free sets generalize this notion to “almost closed” sets [BBR00].

More recently, a more powerful method for pruning frequent sets has been presented, called *non-derivable sets* [CG02]. The main idea is to derive a lower and an upper bound on the support of a set, given the supports of all its subsets. When these bounds are equal (the support of) the set is *derivable*. In this paper, we extend this work in a natural way to association rules: we introduce similar derivation techniques to find tight bounds on the confidence of a rule, given its subrules.

The problem we attack can be formulated as follows. Given the set \mathcal{R} of association rules (with respect to a given frequency threshold, confidence threshold, and database \mathcal{D}), choose a subset $\mathcal{R}' \subset \mathcal{R}$ such that the confidence of every pruned rule $R \in \mathcal{R} \setminus \mathcal{R}'$ can be derived up to a user-specified error limit, possibly zero, from its subrules. Rule $X' \Rightarrow Y'$ is a *subrule* of $X \Rightarrow Y$ iff $X' \subseteq X$ and $Y' \subseteq Y$; selecting only the subrules to derive the confidence of a given rule should improve the understandability of the results. (In this paper, the term subrule will refer to proper subrules, i.e., subrules not equal to the original rule.) In other words, rule $X \Rightarrow Y$ is *derivable* and redundant, if its confidence can be derived from the confidences and supports of its subrules; otherwise it is *non-derivable*. Note that being derivable is a function of the subrules: the actual rule confidence and support are not needed for knowing whether the rule is derivable.

Before going to the methods, we would like to remind the readers that obviously redundancy is not the only reason why some association rules are uninteresting. Interestingness is often subjective, and tools such as templates or other syntactical constraints can be very useful. Subjective interestingness is, however, outside the scope of this paper.

3 Non-Derivable Association Rules

We now show how to derive lower and upper bounds for the confidence of an association rule, given its subrules. We start by reviewing the technique to derive bounds on the support of a set [CG02].

3.1 Sets The main principle behind the support derivation technique used for mining non-derivable sets is the inclusion-exclusion principle [GS00]. For any subset $J \subseteq I$, we obtain a lower or an upper bound on the support of I using one of the following formulas.

If $|I \setminus J|$ is odd, then

$$(3.1) \quad \text{supp}(I) \leq \sum_{J \subseteq X \subset I} (-1)^{|I \setminus X|+1} \text{supp}(X).$$

If $|I \setminus J|$ is even, then

$$(3.2) \quad \text{supp}(I) \geq \sum_{J \subseteq X \subset I} (-1)^{|I \setminus X|+1} \text{supp}(X).$$

For example, in Figure 1, we show all possible rules to derive the bounds for a given set $\{abcd\}$.

When the smallest upper bound equals the highest lower bound, then we have actually obtained the exact support of the set solely based on the supports of its subsets. These sets are called *derivable*, and all other sets *non-derivable*. The collection of non-derivable sets has several nice properties.

PROPERTY 3.1. [CG02] *The size of the largest non-derivable set is at most $1 + \log |D|$ where $|D|$ denotes the total number of transactions in the database.*

PROPERTY 3.2. [CG02] *The collection of non-derivable sets is downward closed. In other words, all supersets of a derivable set are derivable, and all subsets of a non-derivable set are non-derivable.*

A less desirable property is that the number of bounds for a given itemset is exponential in the size of the itemset. For more results and discussions, we refer the interested reader to [CG02].

3.2 Association Rules Now, consider a rule $X \Rightarrow Y$ and assume all its (proper) subrules are known, i.e., their supports and confidences are given and hence, also the support of all proper subsets of $X \cup Y$. In order to compute bounds for the confidence of that rule, we bound the support of $X \cup Y$ using the above described technique and divide the lower and upper bound by the support of X , resulting in a lower and upper bound for the confidence of $X \Rightarrow Y$. The goal is to find and remove all *derivable* association rules, i.e., rules for which the lower and the upper bounds of confidence are equal. From this procedure, the following property is readily verified.

PROPERTY 3.3. *Given all (proper) subrules of association rule $X \Rightarrow Y$: $X \Rightarrow Y$ is derivable if and only if $X \cup Y$ is a derivable set.*

This leads to an association rule pruning method which can be represented as a simple modification to the original association rule generation algorithm in which only non-derivable itemsets are used.

Note that when considered as sets in separation, X can be a non-derivable itemset while the set $X \cup Y$ is a derivable itemset, cfr. Property 3.2. A straightforward application of non-derivability of itemsets to association rule mining would be to output rules in which the condition X is non-derivable (regardless of whether the union $X \cup Y$ is).

We next consider some interesting, more restricted cases of pruning. When considering the possible redundancy of a specific association rule, it is probably natural and easier to focus only on those rules which have exactly the same condition or exactly the same consequent. Such a compromise results in less pruning but is likely to increase the understandability of pruning.

$$\begin{aligned}
\text{supp}(abcd) &\geq \text{supp}(abc) + \text{supp}(abd) + \text{supp}(acd) + \text{supp}(bcd) - \text{supp}(ab) - \text{supp}(ac) - \text{supp}(ad) \\
&\quad - \text{supp}(bc) - \text{supp}(bd) - \text{supp}(cd) + \text{supp}(a) + \text{supp}(b) + \text{supp}(c) + \text{supp}(d) - \text{supp}(\{\}) \\
\text{supp}(abcd) &\leq \text{supp}(a) - \text{supp}(ab) - \text{supp}(ac) - \text{supp}(ad) + \text{supp}(abc) + \text{supp}(abd) + \text{supp}(acd) \\
\text{supp}(abcd) &\leq \text{supp}(b) - \text{supp}(ab) - \text{supp}(bc) - \text{supp}(bd) + \text{supp}(abc) + \text{supp}(abd) + \text{supp}(bcd) \\
\text{supp}(abcd) &\leq \text{supp}(c) - \text{supp}(ac) - \text{supp}(bc) - \text{supp}(cd) + \text{supp}(abc) + \text{supp}(acd) + \text{supp}(bcd) \\
\text{supp}(abcd) &\leq \text{supp}(d) - \text{supp}(ad) - \text{supp}(bd) - \text{supp}(cd) + \text{supp}(abd) + \text{supp}(acd) + \text{supp}(bcd) \\
\text{supp}(abcd) &\geq \text{supp}(abc) + \text{supp}(abd) - \text{supp}(ab) \\
\text{supp}(abcd) &\geq \text{supp}(abc) + \text{supp}(acd) - \text{supp}(ac) \\
\text{supp}(abcd) &\geq \text{supp}(abd) + \text{supp}(acd) - \text{supp}(ad) \\
\text{supp}(abcd) &\geq \text{supp}(abc) + \text{supp}(bcd) - \text{supp}(bc) \\
\text{supp}(abcd) &\geq \text{supp}(abd) + \text{supp}(bcd) - \text{supp}(bd) \\
\text{supp}(abcd) &\geq \text{supp}(acd) + \text{supp}(bcd) - \text{supp}(cd) \\
\text{supp}(abcd) &\leq \text{supp}(abc) \\
\text{supp}(abcd) &\leq \text{supp}(abd) \\
\text{supp}(abcd) &\leq \text{supp}(acd) \\
\text{supp}(abcd) &\leq \text{supp}(bcd) \\
\text{supp}(abcd) &\geq 0
\end{aligned}$$

Figure 1: Bounds on $\text{supp}(abcd)$.

3.3 Fixed Consequent First we consider the case of a fixed consequent. In other words, the derivability (redundancy) of a rule is a function of those subrules that explain the same consequent. We handle this case as two separate subclasses of rules, those with a single item consequent and those with multiple items in the consequent.

First consider rules $X \Rightarrow Y$ with $|Y| = 1$. Given all its subrules with the same consequent and their respective supports and confidences, we immediately obtain the supports of all subsets of $X \cup Y$, except of the sets X and $X \cup Y$ themselves.

EXAMPLE 1. Consider the rule $abc \Rightarrow d$. From each of its subrules, e.g., $ab \Rightarrow d$, we obtain the support of two subsets of $abcd$: the support of abd (the support of the rule) and the support of ab (the support of the rule divided by its confidence).

rule	sets
$ab \Rightarrow d$	ab, abd
$ac \Rightarrow d$	ac, acd
$bc \Rightarrow d$	bc, bcd
$a \Rightarrow d$	a, ad
$b \Rightarrow d$	b, bd
$c \Rightarrow d$	c, cd
$\{\} \Rightarrow d$	$\{\}, d$

The only two subsets of $abcd$ that are missing are abc and $abcd$, i.e., exactly those needed to compute the confidence of the desired rule.

Thus, given the subrules of $X \Rightarrow Y$ with the same consequent, the support of X can be directly bounded.

For bounding the support of $X \cup Y$, however, information about X is missing, and we cannot simply use all derivation formulas. To solve this, we first compute the bounds for X , and then we compute the bounds for $X \cup Y$ for every possible value of X . As a result, we have a set of triples (v, l, u) with v a possible support value for X and l and u the corresponding lower and upper bound for $X \cup Y$ respectively.

EXAMPLE 2. Suppose we want to bound the confidence of the rule $ab \Rightarrow c$, given the following supports.

$\text{supp}(ac)$	$= 3$
$\text{supp}(bc)$	$= 3$
$\text{supp}(a)$	$= 7$
$\text{supp}(b)$	$= 7$
$\text{supp}(c)$	$= 5$
$\text{supp}(\{\})$	$= 10$

Then, bounding ab results in a lower bound of $4 = 7 + 7 - 10 = \text{supp}(a) + \text{supp}(b) - \text{supp}(\{\})$, and an upper bound of $7 = \text{supp}(a) = \text{supp}(b)$. Then for every possible value of the support of ab , we compute the bounds for the support of abc and the corresponding bounds for the confidence of $ab \Rightarrow c$.

	$\text{supp}(abc)$	$\text{conf}(ab \Rightarrow c)$
$\text{supp}(ab) = 4$	$[1, 1]$	$[1/4, 1/4]$
$\text{supp}(ab) = 5$	$[1, 2]$	$[1/5, 2/5]$
$\text{supp}(ab) = 6$	$[2, 3]$	$[2/6, 3/6]$
$\text{supp}(ab) = 7$	$[3, 3]$	$[3/7, 3/7]$

Hence, we can conclude that the confidence interval of $ab \Rightarrow c$ is $[1/5, 1/2]$.

As the example above shows, it is not sufficient to use only values at the lower and the upper bounds of X when computing the bounds for $X \cup Y$: the extreme values for the confidence may occur at intermediate possible values of X .

Also note that a rule $X \Rightarrow Y$ can be derivable even if X is not. This is the case when all the bounds of $X \cup Y$, for every possible value of X , result in the same equal upper and lower bound on the confidence of $X \Rightarrow Y$, as illustrated in the following example.

EXAMPLE 3. Suppose we want to bound the confidence of the rule $ab \Rightarrow c$, given the following supports.

$supp(ac)$	$= 7$
$supp(bc)$	$= 7$
$supp(a)$	$= 7$
$supp(b)$	$= 7$
$supp(c)$	$= 10$
$supp(\{\})$	$= 10$

Then, bounding ab results in a lower bound of $4 = 7 + 7 - 10 = supp(a) + supp(b) - supp(\{\})$, and an upper bound of $7 = supp(a) = supp(b)$. Then for every possible value of the support of ab , we compute the bounds for the support of abc and the corresponding bounds for the confidence of $ab \Rightarrow c$.

	$supp(abc)$	$conf(ab \Rightarrow c)$
$supp(ab) = 4$	$[4, 4]$	$[1, 1]$
$supp(ab) = 5$	$[5, 5]$	$[1, 1]$
$supp(ab) = 6$	$[6, 6]$	$[1, 1]$
$supp(ab) = 7$	$[7, 7]$	$[1, 1]$

Therefore, we can conclude that the confidence of $ab \Rightarrow c$ is 1, and hence, derivable.

When the consequent of a rule $X \Rightarrow Y$ consists of more than one item, then its subrules with the same consequent do no longer provide the supports for all necessary subsets of $X \cup Y$. Although we can still derive tight bounds for X using the usual inclusion-exclusion formulas, it becomes a lot more complex to derive the bounds for $X \cup Y$.

EXAMPLE 4. Consider the rule $abc \Rightarrow de$. From the support and confidence of each of its subrules with the same consequent, we again obtain the support of exactly 2 subsets of $abcde$, i.e., the support of the conditions of the subrules and the support of the sets containing the conditions and the consequent.

$ab \Rightarrow de$	$ab, abde$
$ac \Rightarrow de$	$ac, acde$
$bc \Rightarrow de$	$bc, bcde$
$a \Rightarrow de$	a, ade
$b \Rightarrow de$	b, bde
$c \Rightarrow de$	c, cde
$\{\} \Rightarrow de$	$\{\}, de$

Hence, apart from the missing supports of the subsets abc and $abcde$, we now also don't have any information on the supports of $d, e, ad, ae, bd, be, cd, ce, abd, abe, acd, ace, bcd, bce$.

Since the consequents of all these rules are the same, we can solve this problem by simply considering the consequent as a single item which occurs in a transaction only if all items in the consequent occur in that transaction. In that way, the problem of multiple items in the consequent is reduced to the case in which only a single item occurs in the consequent, and hence, can be solved as described before.

3.4 Fixed Condition or Consequent We now study the case where the considered subrules have either the same condition or the same consequent as the original rule. The motivation for this approach is that it is likely to be easier for the user to understand redundancy with respect to such subrules than all possible subrules.

To find such non-derivable rules, the first observation is that we can divide the problem into two parts: (1) obtain confidence bounds with fixed consequent subrules, as described in the previous subsection, and with fixed condition subrules (to be described below), and then (2) output the intersection of the possible intervals as the result.

To bound the confidence of $X \Rightarrow Y$ when only those subrules are known that have X as the condition, we need to bound the support of $X \cup Y$, as the support of X is given. To find the bounds, we simply restrict ourselves to those inclusion-exclusion formulas containing only terms that are supersets of X .

3.5 Using Only Some Subrules From an intuitive point of view, it makes sense to measure the value or interestingness of an association rule by comparing to its subrules. As described above, this is exactly what happens when we compute the bounds on the confidence of an association rule using the inclusion-exclusion principle. Unfortunately, for larger sets, the inclusion-exclusion formulas can become quite large and complex, and hence, not so intuitive anymore. Therefore, we also consider the case in which only those subrules with a condition of a minimum size are allowed to be used.

More specifically, for any subset $J \subseteq I$, we obtain a lower or an upper bound on the support of I using one of the formulas in (3.1) or (3.2), but now, we only allow the formulas to be used for those subsets $J \subseteq I$ such that $|I \setminus J| \geq k - 1$, for a user given parameter $k > 0$. We also call this parameter the allowable *depth* of the rules to be used. In Figure 1, the formulas are shown in descending order of depth, starting with depth 5.

In our case we bound not one, but two sets which differ by one in size. We use depth $k - 1$ for the condition of the rule and depth k for all items in the rule.

Dataset	#items	trans. size	#trans.	support threshold
chess	76	37	3 196	70% (2238)
connect	130	43	67 557	90% (60802)
mushroom	120	23	8 124	20% (1625)
pumsb	7117	74	49 046	85% (41690)

Table 1: Dataset characteristics

4 Experiments

For an experimental evaluation of the proposed algorithms, we performed several experiments on real datasets also used in [Zak00]. We implemented the proposed algorithms in C++, and for comparison to recent methods we use the original authors’ own implementations [LH04, JS02, Zak00, ZP03].

All datasets were obtained from the UCI Machine Learning Repository. The chess and connect datasets are derived from their respective game steps, the mushroom database contains characteristics of various species of mushrooms, and the pumsb dataset contains census data. Table 1 shows some characteristics of the used datasets; for each dataset, we used the lowest support threshold that was mentioned in [Zak00]. The confidence threshold was set to 0% in all experiments.

Figure 2 shows the effect of pruning for the four datasets, as a function of the width of the bound on confidence. Three different variants are shown in each panel (from top to bottom): the number of non-redundant rules when only subrules with identical consequent are used, when only subrules with either identical consequent or identical condition are used, and when all subrules are used. These variants offer different trade-offs between the amount of pruning and how easy it is for the user to understand what was pruned. For a comparison, the number of (minimal) closed rules is also given. (The numbers of minimal closed rules have been obtained with M. Zaki’s implementation. They differ from those reported by him in reference [Zak00], since in the latter one he was not exactly mining minimal rules [M. Zaki, personal communication].)

The immediate observation is that pruning has a dramatic effect on the number of rules (note that the Y axis has a logarithmic scale). In particular, a large amount of rules can be derived exactly. Some of the results are also given in numerical form in Table 2. The table reports results for exactly derivable rules with identical consequent subrules, with identical condition or consequent subrules, or with all subrules. The row “1% interval” was obtained by pruning rules for which the lower and upper bounds of confidence are at most 1 percentage point apart. Results with minimal closed rules are included for comparison.

The number of non-derivable association rules is less

	chess	connect	mushroom	pumsb
All rules	8160101 100%	3667831 100%	19245239 100%	1429297 100%
Identical consequent	1572360 19%	557579 15%	2829208 15%	695871 49%
Id. condition or consequent	65978 0.81%	11231 0.31%	94860 0.49%	177155 12%
All subrules	4181 0.051%	552 0.015%	7546 0.039%	16345 1.1%
All subrules, 1% interval	718 0.0088%	167 0.0046%	5358 0.028%	543 0.038 %
Minimal closed rules	139431 1.7%	15496 0.42%	6815 0.035%	71813 5.0%

Table 2: Number of rules after different pruning methods (absolute number and percentage of all rules).

than the number of minimal closed rules already when using only subrules with identical consequent or condition in chess and connect datasets. In pumsb the number of non-derivable association rules is less than the number of minimal closed rules if we use all subrules to compute the upper and lower bound. In mushroom the number of minimal closed rules is slightly less than the number of non-derivable association rules.

Relatively small error bounds, already in the order of fractions of percent, can result in significant further pruning. For example in the mushroom dataset, the number of non-derivable association rules when using all subrules becomes less than the number of minimal closed rules when we allow the difference of upper and lower bound to be one percentage unit. In other datasets the effect of allowing a small interval for the confidence bounds is even more radical.

A comparison to the maximum entropy technique [JS02] and basic association rules [LH04] is given in Figure 3. It shows the number of non-redundant rules with exactly one item in the consequent, since the two other techniques only find redundancies in such rules. A comparison to the maximum entropy approach shows that sometimes it is quite competitive, but it is not a very robust approach for pruning in these cases. The algorithm is approximative and iterative. As a compromise between efficiency and accuracy, we used exactly 5000 iterations in these tests; each run then took less than a day except for the chess dataset, for which the execution time was over three days. (The steps visible in some of the maximum entropy graphs are due to a limited accuracy in the output of the implementation, they are not inherent in the method itself.)

The trend seems to be that for very low error bounds, the proposed method is always superior. With a growing error bound, the maximum entropy approach sometimes outperforms non-derivable association rules. The number of basic association rules is considerably greater than the

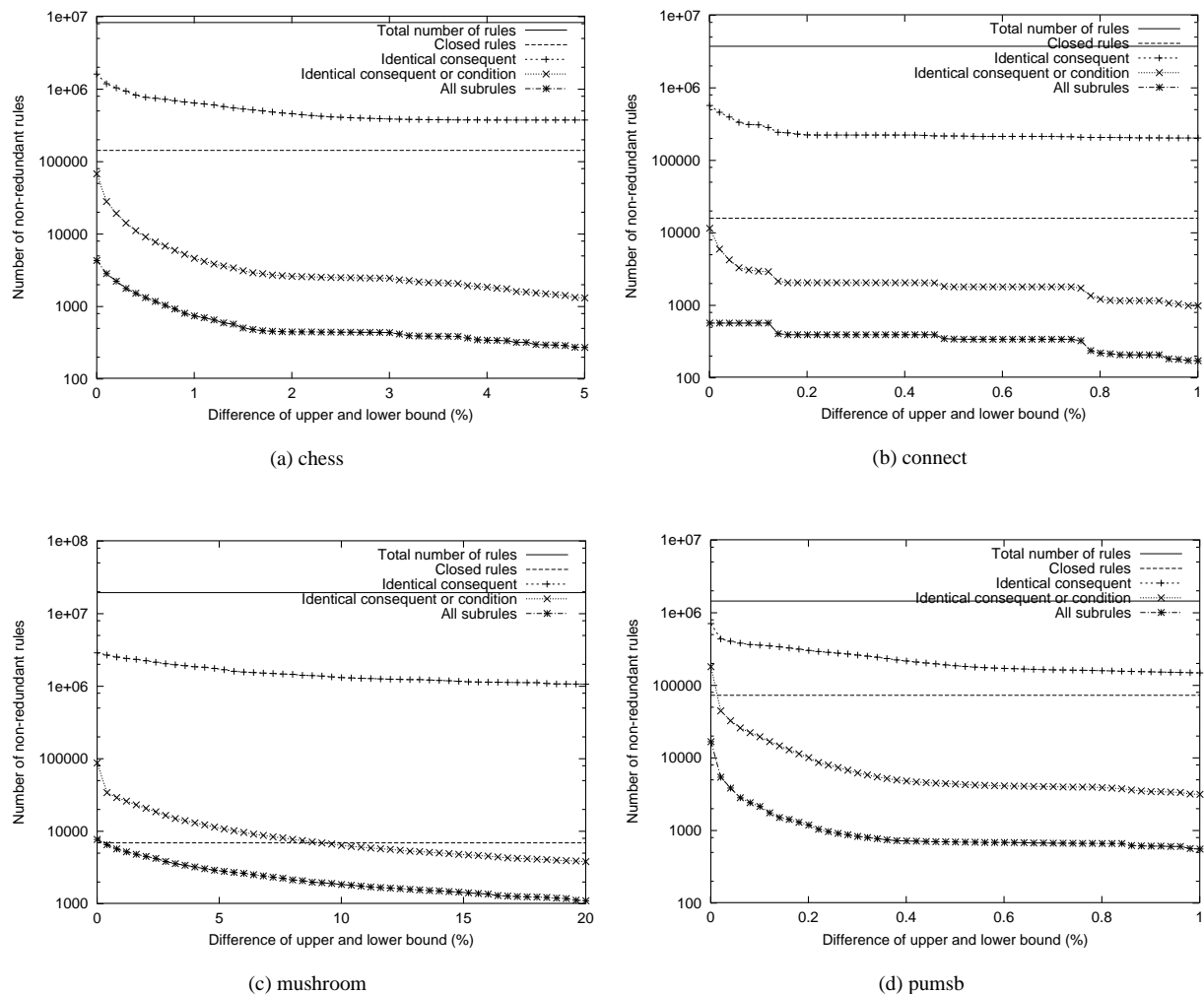


Figure 2: The number of non-derivable and minimal closed association rules.

number of non-derivable rules in all four datasets. As a technique that does not consider error bounds, the basic association rules always outperform the maximum entropy approach in terms of exact inference of rules; sometimes the marginal is quite small, though.

For a further analysis of the proposed method, Figure 4 shows results for different depths of the formulas that were allowed to be used (cf. Section 3.5). This figure only uses association rules with exactly one item in the consequent. The line labeled 'infinite depth' denotes the number of non-derivable rules when all possible formulas are allowed to be used. Additionally, the figure also shows the number of association rules for which the condition is a non-derivable itemset. Since this is a straightforward pruning mechanism based on the notion of non-derivable sets, it shows from where the actual power of the presented confidence derivation tech-

nique starts.

A remarkable result is that most of the derivable rules are already derivable when only the inclusion-exclusion formulas up to depth 3 are allowed to be used. Such a result is particularly nice for the end user, since it means that the reasons for redundancy of a rule are mostly in the most immediate subrules, making the pruning more intuitive and easy to understand.

Finally, Figure 5 shows the number of rules as a function of the support thresholds much lower than those presented in [Zak00]; again with a singular consequent. In these figures, an association rule was considered to be non-redundant if the width of its confidence bound was more than 0.1%. According to the figure, the presented technique scales very well to low support thresholds and achieves roughly similar reductions in the number of association rules across the

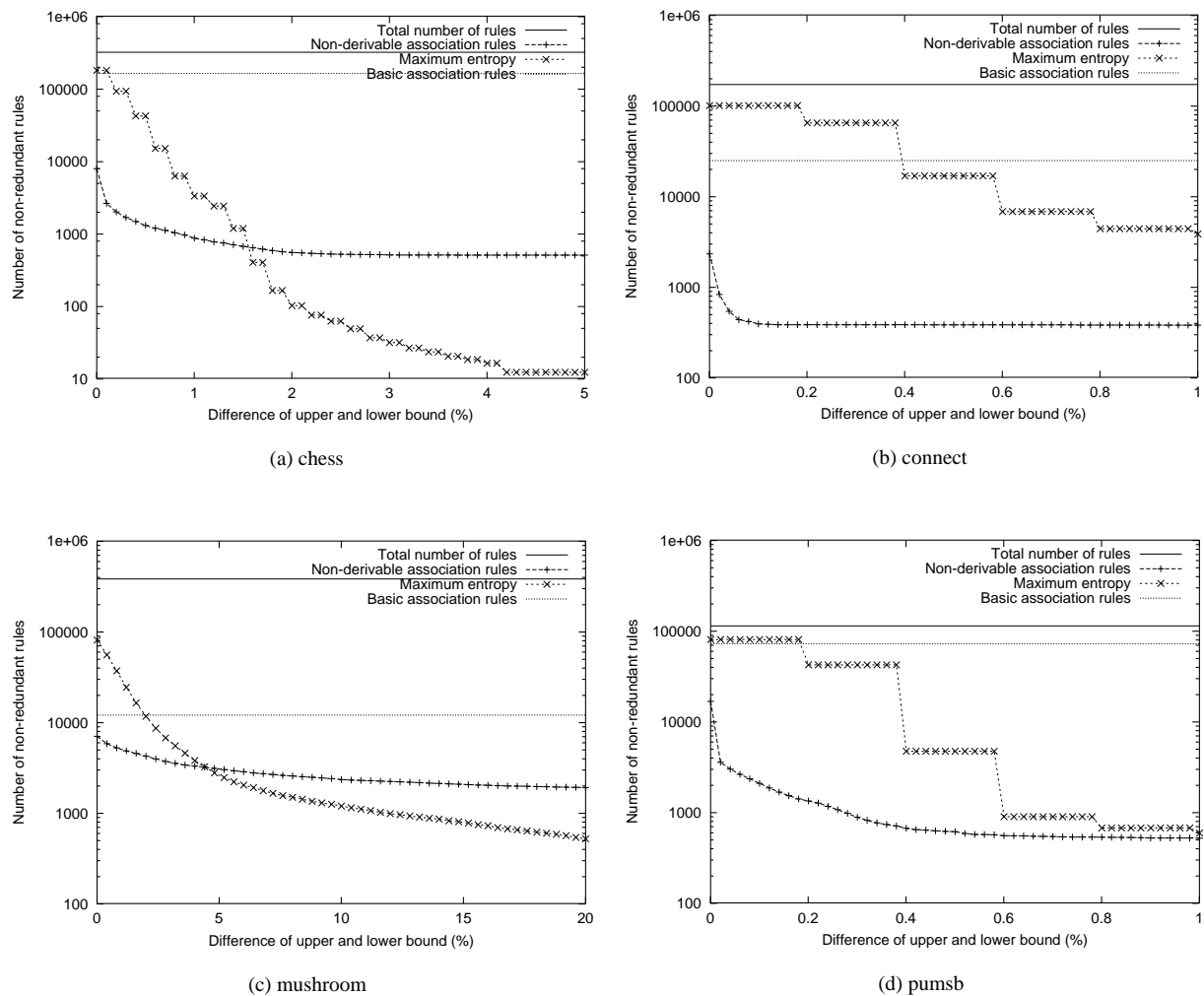


Figure 3: Number of non-derivable and basic association rules and rules produced by maximum entropy method.

ranges tested.

5 Conclusions

We presented a solid foundation for computing upper and lower bounds of the confidence of an association rule, given its subrules. When the upper and lower bounds are equal or almost equal, we call the association rule derivable and consider it to be redundant with respect to its subrules. The presented technique is based on the inclusion–exclusion principle, recently successfully used for bounding the support of sets of items [CG02]. The method is simple, it gives absolute bounds, and it does not assume any specific inference system. The bounds and derivability follow from the definitions of support and confidence: when a rule is pruned as exactly derivable, then there exists only one value for the confidence that is consistent with all the subrules.

Experimental results with real data sets demonstrated very high pruning power. In our experiments, up to 99–99.99% of rules were exactly derivable, and always over 99.96% derivable within $\pm 0.5\%$ points. The amount of pruning depends a lot on data set characteristics as well as on the support threshold: the lower the threshold, the more redundant is the rule set. In absolute terms, the figures indicate great practical significance.

In comparison to related techniques, it is surprising how efficient the proposed simple method is. The related techniques almost invariably make strong assumptions, in the form of fixing an inference system or an estimation method. In the face of the experimental results, our simple and consistent bounding can give much higher pruning factors without any such assumptions.

We gave three different variants of the method, using

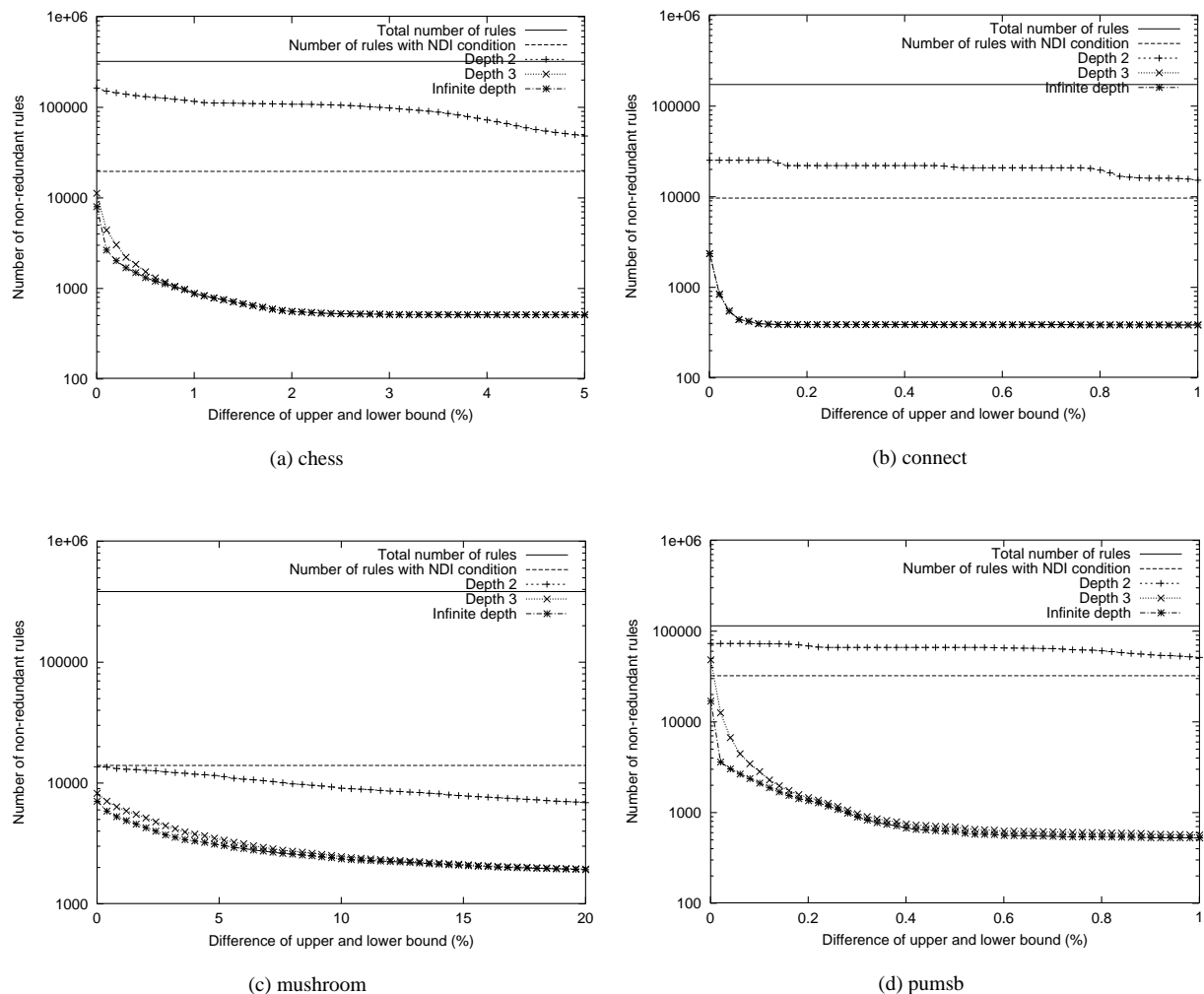


Figure 4: The number of non-derivable association rules with a singular consequent.

different sets of subrules to obtain the confidence constraints. They have different trade-offs between the amount of pruning and understandability of pruning. An evaluation of different pruning mechanisms from the end user point of view is a topic for further work.

An important and valid critique on the proposed techniques is that in practice we do not actually have all subrules of an association rule as some of them might not be confident. Indeed, in our experiments, we never used the confidence threshold for pruning, i.e. it was set to 0. Nevertheless, also for higher minimum confidence thresholds, it is always easy to simply compute the actual confidence of all necessary subrules given the frequent itemsets. Furthermore, our experiments show that the numbers of frequent non-derivable association rules are extremely small without using a confidence threshold. Note that in practice, it is not always clear

which confidence threshold should be used and rules with small confidence can sometimes even be extremely interesting.

Nevertheless, in future work, we will explore a sequential pruning mechanism in which only subrules are used that are confident and that were not already pruned earlier.

Acknowledgements

We would like to thank G. Li and H. Hamilton [LH04], S. Jaroszewicz and D. A. Simovici [JS02] and M. Zaki [Zak00, ZP03] for kindly providing implementations of their methods.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami.

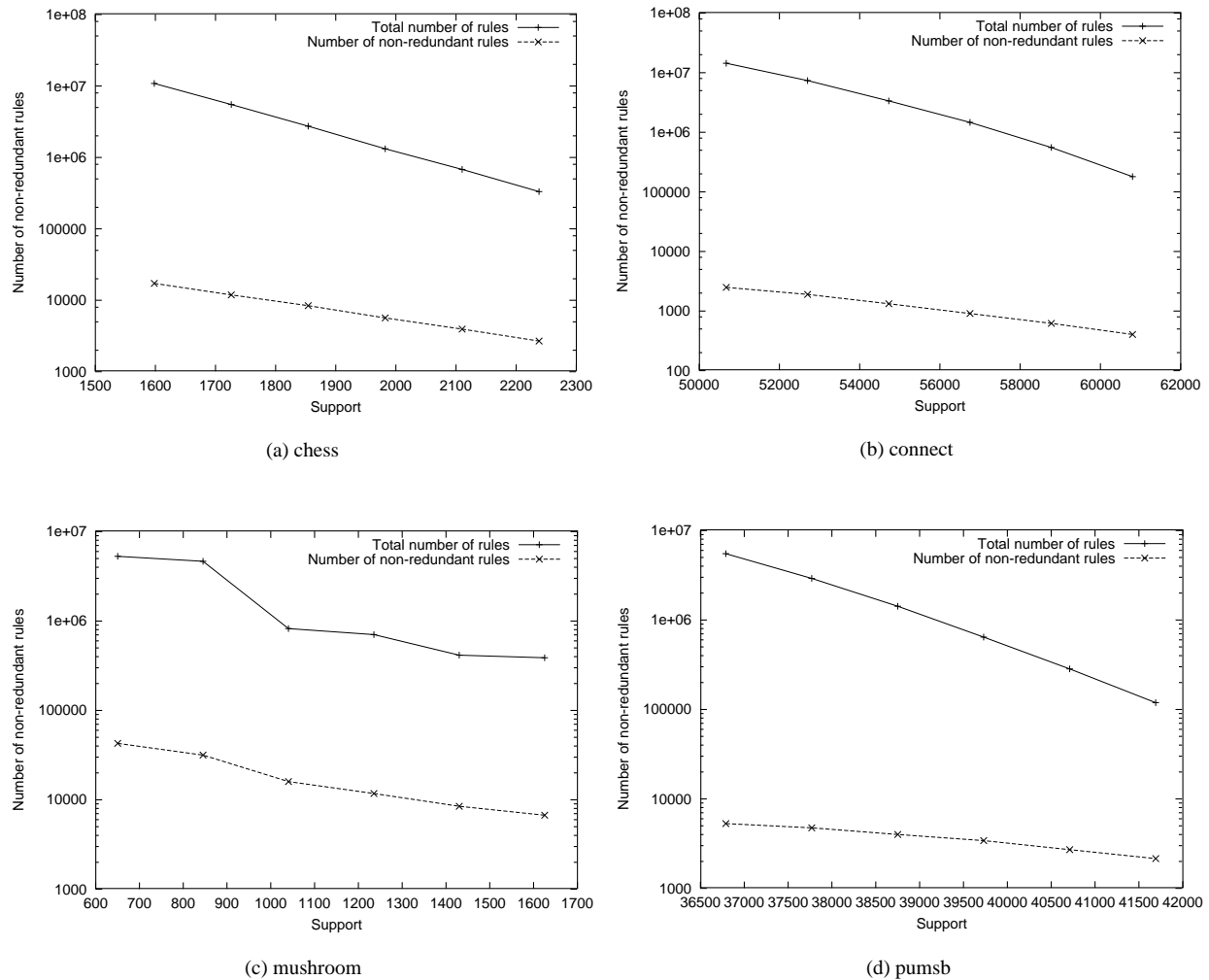


Figure 5: The number of non-derivable association rules for different support thresholds.

- Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914 – 925, December 1993. Special Issue on Learning and Discovery in Knowledge-Based Databases.
- [BBR00] J-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, pages 75–85, Lyon, France, 2000. Springer.
- [BPT⁺00] Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational Logic – CL 2000: First International Conference*, pages 972 – 986, London, UK, 2000.
- [CG02] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, volume 2431 of *Lecture Notes in Computer Science*, pages 74–85. Springer, 2002.
- [GS00] J. Galambos and I. Simonelli. *Bonferroni-type Inequalities with Applications*. Springer, 2000.
- [GVdB00] B. Goethals and J. Van den Bussche. On supporting interactive association rule mining. In Y. Kambayashi, M.K. Mohania, and A.M. Tjoa, editors, *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery*, volume 1874 of *Lecture Notes in Computer Science*, pages 307–316. Springer, 2000.
- [JS02] S. Jaroszewicz and D. A. Simovici. Pruning redundant association rules using maximum entropy principle. In *Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference, PAKDD'02*, pages 135–147, Taipei, Taiwan, May 2002.
- [KMR⁺94] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In

- Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401 – 407, Gaithersburg, MD, USA, November 1994. ACM.
- [LH04] Guichong Li and Howard J. Hamilton. Basic association rules. In *Fourth SIAM International Conference on Data Mining*, Florida, USA, 2004.
- [MPS99] Heikki Mannila, Dmitry Pavlov, and Padhraic Smyth. Prediction with local patterns using cross-entropy. In *Proceedings of the ACM SIGKDD*, pages 357–361. ACM Press, 1999.
- [NLHP98] R.T. Ng, L.V.S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In L.M. Haas and A. Tiwary, editors, *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, volume 27(2) of *SIGMOD Record*, pages 13–24. ACM Press, 1998.
- [PBTL99] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 398–416. Springer, 1999.
- [Zak00] Mohammed J. Zaki. Generating non-redundant association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 34 – 43, Boston, MA, USA, 2000.
- [ZP03] Mohammed Zaki and Benjarath Phoophakdee. MIRAGE: A framework for mining, exploring and visualizing minimal association rules. Technical Report RPI CS Dept Technical Report 03-04, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York, July 2003.

Depth-First Non-Derivable Itemset Mining

Toon Calders*

University of Antwerp, Belgium
toon.calders@ua.ac.be

Bart Goethals†

HIIT-BRU, University of Helsinki, Finland
bart.goethals@cs.helsinki.fi

Abstract

Mining frequent itemsets is one of the main problems in data mining. Much effort went into developing efficient and scalable algorithms for this problem. When the support threshold is set too low, however, or the data is highly correlated, the number of frequent itemsets can become too large, independently of the algorithm used. Therefore, it is often more interesting to mine a reduced collection of interesting itemsets, i.e., a condensed representation. Recently, in this context, the *non-derivable* itemsets were proposed as an important class of itemsets. An itemset is called derivable when its support is completely determined by the support of its subsets. As such, derivable itemsets represent redundant information and can be pruned from the collection of frequent itemsets. It was shown both theoretically and experimentally that the collection of non-derivable frequent itemsets is in general much smaller than the complete set of frequent itemsets. A breadth-first, Apriori-based algorithm, called NDI, to find all non-derivable itemsets was proposed. In this paper we present a depth-first algorithm, dfNDI, that is based on Eclat for mining the non-derivable itemsets. dfNDI is evaluated on real-life datasets, and experiments show that dfNDI outperforms NDI with an order of magnitude.

1 Introduction

Since its introduction in 1993 by Agrawal et al. [3], the frequent itemset mining problem has received a great deal of attention. Within the past decade, hundreds of research papers have been published presenting new algorithms or improvements on existing algorithms to solve this mining problem more efficiently.

The problem can be stated as follows. We are given a set of items \mathcal{I} , and an *itemset* $I \subseteq \mathcal{I}$ is some set of items. A *transaction* over \mathcal{I} is a couple $T = (tid, I)$ where tid is the transaction identifier and I is an itemset. A transaction $T = (tid, I)$ is said to *support* an itemset $X \subseteq \mathcal{I}$, if $X \subseteq I$. A *transaction database* \mathcal{D} over \mathcal{I} is a set of transactions over \mathcal{I} . We omit \mathcal{I} whenever it is clear from the context. The *cover* of an itemset X in \mathcal{D} consists of the set of transaction identifiers of transactions in \mathcal{D} that support X : $cover(X, \mathcal{D}) := \{tid \mid (tid, I) \in \mathcal{D}, X \subseteq I\}$. The

support of an itemset X in \mathcal{D} is the number of transactions in the cover of X in \mathcal{D} : $support(X, \mathcal{D}) := |cover(X, \mathcal{D})|$. An itemset is called *frequent* in \mathcal{D} if its support in \mathcal{D} exceeds the minimal support threshold σ . \mathcal{D} and σ are omitted when they are clear from the context. The goal is now to find all frequent itemsets, given a database and a minimal support threshold.

Recent studies on frequent itemset mining algorithms resulted in significant performance improvements: a first milestone was the introduction of the breadth-first Apriori-algorithm [4]. In the case that a slightly compressed form of the database fits into main memory, even more efficient, depth-first, algorithms such as Eclat [18, 23], and FP-growth [12] were developed.

However, independently of the chosen algorithm, if the minimal support threshold is set too low, or if the data is highly correlated, the number of frequent itemsets itself can be prohibitively large. No matter how efficient an algorithm is, if the number of frequent itemsets is too large, mining all of them becomes impossible.

To overcome this problem, recently several proposals have been made to construct a condensed representation [15] of the frequent itemsets, instead of mining all frequent itemsets. A condensed representation is a sub-collection of all frequent itemsets that still contains all information. The most well-known example of a condensed representation are the *closed sets* [5, 7, 16, 17, 20]. The closure $cl(I)$ of an itemset I is the largest superset of I such that $supp(cl(I)) = supp(I)$. A set I is *closed* if $cl(I) = I$. In the closed sets representation only the frequent closed sets are stored. This representation still contains all information of the frequent itemsets, because for every set I it holds that

$$supp(I) = \max\{supp(C) \mid I \subseteq C, cl(C) = C\}.$$

Another important class of itemsets in the context of condensed representations are the *non-derivable* itemsets [10]. An itemset is called *derivable* when its support is completely determined by the support of its subsets. As such, derivable itemsets represent redundant information and can be pruned from the collection of frequent itemsets. For an itemset, it can be checked whether or not it is derivable by computing bounds on the support. In [10], a method based on the inclusion-exclusion principle is used.

*Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium)(F.W.O. - Vlaanderen).

†Current affiliation: University of Antwerp, Belgium.

It was shown both theoretically and experimentally that the collection of non-derivable frequent itemsets is in general much more concise than the complete set of frequent itemsets. It was proven that for a given database \mathcal{D} , all sets of length more than $\log_2(|\mathcal{D}|) + 1$ are derivable. Hence, especially in databases with numerous items, but with few transactions, the number of non-derivable itemsets is guaranteed to be relatively small compared to the number of frequent itemsets. Many biological datasets, e.g., gene expression datasets, are typical examples of such databases. In experiments in [10], it was shown empirically that the number of non-derivable itemsets is in general orders of magnitudes smaller than the number of frequent itemsets. In most experiments, the number of non-derivable itemsets was even smaller than the number of closed sets.

In [10], a breadth-first, Apriori-based algorithm NDI to find all non-derivable itemsets was proposed. Due to the relatively small number of non-derivable itemsets, the NDI-algorithm almost always outperforms mining all frequent itemsets, independently of the algorithm [8]. When we, however, look at the time and space required by the NDI-algorithm as a function of its output-size, its performance is far below that of state-of-the-art frequent set mining algorithms. The low efficiency of NDI comes mainly from the fact that it is a breadth-first generate-and-test algorithm. All candidates of a certain level need to be processed simultaneously, and the support tests involve repeated scans over the complete database.

In contrast, in the case of mining all frequent itemsets, depth-first algorithms have been shown to perform much better and have far less costly candidate generation phases, and do not require scanning the complete database over and over again. Furthermore, item reordering techniques can be used to avoid the generation of too many candidates.

Unfortunately, depth-first algorithms essentially do not perform the so called Apriori-test, that is, test whether all of subsets of a generated candidate are known to be frequent, as most of them are simply not generated yet. Nevertheless, the supports of the subsets of a given set is exactly what is needed in order to determine whether an itemset is derivable or not. In this paper, we tackle this problem and present a depth-first algorithm, dfNDI, for mining non-derivable itemsets. The dfNDI-algorithm is based on Eclat and the diffset technique as introduced by Zaki et al. [18, 19]. As such, dfNDI combines the efficiency of depth-first itemset mining algorithms with the significantly lower number of non-derivable itemsets, resulting in an efficient algorithm for mining non-derivable itemsets.

The organization of the paper is as follows. In Section 2, the non-derivable itemsets and the level-wise NDI algorithm are revisited. In Section 3, we shortly describe the Eclat-algorithm on which our dfNDI-algorithm is based. Special attention is paid to item reordering techniques. Section

4 introduces the new algorithm dfNDI in depth, which is experimentally evaluated and compared to the level-wise NDI in Section 5.

2 Non-Derivable Itemsets

In this section we revisit the non-derivable itemsets introduced in [10]. In [10], rules were given to derive bounds on the support of an itemset I if the supports of all its strict subsets of I are known.

2.1 Deduction Rules Let a *generalized itemset* be a conjunction of items and negations of items. For example, $G = \{a, b, \bar{c}, d\}$ is a generalized itemset. A transaction (tid, I) contains a general itemset $G = X \cup \bar{Y}$ if $X \subseteq I$ and $I \cap Y = \emptyset$. The *support of a generalized itemset G in a database \mathcal{D}* is the number of transactions of \mathcal{D} that contain G .

We say that a general itemset $G = X \cup \bar{Y}$ is *based on itemset I* if $I = X \cup Y$. From the well known inclusion-exclusion principle [11], we know that for a given general itemset $G = X \cup \bar{Y}$ based on I ,

$$\text{supp}(G) = \sum_{X \subseteq J \subseteq I} (-1)^{|J \setminus X|} \text{supp}(J) .$$

Hence, $\text{supp}(I)$ equals

$$(2.1) \quad \sum_{X \subseteq J \subseteq I} (-1)^{|I \setminus J|+1} \text{supp}(J) + (-1)^{|Y|} \text{supp}(G)$$

Notice that depending on the sign of $(-1)^{|Y|} \text{supp}(G)$, the term

$$\delta_X(I) := \sum_{X \subseteq J \subseteq I} (-1)^{|I \setminus J|} \text{supp}(J)$$

is a lower ($|Y|$ even) or an upper ($|Y|$ odd) approximation for the support of I . In [10], this observation was used to compute lower and upper bounds on the support of an itemset I . For each set I , let l_I (u_I) denote the lower (upper) bound we can derive using the deduction rules. That is,

$$\begin{aligned} l_I &= \max\{\delta_X(I) \mid X \subseteq I, |I \setminus X| \text{ odd}\} , \\ u_I &= \min\{\delta_X(I) \mid X \subseteq I, |I \setminus X| \text{ even}\} . \end{aligned}$$

Since we need the supports of all strict subsets of I to compute the bounds, l_I and u_I clearly depend on the underlying database.

EXAMPLE 1. Consider the following database \mathcal{D} :

TID	Items		
1	a, b, c, d	6	a, b, e
2	a, b, c	7	a, c, e
3	a, b, d, e	8	a, d, e
4	c, e	9	b, c, e
5	b, d, e	10	b, d, e

Some deduction rules for abc are the following:

$$\begin{aligned} \text{supp}(abc) &\geq \text{supp}(ab) + \text{supp}(ac) - \text{supp}(a) = 1 \\ \text{supp}(abc) &\leq \text{supp}(ab) + \text{supp}(ac) + \text{supp}(bc) \\ &\quad - \text{supp}(a) - \text{supp}(b) - \text{supp}(c) \\ &\quad + \text{supp}(\{\}) = 2 \end{aligned}$$

Hence, based on the supports of the subsets of abc , we can deduce that $\text{supp}(abc)$ is in $[1, 2]$.

In this paper, we will use Equation (2.1) to compute the support of I , based on the supports of its subsets, and the support of the generalized itemset G .

EXAMPLE 2. Some equalities for itemset abc :

$$\begin{aligned} \text{supp}(abc) &= \text{supp}(ab) - \text{supp}(ab\bar{c}) \\ &= (\text{supp}(ab) + \text{supp}(ac) + \text{supp}(bc) \\ &\quad - \text{supp}(a) - \text{supp}(b) - \text{supp}(c) \\ &\quad + \text{supp}(\{\})) - \text{supp}(ab\bar{c}) \end{aligned} \quad \square$$

Notice incidentally that the complexity of the term $\delta_X(I)$ depends on the cardinality of $Y = I \setminus X$. The larger Y is, the more complex it will be to compute the support of I based on $X \cup \bar{Y}$. For example, for $abcd$,

$$\begin{aligned} \delta_{abc}(abcd) &= abc \\ \delta_a(abcd) &= abc + abd + acd - ab - ac - ad + a \end{aligned}$$

In general, the number of terms in $\delta_X(I)$ is $2^{|I \setminus X|} - 1$. Therefore, whenever in the algorithm we have the choice between two generalized itemsets, we will always choose the set with the least negations, as this set will result in a more efficient calculation of the support of I .

2.2 Condensed Representation Based on the deduction rules, it is possible to generate a summary of the set of frequent itemsets. Indeed, if $l_I = u_I$, then $\text{supp}(I, \mathcal{D}) = l_I = u_I$, and hence, we do not need to store I in the representation. Such a set I , will be called a *Derivable Itemset* (DI), all other itemsets are called *Non-Derivable Itemsets* (NDIs). Based on this principle, in [10], the following condensed representation was introduced:

$$\text{NDI}(\mathcal{D}, \sigma) := \{I \mid \text{supp}(I, \mathcal{D}) \geq \sigma, l_I \neq u_I\}.$$

In the experiments presented in Section 5, it is shown that the collection of non-derivable itemsets is much more concise than the complete collection of frequent itemsets, and often even more concise than other concise representations. For a discussion on the relation between NDI and the other condensed representation we refer to [9].

2.3 The NDI Algorithm In [10], in a slightly different form, the following theorem was proven:

Algorithm 1 Eclat

Input: $\mathcal{D}, \sigma, I \subseteq \mathcal{I}$

Output: $\mathcal{F}[I](\mathcal{D}, \sigma)$

```

1:  $\mathcal{F}[I] := \{\}$ 
2: for all  $i \in \mathcal{I}$  occurring in  $\mathcal{D}$  do
3:    $\mathcal{F}[I] := \mathcal{F}[I] \cup \{I \cup \{i\}\}$ 
4:   // Create  $\mathcal{D}^i$ 
5:    $\mathcal{D}^i := \{\}$ 
6:   for all  $j \in \mathcal{I}$  occurring in  $\mathcal{D}$  such that  $j > i$  do
7:      $C := \text{cover}(\{i\}) \cap \text{cover}(\{j\})$ 
8:     if  $|C| \geq \sigma$  then
9:        $\mathcal{D}^i := \mathcal{D}^i \cup \{(j, C)\}$ 
10:    end if
11:  end for
12:  // Depth-first recursion
13:  Compute  $\mathcal{F}[I \cup \{i\}](\mathcal{D}^i, \sigma)$ 
14:   $\mathcal{F}[I] := \mathcal{F}[I] \cup \mathcal{F}[I \cup \{i\}]$ 
15: end for
```

THEOREM 2.1. [10] (**Monotonicity**) Let $I \subseteq J$ be itemsets. If $\text{supp}(I) = \delta_X(I)$, then, for all X' such that $X \subseteq X' \subseteq X \cup (J \setminus I)$, $\text{supp}(J) = \delta_{X'}(J)$.

Hence, if I is a derivable, then J is derivable as well.

Based on this theorem, a level-wise Apriori-like algorithm was given in [10]. In fact, the NDI-algorithm corresponds largely to constrained mining algorithms with non-derivability as an anti-monotone constraint. In the candidate generation phase of Apriori, additional to the monotonicity check, the lower and upper bounds on the candidate itemsets are computed. Such a check is possible, since in Apriori a set I can only be a candidate after all its strict subsets have been counted. The candidate itemsets that have an upper bound below the minimal support threshold are pruned, because they cannot be frequent. The itemsets having lower bound equal to the upper bound are pruned since they are derivable. Because of Theorem 2.1, we know that the supersets of a derivable itemset will be derivable as well, and hence, a derivable itemset can be pruned in the same way as an infrequent itemset. Furthermore, from Theorem 2.1, we can derive that if for a set I , $\text{supp}(I) = l_I$, or $\text{supp}(I) = u_I$, then all strict supersets of I are derivable. These properties lead straightforwardly to the level-wise algorithm NDI given in [10].

3 The Eclat Algorithm

In this section we describe the Eclat-algorithm, since our dfNDI-algorithm is based on it. Eclat was the first successful algorithm proposed to generate all frequent itemsets in a depth-first manner [18, 23]. Later, several other depth-first algorithms have been proposed [1, 2, 13].

Given a transaction database \mathcal{D} and a minimal support

threshold σ , denote the set of all frequent itemsets with the same prefix $I \subseteq \mathcal{I}$ by $\mathcal{F}[I](\mathcal{D}, \sigma)$. Eclat recursively generates for every item $i \in \mathcal{I}$ the set $\mathcal{F}[\{i\}](\mathcal{D}, \sigma)$. (Note that $\mathcal{F}[\{\}](\mathcal{D}, \sigma) = \bigcup_{i \in \mathcal{I}} \mathcal{F}[\{i\}](\mathcal{D}, \sigma)$ contains all frequent itemsets.)

For the sake of simplicity and presentation, we assume that all items that occur in the transaction database are frequent. In practice, all frequent items can be computed during an initial scan over the database, after which all infrequent items will be ignored.

In order to load the database into main memory, Eclat transforms this database into its *vertical format*. I.e., instead of explicitly listing all transactions, each item is stored together with its cover (also called *tidlist*). In this way, the support of an itemset X can be easily computed by simply intersecting the covers of any two subsets $Y, Z \subseteq X$, such that $Y \cup Z = X$.

The Eclat algorithm is given in Algorithm 1.

Note that a candidate itemset is represented by each set $I \cup \{i, j\}$ of which the support is computed at line 7 of the algorithm. Since the algorithm doesn't fully exploit the monotonicity property, but generates a candidate itemset based on only two of its subsets, the number of candidate itemsets that are generated is much larger as compared to a breadth-first approach such as apriori. As a comparison, Eclat essentially generates candidate itemsets using only the join step from Apriori, since the itemsets necessary for the prune step are not available.

EXAMPLE 3. Let the minimal support σ be 2. We continue working on \mathcal{D} as given in Example 1. As illustrated in Figure 1, Eclat starts with transforming the database into its vertical format. Then, recursively, the conditional databases are formed. The arcs indicate the recursion. The tree is traversed depth-first, from left to right. For example, the database \mathcal{D}^{ad} is formed using \mathcal{D}^a , by intersection the tid-list of d with the tid-lists of all items that come after d in \mathcal{D}^a . The tid-lists with the item between brackets (e.g., the tid-list for d in \mathcal{D}^c) indicate lists that are computed, but that are not in the conditional database because the item is infrequent. At any time, only the parents of the conditional database being constructed are kept in memory. For example, when \mathcal{D}^{ad} is constructed, the databases \mathcal{D}^a and \mathcal{D} are in memory; once a conditional database is no longer needed, it is removed from memory.

A technique that is regularly used, is to reorder the items in support ascending order. In Eclat, such reordering can be performed at every recursion step between line 11 and line 12 in the algorithm.

The effect of such a reordering is threefold:

- (1) The number of candidate itemsets that is generated is reduced. The generation step of Eclat is comparable to

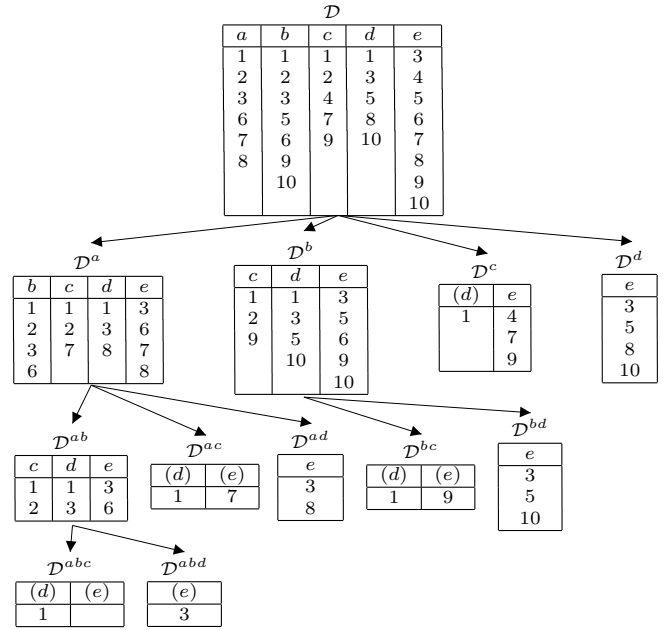


Figure 1: Eclat traversal.

the join-step of Apriori: a set $a_1 \dots a_k$ is generated if both $a_1 \dots a_{k-1}$ and $a_1 \dots a_{k-2}a_k$ are frequent. By re-ordering, the generating sets tend to have lower support which results in less candidates.

- (2) A second effect of the fact that the generating sets tend to have lower support is that their tid-lists are smaller.
- (3) At a certain depth d , the covers of at most all k -itemsets with the same $k - 1$ -prefix are stored in main memory, with $k \leq d$. Because of the item reordering, this number is kept small.

Experimental evaluation in earlier work has shown that reordering the items results in significant performance gains.

EXAMPLE 4. In Figure 2, we illustrate the effect of item reordering on the dataset given in Example 1, with the same support threshold $\sigma = 2$. As compared to Example 3, the tree is more balanced, the conditional databases are smaller, and there are 3 less candidates generated.

3.1 Diffsets Recently, Zaki proposed a new approach to efficiently compute the support of an itemset using the vertical database layout [19]. Instead of storing the cover of a k -itemset I , the difference between the cover of I and the cover of the $k - 1$ -prefix of I is stored, denoted by the *diffset* of I . To compute the support of I , we simply need to subtract the size of the diffset from the support of its $k - 1$ -prefix. This support can be provided as a parameter within the recursive function calls of the algorithm. The diffset of

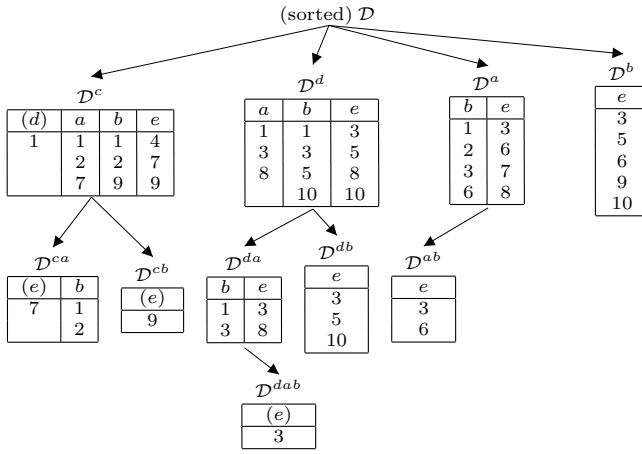


Figure 2: Eclat traversal with reordered items.

an itemset $I \cup \{i, j\}$, given the two diffsets of its subsets $I \cup \{i\}$ and $I \cup \{j\}$, with $i < j$, is computed as follows:

$$\text{diffset}(I \cup \{i, j\}) := \text{diffset}(I \cup \{j\}) \setminus \text{diffset}(I \cup \{i\}).$$

This technique has experimentally shown to result in significant performance improvements of the algorithm, now designated as *dEclat* [19]. The original database is still stored in the original vertical database layout.

Notice incidentally that with item reordering,

$$\text{supp}(I \cup \{i\}) \leq \text{supp}(I \cup \{j\}) .$$

Hence, $\text{diffset}(I \cup \{i\})$ is larger than $\text{diffset}(I \cup \{j\})$. Thus, to form $\text{diffset}(I \cup \{i, j\})$, the largest diffset is subtracted from the smallest, resulting in smaller diffsets. This argument, together with the three effects of reordering pointed out before, makes that reordering is a very effective optimization.

EXAMPLE 5. In Figure 3, we illustrate the *dEclat* algorithm with item reordering on the dataset given in Example 1, with the same support threshold $\sigma = 2$. The diffset of for example dab (entry b in the conditional database \mathcal{D}^{da}), is formed by subtracting the diffset of da from the diffset for db . Notice that the items are ordered ascending w.r.t. their support, and not w.r.t. the size of their diffset. The support of dab is computed as the support of da (3) minus the size of its diffset (1), which gives a support of 2.

4 The dfNDI Algorithm

In this section we describe a depth-first algorithm for mining all frequent non-derivable itemsets. The dfNDI algorithm combines ideas behind the Eclat algorithm, the diffsets and the deduction of supports into one hybrid algorithm.

The construction of the dfNDI-algorithm is based on the following principles:

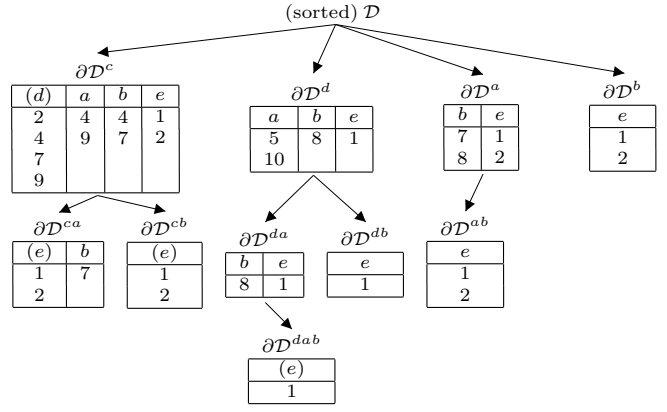


Figure 3: dEclat traversal with reordered items.

1. Just like Eclat, tidlists (and diffsets) will be used to compute the support of itemsets. Recursively, conditional databases will be formed. Hence, computing the support of an itemset will, unlike in the breadth-first version, *not* require a scan over the complete database.
2. There is one problem, however: to compute $\delta_X(I)$, the supports of all sets J such that $X \subseteq J \subset I$ must be known. Since Eclat is a depth-first algorithm, many of these supports are not available. This problem, however, can be solved by changing the order in which the search space is traversed. By changing the order, we can keep a depth-first approach, and still have the property that all subsets of a set I are handled before I itself.
3. Because we need the support of already found sets to compute bounds of their supersets, we will maintain the found frequent non-derivable itemsets in a specialized structure that allows fast lookup. Since the number of non-derivable itemsets is in general much lower than the number of frequent itemsets, the memory requirements for the specialized storage is not too bad; also, it is comparable to the amount of memory used in the ChARM algorithm to store all found frequent closed itemsets [22].
4. The deduction rules allow to extend the diffsets to tidlists of arbitrary generalized itemsets. That is, if we want to determine the support of a set I , we can use the cover of any of the generalized itemsets based on I . Then, based on the support of the generalized itemset $X \cup \bar{Y}$, and on $\delta_X(I)$, the support of I itself can be computed. This flexibility allows us to choose a generalized itemset that has minimal cover size.

In the rest of the section we concentrate on these four points. Then, we combine them and give the pseudo-code of dfNDI, which is illustrated with an example.

4.1 Order of Search Space Traversal We show next how we can guarantee in the Eclat-algorithm that for all sets I the support of its subsets is computed before I itself is handled. In Eclat, the conditional databases are computed in the following order (see Example 3):

$$\begin{array}{lclcl}
\mathcal{D} & \rightarrow & \mathcal{D}^a & \rightarrow & \mathcal{D}^{ab} & \rightarrow & \mathcal{D}^{abc} & \rightarrow & \mathcal{D}^{abcd} \\
& & & & & & \rightarrow & \mathcal{D}^{abd} \\
& & & & & & \rightarrow & \mathcal{D}^{ac} & \rightarrow & \mathcal{D}^{acd} \\
& & & & & & \rightarrow & \mathcal{D}^{ad} \\
& \rightarrow & \mathcal{D}^b & \rightarrow & \mathcal{D}^{bc} & \rightarrow & \mathcal{D}^{bcd} \\
& & & & \rightarrow & \mathcal{D}^{bd} \\
& \rightarrow & \mathcal{D}^c & \rightarrow & \mathcal{D}^{cd} \\
& \rightarrow & \mathcal{D}^d
\end{array}$$

Since the support of a set $a_1 \dots a_n$ is computed as the cardinality of the tidlist of a_n in the conditional database $\mathcal{D}^{a_1 \dots a_{n-1}}$, the supports of the itemsets are compute in the following order:

$$\begin{array}{l}
\{a, b, c, d, e\}, \{ab, ac, ad, ae\}, \{abc, abd, abe\}, \\
\{abcd, abce\}, \{abcde\}, \{abde\}, \{acd, ace\}, \\
\{acde\}, \{ade\}, \{bc, bd, be\}, \{bcd, bce\}, \{bcde\}, \\
\{bde\}, \{cd, ce\}, \{cde\}, \{de\}
\end{array}$$

Hence, for example, when the support of $abcd$ is computed, the supports of bc , bd , cd , acd , and bcd are not counted yet. Therefore, when the search space is explored in this way, all rules for $abcd$ that use one of these five sets cannot be computed.

An important observation now is that for every set I , all subsets either occur *on the recursion path* to I , or *after* I . For example, the support of $abcd$ is computed in \mathcal{D}^{abc} . The supports of the subsets of $abcd$ are either computed on the recursive path: a, b, c, d in \mathcal{D} , ab, ac, ad in \mathcal{D}^a , abc, abd in \mathcal{D}^{ab} , or after $abcd$: acd in \mathcal{D}^{ac} , which is constructed after the recursion below \mathcal{D}^{ab} , bc, bd in \mathcal{D}^b , which comes after the recursion below \mathcal{D}^a has ended, and cd in \mathcal{D}^c , which comes after the recursion below \mathcal{D}^b has ended. We can view the recursive structure between the conditional databases in Eclat as a tree, in which the children are ordered lexicographically, as is illustrated in Figure 4. This tree is by Eclat traversed depth-first, and from left-to-right; that is: in pre-order. Let the node associated with an itemset I be the node of the conditional database in which the support of I is computed. The observation can now be restated as follows: the nodes of the subsets of I are either on the path from \mathcal{D} to the node of I , or are in a branch that comes after I , *never* in a branch that comes *before* the branch of I .

Hence, we can change the order as follows: the same tree is traversed, still depth-first, but, from right to left. We will call this order the *reverse pre-order*. The numbers in the nodes of the tree in Figure 4 indicate the reverse pre-order. In this way, the path from \mathcal{D} to the node of a set I remains

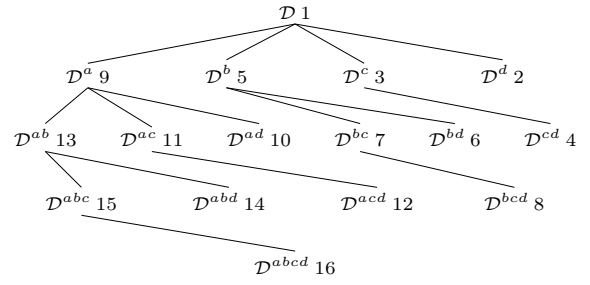


Figure 4: Recursive structure between the conditional databases

the same, but all branches that come after the branch of I , are now handled *before* the branch of I . As such, in reverse pre-order, all subsets of I are handled before I itself.

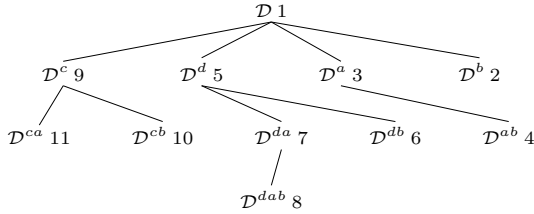
Besides enabling a depth-first algorithm for mining non-derivable itemsets, the reverse pre-order has other applications as well: a similar technique can be used when mining downwards closed collections of itemsets in a depth-first manner while preserving full pruning capabilities. It must be noted however that this ability comes the cost to store all found itemsets in the collection. Hence, the usefulness of this technique can be compromised when the downward closed collection becomes too large. For such large collections it is probably better to sacrifice part of the pruning in order to loose the store of generated itemsets. For small collections though, the reverse pre-order is very useful. Many condensed representations, such as the non-derivable itemsets, are typical examples of relatively small downward closed collections of itemsets.

Notice also that the reverse pre-order has no repercussions on performance or memory usage; the same covers are computed, and the same covers are in memory simultaneously; only the *order* in which they are generated differs. However, now we are guaranteed that *all* subsets of I are handled before I itself.

Another very important remark is that this guarantee remains even when at every step in the recursion the items in the conditional databases are reordered according to their support. Hence, we can still apply the important item reordering techniques.

EXAMPLE 6. Consider the tree in Example 4 that was constructed with item reordering. The numbers in the nodes indicate the reverse pre-order.

The reason that the same relation between a set and its subsets still applies can be seen as follows: the itemset $abde$ is counted in the conditional database \mathcal{D}^{dab} as the cardinality of the tidlist of e . Hence, the order in which the items of $abde$ were selected is: $dabe$. That is: (a) in the database \mathcal{D} , d came before a , b and e ; (b) in \mathcal{D}^d , a came before b and e ; and (c) in \mathcal{D}^{da} , b came before e . Since the



items in a conditional database are processed backwards in the recursion, therefore, (a) a, b, d, e are in \mathcal{D} , and ab, ae, be, abe are handled before \mathcal{D}^d is constructed, (b) ad, bd and de are in \mathcal{D}^d , and bde handled before \mathcal{D}^{da} is constructed, and (c) ade and abd are in \mathcal{D}^{da} .

In general, let I be an itemset and the elements of I are chosen in the order $i_1 \dots i_n$. Let J be a strict subset of I . If J corresponds to a prefix of $i_1 \dots i_n$, then J is computed on the recursive path to $\mathcal{D}^{i_1 \dots i_{n-1}}$. Otherwise, let i_j be the first item of $i_1 \dots i_n$ that is not in J . Then, in the conditional database $\mathcal{D}^{i_1 \dots i_{j-1}}$, all items in $J \setminus \{i_1, \dots, i_{j-1}\}$ come strictly after i_j , and hence, the node for J is on the right of the node for I and is thus visited before I in the reverse pre-order.

4.2 Storing the Found NDIs The frequent non-derivable itemsets are stored together with their supports in a structure that allows for fast look-up of the supports. The itemsets are stored in such a way that fast lookup of the supports is possible. In our implementations, an itemset trie was used for this purpose.

In the extreme case that the number of non-derivable itemsets becomes too large to be maintained in main memory, a condensed representation [15] can be used as well. For example, only the closed itemsets [16] could be stored. Finally, remark that it is not uncommon that an algorithm needs to maintain itemsets in main memory. Most closed itemset mining algorithms need to store the found frequent closed sets in main memory (e.g. Charm [21]).

4.3 Generalizing the Diffsets In Eclat and dEclat, the conditional database $\mathcal{D}^{I \cup \{i\}}$ is generated from the database \mathcal{D}^I if this database contains item i . All items $j > i$ that are in \mathcal{D}^I will be in the conditional database $\mathcal{D}^{I \cup \{i\}}$. In Eclat, the tidlist of j in $\mathcal{D}^{I \cup \{i\}}$ is computed by intersecting the tidlists of i and j in \mathcal{D}^I . In dEclat, not the tidlists are maintained in the conditional databases, but the diffsets. The diffset of j in $\mathcal{D}^{I \cup \{i\}}$ is computed by subtracting the diffset of i in \mathcal{D}^I from the diffset of j in \mathcal{D}^I . Hence, with the generalized itemsets notation, in Eclat the conditional database for $\mathcal{D}^{I \cup \{i\}}$ contains for all items $j > i$, $(j, \text{cover}(I \cup \{i, j\}))$. dEclat maintains for all items $j > i$, $(j, \text{cover}(I \cup \{i, \bar{j}\}))$ in the conditional database $\mathcal{D}^{I \cup \{i\}}$.

In dfNNDI, we will use a similar procedure. First of

Input: \mathcal{D}, σ

Output: \mathcal{D}^l

```

1: for all  $k$  occurring in  $\mathcal{D}$  after  $l$  do
2:   //  $k$  is either  $j$  or  $\bar{j}$ 
3:    $C[k] := \text{cover}(\{l\}) \cap \text{cover}(\{k\})$ 
4:    $C[\bar{k}] := \text{cover}(\{i\}) \setminus \text{cover}(\{k\})$ 
5:   if  $\{i, j\}$  is  $\sigma$ -frequent then
6:     if  $|C[j]| \leq |C[\bar{j}]|$  then
7:        $\mathcal{D}^{\{i\}} := \mathcal{D}^{\{i\}} \cup \{(j, C[j])\}$ 
8:     else
9:        $\mathcal{D}^{\{i\}} := \mathcal{D}^{\{i\}} \cup \{(\bar{j}, C[\bar{j}])\}$ 
10:    end if
11:  end if
12: end for

```

Figure 5: Recursive construction of \mathcal{D}^l in dfNNDI

all, the diffsets are extended to covers of arbitrary generalized itemsets. That is, not only covers of the type $\text{cover}(I \cup \{i, j\})$ and $\text{cover}(I \cup \{i, \bar{j}\})$ will be considered, but also $\text{cover}(X \cup \bar{Y})$ for any generalized itemset $X \cup \bar{Y}$ based on $I \cup \{i, j\}$. Secondly, the choice for which type of cover is not static in dfNNDI. In contrast, in Eclat *always* $\text{cover}(I \cup \{i, j\})$, and in dEclat, *always* $\text{cover}(I \cup \{i, \bar{j}\})$ is used. In dfNNDI, this choice will be postponed to run-time. At run-time both covers are computed, and the one with minimal size is chosen. In this way it is guaranteed that the size of the covers at least *halves* from \mathcal{D} to \mathcal{D}^i . The calculation of both covers can be done with minimal overhead in the same iteration. When iterating over the cover of $\{i\}$, the set of all tid's that are not in $\text{cover}(\{i\}) \cap \text{cover}(\{j\})$ is exactly $\text{cover}(\{i\}) \setminus \text{cover}(\{j\})$.

Let \mathcal{D} be a database that contains tidlists of both items and of negations of items. Let l be the list associated with item i . That is, l is either i or \bar{i} . The procedure used in dfNNDI to construct the conditional database \mathcal{D}^l from \mathcal{D} is given in Figure 5. The procedure is applied recursively. Suppose that we have recursively constructed database \mathcal{D}^G , with $G = X \cup \bar{Y}$ a generalized itemset. For every item i (resp. negated item \bar{i}) in \mathcal{D}^G , the cardinality of the tidlist is in fact the support of the generalized itemset $X \cup \{i\} \cup \bar{Y}$ (resp. $X \cup \bar{Y} \cup \{i\}$). The support test in line 5 is then performed using the deduction rules as explained in Section 2. For example, suppose $l = i$ and $k = \bar{j}$. Let $J = X \cup Y \cup \{i, j\}$. The value $\delta_{X \cup \{i\}}(J)$ is computed (using the stored supports) and from this value and the size of the cover of $X \cup \{i, j\} \cup \bar{Y}$, the support of J can be found:

$$\text{supp}(J) = \delta_{X \cup \{i\}}(J) + (-1)^{|Y|} \text{supp}(X \cup \{i, j\} \cup \bar{Y})$$

Notice also that if $C[j]$ (resp. $C[\bar{j}]$) is empty, $\text{supp}(G \cup \{i, j\}) = 0$ (resp. $\text{supp}(G \cup \{i, \bar{j}\}) = 0$). From Theorem 2.1, we can derive that in that situation, every superset of $X \cup Y \cup \{i, j\}$ is a derivable itemset.

EXAMPLE 7. Consider the following database \mathcal{D} , and the conditional databases \mathcal{D}^a and $\mathcal{D}^{a\bar{b}}$:

\mathcal{D}					\mathcal{D}^a				$\mathcal{D}^{a\bar{b}}$	
a	b	c	d		\bar{b}	c	\bar{d}		c	d
1	1	2	1							
2	2	4	2		5	2	5			
3	3	5	3		8	5	8			
5	7	6	9							
8		7								

\mathcal{D}^a contains \bar{b} because $\text{cover}(a) \setminus \text{cover}(b)$ is smaller than $\text{cover}(a) \cap \text{cover}(b)$. $\mathcal{D}^{a\bar{b}}$ contains d since $\text{cover}(\bar{b}) \setminus \text{cover}(\bar{d})$ is smaller than $\text{cover}(\bar{b}) \cap \text{cover}(\bar{d})$ in \mathcal{D}^a .

The support of abc is counted via the cover of c in $\mathcal{D}^{a\bar{b}}$. Since this cover contains one element, $\text{supp}(abc) = 1$, and hence,

$$\text{supp}(abc) = \text{supp}(ac) - \text{supp}(a\bar{b}c) = 2 - 1 = 1.$$

The support of ab is found in the trie that maintains the found frequent non-derivable itemsets. Notice that the cover of d in $\mathcal{D}^{a\bar{b}}$ is empty. Therefore, $\text{supp}(abd) = 0$, and thus, every superset of abd is derivable.

4.4 The Full dfNDI Algorithm The combination of the techniques discussed in this section give the dfNDI-algorithm described in Algorithm 2. In the recursion, the generalized itemset G that corresponds to the items or negated items that were chosen up to that point, must be passed on. Hence, the set of all frequent NDIs is given by $\text{dfNDI}(\mathcal{D}, \sigma, \{\})$. We assume that \mathcal{D} is given in vertical format, and that all items in \mathcal{D} are frequent.

As already mentioned before, reordering the items in the different conditional databases does not fundamentally change the algorithm. With reordering, we are still guaranteed that all subsets of an itemset I are handled before I itself. Therefore, in the experiments, we will use item reordering techniques to speed up the computation. There are different interesting orderings possible:

1. Ascending w.r.t. support: the advantages of this ordering are that the number of generated candidates is reduced, and that the tree is more balanced. Since for every candidate, lower and upper bounds on the support are computed, generating less candidates can be very interesting.
2. Ascending w.r.t. size of the cover: with this ordering, the size of the covers will in general be smaller, since the covers of the first items in the order are used more often than the covers of the items later in the ordering.

Depending on the application, either the first or the last ordering will be better. Which one is the best probably

Algorithm 2 dfNDI

Input: $\mathcal{D}, \sigma, G = X \cup \bar{Y}$

Output: $\text{dfNDI}(\mathcal{D}, \sigma, G)$

```

1: dfNDI := {}
2: for all  $l$  that occur in  $\mathcal{D}$ , ordered descending do
3:   //  $l = i$  or  $l = \bar{i}$ , for an item  $i$ 
4:   dfNDI := dfNDI  $\cup \{(X \cup Y \cup \{i\})\}$ 
5:   // Create  $\mathcal{D}^l$ 
6:    $\mathcal{D}^l := \{\}$ 
7:   for all  $k$  occurring in  $\mathcal{D}$  after  $l$  do
8:     //  $k = j$  or  $k = \bar{j}$ , for an item  $j$ 
9:     // Let  $J = X \cup Y \cup \{i, j\}$ 
10:    Compute bounds  $[l, u]$  on support of  $J$ ;
11:    if  $l \neq u$  and  $u \geq \sigma$  then
12:      //  $J$  in an NDI
13:      Store  $J$  in the separate trie
14:      // Compute the support of  $J$ 
15:       $C[k] := \text{cover}(\{l\}) \cap \text{cover}(\{k\})$ 
16:       $C[\bar{k}] := \text{cover}(\{l\}) \setminus \text{cover}(\{k\})$ 
17:      //  $|C[j]|$  is  $\text{supp}(X \cup \bar{Y} \cup \{l, j\})$ 
18:      Compute  $\text{supp}(J)$  based on the support of its
19:      subsets and on  $|C[j]|$ 
20:      if  $\text{supp}(J) \geq \sigma$  and  $\text{supp}(J) \neq l$  and
21:       $\text{supp}(J) \neq u$  then
22:        if  $|C[j]| \leq |C[\bar{j}]|$  then
23:           $\mathcal{D}^l := \mathcal{D}^l \cup \{(j, C[j])\}$ 
24:        else
25:           $\mathcal{D}^l := \mathcal{D}^l \cup \{(\bar{j}, C[\bar{j}])\}$ 
26:        end if
27:      end if
28:    end if
29:  end for
30:  // Depth-first recursion
31:  Compute dfNDI( $\mathcal{D}^l, \sigma, G \cup \{l\}$ )
32:  dfNDI := dfNDI  $\cup$  dfNDI( $\mathcal{D}^l, \sigma, G \cup \{l\}$ )
33: end for

```

depends on the selectivity of non-derivability versus the selectivity of frequency. If most sets are pruned due to non-derivability, the second ordering will be more interesting than the first. If frequency is the main pruner, the first order is more interesting.

There are many different variants possible of the dfNDI algorithm. Depending on the situation, one variant may be better than the other.

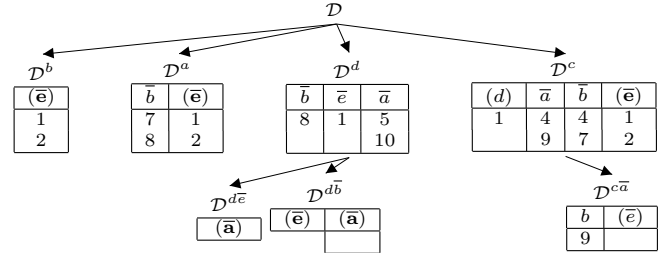
1. Ordering the items in the conditional databases, as described above.
2. Since computing the bounds can be very costly, we want to avoid this work as much as possible. Hence, in situations where frequency is highly selective, it is better to switch the support test and the non-derivability test. That is, lines 10 and 11 in Algorithm 2 are moved to after the if-test on line 19. Evidently, the test $\text{supp}(J) \neq l, u$ is removed from the if-test on line 19 to after the calculation of the bounds. Hence, only bounds are computed on itemsets that are frequent.
3. Another possibility to reduce the amount of work on the calculation of the bounds is not to compute all bounds, but only a limited number. A similar approach already turned out to be quite useful in the context of the breadth-first NDI-algorithm [10]. There, the depth of a rule based on $X \cup \bar{Y}$ was defined as $|Y|$. The lower the depth of a rule, the less complex. Instead of computing all rules, only rules up to a limited depth can be used. In practice it turned out that most of the time using rules up to depth 3 does not affect the precision of the rules. Put elsewhere, most work is done by the rules of limited depth.

EXAMPLE 8. We illustrate the dfNDI algorithm on the database of Example 3. Every conditional database is ordered ascending w.r.t. the size of the cover. The tid-lists with the item between brackets indicate lists that are computed, but that are not in the conditional database, because the itemset I associated with the item is either (a) infrequent (e.g. d in \mathcal{D}^c), or (b) $\text{supp}(I) = l_I$ or $\text{supp}(I) = u_I$. In case (b), I is a frequent NDI, but all supersets of I are derivable. The items in case (b) are indicated in bold.

We start with the database \mathcal{D} . Because of the reverse pre-order that is used, first the database \mathcal{D}^b is constructed. Only item e comes after b in \mathcal{D} . The lower and upper bounds on be are computed. $l_{be} = \text{supp}(b) + \text{supp}(e) - \text{supp}(\emptyset) = 5$, $u_{be} = \text{supp}(b) = 7$. Hence, be is not derivable. Both $\text{cover}(b) \cap \text{cover}(e)$ and $\text{cover}(b) \setminus \text{cover}(e)$ are constructed. $|\text{cover}(b) \cap \text{cover}(e)| = 5$, and hence, $\text{supp}(be) = 5$. Since $\text{supp}(be) = l_{be}$, all supersets of be must be derivable.

Next \mathcal{D}^a is constructed. For ab the following lower and upper bound is computed: $l_{ab} = 3$, $u_{ab} = 6$. The two covers $\text{cover}(\{a\}) \cap \text{cover}(\{b\})$ and $\text{cover}(\{a\}) \setminus \text{cover}(\{b\})$ are

computed. From the size of these covers, the support of ab is computed: $\text{supp}(ab) = 4$. Because the support differs from the lower and the upper bound, b will be in \mathcal{D}^a . Since $\text{cover}(\{a\}) \setminus \text{cover}(\{b\})$ was the smallest of the two covers, $(\bar{b}, \text{cover}(\{a\}) \setminus \text{cover}(\{b\}))$ is added. Then the item e is handled. The bounds for ae are: $l_{ae} = 4$, $u_{ae} = 6$. Thus, ae is non-derivable. The two covers for ae are computed and the support is derived: $\text{supp}(ae) = 4$. Because $\text{supp}(ae) = l_{ae}$, item e will not be added to \mathcal{D}^a , because all supersets of ae must be derivable. This procedure continues until all items are handled. In the construction of $\mathcal{D}^{d\bar{e}}$, it turns out that ade is derivable, since $\delta_d(ade) = 2$, and $\delta_{\emptyset}(ade) = 2$, and hence $l_{ade} = u_{ade} = 2$.



The collection of frequent non-derivable itemsets in this example is:

$\{\emptyset, a, b, c, d, e, ab, ac, ad, ae, bc, bd, be, ce, de, abc, abd\}$

5 Experiments

The experiments were performed on a 1.5GHz Pentium IV PC with 1GB of main memory. To empirically evaluate the proposed dfNDI-algorithm, we performed several tests on the datasets summarized in the following table. For each dataset, the table shows the number of transactions, the number of items, and the average transaction length.

Dataset	# trans.	# items	Avg. length
BMS-POS	515 597	1 656	6,53
Chess	3 196	76	37
Pumsb	49 046	2 112	74
Mushroom	8 124	120	23

These datasets are all well-known benchmarks for frequent itemset mining. The *BMS-POS* dataset contains click-stream data from a small dot-com company that no longer exists. These two datasets were donated to the research community by Blue Martini Software. The *Pumsb*-dataset is based on census data, the *Mushroom* dataset contains characteristics from different species of mushrooms. The *Chess* dataset contains different game configurations. The *Pumsb* dataset is available in the UCI KDD Repository [14], and the *Mushroom* and *Chess* datasets can be found in the UCI Machine Learning Repository [6].

Obviously, as the number of non-derivable frequent itemsets is significantly smaller than the total number of frequent itemsets, it would be unfair to compare the per-

formance of the proposed algorithm with any other normal frequent itemset mining algorithm. Indeed, as soon as the threshold is small enough or the data is highly correlated, it is well known that traditional techniques fail, simply due to the massive amount of frequent sets that is produced, rather than any inherent algorithmic characteristic of the algorithms. Therefore, we merely compare the performance of dfNDI with NDI. Also note that a thorough comparison of the sizes of different condensed representations has already been presented in [10] and will not be repeated here.

In our experiments, we compared the NDI algorithm with three different versions of the dfNDI algorithm. The first version, 'dfNDI', is the regular one as described in this paper. The second version, 'dfNDI - negations', does not allow to use the covers of generalized itemsets, and hence, stores the full cover of every candidate itemset and always computes the covers by intersecting the covers of two subsets. The third version, 'dfndi + support order' is the dfNDI algorithm in which items are dynamically reordered in every recursion step according to their support, while the regular dfNDI algorithm orders the items according to the cover of the generalized itemset it represents. All experiments were performed for varying minimum support thresholds. The result can be seen in Figures 6 and 7.

First, we compared the time performance of these four algorithms. As expected, dfNDI performs much better compared to NDI. Using the smallest cover of a generalized itemset based on the current itemset also has a significant effect, which is especially visible in the pumsb dataset. This is of course mainly due to the faster set intersection and set difference operations on the smaller covers. Ordering the items according to the size of the covers also proves to be better as compared to ordering according to the support of the items, although the difference is never very big.

In the second experiment, we compare the memory usage of the four algorithms. As can be seen, the NDI algorithm needs least amount of memory. Of course, this is expected as the dfNDI also stores parts of the database in main memory. Fortunately, the optimization of storing only the smallest cover of each generalized itemset indeed shows to result in reduced memory usage. Again, the ordering based on the cover seems to result in a little bit better memory usage.

6 Conclusion

In this paper, we presented a new itemset search space traversal strategy, which allows depth-first itemset mining algorithms to exploit the frequency knowledge of all subsets of a given candidate itemset. Second, we presented a depth-first non-derivable itemset mining algorithm, which uses that traversal as it needs for every generated candidate itemset, all of its subsets. From the viewpoint of condensed representations of frequent sets, the non-derivable itemset

collection has already been shown to be superior to all other representations in almost all cases. Third, we generalized the diffsets technique and store the cover of an itemset containing several negated items. These covers are typically much smaller than the usual covers of regular itemsets, and hence, allow faster set intersection and set difference operations. The resulting algorithm, dfNDI, thus inherits all positive characteristics from several different techniques allowing fast discovery of a reasonable amount of frequent itemsets that are not derivable from their subsets. These claims are supported by several experiments on real life datasets.

References

- [1] R. Agarwal, C. Aggarwal, and V. Prasad. Depth first generation of long patterns. In R. Ramakrishnan, S. Stolfo, R. Bayardo, Jr., and I. Parsa, editors, *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 108–118. ACM Press, 2000.
- [2] R. Agarwal, C. Aggarwal, and V. Prasad. A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 61(3):350–371, March 2001.
- [3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, volume 22(2) of *SIGMOD Record*, pages 207–216. ACM Press, 1993.
- [4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, 1994.
- [5] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, 2000.
- [6] C. Blake and C. Merz. *The UCI Repository of machine learning databases* [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
- [7] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PaKDD Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 62–73, 2000.
- [8] T. Calders. Deducing bounds on the support of itemsets. In *Database Technologies for Data Mining*, chapter ?, pages ?–? Springer, 2003.
- [9] T. Calders and B. Goethals. Minimal k -free representations of frequent sets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pages 71–82, 2002.
- [10] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pages 74–85. Springer, 2002.
- [11] J. Galambos and I. Simonelli. *Bonferroni-type Inequalities with Applications*. Springer, 1996.

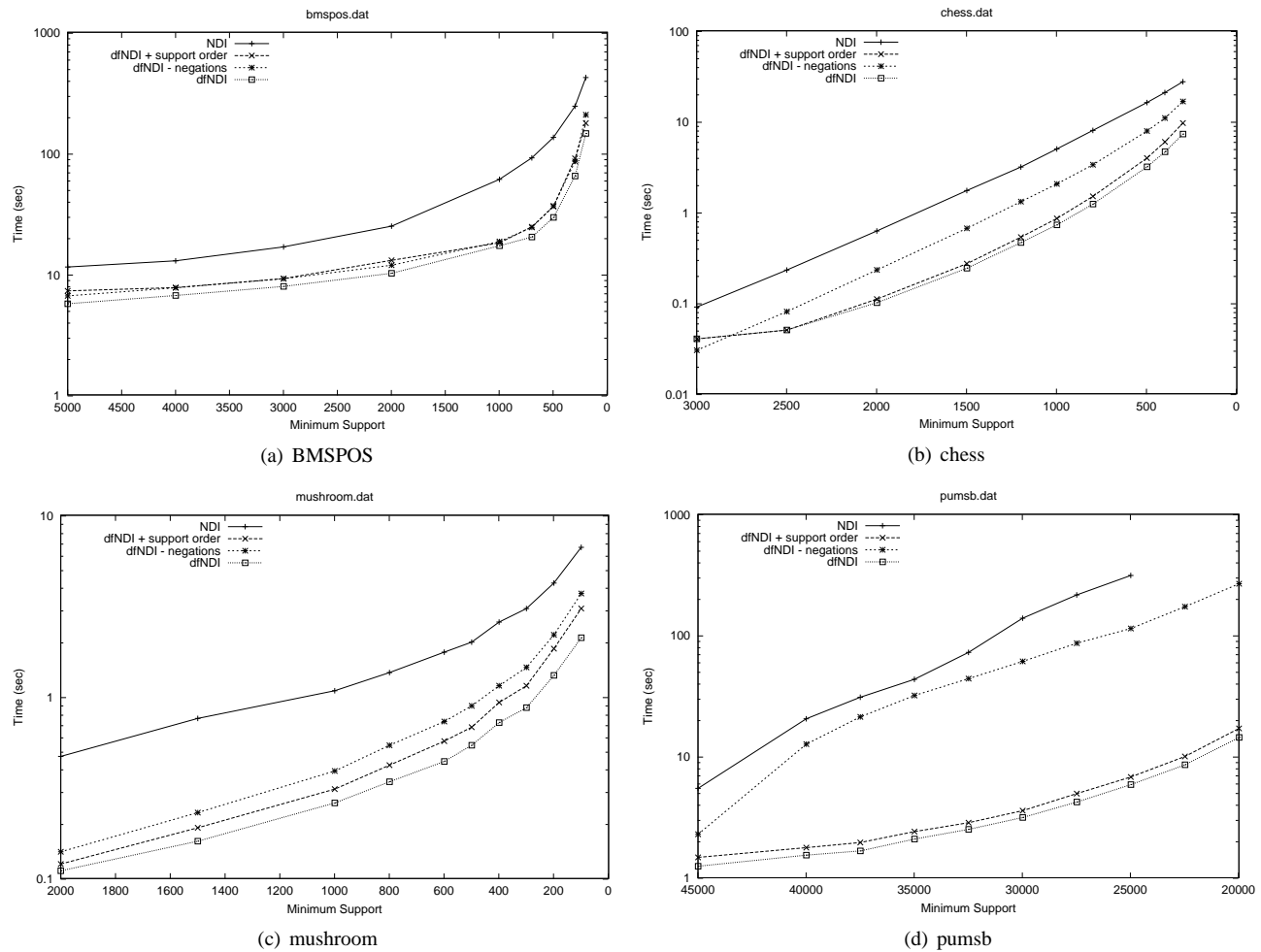
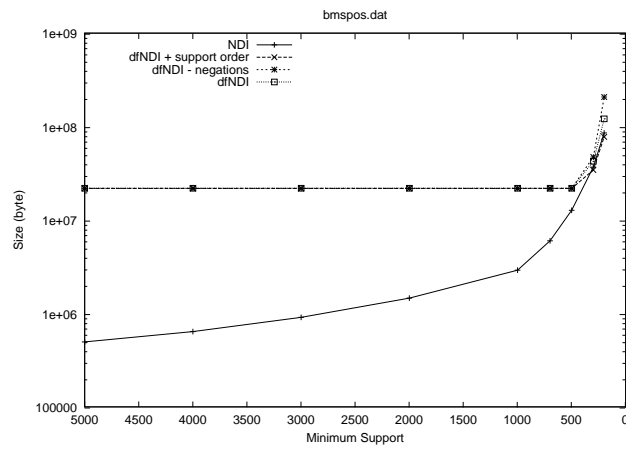
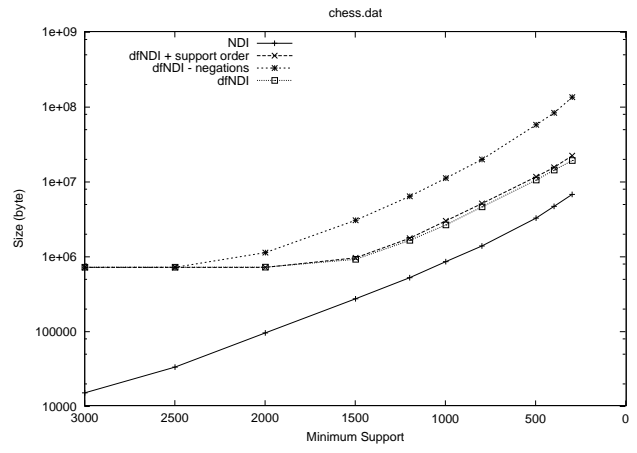


Figure 6: Performance comparison.

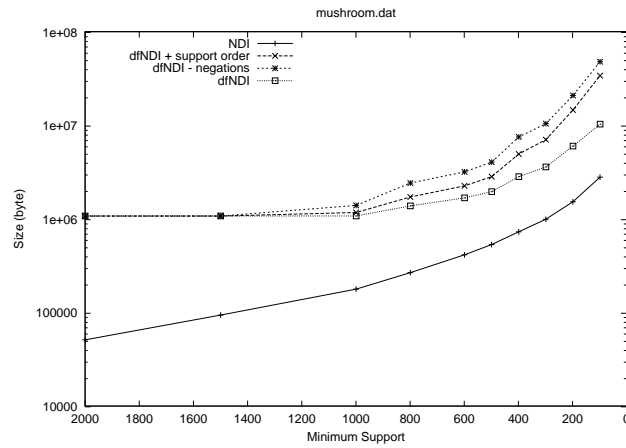
- [12] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 1–12, Dallas, TX, 2000.
- [13] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 2003. To appear.
- [14] S. Hettich and S. D. Bay. *The UCI KDD Archive*. [<http://kdd.ics.uci.edu>]. Irvine, CA: University of California, Department of Information and Computer Science, 1999.
- [15] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. KDD Int. Conf. Knowledge Discovery in Databases*, 1996.
- [16] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT Int. Conf. Database Theory*, pages 398–416, 1999.
- [17] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, TX, 2000.
- [18] M. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, May/June 2000.
- [19] M. Zaki and K. Gouda. Fast vertical mining using diffsets. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2003.
- [20] M. Zaki and C. Hsiao. ChARM: An efficient algorithm for closed association rule mining. In *Technical Report 99-10, Computer Science, Rensselaer Polytechnic Institute*, 1999.
- [21] M. Zaki and C. Hsiao. ChARM: An efficient algorithm for closed association rule mining. In *Proc. SIAM Int. Conf. on Data Mining*, 2002.
- [22] M. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining*, 2002.
- [23] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. Heck-



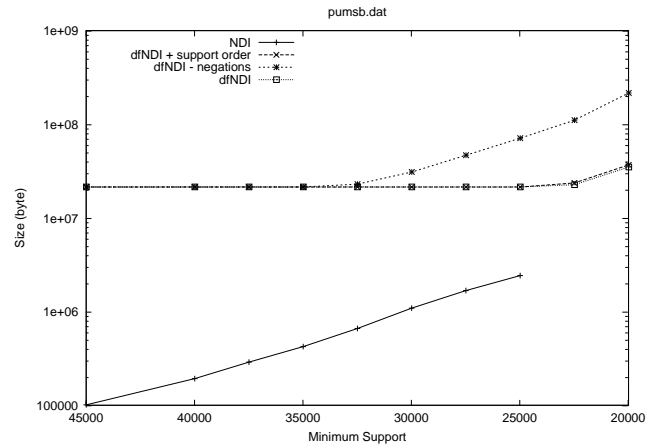
(a) BMSPOS



(b) chess



(c) mushroom



(d) pumsb

Figure 7: Memory usage comparison.

erman, H. Mannila, and D. Pregibon, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.

Exploiting relationships for domain-independent data cleaning^{*†}

Dmitri V. Kalashnikov Sharad Mehrotra Zhaoqi Chen

Computer Science Department
University of California, Irvine

Abstract

In this paper we address the problem of *reference disambiguation*. Specifically, we consider a situation where entities in the database are referred to using descriptions (e.g., a set of instantiated attributes). The objective of reference disambiguation is to identify the unique entity to which each description corresponds. The key difference between the approach we propose (called RelDC) and the traditional techniques is that RelDC analyzes not only object features but also inter-object relationships to improve the disambiguation quality. Our extensive experiments over two real datasets and also over synthetic datasets show that analysis of relationships significantly improves quality of the result.

1 Introduction

Recent surveys [3] show that more than 80% of researchers working on data mining projects spend more than 40% of their project time on cleaning and preparation of data. The data cleaning problem often arises when information from heterogeneous sources is merged to create a single database. Many distinct data cleaning challenges have been identified in the literature: dealing with missing data [20], handling erroneous data [21], record linkage [6, 7], and so on. In this paper we address one such challenge which we refer to as *reference disambiguation*.

The reference disambiguation problem arises when entities in a database contain references to other entities. If entities were referred to using unique identifiers then disambiguating those references would be straightforward. Instead, frequently, entities are represented using properties/descriptions that may not uniquely identify them leading to ambiguity. For instance, a database may store information about two distinct individuals ‘Donald L. White’ and ‘Donald E. White’, both of whom are referred to as ‘D. White’ in another database. References may also be ambiguous due to differences in the representations of the same entity and errors in data

entries (e.g., ‘Don White’ misspelled as ‘Don Whitex’). The **goal** of reference disambiguation is for each reference to correctly identify the unique entity it refers to.

The reference disambiguation problem is related to the problem of *record deduplication* or *record linkage* [7, 6] that often arises when multiple tables (from different data sources) are merged to create a single table. The causes of record linkage and reference disambiguation problems are similar; viz., differences in representations of objects across different datasets, data entry errors, etc. The differences between the two can be intuitively viewed using the relational terminology as follows: while the record linkage problem consists of determining when two records are the same, reference disambiguation corresponds to ensuring that references (i.e., “foreign keys”¹) in a database point to the correct entities.

Given the tight relationship between the two data cleaning tasks and the similarity of their causes, existing approaches to record linkage can be adapted for reference disambiguation. In particular, *feature-based similarity (FBS)* methods that analyze similarity of record attribute values (to determine whether or not two records are the same) can be used to determine if a particular reference corresponds to a given entity or not. This paper argues that the quality of disambiguation can be significantly improved by exploring additional semantic information. In particular, we observe that references occur within a context and define relationships/connections between entities. For instance, ‘D. White’ might be used to refer to an author in the context of a particular publication. This publication might also refer to different authors, which can be linked to their affiliated organizations etc, forming chains of relationships among entities. Such knowledge can be exploited alongside attribute-based similarity resulting in improved accuracy of disambiguation.

In this paper, we propose a domain-independent

^{*}RelDC project (<http://www.ics.uci.edu/~dvk/RelDC>)

[†]This work was supported in part by NSF grants 0331707, 0331690, and IRI-9703120.

¹We are using the term foreign key loosely. Usually, foreign key refers to a unique identifier of an entity in another table. Instead, foreign key above means the set of properties that serve as a reference to an entity.

data cleaning approach for reference disambiguation, referred to as Relationship-based Data Cleaning (RelDC), that systematically exploits not only features but also relationships among entities for the purpose of disambiguation. RelDC views the database as a graph of entities that are linked to each other via relationships. It first utilizes a feature based method to identify a set of candidate entities (choices) for a reference to be disambiguated. Graph theoretic techniques are then used to discover and analyze relationships that exist between the entity containing the reference and the set of candidates.

The primary contributions of this paper are: (1) developing a systematic approach to exploiting both attributes as well as relationships among entities for reference disambiguation (2) establishing that exploiting relationships can significantly improve the quality of reference disambiguation by testing the developed approach over 2 real-world datasets as well as synthetic datasets.

This paper presents the core of the RelDC approach, details of RelDC can be found in [16] where we discuss various implementations, optimizations, computational complexity, sample content and sample graphs for real datasets, and other issues not covered in this paper. The rest of this paper is organized as follows. Section 2 presents a motivational example. In Section 3, we precisely formulate the problem of reference disambiguation and introduce notation that will help explain the RelDC approach. Section 4 describes the RelDC approach. The empirical results of RelDC are presented in Section 5. Section 6 contains the related work, and Section 7 concludes the paper.

2 Motivation for analyzing relationships

In this section we will use an instance of the “author matching” problem to illustrate that exploiting relationships among entities can improve the quality of reference disambiguation. We will also schematically describe one approach that analyzes relationships in a systematic domain-independent fashion. Consider a

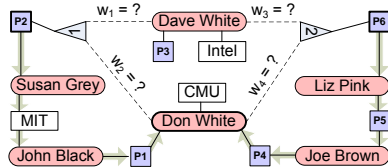


Figure 1: Graph for the publications example

database about *authors* and *publications*. Authors are represented in the database using the attributes $\langle \text{id}, \text{authorName}, \text{affiliation} \rangle$ and information about papers is stored in the form $\langle \text{id}, \text{title}, \text{authorRef1}, \dots, \text{authorRefN} \rangle$. Consider a toy database consisting of the following *authors* and *publications* records.

1. $\langle A_1, \text{'Dave White'}, \text{'Intel'} \rangle$,

2. $\langle A_2, \text{'Don White'}, \text{'CMU'} \rangle$,
3. $\langle A_3, \text{'Susan Grey'}, \text{'MIT'} \rangle$,
4. $\langle A_4, \text{'John Black'}, \text{'MIT'} \rangle$,
5. $\langle A_5, \text{'Joe Brown'}, \text{unknown} \rangle$,
6. $\langle A_6, \text{'Liz Pink'}, \text{unknown} \rangle$.

1. $\langle P_1, \text{'Databases ...'}, \text{'John Black'}, \text{'Don White'} \rangle$,
2. $\langle P_2, \text{'Multimedia ...'}, \text{'Sue Grey'}, \text{'D. White'} \rangle$,
3. $\langle P_3, \text{'Title3 ...'}, \text{'Dave White'} \rangle$,
4. $\langle P_4, \text{'Title5 ...'}, \text{'Don White'}, \text{'Joe Brown'} \rangle$,
5. $\langle P_5, \text{'Title6 ...'}, \text{'Joe Brown'}, \text{'Liz Pink'} \rangle$,
6. $\langle P_6, \text{'Title7 ...'}, \text{'Liz Pink'}, \text{'D. White'} \rangle$.

The goal of the author matching problem is to identify for each **authorRef** in each paper the correct author it refers to.

We can use existing feature-based similarity (FBS) techniques to compare the description contained in each **authorRef** in papers with values in **authorName** attribute in authors. This would allow us to resolve almost every **authorRef** references in the above example. For instance, such methods would identify that ‘Sue Grey’ reference in P_2 refers to A_3 (‘Susan Grey’). The only exception will be ‘D. White’ references in P_2 and P_6 : ‘D. White’ could match either A_1 (‘Dave White’) or A_2 (‘Don White’).

Perhaps, we could disambiguate the reference ‘D. White’ in P_2 and P_6 by exploiting additional attributes. For instance, the titles of papers P_1 and P_2 might be similar while titles of P_2 and P_3 might not, suggesting that ‘D. White’ of P_2 is indeed ‘Don White’ of paper P_1 . We next show that it may still be possible to disambiguate the references ‘D. White’ in P_2 and P_6 by analyzing relationships among entities even if we are unable to disambiguate the references using title (or other attributes).

First, we observe that author ‘Don White’ has co-authored a paper (P_1) with ‘John Black’ who is at MIT, while the author ‘Dave White’ does not have any co-authored papers with authors at MIT. We can use this observation to disambiguate between the two authors. In particular, since the co-author of ‘D. White’ in P_2 is ‘Susan Grey’ of MIT, there is a higher likelihood that the author ‘D. White’ in P_2 is ‘Don White’. The reason is that the data suggests a connection between author ‘Don White’ with MIT and an absence of it between ‘Dave White’ and MIT.

Second, we observe that author ‘Don White’ has co-authored a paper (P_4) with ‘Joe Brown’ who in turn has co-authored a paper with ‘Liz Pink’. In contrast, author ‘Dave White’ has not co-authored any papers with either ‘Liz Pink’ or ‘Joe Brown’. Since ‘Liz Pink’ is a co-author of P_6 , there is a higher likelihood that ‘D. White’ in P_6 refers to author ‘Don White’ compared to author ‘Dave White’. The reason is that often co-author networks form groups/clusters of authors that

do related research and may publish with each other. The data suggests that ‘Don White’, ‘Joe Brown’ and ‘Liz Pink’ are part of the cluster, while ‘Dave White’ is not.

At first glance, the analysis above (used to disambiguate references that could not be resolved using conventional feature-based techniques) may seem ad-hoc and domain dependent. A general principle emerges if we view the database as a graph of inter-connected entities (modeled as nodes) linked to each other via relationships (modeled as edges). Figure 1 illustrates the entity-relationship graph corresponding to the toy database consisting of *authors* and *papers* records. In the graph, entities containing references are linked to the entities they refer to. For instance, since the reference ‘Sue Grey’ in P_2 is unambiguously resolved to author ‘Susan Grey’, paper P_2 is connected by an edge to author A_3 . Similarly, paper P_5 is connected to authors A_5 (‘Joe Brown’) and A_6 (‘Liz Pink’). The ambiguity of the references ‘D. White’ in P_2 and P_6 is captured by linking papers P_2 and P_6 to both ‘Dave White’ and ‘Don White’ via two “choice nodes” (labeled ‘1’ and ‘2’ in the figure). These “choice nodes” serve as OR-nodes in the graph and represent the fact that the reference ‘D. White’ refers to either one of the entities linked to the choice nodes.

Given the graph view of the toy database, the analysis we used to disambiguate ‘D. White’ in P_2 and P_6 can be viewed as an application of the following general principle:

Context Attraction Principle (CAP): *If reference r made in the context of entity x refers to an entity y_j whereas the description provided by r matches multiple entities $y_1, \dots, y_j, \dots, y_N$, then x and y_j are likely to be more strongly connected to each other via chains of relationships than x and y_l ($l = 1, 2, \dots, N; l \neq j$). \square*

The first observation we made, regarding disambiguation of ‘D. White’ in P_2 , corresponds to the presence of the following path (i.e., *relationship chain* or *connection*) between the nodes ‘Don White’ and P_2 in the graph: $P_2 \rightarrow$ ‘Susan Grey’ \rightarrow ‘MIT’ \rightarrow ‘John Black’ $\rightarrow P_1 \rightarrow$ ‘Don White’. Similarly, the second observation, regarding disambiguation of ‘D. White’ in P_6 as ‘Don White’, was based on the presence of the following path: $P_6 \rightarrow$ ‘Liz Pink’ $\rightarrow P_5 \rightarrow$ ‘Joe Brown’ $\rightarrow P_4 \rightarrow$ ‘Don White’. There were no paths between P_2 and ‘Dave White’ or between P_6 and ‘Dave White’ (if we ignore ‘1’ and ‘2’ nodes). So, after applying the CAP principle, we concluded that the ‘D. White’ references in both cases probably corresponded to the author ‘Don White’. In general, there could have been paths not only between P_2 (P_6) and ‘Don White’ but also between P_2 (P_6) and ‘Dave White’. In that case, to determine if ‘D.

White’ is ‘Don White’ or ‘Dave White’ we should have been able to measure whether ‘Don White’ or ‘Dave White’ is more strongly connected to P_2 (P_6).

The generic approach therefore first *discovers connections* between the entity, in the context of which the reference appears and the matching candidates for that reference. It then *measures the connection strength* of the discovered connections in order to give preference to one of the matching candidates. The above discussion naturally leads to two questions:

1. Does the context attraction principle hold over real datasets. That is, if we disambiguate references based on the principle, will the references be correctly disambiguated?
2. Can we design a generic solution to exploiting relationships for disambiguation?

Of course, the second question is only important if the answer to the first is yes. However, we cannot really answer the first unless we develop a general strategy to exploiting relationships for disambiguation and testing it over real data. We will develop one such general, domain-independent strategy for exploiting relationships for disambiguation which we refer to as RelDC in Section 4. We perform extensive testing of RelDC over both real data from two different domains as well as synthetic data to establish that exploiting relationships (as is done by RelDC) significantly improves the data quality. Before we develop RelDC, we first develop notation and concepts needed to explain our approach in Section 3.

3 Problem formalization

3.1 Notation Let \mathcal{D} be the database which contains references that are to be resolved. Let $X = \{x_1, x_2, \dots, x_{|X|}\}$ be the set of all entities in \mathcal{D} . Entities here have the same meaning as in the E/R model. Each entity x_i consists of a set of properties and contains a set of n_{x_i} references $x_i.r_1, x_i.r_2, \dots, x_i.r_{n_{x_i}}$. Each reference $x_i.r_k$ is essentially a description and may itself consist of one or more attributes $x_i.r_k.b_1, x_i.r_k.b_2, \dots$. For instance, in the example from Section 2, *paper* entities contain one-attribute **authorRef** references in the form $\langle \text{author name} \rangle$. If, besides author names, author affiliation were also stored in the *paper records*, then **authorRef** references would have consisted of two attributes – $\langle \text{author name}, \text{author affiliation} \rangle$.

Choice set. Each reference $x_i.r_k$ semantically refers to a single specific entity in X which we denote by $d[x_i.r_k]$. The description provided by $x_i.r_k$ may, however, match a set of one or more entities in X . We refer to this set as the *choice set* of reference $x_i.r_k$ and denote it by $CS[x_i.r_k]$. The choice set consists of all the entities that $x_i.r_k$ could potentially refer to. We assume

$CS[x_i.r_k]$ is given for each $x_i.r_k$. If it is not given, we assume a feature-based similarity approach is used to construct $CS[x_i.r_k]$ by choosing all of the candidates such that FBS similarity between them and $x_i.r_k$ exceed a given threshold. To simplify notation, we will always assume $CS[x_i.r_k]$ has N (i.e., $N = |CS[x_i.r_k]|$) elements y_1, y_2, \dots, y_N .

3.2 The Entity-Relationship Graph RelDC views the resulting database \mathcal{D} as an undirected entity-relationship graph (also known as *Attributed Relational Graph (ARG)*) $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node corresponds to an entity and each edge to a relationship. Notation $v[x_i]$ denotes the vertex in G that corresponds to entity $x_i \in X$. Note that if entity u contains a reference to entity v , then the nodes in the graph corresponding to u and v are linked since a reference establishes a relationship between the two entities. For instance, **authorRef** reference from paper P to author A corresponds to “ A writes P ” relationship.

In the graph G , edges have weights, nodes do not have weights. Each edge weight is a real number in $[0, 1]$, which reflects the degree of confidence the relationship, corresponding to the edge, exists. For instance, in the context of our author matching example, if we are 100% confident ‘John Black’ is affiliated with MIT, then we assign weight of 1 to the corresponding edge. But if we are only 80% confident, we assign the weight of 0.80 to that edge. By default all weights are equal to 1. Notation “edge label” means the same as “edge weight”.

References and linking. If $CS[x_i.r_k]$ has only one element, then $x_i.r_k$ is resolved to y_1 , and graph G contains an edge between $v[x_i]$ and $v[y_1]$. This edge is assigned a weight of 1 to denote that the algorithm is 100% confident that $d[x_i.r_k]$ is y_1 . If $CS[x_i.r_k]$ has

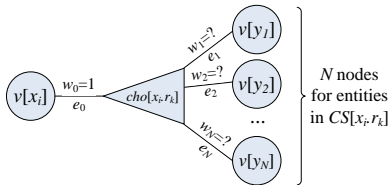


Figure 2: A choice node

more than 1 elements, then graph G contains a **choice node** $cho[x_i.r_k]$, as shown in Figure 2, to reflect the fact that $d[x_i.r_k]$ can be one of y_1, y_2, \dots, y_N . Node $cho[x_i.r_k]$ is linked with node $v[x_i]$ via edge $e_0 = (v[x_i], cho[x_i.r_k])$. Node $cho[x_i.r_k]$ is also linked with N nodes $v[y_1], v[y_2], \dots, v[y_N]$, for each y_j in $CS[x_i.r_k]$, via edges $e_j = (cho[x_i.r_k], v[y_j])$ ($j = 1, 2, \dots, N$). Nodes $v[y_1], v[y_2], \dots, v[y_N]$ are called the *options* of choice $cho[x_i.r_k]$. Edges e_1, e_2, \dots, e_N are called the *option-edges* of choice $cho[x_i.r_k]$. The weights of option-edges

are called *option-edge weights* or simply *option weights*. The weight of edge e_0 is 1. Each weight w_j of edges e_j ($j = 1, 2, \dots, N$) is undefined initially. Since these option-edges e_1, e_2, \dots, e_N represent mutually exclusive alternatives, the sum of their weights should be 1: $w_1 + w_2 + \dots + w_N = 1$.

3.3 The objective of reference disambiguation

To *resolve* reference $x_i.r_k$ means to choose one entity y_j from $CS[x_i.r_k]$ in order to determine $d[x_i.r_k]$. If entity y_j is chosen as the outcome of such a disambiguation, then $x_i.r_k$ is said to be *resolved to y_j* or simply *resolved*. Reference $x_i.r_k$ is said to be *resolved correctly* if this y_j is $d[x_i.r_k]$. Notice, if $CS[x_i.r_k]$ has just one element y_1 (i.e., $N = 1$), then reference $x_i.r_k$ is automatically resolved to y_1 . Thus reference $x_i.r_k$ is said to be *unresolved* or *uncertain* if it is not resolved yet to any y_j and also $N > 1$.

From the graph theoretic perspective, to resolve $x_i.r_k$ means to assign weights of 1 to one edge e_j , $1 \leq j \leq N$ and assign weights of 0 to the other $N - 1$ edges $e_1, e_2, \dots, e_{j-1}, e_{j+1}, \dots, e_N$. This will indicate that the algorithm chooses y_j as $d[x_i.r_k]$.

The **goal** of reference disambiguation is to resolve all references as correctly as possible, that is for each reference $x_i.r_k$ to correctly identify $d[x_i.r_k]$. We will use notation $Resolve(x_i.r_k)$ to refer to the procedure which resolves $x_i.r_k$. The *goal* is thus to construct such $Resolve(\cdot)$ which should be as accurate as possible. The *accuracy* of reference disambiguation is the fraction of references being resolved that are resolved correctly.

The **alternative goal** is for each $y_j \in CS[x_i.r_k]$ to associate weight w_j that reflects the degree of confidence that y_j is $d[x_i.r_k]$. For that alternative goal, $Resolve(x_i.r_k)$ should label each edge e_j with such a weight. Those weights can be **interpreted** later to achieve the main goal: for each $x_i.r_k$ try to identify only one y_j as $d[x_i.r_k]$ correctly. We emphasize this alternative goal since most of the discussion of RelDC approach is devoted to one approach for computing those weights. An interpretation of those weights (in order to try to identify $d[x_i.r_k]$) is a small final step of RelDC. Namely, we achieve this by picking y_j such that w_j is the largest among w_1, w_2, \dots, w_N . That is, the outcome of $Resolve(x_i.r_k)$ is $y_j : w_j = \max_{i=1}^N w_i$.

3.4 Connection Strength and Context Attraction Principle

As mentioned before, RelDC resolves references based on context attraction principle that was discussed in Section 2. We now state the principle more formally. Crucial to the principle is the notion of *connection strength* between two entities x_i and y_j (denoted $c(x_i, y_j)$) which captures how strongly x_i and y_j are connected to each other through relationships. Many differ-

ent approaches can be used to measure $c(x_i, y_j)$ which will be discussed in Section 4. Given the concept of $c(x_i, y_j)$, we can restate the context attraction principle as follows:

Context Attraction Principle: Let $x_i.r_k$ be a reference and y_1, y_2, \dots, y_N be elements of its choice set $CS[x_i.r_k]$ with corresponding option weights w_1, w_2, \dots, w_N (recall that $w_1 + w_2 + \dots + w_N = 1$). The CAP principle states that for all $l, j \in [1, N]$, if $c_l \geq c_j$ then $w_l \geq w_j$, where $c_l = c(x_i, y_l)$ and $c_j = c(x_i, y_j)$.

4 The RelDC approach

We now have developed all the concepts and notation needed to explain RelDC approach for reference disambiguation. Input to RelDC is the entity-relationship graph G discussed in Section 3 in which nodes correspond to entities and edges to relationships. We assume that feature-based similarity approaches have been used in constructing the graph G . The choice nodes are created only for those references that could not be disambiguated using only attribute similarity. RelDC will exploit relationships for further disambiguation and will output a resolved graph G in which each entity is fully resolved.

RelDC disambiguates references using the following four steps:

1. **Compute connection strengths.** For each reference $x_i.r_k$ compute the connection strength $c(x_i, y_j)$ for each $y_j \in CS[x_i.r_k]$. The result is a set of equations that relate $c(x_i, y_j)$ with the option weights: $c(x_i, y_j) = g_{ij}(\bar{w})$. Here, \bar{w} denote the set of all option weights in the graph G .
2. **Determine equations for option weights.** Using the equations from Step 1 and the CAP, determine a set of equations that relate option weights to each other.
3. **Compute weights.** Solve the set of equations from Step 2.
4. **Resolve References.** Utilize/interpret the weights computed in Step 3 as well as attribute-based similarity to resolve references.

We now discuss the above steps in more detail in the following subsections.

4.1 Computing Connection Strength Computation of $c(x_i, y_j)$ consists of two phases. The first phase discovers connections between x_i and y_j . The second phase computes/measures the strength in connections discovered by the first phase.

4.1.1 The connection discovery phase. In general there can be many connections between $v[x_i]$ and $v[y_j]$ in G . Intuitively, many of those (e.g., very long

ones) are not very important. To capture most important connections while still being efficient, the algorithm computes the set of all L -short simple paths $\mathcal{P}_L(x_i, y_j)$ between nodes $v[x_i]$ and $v[y_j]$ in graph G . A path is L -short if its length is no greater than parameter L . A path is *simple* if it does not contain duplicate nodes.

Illegal paths. Not all of the discovered paths are considered when computing $c(x_i, y_j)$ (to resolve reference $x_i.r_k$). Let e_1, e_2, \dots, e_N be the option-edges associated with the reference $x_i.r_k$. When resolving $x_i.r_k$, RelDC tries to determine the weights of these edges via connections that exist in the remainder of the graph not including those edges. To achieve this,

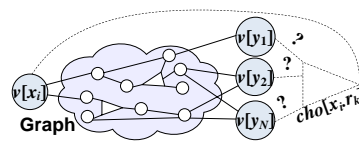


Figure 3: Graph

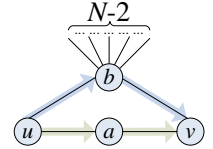


Figure 4: $c(p)$?

RelDC actually discovers paths not in graph G , but in $\tilde{G} = G - cho[x_i.r_k]$, see Figure 3. That is, \tilde{G} is graph G with node $cho[x_i.r_k]$ removed. Also, in general, paths considered when computing $c(x_i, y_j)$ may contain option-edges of some choice nodes. If a path contains an option-edge of a choice node, it should not contain another option-edge of the same choice node. For instance, if a path used to compute connection strength between two nodes in the graph contained an option edge e_j of the choice node shown in Figure 2, it must not contain any of the rest of the option-edges $e_1, e_2, \dots, e_{j-1}, e_{j+1}, \dots, e_N$.

4.1.2 Computing connection strength A natural way to compute the connection strength $c(u, v)$ between nodes u and v is to compute it as the probability to reach node v from node u via random walks in graph G where each step is done with certain probability. Such problems have been studied for graphs in the previous work under Markovian assumptions. The graph in our case is not Markovian due to presence of illegal paths (introduced by choice nodes). So those approaches cannot be applied directly. In [16] we have developed the *probabilistic model (PM)* which treats edge weights as probabilities that those edges exist and which can handle illegal paths. In this section we present the *weight-based model (WM)* which is a simplification of PM. Other models can be derived from [11, 24].

WM is a very intuitive model which is suited well for illustrating issues related to computing $c(u, v)$. WM computes $c(u, v)$ as the sum $\sum_{p \in \mathcal{P}_L(u, v)} c(p)$ of the connection strength $c(p)$ of each path p in $\mathcal{P}_L(u, v)$. The connection strength $c(p)$ of path p from u to v is the probability to follow path p in graph G . Next we

explain how WM computes $c(p)$.

Motivating $c(p)$ formula. Which factors should be taken into account when computing the connection strength $c(p)$ of each individual path p ?

Figure 4 illustrates two different paths (or connections) between nodes u and v : $p_a = u \rightarrow a \rightarrow v$ and $p_b = u \rightarrow b \rightarrow v$. Assume that all edges in this figure have weight of 1. Let us understand which connection is better.

Both connections have an equal length of two. One connection is going via node a and the other one via b . The intent of Figure 4 is to show that b “connects” many things, not just u and v , whereas a “connects” only u and v . We argue the connection between u and v via b is much weaker than the connection between u and v via a : since b connects many things it is not surprising we can connect u and v via b . For example, for the author matching problem, u and v can be two authors, a can be a publication and b a university.

To capture the fact that $c(p_a) > c(p_b)$, we measure $c(p_a)$ and $c(p_b)$ as the probabilities to follow paths p_a and p_b respectively. Notice, measures such as path length, network flow do not capture this fact. We compute those probabilities as follows. For path p_b we start from u . Next we have a choice to go to a or b with probabilities of $\frac{1}{2}$, and we choose to follow (u, b) edge. From node b we can go to any of the $N - 1$ nodes (cannot go back to u) but we go specifically to v . So the probability to reach v via path p_b is $\frac{1}{2(N-1)}$. For path p_a we start from u , we go to a with probability $\frac{1}{2}$ at which point we have no choice but to go to v , so the probability to follow p_a is $\frac{1}{2}$.

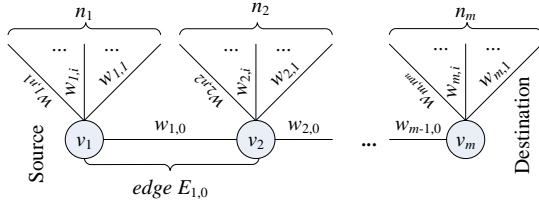


Figure 5: Computing $c(p)$ of path $p = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_m$. Only “possible-to-follow” edges are shown.

General WM formula. In general, each L -short simple path p can be viewed as a sequence of m nodes v_1, v_2, \dots, v_m , where $m \leq L + 1$, as shown in Figure 5. Figure 5 shows that from node v_i it is *possible to follow*² $n_i + 1$ edges labeled $w_{i,0}, w_{i,1}, \dots, w_{i,n_i}$. The probability to follow the edge labeled $w_{i,0}$ is proportional to weight $w_{i,0}$ and computed as $w_{i,0} / (\sum_{j=0}^{n_i} w_{i,j})$. The probability to follow path p is computed as the probability to follow each of its edges:

$$(4.1) \quad c(p) = \prod_{i=1}^{m-1} \frac{w_{i,0}}{\sum_{j=0}^{n_i} w_{i,j}}.$$

²It is not possible to follow zero-weight edges, and edges following which would make the path not simple.

The total connection strength between nodes u and v is computed as the sum of connection strengths of paths in $\mathcal{P}_L(u, v)$:

$$(4.2) \quad c(u, v) = \sum_{p \in \mathcal{P}_L(u, v)} c(p).$$

Measure $c(u, v)$ is the probability to reach v from u by following only L -short simple paths, such that the probability to follow an edge is proportional to the weight of the edge.

For instance, for the toy database we have:

1. $c_1 = c(P_2, \text{'Dave White'}) = c(P_2 \rightarrow \text{Susan} \rightarrow \text{MIT} \rightarrow \text{John} \rightarrow P_1 \rightarrow \text{Don} \rightarrow P_4 \rightarrow \text{Joe} \rightarrow P_5 \rightarrow \text{Liz} \rightarrow P_6 \rightarrow \text{'2'} \rightarrow \text{Dave White}) = \frac{w_2}{2}$.
2. $c_2 = c(P_2, \text{'Don White'}) = c(P_2 \rightarrow \text{Susan} \rightarrow \text{MIT} \rightarrow \text{John} \rightarrow P_1 \rightarrow \text{'Don White'}) = 1$.
3. $c_3 = c(P_6, \text{'Dave White'}) = \frac{w_1}{2}$
4. $c_4 = c(P_6, \text{'Don White'}) = 1$

4.2 Determining equations for option-edge weights

Given the connection strength measures $c(x_i, y_j)$ for each unresolved reference $x_i.r_k$ and its options y_j , we can use the context attraction principle to determine the relationships between the weights associated with the option-edges in the graph G . Note that the context attraction principle does not contain any specific strategy on how to relate weights to connection strengths. Any strategy that assigns weight such that if $c_i \geq c_j$ then $w_i \geq w_j$ is appropriate, where $c_i = c(x_i, y_i)$ and $c_j = c(x_i, y_j)$. In particular, we use the strategy where weights w_1, w_2, \dots, w_N are proportional to the corresponding connection strengths: $w_j \cdot c_i = w_i \cdot c_j$. Using this strategy weight w_j ($j = 1, 2, \dots, N$) is computed as:

$$(4.3) \quad w_j = \begin{cases} c_j / (\sum_{l=1}^N c_l) & \text{if } \sum_{l=1}^N c_l > 0; \\ \frac{1}{N} & \text{if } \sum_{l=1}^N c_l = 0. \end{cases}$$

For instance, for the toy database we have:

1. $w_1 = c_1 / (c_1 + c_2) = \frac{w_3}{2} / (1 + \frac{w_3}{2})$
2. $w_2 = c_2 / (c_1 + c_2) = 1 / (1 + \frac{w_3}{2})$
3. $w_3 = c_3 / (c_3 + c_4) = \frac{w_1}{2} / (1 + \frac{w_1}{2})$
4. $w_4 = c_4 / (c_3 + c_4) = 1 / (1 + \frac{w_1}{2})$

4.3 Determining all weights by solving equations

Given a system of equations relating option-edge weights as derived in Section 4.2, our goal next is to determine values for the option-edge weights that satisfy the equations. Before we discuss how such equations can be solved in general, let us first solve the option-edge weight equations in the toy example. These equations, given an additional constraint that all weights should be in $[0, 1]$, have a unique solution $w_1 = 0$, $w_2 = 1$, $w_3 = 0$, and $w_4 = 1$. Once we have computed the weights, RelDC will interpret these weights to resolve the references. In the toy example, weights $w_1 = 0$,

$w_2 = 1$, $w_3 = 0$, and $w_4 = 1$ will lead RelDC to resolve ‘D. White’ in both P_2 and P_6 to ‘Don White’.

In general case, Equations (4.3), (4.1), and (4.2) define each option weight as a function of other option weights: $w_i = f_i(\overline{w})$. The exact function for w_j is determined by Equations (4.3), (4.1), and (4.2) and by the paths that exist between $v[x_i]$ and $v[y_j]$ in G . Often, in practice, $f_i(\overline{w})$ is constant leading to the equation of the form $w_i = \text{const}$.

The goal is to find such a combination of weights that “satisfies” the system of $w_i = f_i(\overline{w})$ equations along with the constraints on the weights. Since a system of equations, each of the type $w_i = f_i(\overline{w})$, might not have an exact solution, we transform the equations into the form $f_i(\overline{w}) - \delta_i \leq w_i \leq f_i(\overline{w}) + \delta_i$. Here variable δ_i , called *tolerance*, can take on any real nonnegative value. The problem transforms into solving the NLP problem where the constraints are specified by the inequalities above and the objective is to minimize the sum of all δ_i ’s. Additional constraints are: $0 \leq w_i \leq 1$, $\delta_i \geq 0$, for all w_i, δ_i . In [16] we argue that such a system of equations always has a solution.

The straightforward approach to solving the resulting NLP problem is to use one of the off-the-shelf math solver such as SNOPT. Such solvers, however, do not scale to large problem sizes that we encounter in data cleaning as will be discussed in Section 5. We therefore exploit a simple iterative approach, which is outlined below.³ The iterative method first iterates over each reference $x_i.r_k$ and assigns weight of $\frac{1}{|CS[x_i.r_k]|}$ to each w_j . It then starts its major iterations in which it first computes $c(x_i, y_j)$ for all i and j , using Equation (4.2). Then it uses those $c(x_i, y_j)$ ’s to compute all w_j ’s using Equation (4.3). Note that the values of w_j ’s will change from $\frac{1}{|CS[x_i.r_k]|}$ to new values. The algorithm performs several major iterations until the weights converge (the resulting changes across iterations are negligible) or the algorithm is explicitly stopped.

Let us perform one iteration of the iterative method for the example above. First $w_1 = w_2 = \frac{1}{2}$ and $w_3 = w_4 = \frac{1}{2}$. Next $c_1 = \frac{1}{4}$, $c_2 = 1$, $c_3 = \frac{1}{4}$, and $c_4 = 1$. Finally, $w_1 = \frac{1}{5}$, $w_2 = \frac{4}{5}$, $w_3 = \frac{1}{5}$, and $w_4 = \frac{4}{5}$. If we stop the algorithm at this point and interpret w_j ’s, then the RelDC’s answer is identical to that of the exact solution: ‘D. White’ is ‘Don White’.

Note that the above described iterative procedure computes only an *approximate* solution for the system whereas the solver finds the exact solution. Let us refer to iterative implementation of RelDC as *Iter-RelDC* and

denote the implementation that uses a solver as *Solv-RelDC*. For both Iter-RelDC and Solv-RelDC, after the weights are computed, those weights are **interpreted** to produce the final result, as discussed in Section 4. It turned out that the accuracy of Iter-RelDC (with a small number of iterations, such as 10–20) and of Solv-RelDC is practically identical. This is because even though the iterative method does not find the exact weights, those weights are close enough to those computed using a solver. Thus, when the weights are *interpreted*, both methods obtain similar results.

4.4 Resolving references by interpreting weights. When resolving references $x_i.r_k$ and deciding which entity among y_1, y_2, \dots, y_N from $CS[x_i.r_k]$ is $d[x_i.r_k]$, RelDC chooses such y_j that w_j is the largest among w_1, w_2, \dots, w_N . Notice, to resolve $x_i.r_k$ we could have also combined w_j weights with feature-based similarities $FBS(x_i, y_j)$ (e.g., as a weighted sum), but we do not study that approach in this paper.

5 Experimental Results

In this section we experimentally study RelDC using two real (*publications* and *movies*) and synthetic datasets. RelDC was implemented using C++ and SNOPT solver [4]. The system runs on a 1.7GHz Pentium machine. We test and compare the following implementations of RelDC:

1. **Iter-RelDC** vs. **Solv-RelDC**. If neither ‘Iter-’ nor ‘Solv-’ is specified, Iter-RelDC is assumed.
2. **WM-RelDC** vs. **PM-RelDC**. The prefixes indicate whether the weight-based model (WM) from Section 4.1.2 or probabilistic model (PM) from [16], is used for computing connection strengths. By default WM-RelDC is assumed.

In each of the RelDC implementations, the value of L used in computing the L -short simple paths is set to 7 by default. In [16] we show that WM-Iter-RelDC is one of the best implementations of RelDC in terms of both accuracy and efficiency. That is why the bulk of our experiments use that implementation.

5.1 Case Study 1: the publications dataset

5.1.1 Datasets In this section we will introduce RealPub and SynPub datasets. Our experiments will solve *author matching* (AM) problem, defined in Section 2, on these datasets.

RealPub dataset. RealPub is a real dataset constructed from *two* public-domain sources: CiteSeer[1] and HPSearch[2]. CiteSeer can be viewed as a collection of research publications, HPSearch as a collection of information about authors. HPSearch can be viewed as

³Methods different from Iter-RelDC can be used to compute an approximate solution as well: e.g. [16] sketches another solution which is based on computing the bounding intervals for the option weights and then applying the techniques from [9, 8, 10].

a set of $\langle \text{id}, \text{authorName}, \text{department}, \text{organization} \rangle$ tuples. That is the affiliation consists of not just organization like in Section 2, but also of department. Information stored in CiteSeer is in the same form as specified in Section 2, that is $\langle \text{id}, \text{title}, \text{authorRef1}, \dots, \text{authorRefN} \rangle$ per each paper. [16] contains sample content of CiteSeer and HPSearch as well as the corresponding entity-relationship graph.

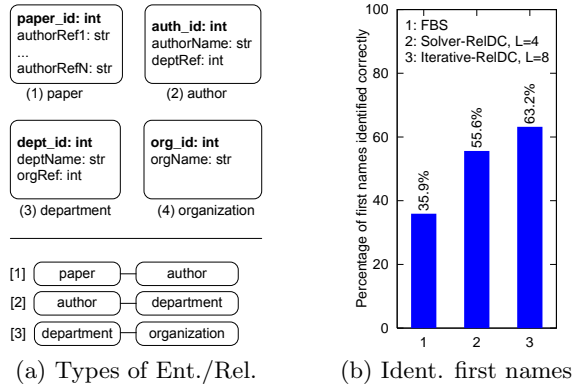


Figure 6: Experiments

The various types of entities and relationships present in RealPub are shown in Figure 6(a). RealPub consists of 4 *types* of entities: papers (255K), authors (176K), organizations (13K), and departments (25K). To avoid confusion we use “authorRef” for author names in *paper* entities and “authorName” for author names in *author* entities. There are 573K authorRef’s in total. Our experiments on RealPub will explore the efficacy of RelDC in resolving these references.

To test RelDC, we first constructed an entity-relationship graph G for the RealPub database. Each node in the graph corresponds to an entity of one of these types. If author A is affiliated with department D , then there is $(v[A], v[D])$ edge in the graph. If department D is a part of organization U , then there is $(v[D], v[U])$ edge. If paper P is written by author A , then there is $(v[A], v[P])$ edge. For each of the 573K authorRef references, feature-based similarity (FBS) was used to construct its choice set.

In the RealPub dataset, the paper entities refer to authors using **only their names (and not affiliations)**. This is because the paper entities are derived from the data available from CiteSeer which did not directly contain information about the author’s affiliation. As a result, only similarity of author names was used to initially construct the graph G .

This similarity has been used to construct choice sets for all authorRef references. As the result, 86.9% (498K) of all authorRef references had choice set of size one and the corresponding papers and authors were linked directly. For the remaining 13.1% (75K) refer-

ences, 75K choice nodes were created in the graph G . RelDC was used to resolve these remaining references. The specific experiments conducted and results will be discussed later in this section. Notice that the RealPub dataset allowed us to test RelDC only under the condition that a majority of the references are already correctly resolved. To test robustness of the technique we tested RelDC over synthetic datasets where we could vary the uncertainty in the references from 0 to 100%.

SynPub dataset. We have created two synthetic datasets SynPub1 and SynPub2, that emulate RealPub. The synthetic datasets were created since, for the RealPub dataset, we do not have the true mapping between papers and the authors of those papers. Without such a mapping, as will become clear when we describe experiments, testing for accuracy of reference disambiguation algorithm requires a manual effort (and hence experiments can only validate the accuracy over small samples). In contrast, since in the synthetic datasets, the *paper-author* mapping is known in advance, accuracy of the approach can be tested over the entire dataset. Another advantage of the SynPub dataset is that by varying certain parameters we can manually control the nature of this dataset allowing for the evaluation of all aspects of RelDC under various conditions (e.g., varying level of ambiguity/uncertainty in the dataset).

Both the SynPub1 and SynPub2 datasets contain 5000 papers, 1000 authors, 25 organizations and 125 departments. The average number of choice nodes that will be created to disambiguate the authorRef’s is 15K (notice, the whole RealPub dataset has 75K choice nodes). The difference between SynPub1 and SynPub2 is that author names are constructed differently: SynPub1 uses unc_1 and SynPub2 uses unc_2 as will be explained shortly.

5.1.2 Accuracy experiments In our context, the *accuracy* is the fraction of all authorRef references that are resolved correctly. This definition includes the references that have choice sets of cardinality 1.

Experiment 1 (RealPub: manually checking samples for accuracy). Since the correct *paper-author* mapping is not available for RealPub, it is infeasible to test the accuracy on this dataset. However it is possible to find a portion of this *paper-author* mapping *manually* for a sample of RealPub by going to authors web pages and examining their publications.

We have applied RelDC to RealPub in order to test the effectiveness of analyzing relationships. To analyze the accuracy of the result, we concentrated only on the 13.1% of uncertain authorRef references. Recall, the cardinality of the choice set of each such reference is at least two. For 8% of those references there were

no $x_i \rightsquigarrow y_j$ paths for all j 's, thus RelDC used only FBS and not relationships. Since we want to test the effectiveness of analyzing relationships, we remove those 8% of references from further consideration as well. We then chose a random samples of 50 papers that were still left under consideration. For this sample we compared the reference disambiguation result produced by RelDC with the true matches. The true matches for **authorRef** references in those papers were computed manually. In this experiment, RelDC was able to resolve **all** of the 50 sample references correctly! This outcome is in reality not very surprising since in the RealPub datasets, the number of references that were ambiguous was only 13.1%. Our experiments over the synthetic datasets will show that RelDC reaches very high disambiguation accuracy when the number of uncertain references is not very high.

Ideally, we would have liked to have performed further accuracy tests over RealPub by either testing on larger samples (more than 50) and/or repeating the test multiple times (in order to establish confidence levels). However, this is infeasible due to the time-consuming manual nature of this experiments. \square

Experiment 2 (RealPub: accuracy of identifying author first names). We conducted another experiment over RealPub dataset to test the efficacy of RelDC in disambiguating references which we describe below.

We first remove from RealPub all the paper entities which have an **authorRef** in format “*first initial + last name*”. This leaves only papers with **authorRef**'s in format “*full first name + last name*”. Then we pretend we only know “*first initial + last name*” for those **authorRef**'s. Next we run FBS and RelDC and see whether or not they would disambiguate those **authorRef**'s to authors whose full first names coincide with the original full first names. In this experiment, for 82% of the **authorRef**'s the cardinality of their choice sets is 1 and there is nothing to resolve. For the rest 18% the problem is more interesting: the cardinality of their choice sets is at least 2. Figure 6(b) shows the outcome for those 18%.

Notice that the reference disambiguation problem tested in the above experiment is of a limited nature – the tasks of identifying the correct first name of the author and the correct author are not the same in general.⁴ Nevertheless, the experiment allows us to test the accuracy of RelDC over the entire database and does show the strength of the approach. \square

Accuracy on SynPub. The next set of experiments tests accuracy of RelDC and FBS approaches on

SynPub dataset. “RelDC 100%” (“RelDC 80%”) means for 100% (80%) of *author* entities the affiliation information is available. Once again, *paper* entities do not have author affiliation attributes, so FBS cannot use affiliation, see Figure 6(a). Thus those 100% and 80% have no effect on the outcome of FBS. Notation ‘L=4’ means RelDC explores paths of length no greater than 4.

Experiment 3 (Accuracy on SynPub1). SynPub1 uses *uncertainty of type 1* defined as follows. There are $N_{auth} = 1000$ unique authors in SynPub1. But there are only $N_{name} \in [1, N_{auth}]$ unique **authorName**'s. We construct the **authorName** of the author with ID of k , for $k = 0, 1, \dots, 999$, as “name” concatenated with $(k \bmod N_{name})$. Each **authorRef** specifies one of those **authorName**'s. Parameter unc_1 is $unc_1 = \frac{N_{auth}}{N_{name}}$ ratio. For instance, if N_{name} is 750, then the authors with IDs of 1 and 751 have the same **authorName**: “name1”, and $unc_1 = \frac{1000}{750} = 1\frac{1}{3}$. In SynPub1 for each author whose name is not unique, one can never identify with 100% confidence any paper this author has written. Thus the uncertainty for such authors is very high.

Figure 7 studies the effect of unc_1 on accuracy of RelDC and FBS. If $unc_1 = 1.0$, then there is no uncertainty and all methods have accuracy of 1.0. As expected, the accuracy of all methods monotonically decreases as uncertainty increases. If $unc_1 = 2.0$, the uncertainty is very large: for any given author there is exactly one another author with the identical **authorName**. For this case, any FBS have no choice but to guess one of the two authors. Therefore, the accuracy of any FBS, as shown in Figures 7, is 0.5. However, the accuracy of RelDC 100% (RelDC 80%) when $unc_1 = 2.0$ is 94%(82%). The gap between RelDC 100% and RelDC 80% curves shows that in SynPub1 RelDC relies substantially on author affiliations for the disambiguation.

Comparing the RelDC implementations. Figure 8 shows that the accuracy results of WM-Iter-RelDC, PM-Iter-RelDC, WM-Solv-RelDC implementations are comparable. Figure 9 shows that Iter-RelDC is the fastest implementation among them. The same trend has been observed for all other tested cases. \square

Experiment 4 (Accuracy on SynPub2). SynPub2 uses *uncertainty of type 2*. In SynPub2, **authorName**'s (in *author* entities) are constructed such that the following holds, see Figure 6(a). If an **authorRef** reference (in a *paper* entity) is in the format “*first name + last name*” then it matches only one (correct) author. But if it is in the format “*first initial + last name*” it matches exactly two authors. Parameter unc_2 is the fraction of **authorRef**'s specified as “*first initial + last name*”. If $unc_2 = 0$, then there is no uncertainty and the accuracy of all methods is 1. Also notice that the case when

⁴It is not enough to determine that ‘J.’ in ‘J. Smith’ corresponds to ‘John’ if there are multiple ‘John Smith’'s in the dataset.

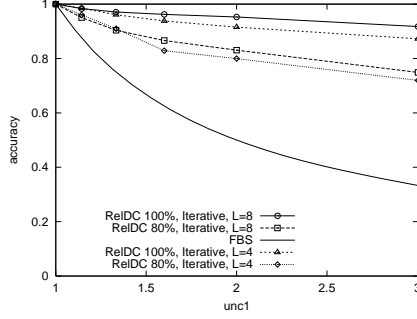


Figure 7: SynPub1: accuracy

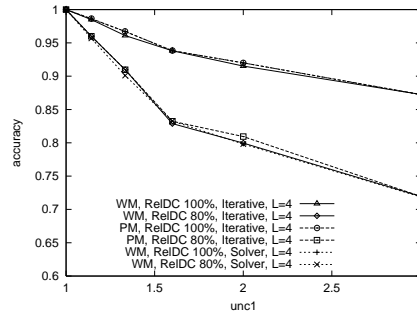


Figure 8: RelDC implementations

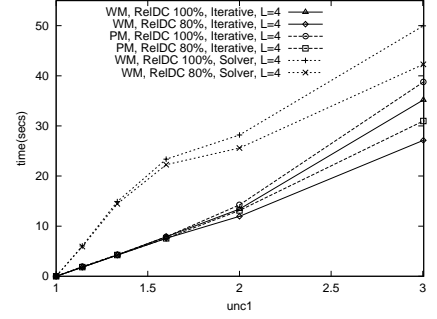


Figure 9: SynPub1: efficiency

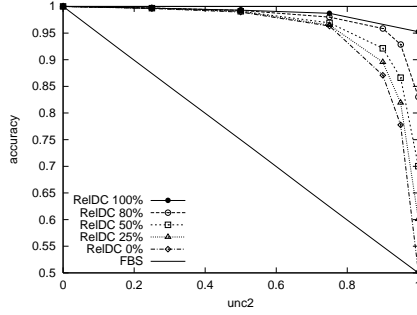


Figure 10: SynPub2: Acc. vs. unc_2

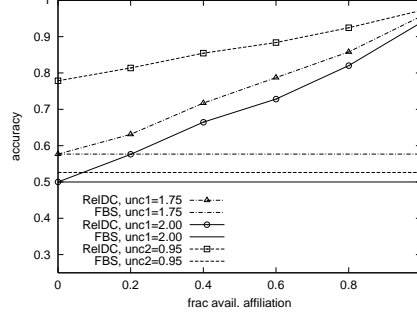


Figure 11: SynPub: affiliation

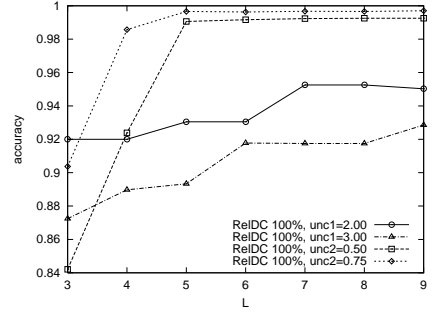


Figure 12: SynPub: Acc vs. L

$unc_2 = 1.0$ is equivalent to $unc_1 = 2.0$.

There is less uncertainty in Experiment 4 than in Experiment 3. This is because for each author there is a chance that he is referenced to by his full name in some of his papers, so for these cases the *paper-author* associations are known with 100% confidence.

Figure 10 shows the effect of unc_2 on the accuracy of RelDC. As in Figure 7, in Figure 10 the accuracy decreases as uncertainty increases. However this time the accuracy of RelDC is much higher. The fact that curves for RelDC 100% and 80% are almost indiscernible until unc_2 reaches 0.5, shows that RelDC relies less heavily on weak author affiliation relationships but rather on stronger connections via papers. \square

5.1.3 Other experiments

Experiment 5 (Importance of relationships). Figure 11 studies what effect the number of relationships and the number of relationship *types* have on the accuracy of RelDC. When resolving *authorRef*'s, RelDC uses three types of relationships: (1) *paper-author*, (2) *author-department*, (3) *department-organization*.⁵ The affiliation relationships (i.e., (2) and (3)) are derived from the affiliation information in *author* entities.

The affiliation information is not always available for each *author* entity in RealPub. In our synthetic

datasets we can manually vary the amount of available affiliation information. The x -axis shows the fraction ρ of *author* entities for which their affiliation is known. If $\rho = 0$, then the affiliation relationships are eliminated completely and RelDC has to rely solely on connections via *paper-author* relationships. If $\rho = 1$, then the complete knowledge of author affiliations is available. Figure 11 studies the effect of ρ on accuracy. The curves in this figure are for both SynPub1 and SynPub2: $unc_1 = 1.75$, $unc_1 = 2.00$, and $unc_2 = 0.95$. The accuracy increases as ρ increases showing that RelDC deals with newly available relationships well. \square

Experiment 6 (Longer paths). Figure 12 examines the effect of path limit parameter L on the accuracy. For all the curves in the figure, the accuracy monotonically increases as L increases with the only one exception for “RelDC 100%, $unc_1=2$ ” and $L = 8$. The usefulness of longer paths depends on the combination of other parameters. Typically, there is a tradeoff: larger values of L lead to higher accuracy of disambiguation but slower performance. The user running RelDC must decide the value of L based on this accuracy/performance tradeoff for the dataset being cleaned. For SynPub, $L = 7$ is a reasonable choice. \square

Experiment 7 (Efficiency of RelDC). To show the applicability of RelDC to a large dataset we have successfully applied an optimized version of RelDC to clean RealPub with L ranging from 2 up to 8. Figure 13 shows the execution time of RelDC as a function of the

⁵Note, a ‘type of relationship’ (e.g., *paper-author*) is different from a ‘chain of relationships’ (e.g., *paper1-author1-dept1-...*).

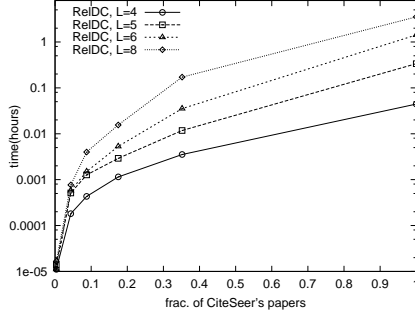


Figure 13: RealPub: efficiency

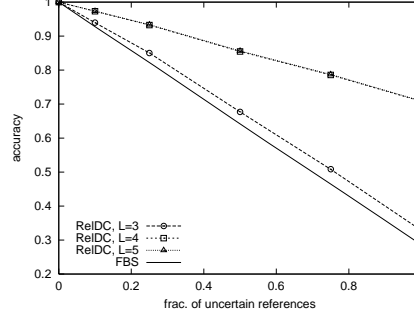


Figure 14: RealMov: *director* refs.

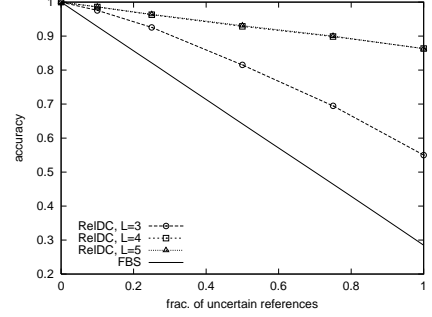


Figure 15: RealMov: *studio* refs.

fraction of papers from RealPub, e.g. 1.0 corresponds to all papers in RealPub (the whole CiteSeer) dataset. Notice, optimizations of RelDC are discussed only in [16], they are crucial to achieve 1–2 orders of magnitude of improvement in performance. \square

5.2 Case Study 2: the movies dataset

5.2.1 Dataset RealMov is a real public-domain movies dataset described in [25] which has been made popular by the textbook [13]. Unlike RealPub dataset, in RealMov all the needed correct mappings are known, so it is possible to test the disambiguation accuracy of various approaches more extensively. However, RealMov dataset is much smaller compared to the RealPub dataset. RealMov contains entities of three types: *movies* (11,453 entities), *studios* (992 entities), and *people* (22,121 entities). There are five types of relationships in the RealMov dataset: *actors*, *directors*, *producers*, *producingStudios*, and *distributingStudios*. Relationships *actors*, *directors*, and *producers* map entities of type *movies* to entities of type *people*. Relationships *producingStudios* and *distributingStudios* map *movies* to *studios*. [16] contains the sample graph for RealMov dataset as well as sample content of *people*, *movies*, *studios* and *cast* tables from which it has been derived.

5.2.2 Accuracy experiments

Experiment 8 (RealMov: Accuracy of disambiguating *director* references). In this experiment we study the accuracy of disambiguating references from movies to directors of those movies.

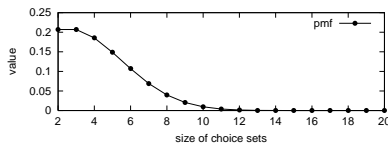


Figure 16: PMF of sizes of choice sets.

Since in RealMov each reference, including each *director* reference, already points directly to the right match, we artificially introduce ambiguity in the refer-

ences manually. Similar approach to testing data cleaning algorithms have also been used by other researchers, e.g. [7]. Given the specifics of our problem, to study the accuracy of RelDC we will simulate that we used FBS to determine the choice set of each reference but FBS was uncertain in some of the cases.

To achieve that, we first choose a fraction ρ of *director* references (that will be uncertain). For each reference in this fraction we will simulate that FBS part of RelDC has done its best but still was uncertain as follows. Each *director* reference from this fraction is assigned a choice set of N people. One of those people is the true director, the rest ($N-1$) are chosen randomly from the set of *people* entities.

Figure 14 studies the accuracy as ρ is varied from 0 to 1 and where N is distributed according to the probability mass function (pmf) shown in Figure 16, see [16] for detail. The figure shows that RelDC achieves better accuracy than FBS. The accuracy is 1.0 when $\rho = 0$, since all references are linked directly. The accuracy decreases almost linearly as ρ increases to 1. When $\rho = 1$, the cardinality of the choice set of each reference is at least 2. The larger the value of L , the better the results. The accuracy of RelDC improves significantly as L increases from 3 to 4. However, the improvement is less significant as L increases from 4 to 5. Thus the analyst must decide whether to spend more time to obtain higher accuracy with $L = 5$, or whether $L = 4$ is sufficient. \square

Experiment 9 (RealMov: Accuracy of disambiguating *studio* references). This experiment is similar to Experiment 8, but now we disambiguate *producingStudio*, instead of *director*, references. Figure 15 corresponds to Figure 14. The RelDC's accuracy of disambiguating *studio* references is even higher. \square

6 Related Work

Many research challenges have been explored in the context of data cleaning in the literature: dealing with missing data, handling erroneous data, record linkage, and so on. The closest to the problem of reference

disambiguation addressed in this paper is the problem of record linkage. The importance of record linkage is underscored by the large number of companies, such as Trillium, Vality, FirstLogic, DataFlux, which have developed (domain-specific) record linkage solutions.

Researchers have also explored domain-independent techniques, e.g. [23, 12, 14, 5, 22]. Their work can be viewed as addressing two challenges: (1) improving similarity function, as in [6]; and (2) improving efficiency of linkage, as in [7]. Typically two-level similarity functions are employed to compare two records. First, such a function computes attribute-level similarities by comparing values in the same attributes of two records. Next the function combines the attribute-level similarity measures to compute the overall similarity of two records. A recent trend has been to employ machine learning techniques, e.g. SVM, to learn the best similarity function for a given domain [6]. Many techniques have been proposed to address the efficiency challenge as well: e.g. using specialized indexes [7], sortings, etc.

Those domain-independent techniques deal only with attributes. To the best of our knowledge, RelDC, which was first publicly released in [15], is the first domain-independent data cleaning framework which exploits relationships for cleaning. Recently, in parallel to our work, other researchers have also proposed using relationships for cleaning. In [5] Ananthakrishna et al. employ similarity of directly linked entities, for the case of hierarchical relationships, to solve the record de-duplication challenge. In [19] Lee et al. develop an association-rules mining based method to disambiguate references using similarity of the context attributes: the proposed technique is still an FBS method, but [19] also discusses concept hierarchies which are related to relationships. Getoor et al. in DKDM04 use similarity of attributes of directly linked objects, like in [5], for the purpose of object consolidation. However, the challenge of applying that technique in practice on real-world datasets was identified as future work in that paper. In contrast to the above described techniques, RelDC utilize the CAP principle to automatically discover and analyze relationship chains, thereby establishing a framework that employs systematic relationship analysis for the purpose of cleaning.

7 Conclusion

In this paper we have shown that analysis of inter-object relationships is important for data cleaning and demonstrated one approach that utilizes relationships. As future work we plan to apply similar techniques to the problem of record linkage. This paper outlines only the core of the RelDC approach, for more details the interested reader is referred to [16]. Another interesting follow-up work [18] addresses the challenge

of automatically adapting RelDC to datasets at hand by learning how to weigh different connections directly from the data. Solving this challenge, in general, not only makes the approach to be a plug-and-play solution but also can improve the accuracy as well as efficiency of the approach as discussed in [18].

References

- [1] CiteSeer. <http://citeseer.nj.nec.com/cs>.
- [2] HomePageSearch. <http://hpsearch.uni-trier.de>.
- [3] Knowledge Discovery. <http://www.kdnuggets.com/polls/2003/data-preparation.htm>.
- [4] SNOPT solver. <http://www.gams.com/solvers/>.
- [5] Ananthakrishna, Chaudhuri, and Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.
- [6] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, 2003.
- [7] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proc. of ACM SIGMOD Conf.*, 2003.
- [8] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proc. ACM SIGMOD Conf.*, 2003.
- [9] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *IEEE TKDE*, 16(9), Sept. 2004.
- [10] R. Cheng, S. Prabhakar, and D. Kalashnikov. Querying imprecise data in moving object environments. In *Proc. IEEE ICDE Conf.*, 2003.
- [11] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.
- [12] I. Fellegi and A. Sunter. A theory for record linkage. *J. of Amer. Stat. Assoc.*, 64(328):1183–1210, 1969.
- [13] H. Garcia-Molina, J. Ullman, and J. Widom. *Database systems: the complete book*. Prentice Hall, 2002.
- [14] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proc. of SIGMOD*, 1995.
- [15] D. Kalashnikov and Mehrotra. Exploiting relationships for data cleaning. *TR-RESCUE-03-02*, Nov. 2003.
- [16] D. Kalashnikov and S. Mehrotra. Exploiting relationships for domain-independent data cleaning. *Extended Version of SIAM Data Mining 2005 publication*, <http://www.ics.uci.edu/~dvk/pub/sdm05.pdf>.
- [17] D. V. Kalashnikov and S. Mehrotra. RelDC project. <http://www.ics.uci.edu/~dvk/RelDC/>.
- [18] D. V. Kalashnikov and S. Mehrotra. Learning importance of relationships for reference disambiguation. *Submitted for Publication*, Dec. 2004. <http://www.ics.uci.edu/~dvk/RelDC/TR-RESCUE-04-23.pdf>.
- [19] M. Lee, W. Hsu, and V. Kothari. Cleaning the spurious links in data. *IEEE Intelligent Systems*, Mar-Apr 2004.
- [20] R. Little and D. Rubin. *Statistical Analysis with Missing Data*. John Wiley and Sons, 1986.
- [21] J. Maletic and A. Marcus. Data cleansing: Beyond integrity checking. In *Conf. on Inf. Quality*, 2000.
- [22] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD*, 2000.
- [23] Newcombe, Kennedy, Axford, and James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.
- [24] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.
- [25] G. Wiederhold. www-db.stanford.edu/pub/movies/.

A Spectral Clustering Approach To Finding Communities in Graphs*

Scott White[†] and Padhraic Smyth[†]

Abstract

Clustering nodes in a graph is a useful general technique in data mining of large network data sets. In this context, Newman and Girvan [9] recently proposed an objective function for graph clustering called the Q function which allows automatic selection of the number of clusters. Empirically, higher values of the Q function have been shown to correlate well with good graph clusterings. In this paper we show how optimizing the Q function can be reformulated as a spectral relaxation problem and propose two new spectral clustering algorithms that seek to maximize Q . Experimental results indicate that the new algorithms are efficient and effective at finding both good clusterings and the appropriate number of clusters across a variety of real-world graph data sets. In addition, the spectral algorithms are much faster for large sparse graphs, scaling roughly linearly with the number of nodes n in the graph, compared to $O(n^2)$ for previous clustering algorithms using the Q function.

1 Introduction

Large complex graphs representing relationships among sets of entities are an increasingly common focus of scientific inquiry. Examples include social networks, Web graphs, telecommunication networks, semantic networks, and biological networks. One of the key questions in understanding such data is “How many communities are there and what are the community memberships?”

Algorithms for finding such communities, or automatically grouping nodes in a graph into clusters, have been developed in a variety of different areas, including VLSI design, parallel computing, computer vision, social networks, and more recently in machine learning. Good algorithms for graph clustering hinge on the quality of the objective function being used. A variety of different objective functions and clustering algorithms have been proposed for this problem, ranging from hierarchical clustering to max-flow/min-cut methods to methods based on truncating the eigenspace of a suitably-defined matrix. In recent years, much attention has been paid to spectral clustering algorithms (e.g., [11],[12],[14]) that, explicitly or implicitly, attempt to

globally optimize cost functions such as the Normalized Cut measure [12]. The majority of these approaches attempt to balance the size of the clusters while minimizing the interaction between dissimilar nodes. However, for the types of complex heterogeneous networks that arise naturally in many domains, the bias that these approaches have towards clusters of equal size can be seen as a drawback. Furthermore, many of these measures, such as Normalized Cut, can not be used directly for selecting the number of clusters, k , since they increase (or decrease) monotonically as k is varied.

Recently, a new approach was developed by Newman and Girvan [9] to overcome limitations of previous measures for measuring community structure. They proposed the “modularity function” Q , which directly measures the quality of a particular clustering of nodes in a graph. It can also be used to automatically select the optimal number of clusters k , by finding the value of k for which Q is maximized, in contrast to most other objective functions used for graph clustering.

Let $G(V, E, W)$ be an undirected graph consisting of the set of nodes V , the set of edges E , and a symmetric weight matrix $W \in \mathbb{R}^{n \times n}$, where n is the number of vertices. The weights $w_{ij} = w_{ji} = [W]_{ij}$ are positive if there is an edge between vertices v_i and v_j , and 0 otherwise. The modularity function Q can be defined as

$$(1.1) \quad Q(\mathcal{P}_k) = \sum_{c=1}^k \left[\frac{\mathcal{A}(V_c, V_c)}{\mathcal{A}(V, V)} - \left(\frac{\mathcal{A}(V_c, V)}{\mathcal{A}(V, V)} \right)^2 \right]$$

where \mathcal{P}_k is a partition of the vertices into k groups and where $\mathcal{A}(V', V'') = \sum_{i \in V', j \in V''} w(i, j)$. Thus, $\mathcal{A}(V_c, V_c)$ measures the within-cluster sum of weights, $\mathcal{A}(V_c, V)$ measures the sum of weights over all edges attached to nodes in cluster c , and $\mathcal{A}(V, V)$ measures the sum of all edge weights in the graph. Considering binary weights for simplicity, the term $\frac{\mathcal{A}(V_c, V_c)}{\mathcal{A}(V, V)}$ is the empirical probability $\hat{p}_{c,c}$ that both ends of a randomly selected edge from G lie in cluster c . Similarly, $\frac{\mathcal{A}(V_c, V)}{\mathcal{A}(V, V)}$ is the empirical probability \hat{p}_c that a specific end of an edge (either one), for a randomly selected edge, lies in cluster c . Thus, under an independence model, Q can be interpreted as a measure of the deviation between (a) the observed edge-cluster probabilities $\hat{p}_{c,c}$ and (b) what one would predict under an independence model: \hat{p}_c^2 .

*The research in this paper was supported by the National Science Foundation under Grant IRI-9703120 as part of the Knowledge Discovery and Dissemination program. SW was also supported by a National Defense Science and Engineering Graduate Fellowship.

[†]Department of Computer Science, University of California, Irvine

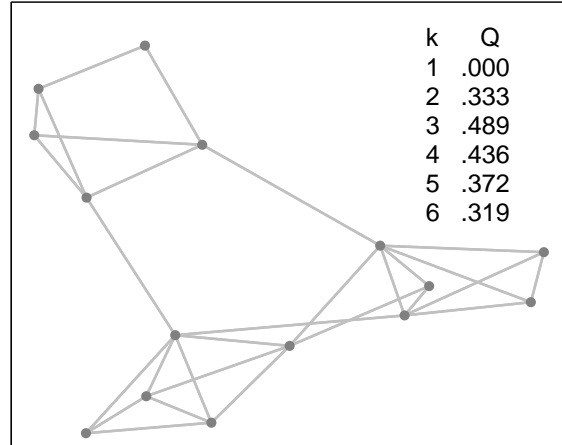


Figure 1: A toy graph showing Q values for different numbers of clusters.

Newman [10] and Newman and Girvan [9] showed across a wide variety of simulated and real-world graphs that larger Q values are correlated with better graph clusterings. In addition, they found that real-world unweighted networks with high community structure generally have Q values within a range from 0.3 to 0.7. Figure 1 shows an example of a simple toy graph with binary weights, where the structure of the graph visually suggests 3 clusters. Also shown are the maximum values of the Q function for different numbers of clusters k , and indeed Q is maximized for $k = 3$.

As pointed out in [10], if no edges exist that connect vertices across clusters then $Q = 1$, and conversely if the number of inter-cluster edges is no better than random then $Q = 0$. We have found empirically that the Q measure works well in practice in terms of both (a) finding good clusterings of nodes in graphs where community structure is evident, and (b) indicating what the appropriate number of clusters k is for such a graph.

In this paper we show how Newman's Q measure can be related to the broader family of spectral clustering methods. Specifically:

- We show how the problem of maximizing the modularity measure Q can be reformulated as an eigenvector problem involving a matrix we call the “ Q -Laplacian.” In this manner we link work on graph clustering using the Q measure to relevant work on spectral clustering (e.g., [11], [12],[14]).
- We use the eigenvector formulation of maximizing Q to derive two new spectral graph clustering algorithms. One of these algorithms directly seeks a global optimum of the Q function. The other algorithm is similar to Newman's agglomerative clustering algorithm [10], in that it attempts to

maximize Q via local iterative improvement.

- We compare the new algorithms with Newman's algorithm on different graph data sets and empirically illustrate that:
 - the spectral approach to maximizing Q produces results that, in terms of cluster quality, are comparable or better than results from Newman's hierarchical algorithm, and
 - the proposed algorithms are linear per iteration in the number of nodes and edges in the graph, compared to quadratic complexity in the number of nodes for the original algorithm proposed by Newman [10].

2 Spectral Approaches to Maximizing the Q Function

Consider for the moment that the number of clusters, k , is fixed. We use the following strategy to address the problem of finding a partitioning that maximizes $Q(P_k)$ as follows:

1. Reformulate the problem of maximizing Newman's Q function as a discrete quadratic assignment problem.
2. Approximate the resulting assignment problem by relaxing it to a continuous one which can be solved analytically using eigen-decomposition techniques.
3. Compute the top $k - 1$ eigenvectors of this solution to form a $k - 1$ -dimensional embedding of the graph into a Euclidean space. Use “hard-assignment” geometric clustering (the k -means algorithm) on this embedding to generate a clustering P_k .

Below we outline each of these steps. In the section which follows, Section 3, we then describe computational details for two proposed clustering algorithms based on this approach.

2.1 Quadratic Assignment We assume G is simple, i.e., G contains no self-loops nor parallel edges, and is connected, i.e., there is a path from any vertex to any other vertex. Let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix having d_i in the i th diagonal entry and 0 everywhere else, where $d_i = \sum_j w_{ij}$. We denote the diagonal matrix derived from an $n \times n$ matrix X as $\text{diag}(X) \in \mathbb{R}^{n \times n}$ where $[\text{diag}(X)]_{ij} = [X]_{ij}$ if $i = j$ and 0 otherwise. Finally, let $\text{tr}(X) = \sum_i [X]_{ii}$ be the trace of matrix X .

To simplify notation, we rewrite Q as follows:

$$(2.2) \quad Q(P_k) \propto \sum_{c=1}^k [\mathcal{A}(V, V) \mathcal{A}(V_c, V_c) - \mathcal{A}(V_c, V)^2]$$

Given a k -partition P_k , define a corresponding $n \times k$ assignment matrix $X = [\mathbf{x}_1, \dots, \mathbf{x}_k]$ with $x_{ic} = 1$ if $v_i \in V_c$, and $x_{ic} = 0$ otherwise, for $1 \leq c \leq k$. Observe that since each vertex can only be in one cluster, $X \mathbf{1}_k = \mathbf{1}_n$. We can reformulate Q in terms of the assignment matrix X as follows:

$$\begin{aligned} Q(P_k) &\propto \sum_{c=1}^k \left[\text{vol}_G \sum_{i,j=1}^n w_{ij} x_{ic} x_{jc} - \left(\sum_{i,j=1}^n w_{ij} x_{ic} \right)^2 \right] \\ &= \sum_{c=1}^k \left[\text{vol}_G \mathbf{x}_c^T W \mathbf{x}_c - \left(\sum_{i=1}^n d_i x_{ic} \right)^2 \right] \\ &= \sum_{c=1}^k \left[\text{vol}_G \mathbf{x}_c^T W \mathbf{x}_c - (\mathbf{d}^T \mathbf{x}_c)^2 \right] \end{aligned}$$

where $\mathbf{d} \in \mathbb{R}^{n \times 1}$ such that component d_i equals the weighted degree of vertex i and we rewrite $\mathcal{A}(V, V)$ as vol_G , the volume of graph G . Thus,

$$\begin{aligned} Q(P_k) &\propto \sum_{c=1}^k [\text{vol}_G \mathbf{x}_c^T W \mathbf{x}_c - \mathbf{x}_c^T \mathbf{d} \mathbf{d}^T \mathbf{x}_c] \\ &= \sum_{c=1}^k [\text{vol}_G \mathbf{x}_c^T W \mathbf{x}_c - \mathbf{x}_c^T \mathcal{D} \mathbf{x}_c] \end{aligned}$$

where $\mathcal{D} = \mathbf{d} \mathbf{d}^T$. Since for any matrix A and assignment matrix X , $\text{tr}(X^T A X) = \sum_{c=1}^k [\mathbf{x}_c^T A \mathbf{x}_c]$, we can further reduce Q as follows:

$$\begin{aligned} Q(P_k) &\propto \text{vol}_G \text{tr}(X^T W X) - \text{tr}(X^T \mathcal{D} X) \\ &= \text{tr}(X^T (\text{vol}_G W - \mathcal{D}) X) \\ &= \text{tr}(X^T (\mathcal{W} - \mathcal{D}) X) \end{aligned}$$

where $\mathcal{W} = \text{vol}_G W$. The problem of maximizing Q can then be expressed as:

$$(2.3) \quad \max_X \{ \text{tr}(X^T (\mathcal{W} - \mathcal{D}) X) \} \text{ s.t. } X^T X = M$$

where $M \in \mathbb{R}^{k \times k}$ is a diagonal matrix with diagonal entry $[M]_{ii} = |V_i|$, where $|V_i|$ is the number of nodes in cluster V_i .

2.2 Spectral Relaxation Finding an assignment matrix X which maximizes (2.3) is NP-complete. To address this we can attempt to derive a good approximation by relaxing the discreteness constraints that the $X_{ij} \in \{0, 1\}$, so that instead the $X_{ij} \in \mathbb{R}^1$. This transforms the discrete optimization problem into one that is continuous. To find the optimal relaxed X , take the derivative of the following expression with respect to X :

$$(2.4) \quad \text{tr}(X^T (\mathcal{W} - \mathcal{D}) X) + (X^T X - M) \Lambda$$

where $\Lambda \in \mathbb{R}^{k \times k}$ is the diagonal matrix of Lagrangian multipliers. Setting this equal to 0 and rearranging terms we have:

$$(2.5) \quad L_Q X = X \Lambda$$

where $L_Q = \mathcal{D} - \mathcal{W}$ and we refer to this diagonal matrix as the “Q-Laplacian”. Aside from noting its similarity in form to the standard Laplacian, we observe that this is a standard matrix eigenvalue problem which can be solved using standard eigendecomposition methods. Furthermore, had we normalized the original matrix W so that all rows sum to one and had we also added back in the normalization constant that we took out from equation (2.2) then we would have the following eigenvalue equation:

$$(2.6) \quad \mathcal{L}_Q X = X \Lambda$$

where $\mathcal{L}_Q = \frac{1}{n^2} E - \frac{1}{n} W'$ where E is a matrix of all ones and W' is the matrix W normalized so all rows sum to one. The first term of this equation can be seen as a damping term that ensures that there are edges between all of the nodes of very small weight and the second term is the original weight matrix after scaling and normalization. As $n \rightarrow \infty$, the first term will approach 0 much faster than the second term, and hence will play a negligible role in determining the eigenspace of the matrix.

Thus, for even moderately large values of n , it seems reasonable that W' will provide a close approximation to \mathcal{L}_Q , which we refer to as the “normalized Q Laplacian.”¹ In this paper, we will adopt the simplest method

¹The minus sign and the constant $\frac{1}{n}$ do not impact the resulting eigenspace.

for normalizing a matrix so its rows sum to one, namely, to left multiply the matrix W by D^{-1} . The advantage of using $W' = D^{-1}W$ as an approximation to \mathcal{L}_Q is that it is easy to compute, it is well studied, especially in relation to Markov chains where it is known as the transition matrix, and it retains sparsity so we can use fast methods for eigendecomposing a sparse matrix.

The final step in this framework is to iterate over different values of k , to search for the best clusterings (highest $Q(P_k)$ scores). For each k , we try to find the optimal partitioning, i.e., a “hard-assignment” of the nodes to k clusters, based on clustering the rows of the matrix X .

3 Two New Graph Clustering Algorithms

In this section, we propose two new algorithms for clustering graphs that build on insights developed in the previous section.

3.1 Computing the embedding Assume that we are seeking up to a maximum of K clusters and that we have a weight matrix $W \in \mathbb{R}^{n \times n}$. Both of our proposed algorithms below begin by computing the top $K - 1$ eigenvectors (ignoring the trivial all-ones eigenvector) corresponding to Equation 2.6. Specifically:

1. Compute the transition matrix $\mathcal{M} = D^{-1}W$
2. Compute the eigenvector matrix $U_K = [\mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_{K-1}]$ from \mathcal{M} using a sparse eigenvector decomposition method such as a variant of the Lanczos method or subspace iteration.

In the experimental results in this paper we compute the $K - 1$ eigenvectors using the Implicitly Restarted Lanczos Method (IRLM) [2]. If one makes the conservative assumption that there are $O(K)$ extra Lanczos steps, then the IRLM has worst-case time complexity of $O(mKh + nK^2h + K^3h)$ where m is the number of edges in the graph, and h is the number of iterations required until convergence. For sparse graphs, where $m \sim n$, and where $K \ll n$, we found the IRLM to be extremely fast, taking near linear time with respect to the number of nodes n .

In the algorithms below, we initialized k-means so that the starting centroids were chosen to be as close to orthogonal as possible. Initializing k-means in this way does not change the time-complexity but can significantly help to improve the quality of the clusterings, as discussed in [11], while at the same reducing the need for multiple random restarts. In addition, both algorithms below can be run for any range of k values between a lower bound k_{\min} and an upper bound k_{\max} . When not stated otherwise, we will

assume in what follows that we have $k_{\min} = 1$ and $k_{\max} = K$. In the case where $k = 1, Q = 0$ and the cluster is just all the vertices in the graph.

3.2 Algorithm Spectral-1 This algorithm takes as input an eigenvector matrix U_K , and consists of the following steps:

1. For each value of k , $2 \leq k \leq K$:
 - (a) Form the matrix U_k from the first $k - 1$ columns of U_K .
 - (b) Scale the rows of U_k using the l^2 -norm so they all have unit length
 - (c) Cluster the row vectors of U_k using k-means or any other fast vector-based clustering algorithm. For $k = 1$, the cluster is just the graph itself.
2. Pick the k and the corresponding partition that maximizes $Q(P_k)$.

This algorithm is similar in spirit to the one developed in [11]. Both algorithms embed the input graph into a Euclidean space by eigendecomposing a suitable matrix and then cluster the embedding using a geometric clustering algorithm. We experimentally validated the claim made in [11] that row-normalizing the matrix of eigenvectors, so that the row vectors are projected onto the unit hypersphere, gives much higher quality results. The Spectral-1 algorithm is different in three key respects to this earlier work (in addition to the minor ontological point that our framework is designed to cluster graphs while theirs is designed to cluster real-valued points):

1. Whereas in [11] the matrix that is eigendecomposed, $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, was implicitly chosen to optimize the Normalized Cut, our algorithm is explicitly designed to optimize the modularity Q .
2. Our algorithm has a natural method for model-selection, the Q measure, which is the same objective function our embedding is based on. Since Normalized Cut is biased by the size of k , it can not be used for choosing the best k .
3. Our algorithm does not require an extra step of model selection to ensure: a) the edge weights are scaled correctly and b) the graph is sparsified. If links are not sparsified in the algorithm in [11], the time complexity is $O(n^3)$.

Table 1: Sample clusters found for the WordNet data.

Hard Science	Qualities	Metabolism	Soft Science	Systems
taxonomy	attribute	regulation	social relation	organism
science	drive	reproduction	profession	body
mathematics	folly	Krebs cycle	social science	hierarchy
pure mathematics	judgment	hypostasis	law	digestive system
applied mathematics	estimate	nutrition	politics	infrastructure
statistics	trait	growth	medicine	network
information theory	personality	anabolism	theology	system
computer science	character	bodily process	opinion	water system
information science	nature	catabolism	explanation	live body
information theory	thoughtfulness	gastrulation	anthropology	sensory system

3.3 Algorithm Spectral-2 The second algorithm we propose is a faster version of the algorithm in the previous section (Spectral-1). It uses a greedy strategy of recursive bisection to search over different values of k . Because of this strategy it need not find as high quality a clustering (as high a Q value) as the other approach, but it will be faster since in going from k clusters to $k+1$ only a portion of the data needs to be clustered rather than all of the data. The algorithm again takes as input the eigenvector matrix U_K as before and consists of the following steps:

1. Initialize k , the current number of clusters, to k_{min} . Initialize P , the best clustering assignment seen so far, to the clustering produced by running k-means with k set to k_{min} clusters. If $k_{min} = 1$, then simply initialize P to be one cluster containing all nodes in the graph.
2. Repeat until $k > K$ or no more splits are possible:
 - (a) Set $P_{new} = P$
 - (b) For each cluster V_c in P :
 - i. If not already formed, form the matrix U_k from the first $k - 1$ columns of U_K and scale the rows using the l^2 -norm so they all have unit length
 - ii. Form the matrix $U_{k,c}$ from U_k by keeping only rows corresponding to nodes in V_c
 - iii. Run k-means with 2 clusters on $U_{k,c}$ to get two new sub-clusters, $V_{c,1}$ and $V_{c,2}$.
 - iv. Form a new partition, P' by setting $P' = P$ and replacing the corresponding V_c with $V_{c,1}$ and $V_{c,2}$
 - v. If $Q(P') > Q(P)$, accept the split by replacing the corresponding V_c in P_{new} with $V_{c,1}$ and $V_{c,2}$, otherwise reject it and leave P_{new} unchanged.

vi. Assign k to be the (possibly new) number of clusters in P_{new}

(c) Set $P = P_{new}$

The idea behind this algorithm is to start with k_{min} clusters and instead of rerunning k-means on the entire graph for subsequent values of k as we did in the previous algorithm, we instead try recursively splitting each cluster into two child clusters if the split produces a higher value of Q . By continuing this procedure until no more splits are possible or until K clusters have been found, we end up with a clustering with the highest value of Q encountered along the way. The particular algorithm above is order-sensitive in the sense that cycling through the clusters in a different order could produce different results—however, we have not noticed any particular sensitivity to order in the data sets described later in the paper. Unlike many other recursive bisection methods, model selection here is natural and straightforward. We choose to accept a split if the split results in a higher value of Q . Of course, the drawback with this algorithm is that we tradeoff speed with accuracy. The algorithm uses a greedy strategy, where a split can never be revoked and so one bad choice negatively affects all other choices further down the same branch. Thus, the quality of the results are not necessarily as good as with Spectral-1, although they are still generally competitive as we will see below. There is a subtlety in this algorithm in that, if left unchecked, each cluster for which a split attempt was made but failed would be retried again the next time through the loop. For this reason, we only allow a single attempt to split a given cluster.

3.4 Computational Complexity For the Spectral-1 algorithm, we run k-means K times, where in each case the dimensionality is $d = k - 1$. Standard k-means with a Euclidean distance metric has time complexity

$O(ndke)$ where n is the number of data points, d is the dimensionality of each point, and e is the number of iterations required for k-means to converge. Elkan [5] proposed a much faster version of k-means. It produces exactly the same results as standard k-means but uses various geometric inequalities to significantly reduce the number of distance computations required. Elkan found that with his proposed algorithm, the overall time complexity is roughly $O(nke)$ where e is the number of iterations. We use this algorithm for our implementation of k-means in both Spectral-1 and Spectral-2.

The resulting complexity for clustering in Spectral-1, using Elkan’s fast k-means algorithm, is roughly $O(nK^2e)$. For Spectral-2 the computational complexity is not as easy to estimate. In the worst case (completely imbalanced clusters where the largest cluster is split at each iteration) it will have the same complexity as Spectral-1. However, in practice we have found that it is considerably faster than Spectral-1 and will show experimental results later in the paper that illustrate this.

In addition, for both algorithms there is the additional complexity of $O(mKh + nK^2h + K^3h)$ for computing the matrix of eigenvectors U_K , using the IRLM. Thus, Spectral-1 and Spectral-2 have an overall worst-case time complexity of $O(mKh + nK^2h + K^3h + nK^2e)$. Thus, for sparse graphs, where $m \sim n$, the algorithms will scale roughly linearly as a function of the number of nodes n . This is in contrast to Newman’s algorithm which has complexity $O(n^2)$ even for sparse graphs, and thus does not scale up as well to large values of n .

4 Experimental Results

4.1 Clustering words from WordNet We first illustrate how different choices for the graph embedding can affect the quality of clustering. We use a relatively small unweighted graph extracted from the WordNet database of word meanings and semantic relationships [8]. The reason we chose this data set was because one can immediately judge the quality of the clusters since intuitively clusters should contain words that share common semantic features which are recognizable. We created an unweighted undirected graph where nodes represent words and an edge exists between two nodes if any of the following semantic relationships exist between them: synonymy, antonymy, hypernymy, hyponymy, meronymy, troponymy, causality, and entailment. The entire graph contains 82670 nodes. We extracted a subgraph comprised of all nodes whose shortest path distance away from the word “Science” is no more than 3. We also removed all nodes with degree 1 so that the graph layout would not be too cluttered. Adding this

constraint had little effect on the quality of the clusters.

The resulting subgraph contains 230 nodes and 389 edges. Figure 2 shows the best clustering found by the Spectral-1 algorithm.² For this clustering there were 12 clusters and $Q = 0.696$. Table 1 shows ten representative words from five random clusters.

Figure 3 shows how the modularity Q varies with k when each of the following three types of matrices is used in step 1 of the first algorithm and the eigenvectors are computed exactly: standard Q Laplacian L_Q , normalized Q Laplacian \mathcal{L}_Q , and the transition matrix $D^{-1}W$. We can see that using the transition matrix in step 1 provides a very good approximation to the normalized Q matrix. We can also see that the standard Q Laplacian slightly underperforms both of these matrices. This agrees with observations by other authors that the normalized Laplacian gives better results than the standard Laplacian (e.g., [12],[14]).

4.2 Clustering American college football teams

Our next example demonstrates the ability of both of our algorithms to identify known clusters. The unweighted network was drawn from the schedule of games played between 115 NCAA Division I-A American college football teams in the year 2000. Each node in this network represents a college football team and each edge represents the fact that two teams played together. Because the true conference to which each team belongs is known a-priori, and because inter-conference games are played more often than intra-conference games, the groups of teams that form conferences correspond to natural clusters that should be identifiable in the graph. Figure 4 shows that this is indeed the case where the Spectral-1 algorithm identified the correct number of clusters and, furthermore, each team assignment to a cluster made by the algorithm was correct.³ Our second algorithm, Spectral-2, did almost as well making very few mistakes. The mistakes were:

1. North Texas was put into the Big 12 conference instead of Big West.
2. Arkansas State was put in Western Athletic instead of Big West.
3. EastCarolina was placed in Atlantic Coast instead of Conference USA
4. BigTen was split into two equal sized clusters: {Michigan, Ohio State, Wisconsin, Iowa, Illinois,

²Graphs are shown using the Fruchterman-Reingold layout.

³For the group of eight teams that do not belong to any conference, each was assigned to one of the conferences, e.g., Notre Dame, Navy, and Florida (Miami) to the Big East Conference.

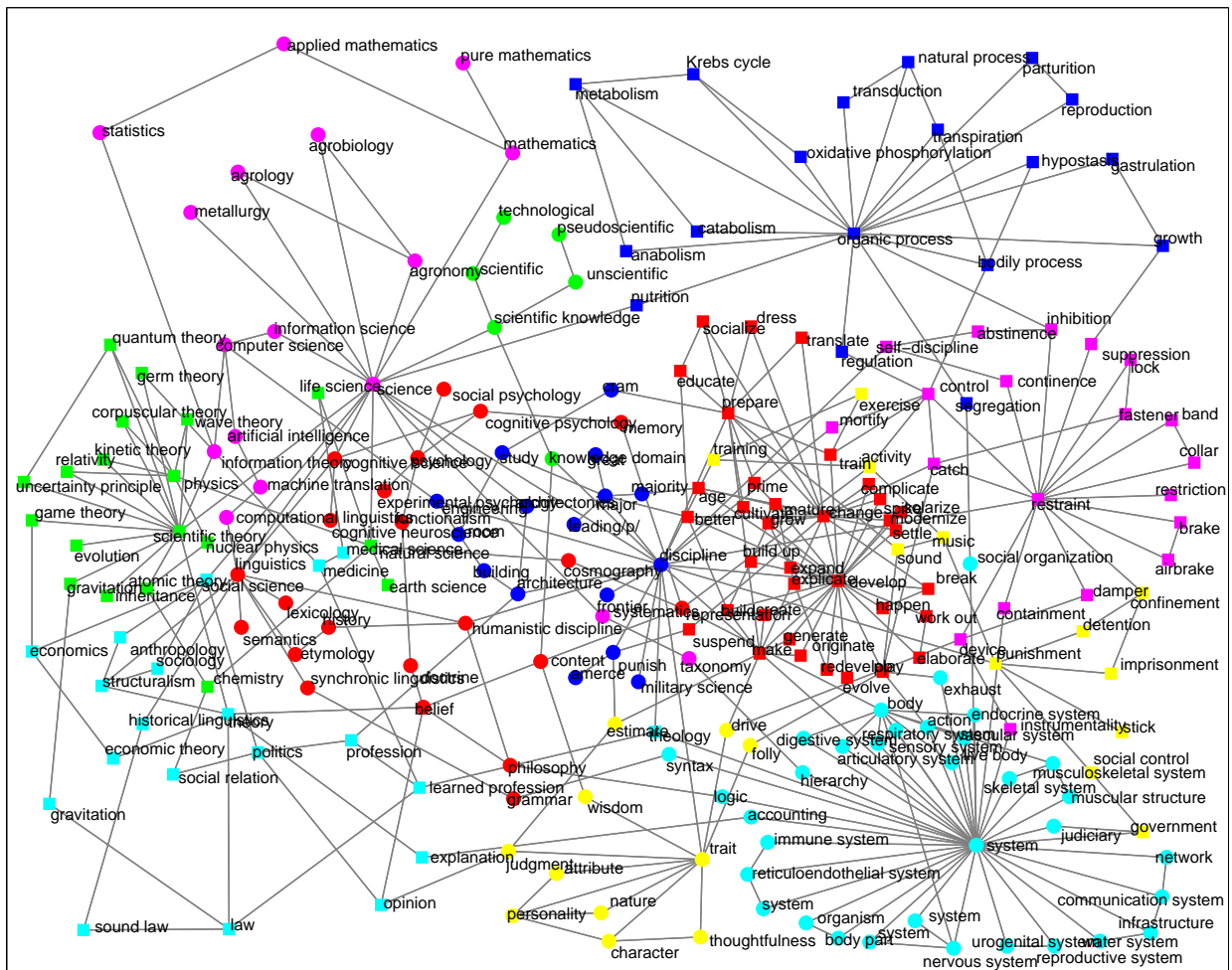


Figure 2: Clusters for WordNet data, $k = 12$ (best viewed in color).

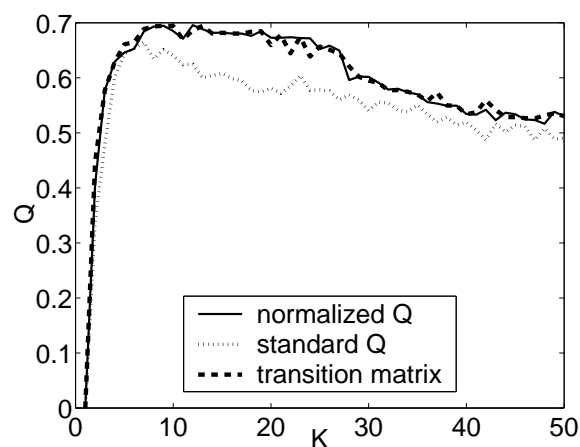


Figure 3: Q versus k for the WordNet data.

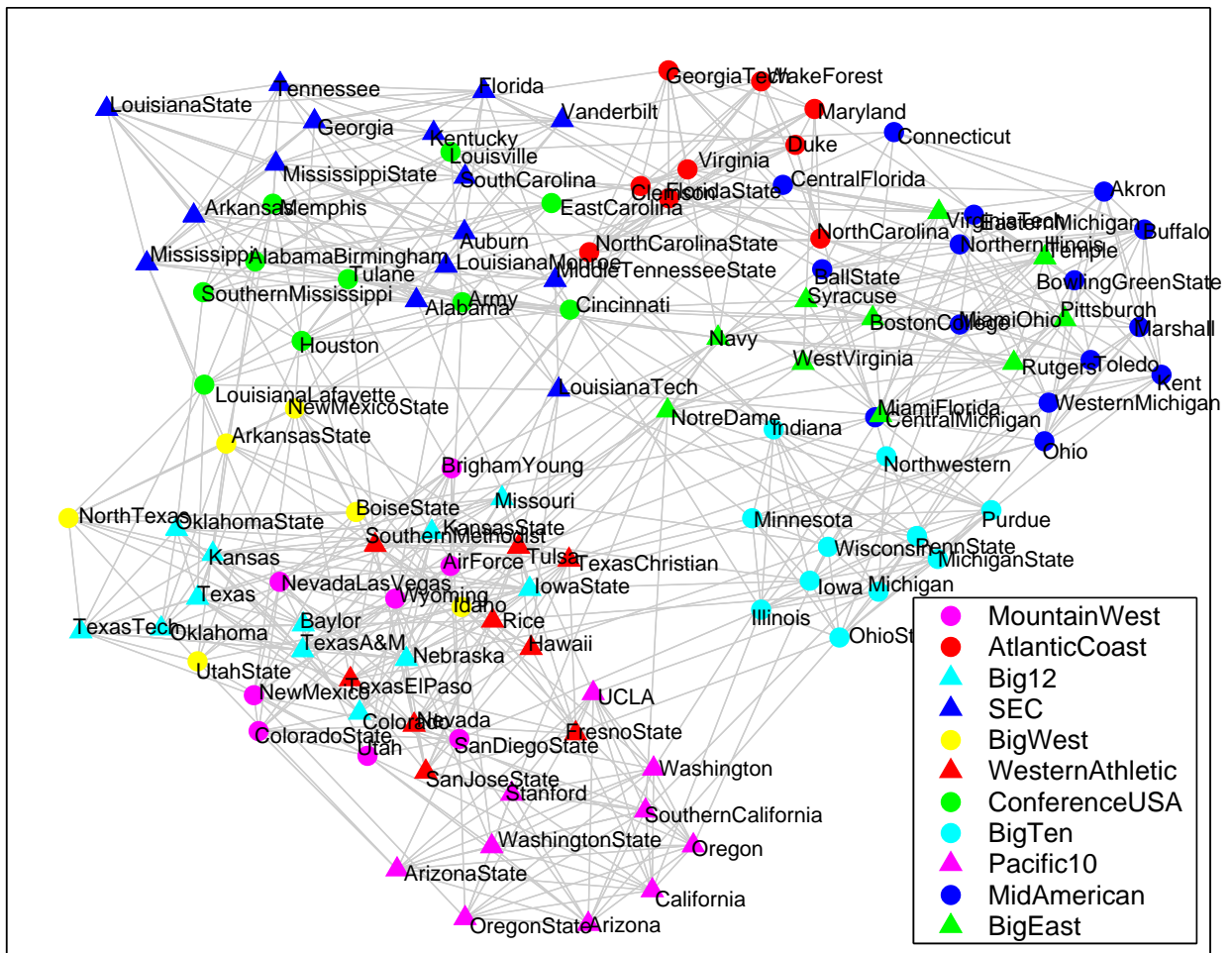


Figure 4: Clusters for NCAA Division I-A football teams, $k = 11$ (best viewed in color).

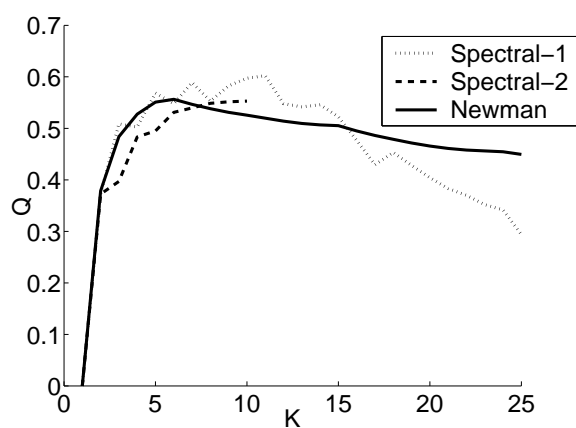


Figure 5: Q versus k for NCAA Division I-A football teams.

Michigan State} and {Northwestern, Purdue, Minnesota, Penn State, Indiana}

5. Mountain West, Pacific 10 and Big West were merged into one cluster

To summarize there were three individual mistakes where a single team was misplaced in the wrong conference, as well as one incorrect split and two splits that should have happened but did not.

Newman [10] proposed a hierarchical agglomerative clustering method for finding graph clusterings based on Q , with a worst-case time complexity of $O(n^2)$. At each step of the algorithm the two clusters are merged that result in the largest increase in Q , providing local iterative improvements to the overall Q score. The number of clusters k can be obtained from the resulting dendrograms by selecting the level of the tree for which $Q(\mathcal{P}_k)$ is highest. Figure 5 shows how the modularity Q varied with k for both of our algorithms as well as Newman’s original greedy algorithm which uses the modularity Q as a distance measure to do agglomerative clustering. The peak for Spectral-1 was at $k=11$, $Q=0.602$, which corresponds precisely with the actual number of conferences in the NCAA Division I-A American college football league. We can see that Newman’s algorithm underperforms this algorithm on this data set. The best clustering found by Newman’s algorithm was for $k=6$, $Q=0.556$. As Newman [10] points out and we see in this example, his algorithm can miss the best clustering since it makes decisions purely at a local level whereas Q is inherently a global measure. The same is true for our faster, greedy algorithm. The best clustering found by Spectral-2 was $k=10$, $Q=0.553$. This is competitive with Newman’s algorithm but as we will see in the timing experiments this algorithm runs significantly faster than Newman’s algorithm.

4.3 Clustering authors publishing in NIPS For our final experiment we extracted a weighted coauthorship network from volumes 0-12 of the NIPS conference papers. The raw data contains 2037 authors and 1740 papers from which we created a coauthorship graph where nodes represent authors and edges represent coauthorship between the given pair of authors. We weighted each edge using the following weighting scheme: $w_{ij} = \sum_k \frac{1}{n_k - 1}$ where n_k is the number of coauthors for paper k , and w_{ij} represents the weight assigned to the edge between nodes i and j for which there was a coauthor relation. Building a coauthorship network from this data yields many disconnected components so we used only the dominant component which has 1061 nodes (authors) and 4160 edges (coauthorship pairs).

We ran both of our algorithms with $K=100$. Figure 6 shows the best clustering found by Spectral-1 where $k=31$ and $Q=0.874$. In this figure, the original coauthorship graph was shrunk so that now each node represents a cluster and the size of the node roughly indicates the size of each cluster. PageRank with Priors was used to label each cluster with the three authors whose importance was highest relative to the cluster to which they belong [13]. The resulting clusters clearly reflect various subcommunities of NIPS authors based on the first 12 years of the conference. Figure 7 shows how the modularity Q varies with k . We found that on this data set Newman’s algorithm marginally outperformed our Spectral-1 algorithm, although both algorithms were very close in terms of the optimal value of Q found: $Q=0.874$ and $k=31$ for Spectral-1 vs. $Q=0.876$ and $k=33$ for Newman’s algorithm. Spectral-2 did not do as well although it still gave competitive results finding a clustering for $k=44$ and $Q=0.861$.

This brings up an important point about the difference between Newman’s algorithm and our faster, greedy Spectral-2 algorithm. Newman’s algorithm starts with n clusters, one for each node in the graph, and continues to merge clusters one at a time whereas our faster algorithm starts with one cluster, all nodes in the graph being assigned to it, and continues to split clusters one at a time. Thus, as we saw in the previous example with the college football data set, Newman’s algorithm made some mistakes early on in the merging process which caused Q to reach a maximum only after k was already much smaller than the correct solution for $k = 11$. Our faster algorithm, Spectral-2, also made mistakes early on, in the splitting process, causing it to overshoot the more optimal values of Q found by the other two algorithms in the vicinity of $32 \leq k \leq 33$ and instead find a maximum value of Q for much larger k , in this case $k = 44$. Although both algorithms are able to find clusterings that are quite competitive with Spectral-1, they both potentially suffer from the problem of overshooting, although each in opposite directions, because of the potential limitations of the greedy strategy. Nevertheless, this example also highlights the fact that there are many graphs for which a greedy strategy can perform quite well (as also documented by Newman [10]).

4.4 Timing Experiments In this section, we present timing results for each of the experiments conducted in the experiments just described. In addition, we reran Spectral-1 and Spectral-2 for $K = 25$ on the “Science” network and for $K = 50$ on the NIPS coauthorship network to highlight how the choice of K affects performance. All algorithms were implemented in

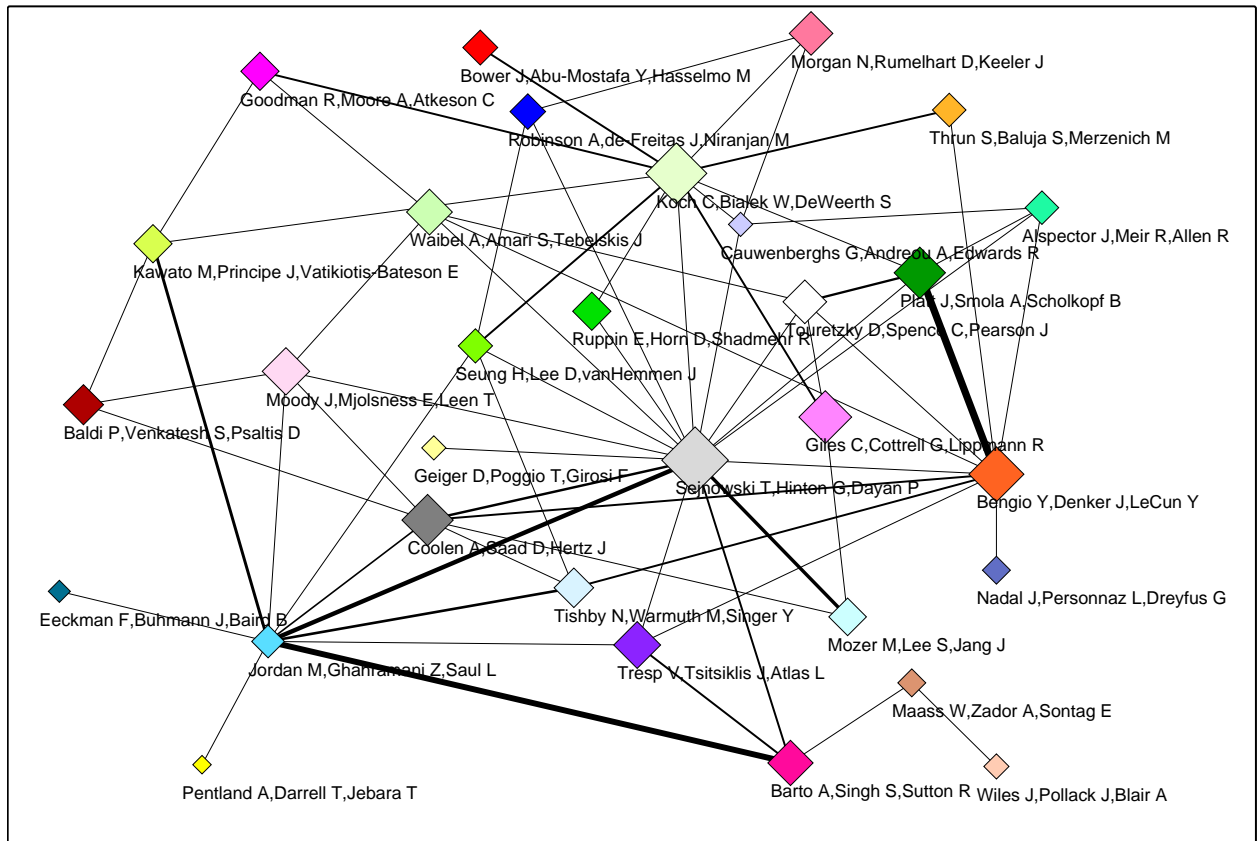


Figure 6: Clustering for NIPS co-authorship data, $k = 31$ (best viewed in color).

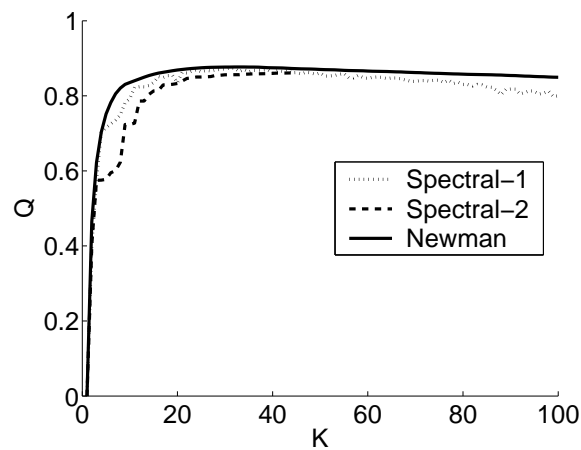


Figure 7: Q versus k for NIPS coauthorship data.

Matlab and run on a 1.2 GHz Pentium II laptop. We used the sparse eigendecomposition routine *eigs* in Matlab to compute the eigenvectors using the IRLM.

Table 2: Timing results for separate components of Spectral-1 in seconds.

Name	n	K	<i>eigs</i>	Clustering
Football	115	25	0.16	2.29
Science	230	25	0.38	3.43
Science	230	50	0.67	12.36
NIPS	1061	50	6.53	40.41
NIPS	1061	100	15.49	292.15

Table 2 shows the timings for each of the key components of the Spectral-1 algorithm. Clustering takes most of the time, especially as K and n increase.

Table 3: Timing results for separate components of Spectral-2 in seconds.

Name	n	K	<i>eigs</i>	Clustering
Football	115	25	0.16	0.15
Science	230	25	0.38	0.23
Science	230	50	0.67	0.30
NIPS	1061	50	6.53	1.33
NIPS	1061	100	15.49	2.01

Table 3 shows the timings for each of the key components of the Spectral-2 algorithm. For Spectral-2, as K increases, running *eigs* becomes an increasingly large performance bottleneck. The time taken for clustering is low since we never have to re-run the k-means algorithm on the entire graph. This includes the time taken to determine whether or not to accept a split which involves computing Q which takes little time since we don't recompute Q from scratch but instead update Q based on the edges that have been reassigned.

Table 4: Overall timing results in seconds.

Name	n	K	Spec-2	Spec-1	Newman
Football	115	25	0.41	3.11	7.74
Science	230	25	0.77	4.23	8.38
Science	230	50	1.06	13.96	8.38
NIPS	1061	50	10.57	51.57	387.15
NIPS	1061	100	22.14	321.14	387.15

Table 4 shows the overall times (in seconds) for each of the five experiments.⁴ Perhaps most interesting is to observe how the interaction between K and n affects the results. Spectral-1 is generally faster than Newman's algorithm although for large enough values of K and small enough values of n , Newman's algorithm can be faster. Spectral-2 is generally at least an order of magnitude faster than the other two algorithms although as K gets larger the difference in speed is less pronounced.

5 Discussion

The idea of reducing a combinatorial graph partitioning problem to a geometric vector space partitioning problem using spectral techniques is by no means new. Some of the earliest breakthroughs can be attributed to Hall [7] and Fiedler [6]. Alpert and Yao [1] showed that when the full eigenspace is used, certain graph partitioning problems exactly reduce to vector partitioning ones. More recently, Brand and Huang [3] presented theoretical results precisely characterizing how compacting the eigenbasis is able to magnify structure in the data. Furthermore, Chung [4] and others have laid much of the foundational work in spectral graph theory, on which a large part of the subsequent theoretical analysis of spectral clustering methods is based.

The key idea in this paper is to reverse engineer Newman's Q function into a spectral framework in which any input graph can be optimally embedded into Euclidean space. Once the input graph is represented in a Euclidean space, we can then use fast geometric clustering algorithms such as k-means to identify the clusters. Any algorithmic framework developed in this way faces a large search problem since both k (the number of clusters) and the dimensionality of the embedding which maximize Q need to be explored. Both algorithms for fixed k choose the dimensionality of the embedding to be $k-1$ (e.g. for $k=2$, we just use the top eigenvector). The assumption here is that while it may be possible, in some cases, to use fewer dimensions and still find a good clustering for fixed k , while also making the algorithm even faster, it is better to be conservative. Experiments have shown that the higher the dimensionality (the more eigenvectors), the better the clusterings produced, although we did not find that having the dimensionality of the embedding exceed $k-1$ helped in any way.

Both algorithms empirically track the performance of Newman's algorithm quite closely. The slower, more

⁴Note that the times for Spectral-1 and Spectral-2 are slightly larger than the sums of the corresponding times in Tables 2 and 3 due to additional overhead in the algorithms.

accurate algorithm (Spectral-1) can produce higher-quality clusterings than Newman's because of the non-greedy search heuristic. The Spectral-2 algorithm could be viewed as a top-down divisive search alternative to Newman's bottom-up agglomerative search, in a general hierarchical clustering context, with attendant advantages and disadvantages to each in terms of the greedy search strategy, as well as having significant differences in their computational characteristics.

Other search heuristics approaches are also possible and may lead to different trade-offs between cluster quality and computation time. For example, combining both of our algorithms into a hybrid algorithm may yield a fruitful trade-off between speed and cluster quality. For graphs where the number of clusters to search over is large, Newman's hierarchical clustering approach may be the preferred method given that it operates directly on the graph without any need for embedding the graph into a Euclidean vector space. Our algorithms, in contrast, use sparse eigenvector techniques which scale quadratically with the number of clusters to search over. However, when the number of clusters to search over is small, and n the number of nodes increases, the $O(n^2)$ complexity of hierarchical clustering can quickly become intractable. In contrast, the two algorithms we propose here will scale relatively well to large graphs.

6 Conclusions

In this paper we have shown how the recently proposed Q function can be used to find high quality graph clusterings. We give a precise analytical expression which when maximized returns a discrete assignment matrix X that represents the optimal partitioning of a graph according to the Q function for fixed k . Because maximizing this expression is NP-Complete, we show how the discrete maximization can be approximated as a continuous one that is easily solvable by performing eigenvector decomposition on a matrix L_Q , which we call the Q -Laplacian. We present two algorithms which attempt to search over different values of k to find the best value of k and the accompanying best clustering. The first algorithm we present searches independently for a best clustering for each value of k . Unlike Newman's algorithm, which optimizes Q by local iterative improvement, this algorithm seeks a direct global maximum of Q . The second algorithm we present is similar to Newman's algorithm in that it uses a local greedy search heuristic; however, it is based on a top-down strategy of splitting clusters that lead to higher values of Q and is thus much faster than the other two algorithms for $K \ll n$. Empirical results suggest that both methods provide high quality graph clusterings on a variety of graphs that exhibit community structure, and both methods scale lin-

early in the number of edges, allowing for applications to large sparse graphs.

Acknowledgements The data used were generously made available by the Cognitive Science Laboratory at Princeton University (WordNet), Mark Newman (College Football) and Sam Roweis (NIPS).

References

- [1] C. Alpert and S. Yao, Spectral partitioning: the more eigenvectors the better. In *Proceedings of 32nd ACM/IEEE Design Automation Conference*, 1995, pp. 195-200.
- [2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Vorst, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.
- [3] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. *9th International Conference on Artificial Intelligence and Statistics*, 2002.
- [4] F. Chung. Spectral graph theory. Number 92 in *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.
- [5] C. Elkan. Using the triangle inequality to accelerate k-Means. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 147-153.
- [6] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23 (1973), pp. 298-305.
- [7] K. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 11(3)(1970), pp. 219-229.
- [8] G. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3 (1990), pp. 235-312.
- [9] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 026113 (2004).
- [10] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 066133 (2004).
- [11] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2002, pp. 849-856.
- [12] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (2000), pp. 888-905.
- [13] S. White and P. Smyth. Algorithms for discovering relative importance in graphs. In *Proceedings of Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 266-275.
- [14] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings OF IEEE International Conference on Computer Vision*, 1999, pp. 975-982.

Mining Behavior Graphs for “Backtrace” of Noncrashing Bugs*

Chao Liu[†]

Xifeng Yan^{†‡}

Hwanjo Yu[†]

Jiawei Han[†]

Philip S. Yu[‡]

[†]Department of Computer Science
University of Illinois at Urbana-Champaign
{chaoliu, xyan, hwanjoyu, hanj}@cs.uiuc.edu

[‡]IBM T. J. Watson Research Center
psyu@us.ibm.com

Abstract

Analyzing the executions of a buggy software program is essentially a data mining process. Although many interesting methods have been developed to trace crashing bugs (such as memory violation and core dumps), it is still difficult to analyze noncrashing bugs (such as logical errors). In this paper, we develop a novel method to classify the structured traces of program executions using *software behavior graphs*. By analyzing the correct and incorrect executions, we have made good progress at the isolation of program regions that may lead to the faulty executions. The classification framework is built on an integration of closed graph mining and SVM classification. More interestingly, suspicious regions are identified through the capture of the classification accuracy change, which is measured incrementally during program execution. Our performance study and case-based experiments show that our approach is both effective and efficient.

Keywords. data mining, software reliability, debugging, noncrashing bugs, closed pattern, SVM.

1 Introduction

Software reliability is a top concern in modern industry. Software bugs cost the U.S. economy an estimated 59.5 billion dollars annually, or approximately 0.6% of the GDP, according to a report from the National Institute of Standards and Technology (NIST). As software becomes increasingly bulky in size, sophisticated in complexity, and originated by integration of multiple components, it is an increasingly challenging task to ensure software robustness and reliability.

As well-known in software engineering, better understanding of program behavior can be invaluable to build reliable systems. Extensive research has been conducted on software reliability, ranging from static source code checking [4, 6] to dynamic program verification [5, 18]; and from low-level program execution profiling [9, 7] to high-level behavior analysis [5, 20]. Related achievements have motivated practices in abnormality detection [9, 25] and computer-aided debugging [26, 18, 2].

From a knowledge discovery point of view, the analysis of executions of a buggy program is essentially a data mining process—tracing the data generated during program executions may disclose important patterns and outliers that may help the discovery of software bugs. Thus, we believe that recently developed data mining technology can improve software reliability. In this paper, we investigate the application of data mining methods to program bug analysis. By treating program executions as software behavior graphs, a new method is developed to integrate closed graph mining and SVM classification for the isolation of suspicious regions of noncrashing bugs.

In program analysis, software bugs can be classified into two categories: *crashing bugs* and *noncrashing bugs*. The former refers to the bugs that crash the program execution, such as core dumps or segmentation faults. One can trace back the function call stack from the crashing point for debugging. The latter refers to the bugs that do not incur crashes, such as logic bugs, which are difficult to locate since no crashing point, hence no backtrace, is available.

In this study, we develop a novel classification method for backtracing noncrashing bugs. Our methodology can be outlined as follows.

First, we summarize each execution of a program as a concise but informative *behavior graph*. Fig. 1 shows an example of behavior graphs, which is excerpted from

*This work was supported in part by the U.S. National Science Foundation NSF ITR-03-25603, an IBM Faculty Award, and an IBM Summer Internship. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

two different runs of *ccrypt-1.2*, a utility program for encrypting and decrypting files. Behavior graphs summarize program execution at function level with each node for one function. Solid arrows represent the calling relationship and dashed ones for transitions. As one can see, behavior graphs only preserve function-level sequential information and are thus compact. Despite of its succinctness, it does manifest the behavior abnormalities corresponding to incorrect runs. For example, *ccrypt-1.2* has one bug that is triggered when a user corresponds to the prompt for overwriting an existing file with EOF, rather than as expected ‘Y(es)’ or ‘N(o)’. As shown in Fig. 1, the correct and incorrect runs diverge at the transition edges emitted from function `file_exists`, which is a strong indicator for classification.

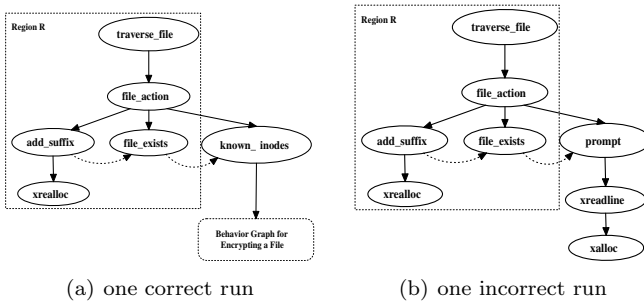


Figure 1: Software Behavior Graphs

Second, based on the behavior graph representation of program runs, the classification of program runs can be formulated as a graph classification problem: *Given a set of behavior graphs that are labelled either positive or negative, can we train a classifier to identify unknown behavior graphs?*

In our study, we use support vector machine (SVM) [13] with linear kernel to do classification. Inspired by the better scalability of closed subgraphs over frequent ones and their stronger expressibility over raw edges as features, we explore the benefits by incorporating closed subgraphs as classification features, which, as shown, has higher classification accuracy as well as better scalability. Interestingly, we also explore the relationship between closed and frequent graph-based SVM classifiers, which sheds light on the inherent relationship between these two related methods.

Third, for effective classification, we develop a novel method to uncover the “backtrace” for noncrashing bugs. Recall that backtrace usually refers to the function call stack at the time a program crashes (i.e., core dump or segmentation fault), based on which debugging can be easy to start. Unfortunately, for noncrashing bugs, their backtraces are no longer available. To help locate this kind of bugs, we attempt to uncover a virtual

“backtrace” for noncrashing bugs, which is essentially a series of bug-relevant functions. We believe that *the functions, whose execution behavior promotes the classification accuracy of distinguishing incorrect runs from correct runs, are likely suspicious functions*. Taking Fig. 1 as an example, a classifier can be trained at the return of function `file_exist`, but its accuracy cannot be high because behavior graphs up to this point (i.e., the subgraph within region R) are almost identical for both incorrect and correct runs. However, if we train another classifier at the return of `file_action` (recall that `file_action` returns later than `file_exist`), the accuracy will be much higher since the training behavior graphs do include the traces that differentiate correct and incorrect runs.

In summary, we make the following contributions:

1. We investigate the application of recently developed data mining techniques to software robustness enhancement and show that data mining may help backtrace noncrashing bugs.
2. We have proposed *software behavior graph* as a concise but informative summary of program executions and developed an efficient mining algorithm, *CloseMine*, to uncover closed frequent subgraphs from behavior graphs, which has been proven effective at identifying failing runs. We further explored the connection between closed frequent graph based and frequent graph based SVM classifiers.
3. We developed a novel classification method to uncover the backtrace for noncrashing bugs, which, as shown through a detailed case study, can be effective in assistance to debugging.

The remainder of the paper is organized as follows. We first introduce preliminary concepts in Section 2. The classification framework is laid out in Section 3, within which both the mining algorithm design and the relationship between frequent graph-based and closed graph-based SVMs are examined. Section 4 describes how to uncover a backtrace based on behavior graphs. Experimental evaluations of classification quality and a case study are presented in Section 5. We discuss the related work in Section 6, and conclude our study in Section 7.

2 Preliminaries

A software execution can be summarized into a *behavior graph*, which consists of its *call graph* and *transition graph*. A *call graph* $G_c(\alpha)$ is a directed graph displaying the function calling relationship in a program run α . The vertex set $V(G_c(\alpha))$ includes all the functions involved in α . Edge (v_i, v_j) belongs to $E(G_c(\alpha))$ if and

only if function i calls function j in α . Transition graph $G_t(\alpha)$ is also a directed graph, exhibiting the transition relationships in α . Edge (v_i, v_j) belongs to $E(G_t(\alpha))$ if and only if function j is called immediately after function i returns. It is also required that functions i and j are called by the same caller function. The superposition of $G_c(\alpha)$ and $G_t(\alpha)$ forms the behavior graph $G(\alpha)$ of run α . Fig. 2 shows three behavior graphs, where solid and dashed arrows represent call relation and transition relation respectively.

We use behavior graphs to model program executions. Call graphs represent the task-subtask relationship, while transition graphs record the sequential order of the subtasks. Behavior graph only preserves the first-order transition and is thus succinct compared with the entire execution sequences. This is necessary for a scalable mining and classification method.

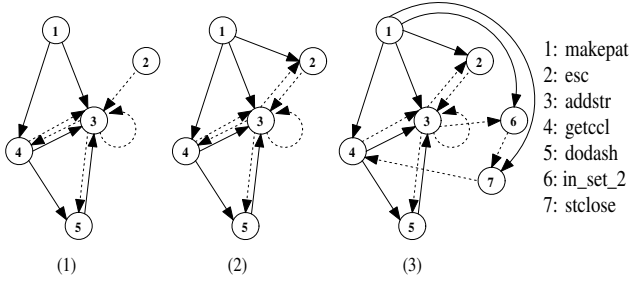


Figure 2: A Behavior Graph Dataset

EXAMPLE 1. Fig. 2 shows behavior graph segments derived from three different runs of our subject program “replace”, a regular expression matching and substitution utility software. Taking the run corresponding to the third graph for instance, `getccl`, `addstr`, `esc`, `in_set_2` and `stclose` are subtasks of function `makepat`. They work together to complete the task associated with `makepat`. As to transition, the dashed arrow from `getccl` to `addstr` means that `addstr` is called immediately after `getccl` returns. ■

If a behavior graph G is a *subgraph* of G' , then G' is a *supergraph* of G , written as $G \subseteq G'$. G' is the *proper supergraph* of G if $G \subset G'$. In the following discussion, we introduce the concepts of frequent and closed frequent graphs.

DEFINITION 1. (FREQUENT (CLOSED) GRAPH) *Given a graph dataset D , $\text{support}(g)$ (or $\text{frequency}(g)$) is the percentage (or number) of graphs in D , of which g is a subgraph. A graph is frequent if its support is no less than a minimum support threshold, min_sup . A frequent graph is closed if there exists no supergraph that has the same support.* ■

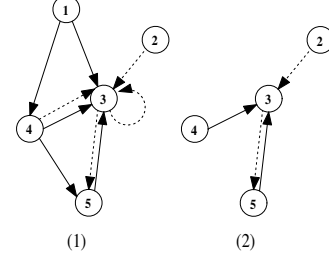


Figure 3: Frequent Graphs

EXAMPLE 2. Fig. 3 depicts two of frequent subgraphs in the dataset shown in Fig. 1, assuming that min_sup is equal to 66.6%. In Fig. 3, the first graph is closed while the second is not since the latter is a subgraph of the former and both of them have the same support. ■

3 The Classification Framework

Given a set of behavior graphs that are labelled either positive (for incorrect runs) or negative (for correct runs), we intend to train a classifier to identify new behavior graphs with unknown labels. The dynamics of classification accuracy will be analyzed to identify the backtrace of non-crashing bugs. In our study, we use support vector machine (SVM) [13] with linear kernel to do classification. The classification framework consists of three steps:

1. extract features from behavior graphs (training dataset),
2. learn an SVM classifier using these features, and
3. classify new behavior graphs.

In order to apply SVM in behavior graph classification, we represent graphs as vectors in a feature space. A naive representation is to treat *edges* as features and a graph as a *vector of edges*. The vector is $\{0, 1\}$ valued. If a graph has a specific edge, it has value “1” in the corresponding dimension, otherwise “0”. Using this representation, the dot product of two feature vectors is the number of common edges that two graphs have

$$(3.1) \quad \mathbf{x}_i \cdot \mathbf{x}_j = |E(g_i) \cap E(g_j)|,$$

where \mathbf{x}_i and \mathbf{x}_j are the vector representation of graphs g_i and g_j . For example, the dot product of the first two graphs in Fig. 2 is 10.

The similarity measure given in Eq. (3.1) is meaningful since it captures the relationship between two behavior graphs. As shown in our experiments, SVMs trained by the above measure work well in identifying some incorrect runs. Unfortunately, the hyperplane

learned in this way will be a linear combination of edges. Thus, it may not achieve good accuracy when a bug is characterized by multiple connected call and transition structures.

As shown in Fig. 2, the major portions of these graphs are very similar to each other although various incorrect runs may behave differently. In well-designed programs, functions usually exhibit strong modularity in source code and in dynamic executions. They are often grouped together to perform a specific task. Hence, the calls and transitions of these functions will be tightly related in the whole behavior graph. The buggy code may first disturb the local structure of a run and then have an effect on its global structure. This intuition inspires us to use recurrent local structures as features.

The classification process based on frequent graphs shares the same framework as the edge-based approach. Each frequent graph is treated as a separate feature in the feature vector. A behavior graph G is first transformed into a feature vector whose i -th dimension is instantiated to 1 if G contains the i -th frequent graph or 0 otherwise.

Unfortunately, due to the explosive number of frequent graphs in behavior graphs, it is often intractable to mine all of them. According to the Apriori property, all the subgraphs of a frequent graph must be frequent. A large frequent graph may generate a huge number of frequent subgraphs. When the number of frequent graphs increases, the performance at mining, training and classifying will drop dramatically. Thus, Deshpande et al. [3] propose a feature selection scheme to screen frequent graphs and Huan et al. [11] introduce the concept of coherent subgraphs to shrink the feature set. These approaches are successful in their problem domains. However, in our problem setting, they are not scalable. For example, in the “replace” program, if the minimum support is set at 40%, which is pretty high, there are still millions of frequent graphs. This renders the classification nearly impossible because it cannot even finish the feature extraction step.

As an alternative, closed frequent graph mining can complete in several orders of magnitude faster than frequent graph mining. Moreover, it commonly generates much less features for classification purpose. Taking the “replace” program as an example, among the millions of frequent graphs, only around 1,000 are closed frequent graphs. This makes the closed frequent graph-based classification more appealing than the frequent graph-based one. Furthermore, since closed frequent graphs is a lossless compression of frequent graphs, the classifier based on closed frequent graphs should have similar performance as the frequent graph based

classifier. Our empirical study suggests that the former is better.

3.1 Mining Closed Frequent Graphs. The first step in our classification framework is to mine closed frequent graphs from a set of behavior graphs and then use them as features. Behavior graphs can be transformed to *labelled undirected pseudographs*. A *pseudograph* is a non-simple graph in which both loops and parallel edges are permitted. A *labelled graph* has labels associated with its vertices and edges. Since behavior graphs have distinct labels for each vertex, we can treat them as sets of 3-tuples $(v_i, v_j, elabel)$, where $i < j$. Edge label *elabel* has four types: (i) *uplink_call*, (ii) *downlink_call*, (iii) *uplink_transition*, and (iv) *downlink_transition*, where “uplink” means that the edge direction is from v_i to v_j whereas “downlink” means the direction is from v_j to v_i . In this way, each behavior graph is regarded as a set of distinct edges. Traditional closed graph mining algorithms, such as CloseGraph [24], do not take advantage of this property. In the following discussion, we develop a simpler graph mining algorithm that fits behavior graphs better.

We apply the pattern-growth methodology to mine closed frequent graphs¹: Whenever a new frequent graph is uncovered, we extend this graph as much as possible until the maximum one is found. Let g be a frequent subgraph with n edges. Suppose g is extended in a series of g_1, g_2, \dots, g_n ($g_1 = \emptyset, g_n = g$), where g_i is a graph formed from g_{i-1} by adding one new edge. If graphs g_i, g_{i+1}, \dots , and g_n have the same support, one could skip the search space between g_i and g_n . That is, whenever g_i is found, g_i should be directly extended to g_n through g_{i+1} to g_n . Any other graph that is a supergraph of g_i and a subgraph of g_n should not be enumerated except g_i, g_{i+1}, \dots , and g_{n-1} . We call it *search space skipping*. However, as illustrated in [24], CloseGraph has to miss some skipping in order to preserve the depth-first search order. The miss of search space skipping may cause problem when the closed frequent subgraphs are very large. Therefore, the naive search order [23] is adopted in our mining algorithm to skip the search space as much as possible.

Algorithm 1 (CloseMine) describes the pseudo code of our closed frequent graph mining algorithm. At each iteration of CloseMine, it first extends a newly discovered frequent graph with one more edge. Then CloseMine checks whether this graph has already been discovered (Line 1 in Algorithm 1). If not, it continues searching its supergraphs. CloseMine adopts an optimization

¹Note that all the closed frequent graphs under examination are connected graphs.

Algorithm 1 CloseMine(g, D, minsup, S)

Input: A graph g , a graph dataset D , a minimum support threshold minsup .

Output: The closed frequent graph set S .

```
1: if  $\exists g' \in S$  s.t.  $g \subset g'$  and  $\text{support}(g) = \text{support}(g')$ 
   then return;
2: extend  $g$  to  $g'$  as long as  $\text{support}(g) = \text{support}(g')$ ;
3: insert  $g'$  to  $S$ ;
4: scan  $D$  once, find edge  $e$  s.t.  $g' \cup \{e\}$  is frequent;
5: for each frequent  $g' \cup \{e\}$  do
6:   CloseMine( $g' \cup \{e\}, D, \text{minsup}, S$ );
7: return;
```

(Line 2) that extends a frequent graph as much as possible until there is no supergraph having the same support.

3.2 Relationship between Closed and Frequent Graph-based Classification. In this section, we examine the relationship between the frequent graph-based and the closed frequent graph-based classification.

Since the whole set of frequent graphs can be reconstructed from closed frequent graphs, a potential question is whether frequent graph-based SVMs can be exactly constructed through a closed frequent graph-based training process? The answer is “yes”. Actually, the concept discussed here can also be generalized to other kinds of frequent patterns like itemsets and sequences. Let us first examine how to build a mapping from frequent graphs to closed frequent graphs.

LEMMA 3.1. *Given a behavior graph G , there is one and only one closed behavior graph G' such that $G \subseteq G'$ and $\text{support}(G) = \text{support}(G')$.*

Proof. Assume to the contrary that there is another closed graph G'' s.t. $G \subset G''$ and $\text{support}(G) = \text{support}(G'')$. Let G^* be the graph formed by $G' \cup G''$. G^* is a connected graph since G' and G'' share a common subgraph G . Therefore, $G'' \subset G^*$ and $\text{support}(G'') = \text{support}(G^*)$, contradicting our assumption. ■

Note that Lemma 3.1 only holds for graphs that have distinct label for each node. Fortunately, behavior graph has this property. Lemma 3.1 shows that there exists one function $f: \mathbb{F} \mapsto \mathbb{C}$, which maps any frequent graph in a frequent graph set \mathbb{F} to one and only one closed graph in a closed frequent graph set \mathbb{C} . Thus, given a graph dataset D and a pre-defined minimum support threshold δ , the above mapping function can be obtained by mining closed frequent graphs from the

dataset and constructing frequent graphs from closed frequent graphs.

In the *frequent or closed feature space*, a graph instance G is represented by a feature vector whose i -th dimension is instantiated to 1 if G contains the i -th feature (frequent graph or closed frequent graph) or 0 otherwise. Given a graph G , the vectors of G in the frequent and closed feature space can be transformed with each other through the mapping function f as described before. The number of frequent graphs that map to the same closed frequent graph g is written as $c(g)$.

In the following discussion, we will show that an SVM trained in the frequent feature space for a training dataset can be constructed in the closed feature space. That is, we may solve the quadratic programming problem for a frequent graph-based SVM in the closed feature space.

Let \mathbf{x} be the feature vector of a graph instance G in the frequent feature space and \mathbf{z} be the vector of G in the closed feature space. Let d be the number of dimensions in the closed feature space and M be a diagonal matrix,

$$M = \begin{pmatrix} \sqrt{c(g_1)} & \dots & \dots & 0 \\ 0 & \sqrt{c(g_2)} & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & \sqrt{c(g_d)} \end{pmatrix}.$$

If we train a linear SVM in the frequent feature space, then $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$, which is equal to $(M\mathbf{z}_i) \cdot (M^T\mathbf{z}_j)$. Let $\mathbf{z}' = M\mathbf{z}$. Vector \mathbf{z}' is in a new feature space \mathbb{C}' , which is formed by scaling the original closed feature space with M . Since $\mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{z}'_i \cdot \mathbf{z}'_j$, the solution of the quadratic programming problem in this new space will be exactly the same as that of the quadratic programming problem in the frequent feature space. Thus, we may use closed frequent graphs as features with the scaling matrix M to learn an equivalent SVM in the frequent feature space.

We further found that if two frequent graphs g_i and g_j , $g_i \subset g_j$, are mapped to the same closed frequent graph, their weights in the optimal hyperplane are the same. That means SVMs cannot distinguish graphs g_i and g_j from the training set. In the closed frequent graph-based classification, we only treat graph g_j as a feature (if g_j is closed), while the frequent graph-based approach also counts g_i as a feature. It is difficult to tell which method is better. However, our experiments indicate that the closed graph-based approach can achieve the similar or even better accuracy in comparison with the frequent graph-based approach.

4 Uncover “Backtrace” for Noncrashing Bugs

With the classification technique developed in Section 3, we here illustrate how to assist programmers in debugging noncrashing bugs.

Software bugs can be classified into two categories, according to their running behaviors. The first one is crashing bugs, which terminate the program execution abnormally with segmentation faults. For instance, illegal memory access and dereference to null pointers are two typical cases. Although crashing bugs happen quite often, they are not too difficult to tackle. At the crashing point, developers can obtain the backtrace, the snapshot of function call stack, based on which tracing back is straightforward. For example, in Fig. 1(b), the program crashes in `prompt`, then we have a function call stack, `traverse_file` \rightarrow `file_action` \rightarrow `prompt`. Programmers may carefully check the logic in these functions first. On the other hand, the other type is noncrashing bugs, which, as suggested through the name, do not incur program crashes. Noncrashing bugs are usually detected in software testing phase. Specifically, when a set of test suites are applied, some of outputs fail to match the expected. In general, fighting noncrashing bugs is harder than crashing ones. Few clues are available for programmers to debug noncrashing bugs.

Through comparison, we notice that this extra difficulty for noncrashing bugs partially comes from the absence of “backtrace”-like information. Suppose a “backtrace” is available for noncrashing buggy runs, which shows what functions are bug relevant, developers could be hinted to focus initial emphasis on those suspected functions. Therefore, we then aim at *identifying suspicious functions that are relevant to incorrect runs*. These functions may provide information to programmers in a way similar to “backtrace”.

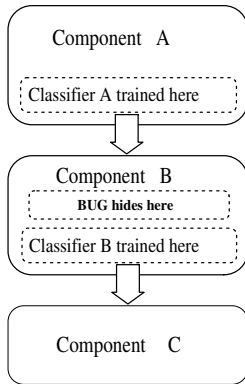


Figure 4: Classification Accuracy Boost

Our method is based on the analysis of the classification accuracy boost. Generally, the classification ac-

curacy should not decrease while more and more trace data become available; especially, accuracy will improve once the execution data contain buggy behaviors. This is illustrated in Fig. 4. Suppose a program runs through components A , B and C in sequence and a noncrashing bug resides in component B as shown. Classifier f_A is trained at the end of execution of component A . As expected, its accuracy cannot be high since it knows few, if any, behaviors induced by the bug. In contrast, classifier f_B that is trained after component B , is expected to have a much higher accuracy than f_A because it does have behavior graphs induced by the bug in incorrect runs. Therefore, as long as f_B has a classification accuracy boost in comparison with f_A , it is more likely that the bug is located in Component B than Component A. This inspires us to uncover “backtrace” for noncrashing bugs by detecting the accuracy change in a series of classifiers incrementally trained along the execution axis.

Specifically, for each function, F_i , two checkpoints B_{in}^i and B_{out}^i are placed at the entrance and the exit of F_i respectively. At each checkpoint, a set of behavior graphs are collected, each of which corresponds to one test case running up to this checkpoint. Then using the classification technique developed in Section 3, a classifier can be trained at each checkpoint with accuracy (precision and recall) evaluated through cross-validation. In our experiments, we choose the highest precision as the accuracy measure while keeping recall no less than 90%. This guarantees that only few incorrect runs are missed and hence precision is a fair measure for comparison. In this way, each function is attached with a precision pair $[P_{in}^i, P_{out}^i]$. If there is a significant precision boost from P_{in}^i to P_{out}^i , we would think function F_i as bug-relevant. Its formal definition is given as follows.

DEFINITION 2. (BUG-RELEVANT) *Given a significance level of precision boost θ ($0 < \theta \leq 1$), a function F_i is bug-relevant if $P_{out}^i - P_{in}^i \geq \theta$.* ■

Consequently, bug-relevant function set (BRFS) refers to the set of functions that are bug-relevant with respect to a significance level θ .

In general, BRFS is a smaller subset of all the functions, and hence it will be effective in helping programmers at debugging; otherwise, all functions are conceptually suspicious.

Furthermore, through experimental studies, we found that BRFS has several nice properties, which further enhance its applicability in debugging. For instance, it is easy to choose a proper cutoff θ , distinguishing bug-relevant from “bug-irrelevant”. In addition, due to the nested structure of function executions, BRFS is

Version	Incorrect Runs	Correct Runs	Buggy Line #	Bug Description
3	130	5412	493	missing one boolean subclause in <i>if</i> statement
4	143	5399	493	misuse of variable
5	271	5271	117	misuse of $<$ while \leq is expected
14	137	5405	369	missing one boolean subclause in <i>if</i> statement
26	198	5344	369	misuse of j while $j+1$ is expected

Table 1: Summary of Buggy Versions

likely to line up in a form quite similar to “backtrace”. However, since we are not yet very clear about the underlying model governing program executions, we refrain from presenting these properties formally. As an alternative, we examine a detailed case study in Section 5.4 together with reasonings about its soundness.

5 Experiments and Case Study

In this section, we evaluate the effectiveness and efficiency of closed frequent graph-based classification. A detailed case study is also given to illustrate its usage in uncovering backtrace of noncrashing bugs.

For classification evaluation, we designed three methods for comparison.

1. *edge*: Edges of a behavior graph are treated as features.
2. *frequent+*: In addition to *edge*, frequent graphs are treated as additional features. The symbol ‘+’ means the classifier also uses edges as features.
3. *close+*: In addition to *edge*, closed frequent graphs are treated as additional features.

All of our experiments were carried out on a 3.2GHz Intel Pentium 4 PC with 1GB physical memory, running Redhat Linux 9.0. *SVM^{light}* [13] was chosen in our implementation due to its good scalability.

5.1 Subject Programs. We took *Siemens Programs* as our testbed, which are widely used in software research [12, 21, 8, 10] because of its artificially instrumented but “realistic” enough software bugs. Readers interested in how Siemens researchers simulated realistic software bugs are referred to [12]. In our experiments, we chose *replace*, one of *Siemens Programs*, as our subject program. It performs regular expression matching and substitution. We chose it because the correctness of an execution is easy to label given the availability of a bug-free version.

Replace program in our study contains 32 versions in total, among which Version 0 is a bug-free version and other versions have one bug each. In this setting, Version 0 serves as the oracle in labelling whether a run

Version	for incorrect runs	for correct runs
3	15	549
4	22	547
5	74	538
14	39	604
26	50	543

Table 2: Number of Distinct Behavior Graphs

is “correct”. We conducted experiments on five buggy versions, which, in our point of view, nicely mimic the typical noncrashing bugs in reality. Table 1 shows the characteristics of these five buggy versions and their bug descriptions.

In order to objectively evaluate the effectiveness of classification, we remove duplicated behavior graphs within the set of correct and incorrect runs respectively. This is based on the consideration that two different but similar inputs may result in the same behavior graph. Table 2 lists the number of distinct graphs in the five versions.

5.2 Effectiveness. In our experiment, incorrect runs are labelled as positive samples and correct ones as negatives. As shown in Table 2, the numbers of positives and negatives are highly imbalanced, suggesting that we should evaluate the effectiveness through precision and recall, rather than pure accuracy.

Recall is defined as the fraction of the total number of incorrect runs that are classified right. *Precision* refers to the fraction of incorrect runs classified that are actually incorrect runs. Though it is highly desirable to achieve both high precision and recall, these two measures are usually contrary to each other. In practice, higher recall means low rate of missing incorrect runs while high precision means high hit rate and low rate of false alarms. In assistance to programmers’ debugging, high precision with reasonably high recall means that the classification features are of high quality in discriminating incorrect runs from correct ones.

We perform five-fold cross validation and plot the result of each method in a recall-precision curve. Intuitively, a better method should have the recall-precision

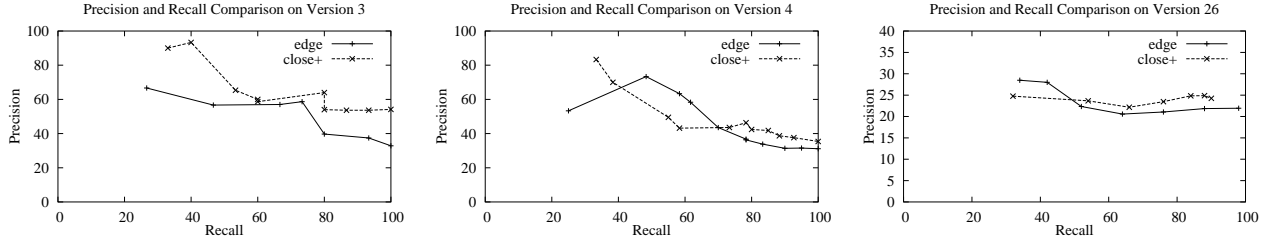


Figure 5: Precision and Recall: *close+* vs. *edge*

curve nearer to the upper right corner.

5.2.1 Effectiveness in Detecting Failing Runs.

Fig. 5 shows the classification results of *edge* and *close+* on Versions 3, 4 and 26. The other two versions have the similar trend.

In classification of program runs, high recall is required due to the high cost of bugs [16, 1]. Thus we emphasize the precision when the recall is at a high level, e.g., 70% and higher. Version 3 in Fig. 5 has the best accuracy: with the 100% recall, the precision can be as high as 50%. It indicates that the classifier does not miss any real incorrect runs and at least one of two alarms is hit on average. Table 2 shows that the ratio between positives and negatives is about 1:37 (i.e., 15:549), which implies that random guessing according to this prior distribution would result in a precision around 2.7% (i.e., 15/(15+549)). The 20-times promotion of precision reaffirms our belief that behavior graphs are informative as to correctness of program executions. Similar conclusions can also be drawn on Version 4 and Version 26 depicted in Fig. 5. Generally, when the recall is as high as above 90%, our classifiers can still maintain a precision no lower than 25%. Considering the highly skewed distribution of positives and negatives in Table 2, we believe SVMs on behavior graphs perform well in the identification of incorrect runs.

Fig. 5 shows that *close+* generally outperforms *edge*, especially when a high recall is a must. This indicates that the addition of closed frequent graphs as features can leverage the classifier quality. In Versions 4 and 26, *edge* also achieves good performance. These are the cases where edges can be rather discriminative in revealing program correctness.

5.2.2 Closed vs. Frequent Graph-Based Classification. Next, we compare the classification accuracy between *frequent+* and *close+*. In Section 3, we show that frequent graph-based SVMs can be trained in the closed feature space. Therefore, we conjecture that

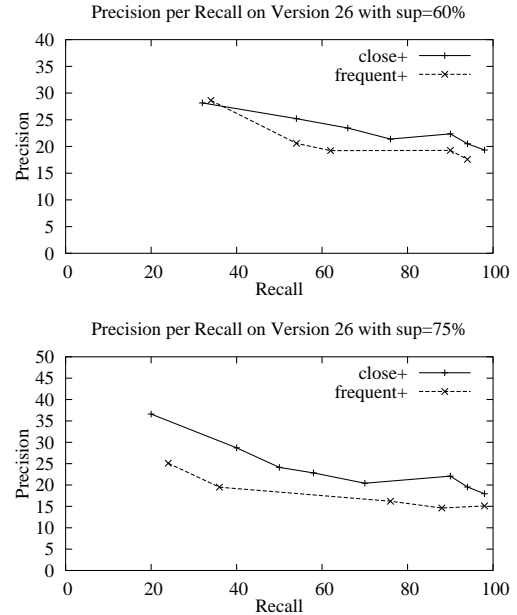


Figure 6: Precision and Recall: *close+* vs. *frequent+*

closed and frequent graph-based SVMs would probably have similar classification performance.

Fig. 6 presents the accuracy of *close+* and *frequent+* on Version 26, which also suggests a little bit better performance of *close+*. Note that the minimum support is set at 60% and 75% respectively, rather than 25% as used in Fig. 5. Under the 50% threshold, *frequent+* failed to complete the mining process.

5.3 Scalability. Figs. 7 and 8 compare *close+* and *frequent+* in terms of mining and training time. It indicates the better scalability of *close+* over *frequent+*. We only plotted the results from Version 3 for examination since others have the similar characteristics.

It becomes obvious that the computational cost of *frequent+* is exponential with regard to the minimum support threshold. For example, *frequent+* cannot finish in 10 hours when the support threshold is at 50%. On

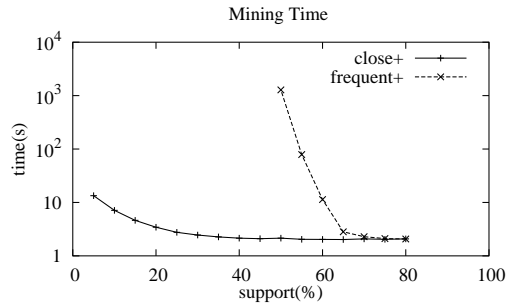


Figure 7: Mining Time w.r.t. Support

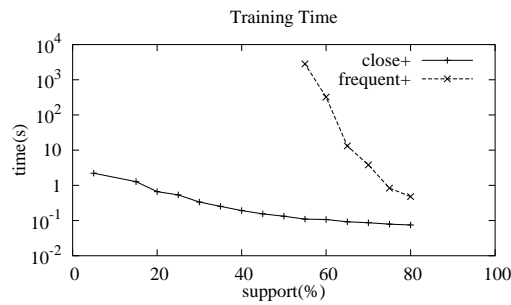


Figure 8: Training Time w.r.t. Support

the contrary, *close+* ran smoothly when the support was gradually lowered down. In practice, a reasonably low threshold is preferred since more patterns can be explored as potential features. When the support is at 5%, *close+* only takes around 15 seconds to learn a classifier (i.e. mining time + training time), which is surprisingly fast.

5.4 Case Study. In this subsection, we illustrate how to backtrace a noncrashing bug through a detailed case study. Section 5.4.1 describes what the bug is, followed by an examination of our approach in Section 5.4.2. We discuss its validity in Section 5.4.3.

5.4.1 Case Description. The buggy code we studied is shown in Program 1, which comes from Version 3 of the “*replace*” program. Within the if-statement at line 9, the subclause “(lastm != m)” is missed for some reason. This “miss of corner case” logic bug causes more than expected runs fall into the condition block between Lines 9 and 12, which in consequence induces incorrect outputs. In this buggy program, programmers may feel confused about where to start debugging since incorrect runs will finish smoothly. Usually they have to verify the code step by step, which is very time-consuming.

Program 1 Buggy Code - Subline Function

```

1 void
2 subline(char *lin, char *pat, char *sub)
3 {
4     int i, lastm, m; 8
5     lastm = -1;
6     i = 0;
7     while ((lin[i] != ENDSTR)) {
8         m= amatch(lin, i, pat, 0);
9         if (m >= 0) /* && (lastm != m) BUG!!!*/{
10             putsb(lin, i, m, sub);
11             lastm = m;
12         }
13         if ((m == -1) || (m == i)){
14             fputc(lin[i], stdout);
15             i = i + 1;
16         } else
17             i = m;
18     }
19 }

```

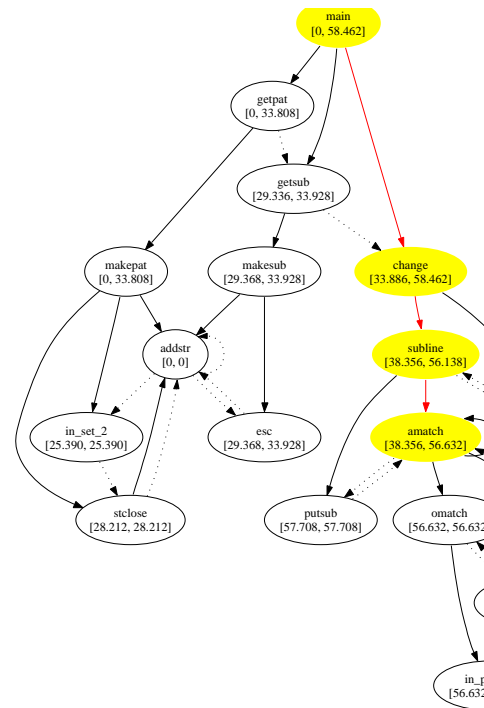


Figure 9: Entrance Precision and Exit Precision

5.4.2 How It Works. Fig. 9 shows the experimental results using our approach that helps narrow down the suspicious bug region. The classifiers are trained on behavior graphs from various program runs. Our classification method is applied at the entrance and the exit of each function. So each function has two precision values – entrance precision and exit precision.

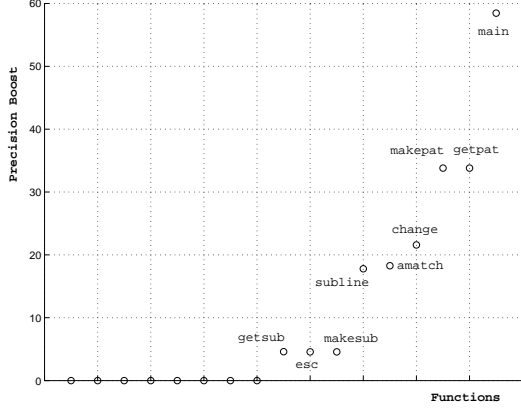


Figure 10: Precision Boost of Functions

Precisions depicted here are with recall at least 95%. We sort functions in increasing order of precision boosts in Fig. 10.

According to the method laid out in Section 4, the first task is to choose a proper significance level θ to identify bug-relevant functions. Seen from Fig. 10, eight functions induce no precision boost while another three only cause less than 5% precision increase. In contrast, the remaining six functions possess more than 17% boost. Therefore, it is easy to choose a safe cutoff in differentiating bug-relevant functions from irrelevant ones. The wide range of cutoffs clearly shows that bug relevance is an objective fact, rather than a subjective judgement. In addition, we believe this property should hold in general because functions that have nothing to do with the incorrect executions are less likely to cause significant precision boost. As a result, six out of the entire 17 functions are identified as bug-relevant. The result is summarized in Table 3.

function name	$Precision_{in}$	$Precision_{out}$
main	0	58.462
getpat	0	33.808
makepat	0	33.808
change	33.886	58.462
subline	38.356	56.318
amatch	38.356	56.632

Table 3: Bug-Relevant Functions with $\theta = 20\%$

Table 3 together with Fig. 10 exposes the following interesting results.

First, **main** function always has the highest precision and precision boost. This makes sense because P_{out}^{main} measures the classifier that uses the information of the whole run, hence achieves the best precision. Meanwhile, P_{in}^{main} is always 0% since no information is

available in the entrance to the **main** function. Therefore, the **main** function always has the highest precision boost. Since **main** is the only entrance of a program, it is trivial to be regarded as bug-relevant.

Second, we can divide the six functions in Table 3 into two groups and rank them by their exit precision (i.e. P_{out}^i). Clearly, functions **main**, **change**, **subline** and **amatch** form a group with the highest exist precision, which actually reveals the backtrace to the buggy code.

Finally, bug-relevant functions tend to line up to form a backtrace. As shown in Fig. 9, the identified bug-relevant functions, namely **main**, **change**, **subline** and **amatch**, form the backtrace for this noncrashing bug. Again, we think this property should hold in general because the nested calling structure is typical in program executions. For instance, if function A calls B and B is regarded as bug-relevant, A would also be bug-relevant because A exits later than B and hence has more bug-relevant information.

In summary, through the above analysis we have uncovered the “backtrace” for this noncrashing bug. Taking this “backtrace” as hints, a programmer can start debugging in a similar way as facing the real backtrace. It is expected that a programmer could pay more attention on this backtrace rather than suspecting all the functions.

5.4.3 Discussion on General Validity. Although our method works reasonably well in the above case, we are not going to claim its general applicability. Due to the wide variety of software bugs, it is unlikely for a method to work well in all cases. In this study, we have been exerting great efforts to narrow down suspicious bug trace by using data mining techniques. The entire framework of exploiting classification dynamics to uncover “backtrace” makes sense by intuition and reasoning. Furthermore, our case study does capture a kind of common bugs, which may imply its applicability beyond this particular case.

We note that our method can only provide programmers with the “backtrace”, a set of bug-relevant functions, which hopefully can assist programmers in a similar way as debugger-provided backtraces for crashing bugs. However, just as a real backtrace may not immediately lead to the discovery of the bug root for a crashing bug, neither does our method. Still a programmer has to scrutinize the source code and figure out a way to fix.

Computer-aided debugging is profound and hence hard to be solved thoroughly in one shot. To the best of our knowledge, it is less likely, if not impossible, to devise a fully-automated debugger, which detects and fixes

bugs without the involvement of human intelligence. We are looking forward to more debates and insights on this interesting and challenging problem.

6 Related Work

Previous related work falls into two fields: frequent pattern-based classification and software debugging.

6.1 Frequent Pattern-Based Classification. Statistical significance of frequent patterns motivates their applicability in classification problems, which is based on the belief that frequent patterns can embody significant and discriminative features. *Associative classification* [19, 17] tries to find a set of association rules based on frequent patterns, from which high quality rules are selected as meta-rules for classification. In contrast, we explore the potentials of *all* the patterns and use sophisticated learning algorithms, such as SVMs, to combine their discriminative power smoothly. In addition, *pattern-based classification* has been successfully applied to chemical and biological domains, such as classification of outer membrane proteins [22] and chemical compounds [3]. In this paper, we not only apply data mining techniques in software engineering, but also demonstrate the power by incorporating closed frequent patterns as features. As shown through experiments, our method has better scalability and meanwhile uplifts the classification accuracy. To the best of our knowledge, this is the first piece of work on using closed frequent patterns in classification and demonstrating their usage in software engineering.

6.2 Software Bug Detection. Software reliability is actively pursued in software engineering and computer system research from various angles. *Static analysis* [4, 6] aims at detecting program abnormalities from the source code level without running the programs. *Dynamic analysis* [2, 5, 20, 18, 26], on the contrary, usually instruments subject programs to dump runtime information during their execution for further analysis. In addition, *model checking* [15] and *fault injection and analysis* [14] also work towards better software reliability through their own approaches.

Our work is in the category of dynamic program analysis, within which the following studies are the most related. Program invariants [7] are used to assist programmers in debugging [2, 18, 26]. Logistic regression is adopted in [18, 26] to single out discriminative invariants while Brun and Ernst use SVMs [2]. Researchers also explore the possibility of clustering incorrect runs based on software behaviors [5, 20]. We approach the software reliability problem through a classification method.

7 Conclusions

In this paper, we investigate the capability for computers to classify incorrect and correct executions based on observations of program behaviors. We develop a classification framework by summarizing program executions as behavior graphs. As demonstrated through experiments, the classification can be both effective and efficient. Moreover, we propose a novel method to exploit the classification accuracy boost and help programmers debug noncrashing buggy code, which otherwise may be elusive to handle. By examining software reliability from a data mining point of view, we make our initial efforts to explore how data mining techniques can contribute to software reliability, a hard but invaluable problem.

There are many issues that need to explore further. For example, it is not clear whether our method can be effective at tracing large software programs with the existence of multiple bugs in different program modules, how to further develop our method to make the trace deeper with finer granularity (such as a small set of program lines), and how to integrate this new approach with other existing software debugging methods. These are a set of issues for our future research.

Acknowledgement

We would like to thank Professor Gregg Rothmel and his colleagues at University of Nebraska - Lincoln for providing us Subject Infrastructure Repository.

References

- [1] European Space Agency. Ariane-5 flight 501 inquiry board report. In <http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf>.
- [2] Y. Brun and M. Ernst. Finding latent code errors via machine learning over program executions. In *Proc. of the 26th Int. Conf. on Software Engineering (ICSE'04)*, pages 480–490, 2004.
- [3] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. In *Proc. of the 3rd IEEE Int. Conf. on Data Mining (ICDM'03)*, pages 35–42, 2003.
- [4] D. Dhurjati, S. Kowshik, V. Adve, and C. Lattner. Memory safety without runtime checks or garbage collection. In *Proc. of Languages Compilers and Tools for Embedded Systems*, pages 69–80. ACM Press, 2003.
- [5] W. Dickinson, D. Leon, and A. Podgurski. Finding failures by cluster analysis of execution profiles. In *Proc. of the 23rd Int. Conf. on Software Engineering (ICSE'01)*, pages 339–348. IEEE Computer Society, 2001.
- [6] N. Dor, M. Rodeh, and M. Sagiv. Csvg: towards a realistic tool for statically detecting all buffer overflows

- in c. In *Proc. of the ACM SIGPLAN 2003 Int. Conf. on Programming Language Design and Implementation (PLDI'03)*, pages 155–167. ACM Press, 2003.
- [7] M. Ernst, J. Cockrell, W. Griswold, and D. Notkin. Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering*, 27(2):1–25, 2001.
- [8] P. Frankl and O. Iakounenko. Further empirical studies of test effectiveness. In *Proc. of the 6th ACM SIGSOFT Int. Symp. on Foundations of Software Engineering (FSE'98)*, pages 153–162. ACM Press, 1998.
- [9] S. Hangal and M. Lam. Tracking down software bugs using automatic anomaly detection. In *Proc. of the 24th Int. Conf. on Software Engineering (ICSE'02)*, pages 291–301. ACM Press, 2002.
- [10] M. Harrold, G. Rothermel, R. Wu, and L. Yi. An empirical investigation of program spectra. In *Proc. of the ACM SIGPLAN-SIGSOFT workshop on Program Analysis for Software Tools and Engineering*, pages 83–90. ACM Press, 1998.
- [11] J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha. Mining spatial motifs from protein structure graphs. In *Proc. of the 8th Annual Int. Conf. on Research in Computational Molecular Biology (RECOMB'04)*, 2004.
- [12] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proc. of the 16th Int. Conf. on Software Engineering (ICSE'94)*, pages 191–200. IEEE Computer Society Press, 1994.
- [13] T. Joachims. Advances in kernel methods: Support vector learning. In *chapter "Making Large-scale SVM Learning Practical"*. MIT Press, 1999.
- [14] G. Kanawati, N. Kanawati, and J. Abraham. Ferrari: A flexible software-based fault and error injection system. *IEEE Transactions on Computers*, 44:248–260, 1995.
- [15] S. Kumar and K. Li. Using model checking to debug device firmware. *SIGOPS Oper. Syst. Rev.*, 36(SI):61–74, 2002.
- [16] N. Leveson and C. Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.
- [17] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proc. 2001 Int. Conf. on Data Mining (ICDM'01)*, pages 369–376, San Jose, CA, 2001.
- [18] B. Liblit, A. Aiken, A. Zheng, and M. Jordan. Bug isolation via remote program sampling. In *Proc. of the ACM SIGPLAN 2003 Int. Conf. on Programming Language Design and Implementation (PLDI'03)*, pages 141–154. ACM Press, 2003.
- [19] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of the 4th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD'98)*, pages 27–31. ACM Press, 1998.
- [20] A. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun, and B. Wang. Automated support for classifying software failure reports. In *Proc. of the 25th Int. Conf. on Software Engineering (ICSE'03)*, pages 465–475. IEEE Computer Society, 2003.
- [21] G. Rothermel and M. J. Harrold. Empirical studies of a safe regression test selection technique. *IEEE Transaction on Software Engineering*, 24(6):401–419, 1998.
- [22] R. She, F. Chen, K. Wang, M. Ester, J. Gardy, and F. Brinkman. Frequent-subsequence-based prediction of outer membrane proteins. In *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*, pages 436–445. ACM Press, 2003.
- [23] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 721–724, 2002.
- [24] X. Yan and J. Han. CloseGraph: Mining closed frequent graph patterns. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 286 – 295, 2003.
- [25] T. Zhang, X. Zhuang, S. Pande, and W. Lee. Hardware supported anomaly detection: down to the control flow level. In *Technical Report, Center for Experimental Research in Computer System, GIT-CERCS-04-11, Georgia Institute of Technology*, 2004.
- [26] A. Zheng, M. Jordan, B. Liblit, and A. Aiken. Statistical debugging of sampled programs. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Learning to Refine Ontology for a New Web Site Using a Bayesian Approach*

Tak-Lam Wong and Wai Lam

Department of Systems Engineering And Engineering Management
The Chinese University of Hong Kong
Hong Kong
{wongtl,wlam}@se.cuhk.edu.hk

Keywords: Text Mining, Knowledge Adaptation, Ontology Refinement

Abstract

We develop a probabilistic framework which can refine an existing ontology from a source Web site to new unseen sites. One characteristic of our framework is to consider several clues related to how an ontology influences the text content and the visual layout of the Web pages. The first clue is the text fragments regarding the content of the concepts previously collected or extracted from the source Web site. The second clue is the text fragments regarding the header labels of the concepts. The third clue is the visual layout of the text fragments regarding the content of the concepts and the header labels of the concepts in the unseen site. To harness the uncertainty involved in a rigorous manner, we formalize these clues by a generative model to represent the generation of text fragments regarding the concepts and the ontology corresponding to the Web page. Bayesian learning technique and expectation-maximization (EM) algorithm are employed to accomplish the task. Extensive experiments on several real-world Web sites from two different domains have been conducted to demonstrate the effectiveness of our framework.

1 Introduction

Recently, ontology has become one of the important research topics in Web technology. This is mainly due to the emergence of semantic Web which attempts to give well-defined and machine-understandable meaning to Web resources for a specified domain [20]. Ontology is also employed in various applications. For example, it can be used to enhance the performance of Web personalization [6]. Web personalization is to intelligently provide users with tailor-made information. By annotating the Web page content semantically with a conceptual hierarchy, precise Web content can be processed in a flexible manner. Ontology is also used extensively in bioinformatics [13]. Due to the terminological difference in different biological repositories, data cannot

*The work described in this paper was substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CUHK 4187/01E, CUHK 4179/03E, and CUHK 4193/04E) and CUHK Strategic Grant (No: 4410001).



Figure 1: An example of Web page about book catalog.

be shared easily. Gene Ontology attempts to solve this problem by introducing a set of controlled vocabularies for annotating a gene product with its molecular functions, the biological process in which it is involved, and the cellular locations in which it is found [14].

Ontology defines the concepts and their relationships in a given domain. It is also regarded as a conceptual hierarchy of the domain since it is normally expressed as a tree-like structure. For example, Figure 1 shows a sample of a Web page from a Web site about book catalog¹. There are several book records in this page. Each book record contains concepts such as “title”, “author”, “published”, “list price”, “you save”, and “our price”. Figure 2 depicts a sample ontology representing a book in this Web site. In this ontology, there is a root node called “book”. The internal nodes such as “title” represent concepts associated with a book. “List price”, “you save”, and “our price” are the subconcepts of the concept “price”.

Manual effort is required to construct an ontology for a given site. However, if we need to deal with a large number of different sites, this task becomes tedious, error-prone, and requires high level of expertise. Recently, several research groups attempted to reduce human effort in ontology construction by applying machine

¹The URL of the Web site is <http://www.halfpricecomputerbooks.html>.

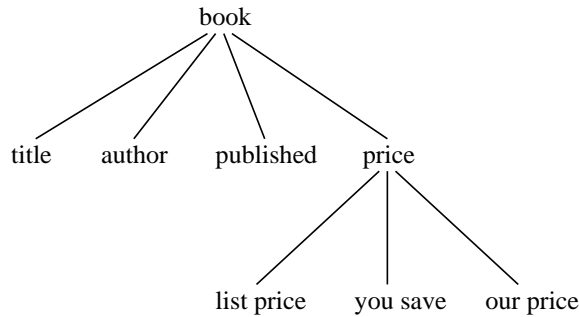


Figure 2: The ontology describing the relationship between the concepts of a book in the Web page shown in Figure 1



Figure 3: An example of Web page about book catalog collected from a different Web site shown in Figure 1.

learning techniques for ontology learning [9, 11, 15]. Of these proposed methods, some of them require interactions with the users during the construction process. Some methods require that the raw text data is organized in a specified format such as tables in Web pages. This poses limitations in current ontology learning techniques. Moreover, the ontology constructed for a particular Web site may not be able to effectively apply to another Web site even in the same domain. Consider the Web page shown in Figure 3². It is collected from a Web site different from the one shown in Figure 1. Figure 4 depicts the ontology describing the concepts of a book in this Web site. Although both ontologies in Figures 2 and 4 define a book, there are several differences. First, some concepts such as “published” and “ISBN” are present in only one of the ontologies. Second, “list price” in Figure 2 and “MSRP” in Figure 4 refer to the same concept, but in different terminology. Therefore, the ontology constructed for one Web site typically cannot be reused in another Web site. A separate effort is required to construct an ontology for the new site.

We develop a probabilistic framework which can

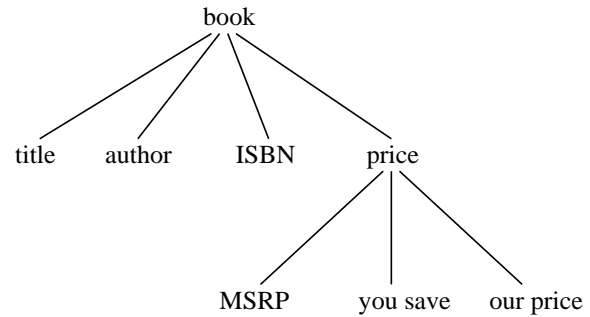


Figure 4: The ontology describing the relationship between the concepts of a book in the Web page shown in Figure 3

automatically refine an existing ontology tailored to a new unseen Web site in the same domain. For example, suppose the source Web site is the one shown in Figure 1 which is associated with the ontology shown in Figure 2. Our framework can automatically refine this existing ontology to suit the new unseen site as shown in Figure 3. The resulting ontology after the refinement will be the one depicted in Figure 4. This is achieved by considering several clues related to how the ontology (concept and structure) influences the text content and visual layout of the Web pages. The first clue is the text fragments corresponding to the content of the concepts previously collected or extracted from the source Web sites. For example, text fragment samples corresponding to the content of the concept “title” on the Web page in Figure 1 include “Game Programming Gems 2”, “Microsoft Excel 2003 Programming Inside Out”, and “Practical C++ Programming 2nd Edition”. These text fragments can be easily collected or extracted by using automatic information extraction methods such as wrappers [3, 7, 17]. Our framework analyzes the characteristics of these text fragments. As the Web sites belong to the same domain, the content of the text fragments regarding the same concept shares some characteristics and provides useful information to identify the same concept in the new unseen site.

The second clue is the text fragments regarding the header labels of the concepts. The text fragments “Author”, “List Price”, and “You Save” in Figure 1 are examples of header labels of the concepts “author”, “list price”, and “you save” respectively. The header labels are useful because their semantic meaning and orthographic features can indicate the relationship between the concepts. For example, the words “Save” and “Price” are semantically related. Hence, it is likely that the concepts “list price” and “you save” are subconcepts of the concept “price”.

The third clue we consider is the visual layout of the text fragments regarding the content of concepts and the header labels of the concepts. The visual layout

²The URL of the Web site is <http://www.discount.com>.

provides information in two aspects. One aspect is the association between the content and the header label of a particular concept. For example, the header label “Author” in Figure 3 is bolded and is located on the left to the text fragments regarding the content of “author”. We can infer that similar patterns in the Web page are probably an indication of new or existing concepts. Another aspect is the location of the concepts relative to each other. We observe that concepts related to each other are normally located in a nearby position. For example, the concepts “list price”, “you save”, and “our price” in Figure 1 are located within the same row.

The clues described above involve uncertainty. To cope with the problem in a formal manner, we develop a generative model to represent the generation of text fragments regarding the concepts and the ontology corresponding to the Web page. Bayesian learning technique and expectation-maximization (EM) algorithm are then employed to achieve this task.

2 Related Work

Ontology plays an important role in semantic Web [20] since it provides a way to express the meaning and knowledge contained in the Web resources such as Web pages. With the semantic Web, software agents can then share knowledge in the Web. Another application of ontology is in the area of bioinformatics. Gene Ontology and RiboWeb are two examples for applying ontologies to semantically describe the knowledge in bioinformatics resources [1, 14]. Stevens et al. proposed to use the ontology language DAML+OIL to construct the bioinformatics concepts [13]. The constructed ontology can support inference and be shared in the semantic Web.

In spite of the increasing popularity of ontology in expressing knowledge, ontology construction is a tedious task and requires high level of expertise. Some methods have been proposed to reduce the human effort in constructing ontology. van der Vet and Mars proposed a bottom-up methodology for ontology construction [16]. Their idea is to perform generalization to form some concepts from the primitive concepts. Maedche and Staab developed a system with graphical user interface for users to easily construct ontologies [9]. OntoLearn [11] is another system to assist ontology construction. It makes use of natural language processing and machine learning techniques to extract terms in domain texts. The semantic meaning of the extracted term is interpreted by WordNet and the ontology is enriched by these extracted terms. The resulting ontology can then be edited, validated by other ontology management tools.

All the methods proposed above aim at reducing

the human effort in ontology construction. However, they all require user interaction during the construction process. A system known as TANGO [15] attempts to generate the ontology from data in table format in a semi-automatic fashion. In TANGO, an ontology engineer first constructs a kernel ontology to the system. Then, the system analyzes the content of each table such as its caption, attribute-value pairs in the Web page to form a mini-ontology. Next, the set of mini-ontologies will be integrated into the kernel ontology. One limitation of TANGO is that the data must be in table format.

Another common shortcoming for the above approaches is that the ontology constructed can only represent a particular Web site. If we want to construct the ontology for a new Web site, a separate effort is required. Maedche et al. investigated the problem of ontology reuse [8]. They found out that the ontology defined for one Web site may not be applicable to another site. They proposed a framework to solve this problem by providing a mathematical model to retain the consistency in the reused ontology. They also developed a tool for managing the ontologies from different Web sites. However, their approach still requires a considerable amount of human effort in practice. Doan et al. proposed a method to solve the ontology matching problem which aims at matching the concepts of two ontologies [5]. For example, the concept “associate professor” in one ontology may be equivalent to the concept “senior lecturer” in another ontology. However their approach requires human effort to prepare training documents in each concept. Their objective is also different from ontology construction or refinement.

3 Overview of Our Framework

The rationale of our framework is to consider several clues concerned with how the ontology affects the text content and visual layout of the Web pages. The first clue is the text fragments regarding the content of the concepts. For example, “Stephen Randy Davis” is a text fragment regarding the concept “author” in the Web page shown in Figure 1. This text fragment possesses characteristics of the content such as the first character in each word is capitalized. These characteristics help identify the text fragments corresponding to the content of similar concepts in the new unseen Web page. The second clue is the text fragments regarding the header label of the concepts. For instance, “You Save” is an example of a header label in Figure 1. The word “save” indicates that this header label is related to the concept “price” of a book. Similarly, the header labels “List Price” and “Our Price” of the concepts “list price” and “our price” respectively are also related to the concept

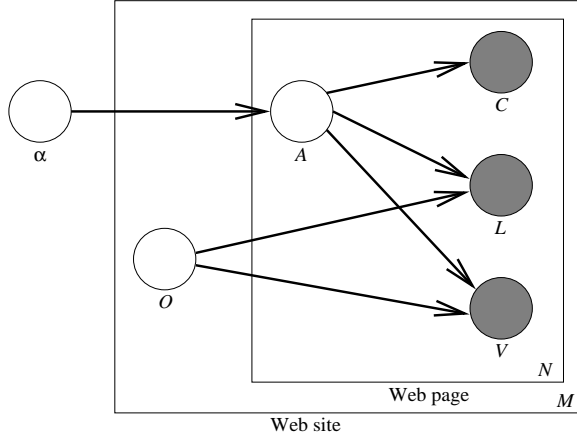


Figure 5: The generative model for generation of text fragments regarding the concepts from a domain ontology. α denotes the concept generation parameter. A and O denote the concept and the ontology respectively. C and L denote the content and the header label of a concept respectively. V denotes visual layout information. N is the number of text fragments regarding the concepts within the Web page and M is the number of Web pages in the Web site.

“price”. Therefore, it increases the likelihood that these concepts should be the subconcepts of the concept “price”. The third clue we consider is the visual layout of the text fragments regarding the content of concepts and the header labels of the concepts. One kind of visual layout information is the association between the content and the header label of a particular concept. The second kind of visual layout information is the location of the concepts relative to each other. Visual layout information helps discover new concepts in the unseen site. For example, suppose we wish to induce an ontology for the new Web page in Figure 3 given an existing concept “title” and some book title names previously collected or extracted from a site such as the one in Figure 1. Based on this evidence, the book title content on the new Web page can be identified. Therefore, the concept “title” can be induced for the new site. Similarly the existing concept “author” can be induced for the new site. Since the header label “Author” of the concept “author” on the new Web page makes use of certain font and boldness, it can be inferred that the text fragments “ISBN” on the new Web page is likely to be the header label of a new concept. Another example is that the concepts “MSRP”, “your price”, and “you save” are located in the same row in Figure 3. It increases the likelihood that these three concepts are the subconcepts of the concept “price”.

To harness the uncertainty involved in a rigorous manner, we investigate a generative model for the generation of text fragments regarding the concepts in a Web page. Figure 5 illustrates the graphical representation of our model. Shaded nodes and unshaded nodes

refer to observable variables and unobservable variables respectively. The arrows represent the dependence of the variables. Precisely, the variables at the head depend on the variables at the tail. In a particular domain, there is a concept generation parameter called α . This parameter is domain dependent and site invariant because the Web sites collected in the same domain contain similar kinds of concepts. This parameter controls the concepts, denoted by A , contained in the Web pages. For each of the N Web pages of the M Web sites, the concepts contained in the Web pages are generated by a certain probability distribution dependent on α , i.e., $P(A; \alpha)$. For example, the Web page in Figure 1 contains the concepts “title”, “author”, “published”, “list price”, “you save”, and “our price”. The content of a concept, denoted by C , is then generated according to the probability distribution $P(C|A)$. For instance, the content of “title” in the first record in Figure 1 is “Game Programming Gems 2”, which is generated from $P(C|A = \text{“title”})$. Since C is dependent on A which in turn depends on α , C is domain dependent and site invariant. Therefore, the text fragments regarding the content of the concepts are similar in the Web pages of the Web sites from the same domain. Another parameter in our model is the ontology associated with the Web site and it is denoted by O in Figure 5. This parameter is site dependent and influences the header label and the visual layout of a concept in the Web pages of a particular Web site.

The header label and the visual layout are denoted by L and V in Figure 5 respectively. The header labels are generated according to the probability distribution $P(L|A; O)$. The ontology also affects the visual layout of a Web page. For example, the concepts “list price”, “you save”, and “our price” are likely to be the subconcepts of “price” and they are located in the same row in the Web page. The visual layout of the concepts is generated by the probability distribution $P(V|A; O)$.

Based on this generative model, we can derive the following joint probability:

$$P(C, L, V, A; \alpha, O) = P(C|A)P(L|A; O)P(V|A; O)P(A; \alpha) \quad (3.1)$$

Using the above notation, a text fragment within a Web page can be represented by the attributes C , L , and V . To predict the concept of a particular text fragment, we compute the probability $P(A|C, L, V; \alpha, O)$. From Equation 3.1, we can derive that:

$$P(A|C, L, V; \alpha, O) = \frac{P(C|A)P(L|A; O)P(V|A; O)P(A; \alpha)}{\sum_A P(C|A)P(L|A; O)P(V|A; O)P(A; \alpha)} \quad (3.2)$$

In practice, the text fragments regarding to the con-

tent of concepts in the new unseen site are required to be identified in advance. We called it *informative text fragment*. We develop an information-theoretic approach to analyze the Document Object Model³(DOM) structure of the Web pages of the unseen Web site. The informative nodes in the DOM structure can be identified. The text fragments within the informative nodes become the informative text fragments.

The parameters of our probabilistic model is inferred from a set of informative text fragments regarding the concepts collected from the source Web site and the new unseen site. The text fragments collected from the source Web site are useful because their content share some characteristics with those text fragments in the new unseen site. Our framework can conduct this inference process to identify the concepts contained in the unseen site. For the informative text fragments collected in the new unseen site, their concepts are uncertain and cannot be used for inference directly. By modeling a concept (A) by an unobservable variable in the generative model, we can employ expectation-maximization (EM) algorithm in the inference of the parameters α and O . We derive several content features and make use of the text fragments collected from both of the source Web sites and the new unseen sites to estimate $P(C|A)$ in Equation 3.2. We can estimate $P(L|A; O)$ and $P(V|A; O)$ from the text fragments collected from the new unseen site since O is site dependent. However, the parameter O cannot be directly obtained because the text fragments collected do not come with a complete and correct ontology for the new unseen site. Potential ontology candidates are constructed by considering the concepts of the text fragments, the semantic meaning, and the orthographic features of the header labels. We design a metric to estimate $P(L|A; O)$ and $P(V|A; O)$ based on the relationship between each of the potential ontology candidates, the conditional probability $P(L|A)$, and the relationship between $P(V|A)$. The ontology can then be refined iteratively to adapt to the new unseen site in our EM framework. The learned model and the refined ontology can then be applied to compute the probability $P(A|C, L, V; \alpha, O)$ of each text fragment in the new unseen site.

4 Bayesian Learning Approach with EM

In our framework, the informative text fragments in the new unseen Web site under consideration belong to one of the concepts, either an existing concept from the source Web site or a new concept. In essence, the joint

probability in Equation 3.1 represents the probability for generating the text fragment (C, L, V) which belongs to the concept A . Suppose there are N text fragments in the Web page and let Λ be this set of N text fragments. The joint probability for generating Λ is as follows:

$$P(\Lambda; \alpha, O) = \prod_{i=1}^N P(C_i, L_i, V_i, A_i; \alpha, O) \quad (4.3)$$

As the concept of the text fragment is uncertain in the new unseen site, Equation 4.3 can be rewritten as follows:

$$P(\Lambda; \alpha, O) = \prod_{i=1}^N \sum_j P(C_i, L_i, V_i, A_{ij}; \alpha, O) \quad (4.4)$$

where A_{ij} represents the i -th text fragment belonging to j -th concept. Combining Equations 3.1 and 4.4, we can derive the log likelihood function $L(\alpha, O; \Lambda)$ as follows:

$$L(\alpha, O; \Lambda) = \sum_{i=1}^N \log \sum_j P(C|A)P(L|A; O)P(V|A; O)P(A|\alpha) \quad (4.5)$$

According to the principle of maximum likelihood estimation of parameters, our goal is to find the set of parameters (α, O) which maximizes the function depicted in Equation 4.5. However, the summation inside the logarithm makes such inference intractable. As A_{ij} is an unobservable variable in Equation 4.5, we can derive the following expected log likelihood function:

$$L'(\alpha, O; \Lambda) = \sum_{i=1}^N \sum_j P(A|\alpha) \log P(C|A)P(L|A; O)P(V|A; O) \quad (4.6)$$

which is tractable. By Jensen's inequality and the concavity of the logarithmic function, it can be proved that $L(\alpha, O; \Lambda)$ is bounded below by $L'(\alpha, O; \Lambda)$ [10]. Therefore, maximizing $L'(\alpha, O; \Lambda)$ is equivalent to maximizing $L(\alpha, O; \Lambda)$. The EM algorithm [4] is employed to iteratively increase $L'(\alpha, O; \Lambda)$ until convergence is reached. The E-Step and M-Step are as follows:

E-Step:

$$P(A|C, L, V; \alpha_t, O_t) \propto P(C|A)P(L|A; O_t)P(V|A; O_t)P(A|\alpha_t)$$

M-Step:

$$(\alpha_{t+1}, O_{t+1}) = \arg \max_{\alpha, O} L'(\alpha, O; \Lambda, \alpha_t, O_t)$$

We can express the terms $P(C|A)$ with probabilities related to the set of features related C . Assume that these features are independent to each other, $P(C|A)$

³The details of the document object model can be found in <http://www.w3.org/DOM/>.

can be expanded as follows:

$$P(C|A) = \prod_{k=1}^{Z_c} P(f_k^c(C)|A) \quad (4.7)$$

where Z_c is the number of content features and $f_k^c(C)$ is the k -th feature of the content C . We design the following features to characterize the content of a concept.

- f_1^c : the number of characters in the content
- f_2^c : the number of tokens in the content
- f_3^c : the average number of characters per token in the content
- f_4^c : the number of digits in the content
- f_5^c : the number of floating points in the content
- f_6^c : the number of alphabets in the content
- f_7^c : the number of upper case alphabets in the content
- f_8^c : the number of lower case alphabets in the content
- f_9^c : the number of punctuations in the content
- f_{10}^c : the number of HTML tags in the content
- f_{11}^c : the number of tokens starting with capital letter in the content

We assume that each of the above features is normally distributed with mean (μ_k^c) and variance (σ_k^c) for the k -th content feature. In the EM framework, they are calculated as follows:

$$\mu_k^c = \frac{\sum_i^N P(A|C, L, V; \alpha_t, O_t) f_k^c(C)}{\sum_i^N P(A|C, L, V; \alpha_t, O_t)} \quad (4.8)$$

$$\sigma_k^c = \frac{\sum_i^N P(A|C, L, V; \alpha_t, O_t) (f_k^c(C) - \mu_k^c)^2}{\sum_i^N P(A|C, L, V; \alpha_t, O_t)} \quad (4.9)$$

Similarly, we can expand the terms $P(L|A; O)$ and $P(V|A; O)$. However, since the text fragments under consideration do not come with the ontology of the new unseen Web site, $P(L|A; O)$ and $P(V|A; O)$ cannot be found directly. We propose a metric (λ) to estimate $P(L|A; O)$ and $P(V|A; O)$ from $P(L|A)$ and $P(V|A)$. The idea is to model the observation that concepts with a common parent are generally located in a nearby position and their header labels are related. Therefore, $P(L|A; O)$ and $P(V|A; O)$ are expanded as follows:

$$P(L|A; O) = \lambda_l(A, L, O) \prod_{k=1}^{Z_l} P(f_k^l(L)|A) \quad (4.10)$$

$$P(V|A; O) = \lambda_v(A, V, O) \prod_{k=1}^{Z_v} P(f_k^v(V)|A) \quad (4.11)$$

where Z_l and Z_v are the number of features for header label and visual layout respectively. $f_k^l(L)$ and $f_k^v(V)$ are the k -th header label feature and the k -th visual layout feature respectively. λ will be discussed in the later part of this section. Similarly we derive several features related to header labels and the visual layout. Unlike the content features, some of the features derived for header labels and the visual layout such as the color, boldness, and the occurrence of a certain word are discrete. Instead of calculating the mean and variance,

we can compute the $P(f_k^l(L)|A; O)$, $P(f_k^v(V)|A; O)$ for the discrete features as follows:

$$P(f_k^l(L) = \gamma|A) = \frac{1 + \sum_{i=1}^N \#(f_k^l(L) = \gamma) P(A|C, L, V; \alpha_t, O_t)}{|f_k^l| + \sum_{s \in f_k^l} \sum_{i=1}^N \#(f_k^l(L) = s) P(A|C, L, V; \alpha_t, O_t)} \quad (4.12)$$

$$P(f_k^v(V) = \omega|A) = \frac{1 + \sum_{i=1}^N \#(f_k^v(V) = \omega) P(A|C, L, V; \alpha_t, O_t)}{|f_k^v| + \sum_{s \in f_k^v} \sum_{i=1}^N \#(f_k^v(V) = s) P(A|C, L, V; \alpha_t, O_t)} \quad (4.13)$$

where $|f_k^l|$ and $|f_k^v|$ are the number of possible values of the k -th header label feature and the k -th visual layout feature respectively, $\#(f_k^l(L) = \gamma)$ and $\#(f_k^v(V) = \omega)$ are the count of the number of times $f_k^l(L)$ equals to γ and the count of the number of times $f_k^v(V)$ equals to ω respectively.

The term $P(A; \alpha)$ in Equation 3.2 can be estimated by:

$$P(A = \psi; \alpha_t, O_t) = \frac{1 + \sum_{i=1}^N P(A = \psi|C, L, V; \alpha_t, O_t)}{|A| + N} \quad (4.14)$$

where $|A|$ is the number of possible values of A .

Recall that concepts with common parents are likely to have similar tokens in their header labels. The term $\lambda_l(A, L, O)$ in Equation 4.10 is proposed to model this observation. $\lambda_l(A, L, O)$ is defined as follows:

$$\lambda_l(A, L, O) = \frac{1}{L} \max_H \lambda(\tau^{\delta_O(A, H)}, D'(L, L_H)) \quad (4.15)$$

where L is a normalizing factor; H is a concept in the ontology O ; τ is a factor between 0 and 1; L_H is the header label for concept H ; $\delta_O(A, H)$ is the distance between the concept A and H in the ontology O ; and $D'(L, L_H)$ is the modified edit distance defined in Section 5.1; $\lambda(x, y)$ is a function whose image is from 0 to 1 if x and y are between 0 and 1. The definition of $\delta_O(A, H)$ is the minimum depth among A and H in the subtree rooted from their common parent in O . For example, in the ontology depicted in Figure 3, $\delta_O(\text{"you save"}, \text{"our price"})$ is 1, while $\delta_O(\text{"you save"}, \text{"ISBN"})$ is 2. $\lambda(x, y)$ is a function defined as follows:

$$\lambda(x, y) = \frac{2xy}{x^2 + y^2} \quad (4.16)$$

The property of this function is that if the values of x and y are more similar, the value of the function is higher. In essence, λ returns a large value if the two concepts under consideration are close in the ontology and their labels are similar. As $\lambda_l(A, L, O)$ is interpreted as probability, a normalizing factor L is involved in Equation 4.15. $\lambda_v(A, V, O)$ in Equation 4.11 is to model the observation that concepts with common parents are likely to be located in a nearby position. The definition of $\lambda_v(A, V, O)$ is similar to the definition of $\lambda_l(A, L, O)$ shown in Equation 4.15, but replacing the them $D'(L, L_H)$ with the inverse of the distance between the two concepts within the Web page.

5 Adapting Ontology to the New Unseen Site

Our ontology refinement framework mainly consists of three components, namely, existing concepts discovery

component, new concepts discovery component and ontology refinement component. These components are designed based on the EM algorithm described above.

5.1 Discovering Existing Concepts The first component of our framework is to discover existing concepts in the new unseen Web site. This is achieved by calculating $P(A|C, L, V; \alpha, O)$ through our EM algorithm. To initialize the EM algorithm, $P(A|C, L, V; \alpha_0, O_0)$ needs to be computed. Recall that the contents of a concept share similar characteristics in both of the source Web site and the new unseen site. Therefore, we can make use of the content of the existing concepts collected in the source Web sites to obtain the initial estimation of $P(A|C, L, V; \alpha_0, O_0)$ for the same concept in the new unseen site. We design a modified edit distance approach to determine the initial estimation denoted by $P_0(A|C, L, V; \alpha_0, O_0)$. The details of the modified edit distance algorithm can be found in our previous work [19]. Suppose $D(C, l_i^j)$ is the distance between the content C and the i -th entry of the content for the j -th concept collected in the source Web site. $P_0(A|C, L, V; \alpha_0, O_0)$ is then estimated by:

$$P_0(A|C, L, V; \alpha_0, O_0) \sim \frac{1}{K} (\max_i \{D'(C, l_i^j)\}) \quad (5.17)$$

where $D'(C, l_i^j) = 1 - D(C, l_i^j)$ and K is a normalization factor. With this initial estimation, the EM algorithm can iteratively estimate the parameter α until convergence. To discover the existing concepts in the new unseen site, we fix O as the existing ontology from the source Web site. After obtaining the parameters, $P(A|C, L, V; \alpha, O)$ can be calculated according to Equation 3.2. For each of the existing concepts, those informative text fragments with probability of belonging to this concept higher than a threshold will be classified as the corresponding concept in the new unseen site. It is possible that no text fragments in the new unseen site belong to a particular existing concept in the source Web site. This kind of concept will then be removed from the existing ontology.

5.2 Discovering New Concepts As described in the previous section, those informative text fragments which are not classified as any existing concepts will be considered for some new concepts in the unseen site. We attempt to label these new concepts by discovering their header labels from the Web page. We make use of the classified text fragments and their header labels to accomplish this task. The idea is to train a binary classifier to predict if a certain text fragment in the new unseen site is a header label of the new concept.

The header label of a concept is one of the surrounding texts of the text fragments regarding the content of the concept. We can construct a set of training examples by pairing the classified text fragments and each of their surrounding texts. If the surrounding text is

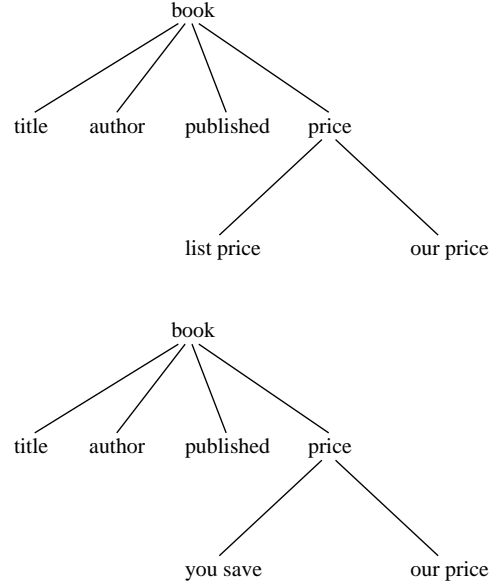


Figure 6: Two samples of potential ontology candidates for the Web site shown in Figure 1.

the header label, this pair becomes the positive training example, otherwise it becomes the negative training example. Next, we can train a binary classifier using this set of training examples. This classifier can predict whether the surrounding text is the header label of a concept, given a text fragment regarding the content of a concept and its surrounding text. Several features are considered in the learning of the classifier. For example, the relative distance between the text fragments and their surrounding text, the characteristics of the surrounding text such as the color, boldness, capitalization are also considered. We employ a naive Bayesian approach and maximum likelihood estimation, in a fashion similar the approach described in Section 4, for this classification task.

Similarly, we can pair the unclassified text fragments generated from Section 5.1 with each of their surrounding texts. Each pair is then classified by the learned classifier to predict the probability that the surrounding text is the header label. The new concepts are then labeled with the discovered header labels. The newly discovered concepts will then be added to the ontology according to the method described in Section 5.3.

5.3 Refining Ontology The newly discovered concepts will be added to the ontology. The location at which the new concept should be placed in the ontology is determined by considering the semantic meaning and the orthographic characteristics of the header labels. We construct a set of potential ontology candidates which form the search space of the parameter O in our framework described in Section 4. Figure 6 shows

two samples of potential ontology candidate for the Web site shown in Figure 1. The ontology of the Web site is then inferred iteratively in the EM algorithm.

We construct the potential ontology candidate using two different methods. One method is to consider the orthographic characteristics of the header labels. As mentioned before, similar concepts are likely to share some similarity in their header labels. For example, “list price” and “our price” are the header labels of two related concepts. We compare the header labels of the concepts. If they have common suffix or common prefix, they will be considered as the subconcepts of a concept labeled with their common suffix or common prefix. Another evidence is to consider semantic meaning of the header labels. For example, the phrase “list price” and the word “save” are semantically related to the concept “price”. They are likely to be the subconcepts of price. To find the semantic relationship between the header labels, we make use of Lexical FreeNet which is a query system for finding the semantic relationship between two phrases⁴ [2]. For example, If querying Lexical FreeNet with the phrases “list price” and “save”, it returns the fact that “price” is the generalization of the phrases “list price” and “save”. Consequently this information is utilized to increase the likelihood that “list price” and “save” are the subconcepts of the concept “price”.

A number of potential ontology candidates are constructed using the two methods described. The idea of ontology refinement is to conduct the EM algorithm described in Section 4 with the following changes in the M-Step:

$$\alpha_{t+1} = \arg \max_{\alpha} L'(\alpha, O; \Lambda, \alpha_t, O_t)$$

$$O_{t+1} = \arg \max_O L'(\alpha, O; \Lambda, \alpha_{t+1}, O_t)$$

After convergence, the parameter O is the potential ontology candidate that maximizes the expected log likelihood function in Equation 4.6. Therefore, this ontology is the most probable ontology that generates the text fragments regarding the concepts in the new unseen site.

6 Identifying Informative Text Fragments

A Web page is made of text fragments. Some text fragments are concerned with the layout format such as HTML tags. Some text fragments are related to the contents of the concepts in an ontology. We call these informative text fragments. We develop a method which can precisely identify the informative text fragments in the new unseen site. The text fragments identified are

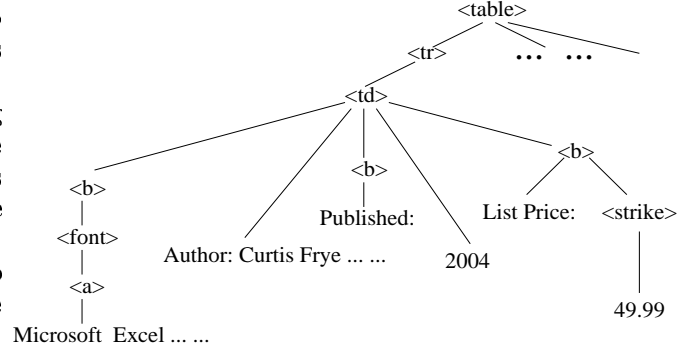


Figure 7: Part of the DOM structure representation for the Web page shown in Figure 1.

then utilized to adapt the ontology to the new site as described in Sections 4 and 5.

The idea of our method is to analyze the Document Object Model (DOM) structure of a Web page. A Web page can be represented by a DOM structure which is essentially an ordered tree consisting of two types of nodes. The first type of node is called element node which is used to represent HTML tag information. These nodes are labeled with the element name such as “<table>”, “<a>”, etc. The other type of node is called text node which includes the text displayed in the browser and labeled simply with the corresponding text. Figure 7 shows part of the DOM structure representation for the Web page shown in Figure 1.

We develop an algorithm that can effectively locate the informative text nodes in the DOM structure. The text fragments within the informative text nodes correspond to the informative text fragments. For each of the text nodes in the DOM structure, we define the path as the string created by concatenating the node labels from the first ancestor to the n -th ancestor where n is a pre-defined value. For example, as shown in Figure 7, the path for the text nodes labeled with “Published:” and “List Price:” are both equal to “<table> <tr> <td> ” and the path for the text nodes labeled with “Microsoft Excel 2003 Programming inside Out” is “<td> <a>” when n is set to 4. Note that each path may locate more than one text node in the DOM structure. We define the probability that the term w_i occurs in the text nodes located by the path p as:

$$P(w_i, p) = \frac{\Gamma(w_i, p)}{\sum_j \Gamma(w_j, p)}$$

where $\Gamma(w_i, p)$ is the number of occurrence of w_i in all the text nodes located by p . Next, we define the *path entropy*, $E(p)$, as follows:

$$E(p) = - \sum_i P(w_i, p) \log P(w_i, p) \quad (6.18)$$

Note that $E(p)$ can be calculated from more than one DOM structure by treating all the DOM structures as a forest and each $P(w_i, p)$ is calculated by considering all the text nodes located by p in the forest. The idea of our algorithm is that the text fragments regarding the con-

⁴The URL of Lexical FreeNet is <http://www.lexfn.com/>.

	Web site (URL)
S1	Half Price Computer Books (http://www.halfpricecomputerbooks.com)
S2	Discount-PCBooks.com (http://www.discount-pcbooks.com)
S3	mmistore.com (http://www.mmistore.com)
S4	Amazon.com (http://www.amazon.com)
S5	1Bookstreet.com (http://www.1bookstreet.com)
S6	Barnes & Noble.com (http://www.barnesandnoble.com)
S7	bookpool.com (http://www.bookpool.com)
S8	half.com (http://half.ebay.com)
S9	DigitalGuru Technical Bookshops (http://www.digitalguru.com)
S10	Canon USA Consumer Product (http://consumer.usa.canon.com)
S11	Kodak (http://www.kodak.com)
S12	Panasonic USA (http://www.panasonic.com)
S13	Olympus America Inc. (http://www.olympusamerica.com/)
S14	Konica Minolta Photo Imaging USA, Inc. (http://www.konica.com)

Table 1: Web sites collected for experiments. S1 - S9 are collected from the book catalog domain. S10 - S14 are collected from the digital camera specification domain.

tent of the concepts are generally different in different Web pages. Therefore, we compute the entropy, which represents the randomness of the term distribution, of a node. If the term distribution in the text nodes is more random, it is likely that this text node contains text fragments regarding the content of the concepts. The details of our algorithm can be found in [18].

7 Case Study

Consider the Web site shown in Figure 1 which is associated with the ontology depicted in Figure 2. In this Web site, we collected a set of informative text fragments regarding the content of the concepts using an automatic information extraction method. For example, we collected the text fragments “Game Programming Gems 2”, “Microsoft Excel 2003 Programming inside Out”, and “Practical C++ Programming, 2nd Edition” for the concept “title”, and the text fragments “Mark Deloura”, “Curtis Frye”, and “Steve Oualline” for the concept “author”.

Although the Web sites in Figures 1 and 3 belong to the same domain, the existing ontology in Figure 2 is not suitable for the Web site in Figure 3. We apply our ontology refinement framework to adapt

the existing ontology from the Web site in Figure 1 to the new Web site in Figure 3. Based on the text fragments regarding the content of the concepts from the source Web site, our framework discovers that the text fragments “Oracle Advance: PL/SQL Programming with CD-ROM”, “Palm OS Programming from the Ground Up”, etc. are the book title names, the text fragments “Scott Urman”, “Robert Mykland”, etc. are the author names in new site. Similarly, it can be discovered that the concepts “you save”, and “our price” are also contained in the new site. There are in total four existing concepts involved in both Web sites and our framework can precisely discover them in the new unseen site.

The discovered existing concepts are retained in the ontology while those concepts such as “published” that cannot be found in the new site are removed. Some of the existing concepts discovered have different header labels in the new unseen site. For example, the concept “author” is associated with the header label “Author”. The concept “you save” is associated with “You Save” in Figure 3. Based on the characteristics of the header labels and the text fragments regarding the content of the concepts, our framework discovers that “ISBN” and “MSRP” are the new concepts in the new site. There are in total two new concepts in Figure 3 and our framework can correctly identify them.

The newly discovered concepts are added to the ontology by the method described in Section 5.3 to form a set of potential ontology candidates. Our framework then iteratively refines the ontology and obtain the one shown in Figure 4. This refined ontology is the same as the manually constructed ontology for the Web site in Figure 3. Therefore starting from the existing ontology in the source Web site, our framework is able to automatically refine the ontology and adapt to a new unseen Web site.

8 Experimental Results

We conducted extensive experiments on several real-world Web sites in two domains, namely book catalog domain and digital camera specification domain, to demonstrate the performance of our ontology refinement framework. Table 1 shows the Web sites used in our experiment. The first column shows the Web site label and the second column shows the name of the Web sites and the corresponding URL. S1 - S9 are Web sites collected from the book catalog domain. S10 - S14 are Web sites collected from the digital camera specification domain.

To evaluate the performance of our ontology refinement framework, we first manually construct the ontology for each Web site. This manually constructed

	S1		S2		S3		S4		S5		S6		S7		S8		S9	
	p	r	p	r	p	r	p	r	p	r	p	r	p	r	p	r	p	r
S1	-	-	1.00	1.00	0.83	1.00	1.00	1.00	1.00	0.80	0.80	0.80	0.80	0.80	0.80	0.80	1.00	1.00
S2	1.00	1.00	-	-	0.80	1.00	0.67	1.00	0.67	0.67	0.75	0.75	1.00	1.00	0.80	1.00	1.00	0.75
S3	0.67	0.80	1.00	1.00	-	-	0.57	1.00	0.83	0.83	1.00	0.67	1.00	0.86	0.75	0.60	1.00	0.80
S4	1.00	1.00	1.00	1.00	0.75	0.75	-	-	1.00	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
S5	0.83	1.00	0.83	0.83	0.67	0.67	0.60	0.75	-	-	1.00	0.80	1.00	0.83	1.00	0.60	1.00	0.75
S6	0.67	1.00	0.71	1.00	1.00	0.71	0.67	1.00	0.83	0.83	-	-	0.83	0.71	0.83	0.83	0.83	0.83
S7	1.00	1.00	1.00	1.00	0.57	0.57	0.60	0.75	0.50	0.50	0.57	0.57	-	-	0.67	0.57	1.00	0.71
S8	1.00	1.00	1.00	1.00	0.80	0.67	0.80	1.00	0.67	0.80	0.75	0.50	0.75	0.50	-	-	1.00	0.60
S9	0.83	1.00	1.00	1.00	0.60	0.50	0.75	0.75	0.50	0.50	0.50	0.33	0.40	0.33	0.80	0.67	-	-

Table 2: Performance of our ontology refinement framework on discovering existing concepts in the new unseen site in the book catalog domain. P and r refer to precision and recall for discovering the existing concepts in the new unseen site respectively.

ontology is considered as the gold standard for evaluation. Next our ontology refinement method was conducted to adapt the ontology from a source Web site to each of the other sites. For example, the ontology from S1 is adapted to S2 to S9 using our ontology refinement method. Each of the resulting ontologies is then compared with the corresponding manually constructed ontology for evaluation. Three aspects are examined in evaluating the performance. The first aspect is to evaluate the capability of discovering the existing concepts in the new unseen site. This is achieved by calculating the recall and precision of the discovery of existing concepts. Recall is calculated by dividing the number of existing concepts for which the system correctly discovered by the total number of actual existing concepts. Precision is calculated by dividing the number of existing concepts for which the system correctly discovered by the total number of existing concepts the system discovered. The second aspect is to evaluate the capability of discovering the new concepts in the new unseen site. This is achieved by calculating the recall and precision of the discovery of new concepts. The third aspect is to evaluate the capability of refining the ontology structure. This is achieved by calculating the tree edit distance between the adapted ontology and the manually constructed ontology. The tree edit distance is defined as the minimum cost of an edit operation sequence that transforms one tree to the other. There are three kinds of edit operations. The first operation is to change the label of a node n . The second operation is to delete a node n , and make its children become the children of the original parent of n . The third operation is to insert a node n as the child of another node m , and make any child become the child of n . We fix the costs of all these edit operations to 1. The smaller the tree edit distance between the two ontologies, the higher their similarity. Readers can refer to [12] for the details of the tree edit distance.

8.1 Evaluation on the Book Catalog Domain

Table 2 shows the results of discovering existing concepts in the new unseen site. The first column shows the Web sites (source sites) from which the ontologies are

given. The first row shows the Web sites (new unseen sites) to which the ontologies are adapted. For example, the row labeled with S1 refers to the set of eight runs where the ontology from S1 is refined to adapt to S2 - S9. Each cell in Table 2 is divided into two sub-columns representing the precision and recall of discovering existing concepts respectively. Our method achieves a very satisfactory performance. In most of the runs, the precision and recall are above 80%. This shows that our framework can effectively discover the existing concepts in the new unseen site.

Table 3 shows the results of discovering new concepts in the new unseen site by adapting the ontology from the source Web site using our framework. The format of the table is similar to that of Table 2. Each cell in Table 3 is divided into two sub-columns representing the precision and recall of discovering new concepts respectively. Some of the cells have a value of "N/A" because there is no new concept in the unseen unseen site. The results show that our framework can discover new concepts in most of the new sites. However, some runs such as the one discovering new concepts in S8 using the ontology from S2 have less satisfactory results. The reason is that some of the concepts are not associated with header labels and hence the new concept cannot be labeled. Nevertheless, our framework can still identify the text fragments regarding the content of new concepts and users can manually interpret the meaning of the new concepts. However, we consider that it fails to discover these new concepts in our experiments.

Table 4 shows the results of comparing the refined ontology with the manually constructed ontology in the new unseen site. The format of the table is similar to that of Table 2. Each cell in Table 4 is divided into two sub-columns representing the edit distance between the adapted ontology and the manually constructed ontologies (ϵ) and the edit distance between the adapted ontology and the manually constructed ontology, normalized by the total number of concepts (ϵ'). Note that the smaller the distance, the better is the performance. The results indicate that our framework achieves a very satisfactory result in refining the structure of the ontology. In most cases, the errors are mainly due to the

	S1		S2		S3		S4		S5		S6		S7		S8		S9	
	p	r	p	r	p	r	p	r	p	r	p	r	p	r	p	r	p	r
S1	-	-	1.00	1.00	0.67	0.67	N/A	N/A	1.00	0.50	1.00	1.00	1.00	0.67	0.80	0.80	1.00	1.00
S2	1.00	1.00	-	-	0.67	0.67	N/A	N/A	0.00	0.00	1.00	0.67	1.00	0.50	0.00	0.00	1.00	1.00
S3	1.00	1.00	1.00	1.00	-	-	N/A	N/A	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
S4	1.00	1.00	1.00	1.00	0.80	0.75	-	-	1.00	0.50	1.00	0.67	1.00	0.50	0.00	0.00	1.00	0.67
S5	1.00	1.00	N/A	N/A	1.00	0.50	N/A	N/A	-	-	1.00	0.50	1.00	0.50	0.00	0.00	1.00	0.67
S6	1.00	1.00	0.50	1.00	1.00	1.00	N/A	N/A	1.00	1.00	-	-	1.00	1.00	N/A	N/A	1.00	1.00
S7	1.00	1.00	1.00	1.00	N/A	N/A	N/A	N/A	1.00	1.00	N/A	N/A	-	-	N/A	N/A	1.00	1.00
S8	1.00	1.00	0.50	1.00	1.00	0.50	N/A	N/A	1.00	0.33	0.00	0.00	1.00	0.50	-	-	1.00	1.00
S9	1.00	1.00	1.00	1.00	0.00	0.00	N/A	N/A	1.00	0.33	0.00	0.00	0.00	0.00	0.80	0.67	-	-

Table 3: Performance of our ontology refinement framework on discovering new concepts in the new unseen site in the book catalog domain. P and r refer to precision and recall for discovering the new concepts in the new unseen site respectively.

	S1		S2		S3		S4		S5		S6		S7		S8		S9	
	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'
S1	-	-	0.00	0.00	2.00	0.25	0.00	0.00	3.00	0.43	2.00	0.25	3.00	0.38	2.00	0.33	0.00	0.00
S2	0.00	0.00	-	-	2.00	0.25	0.00	0.00	4.00	0.57	3.00	0.38	2.00	0.25	2.00	0.33	1.00	0.14
S3	0.00	0.00	0.00	0.00	-	-	0.00	0.00	2.00	0.29	3.00	0.38	2.00	0.25	2.00	0.33	1.00	0.14
S4	0.00	0.00	0.00	0.00	2.00	0.25	-	-	3.00	0.43	3.00	0.38	3.00	0.38	2.00	0.33	1.00	0.14
S5	0.00	0.00	0.00	0.00	3.00	0.38	1.00	0.20	-	-	3.00	0.38	3.00	0.38	3.00	0.50	2.00	0.29
S6	0.00	0.00	0.00	0.00	4.00	0.50	0.00	0.00	2.00	0.29	-	-	3.00	0.38	1.00	0.17	1.00	0.14
S7	0.00	0.00	0.00	0.00	3.00	0.38	1.00	0.20	4.00	0.57	4.00	0.50	-	-	2.00	0.33	2.00	0.29
S8	0.00	0.00	0.00	0.00	4.00	0.50	0.00	0.00	4.00	0.57	5.00	0.63	5.00	0.63	-	-	2.00	0.29
S9	0.00	0.00	0.00	0.00	3.00	0.38	1.00	0.20	5.00	0.71	6.00	0.75	5.00	0.63	2.00	0.33	-	-

Table 4: Performance of our ontology refinement framework on refining the structure of the ontology in the book catalog domain. ϵ refers to the tree edit distance between the refined ontology and the manually constructed ontology in the new unseen site. ϵ' refers to the tree edit distance between the refined ontology and the manually constructed ontology normalized by the total number of concept in the new unseen site. (Note that the smaller the distance, the better is the performance.)

	S10		S11		S12		S13		S14	
	p	r	p	r	p	r	p	r	p	r
S10	-	-	1.00	0.85	1.00	1.00	1.00	1.00	1.00	1.00
S11	1.00	1.00	-	-	1.00	1.00	1.00	1.00	1.00	1.00
S12	1.00	1.00	1.00	0.93	-	-	1.00	1.00	1.00	0.83
S13	1.00	1.00	1.00	0.93	1.00	-	-	1.00	1.00	0.83
S14	1.00	1.00	1.00	0.93	1.00	1.00	1.00	1.00	-	-

Table 5: Performance of our ontology refinement framework on discovering existing concepts in the new unseen site in the digital camera specification domain. P and r refer to precision and recall for discovering the existing concepts in the new unseen site respectively.

undiscovered concepts in the previous stage.

	S10		S11		S12		S13		S14	
	p	r	p	r	p	r	p	r	p	r
S10	-	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.74
S11	1.00	1.00	-	-	1.00	1.00	1.00	1.00	1.00	0.64
S12	1.00	1.00	1.00	0.75	-	-	1.00	1.00	1.00	0.78
S13	1.00	1.00	1.00	0.81	1.00	-	-	1.00	1.00	0.71
S14	1.00	1.00	1.00	0.79	1.00	1.00	1.00	1.00	-	-

Table 6: Performance of our ontology refinement framework on discovering new concepts in the new unseen site in the digital camera specification domain. P and r refer to precision and recall for discovering the new concepts in the new unseen site respectively.

8.2 Evaluation on the Digital Camera Specification Domain We conducted experiments in the digital camera specification domain similar to the experiments described in Section 8.1. Unlike the book catalog domain, in which most of the Web pages contain more than one book records, each Web page in this domain contains only one record and each record consists of more than thirty concepts. Tables 5 and 6 show the results of discovering existing concepts and new concepts in the new unseen site respectively. The format

	S10		S11		S12		S13		S14	
	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'	ϵ	ϵ'
S10	-	-	4.00	0.08	2.00	0.03	6.00	0.10	8.00	0.11
S11	0.00	0.00	-	-	4.00	0.06	8.00	0.13	15.00	0.20
S12	0.00	0.00	9.00	0.20	-	-	6.00	0.10	10.00	0.14
S13	0.00	0.00	7.00	0.16	6.00	0.09	-	-	13.00	0.18
S14	0.00	0.00	9.00	0.20	6.00	0.09	6.00	0.10	-	-

Table 7: Performance of our ontology refinement framework on refining the structure of the ontology in the digital camera specification domain. ϵ refers to the tree edit distance between the refined ontology and the manually constructed ontology in the new unseen site. ϵ' refers to the tree edit distance between the refined ontology and the manually constructed ontology normalized by the total number of concept in the new unseen site. (Note that the smaller the distance, the better is the performance.)

of the tables are the same as that of Tables 2 and 3 respectively. Our framework achieves a very good performance in discovering concepts in this domain. The main reason is that the data are organized in a more rigid table format in the Web pages and all the concepts have their header labels. Our framework can exploit such kind of regularity to discover the concepts. Table 7 shows the results of comparing the refined ontology with the manually constructed ontology in the new unseen site. The format of this table is same as that Table 4. The results indicate that our framework achieves a very promising result in ontology refinement. The header labels and the visual layout of the concepts provide very useful information to refine the ontology. For example, the header labels “LCD Monitor”, “LCD Pixels”, and “LCD Coverage” in S10 indicate the relationship of the three concepts. These three concepts are also located in three consecutive rows. Our framework can effectively refine the ontology from these clues.

9 Conclusions and Future Work

We have developed a probabilistic framework for refining an existing ontology from a source Web site to new unseen sites. Several clues are considered in our framework. The first clue is the text fragments regarding the content of the concepts extracted in the source Web site. The second clue is the text fragments regarding the header labels of the concepts. The third clue is the visual layout of the text fragments regarding the content and the header labels of the concepts. To cope with the uncertainty involved in these clues, we design a generative model representing the generation of text fragments regarding the concepts and the ontology corresponding to the Web page. We employ Bayesian learning techniques and expectation-maximization algorithm to achieve the goal. We have conducted extensive experiments to demonstrate the performance of our approach.

We intend to extend our framework in several directions. One possible direction is to incorporate the domain knowledge of the users. Sometimes, users have some knowledge about the ontology such as some constraints between concepts. Such domain knowledge is useful in the refinement task. Another possible direction is to integrate our refined ontology into other application such as information retrieval systems. The refined ontology may contain errors and therefore a more sophisticated approach is needed.

References

- [1] R. Altman, M. Bada, X. Chai, M. Carillo, R. Chen, and N. Abernethy. Riboweb: An ontology-based system for collaborative molecular biology. *IEEE Transactions on Intelligent Systems*, 14(5):68–76, 1999.
- [2] D. Beeferman. Lexical discovery with an enriched semantic network. In *Proceedings of the Workshop on Applications of WordNet in Natural Language Processing Systems, ACL/COLING 1998*, 1998.
- [3] W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 232–241, 2002.
- [4] A. Dempster, A. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- [5] A. Doan, J. Madhavan, R. Dhamanker, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.
- [6] M. Eirinaki, M. Vazigiannis, and I. Varlamis. SEWeP: Using site semantics and a taxonomy to enhance the web personalization process. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–107, 2003.
- [7] N. Kushmerick and B. Thomas. Adaptive information extraction: Core technologies for information agents. In *Intelligent Information Agents R&D In Europe: An AgentLink Perspective*, pages 79–103, 2002.
- [8] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302, 2003.
- [9] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
- [10] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, Inc., 1997.
- [11] R. Navigli, P. Velardi, and A. Gangemi. Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, 18(1):22–31, 2003.
- [12] D. Shasha and K. Zhang. *Pattern Matching Algorithms*. Oxford University Press, 1997.
- [13] R. Stevens, C. Goble, I. Horrocks, and S. Bechhofer. OILing the way to machine understandable bioinformatics resources. *IEEE Transactions on Information Technology in Biomedicine*, 6(2):129–134, 2002.
- [14] The Gene Ontology Consortium. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.
- [15] Y. Tijerino, D. Embley, D. Lonsdale, and G. Nagy. Ontology generation from tables. In *Proceedings of the Forth International Conference on Web Information Systems Engineering*, pages 242–249, 2003.
- [16] P. van der Vet and N. Mars. Bottom-up construction of ontologies. *IEEE Transaction on Knowledge and Data Engineering*, 10(4):513–525, 1998.
- [17] T. L. Wong and W. Lam. A probabilistic approach for adapting information extraction wrappers and discovering new attributes. In *Proceedings of the 2004 IEEE International Conference on Data Mining*, pages 257–264, 2004.
- [18] T. L. Wong and W. Lam. A probabilistic approach for adapting wrappers and discovering new attributes. In *The Chinese University of Hong Kong, Department of Systems Engineering and Engineering Management Technical Report*, 2004.
- [19] T. L. Wong and W. Lam. Text mining from site invariant and dependent features for information extraction knowledge adaptation. In *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM-2004)*, pages 45–56, 2004.
- [20] World Wide Web Consortium (W3C). Semantic web. In <http://www.w3.org/2001/sw/>, 2001.

Exploiting Parameter Related Domain Knowledge for Learning in Graphical Models

Radu S. Niculescu and Tom M. Mitchell
Dept of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{stefann,tom.mitchell}@cs.cmu.edu

R. Bharat Rao
Clinical CAD
Siemens Medical Solutions
Malvern, PA 19355
bharat.rao@siemens.com

Abstract

Building accurate models from a small amount of available training data can sometimes prove to be a great challenge. Expert domain knowledge can often be used to alleviate this burden. Parameter Sharing is one such important form of domain knowledge. Graphical models like HMMs, DBNs and Module Networks use different forms of Parameter Sharing to reduce the variance in the parameter estimates. The goal of this paper is to present a theoretical approach for learning in presence of several other types of Parameter Related Domain Knowledge that go beyond the ones in the above models. First, we introduce a General Parameter Sharing Framework that describes the models just mentioned, but allows for much finer grained parameter sharing assumptions. In this framework, we present sound procedures for parameter learning from both a Frequentist and a Bayesian point of view, from both complete and incomplete data, in the case where a domain expert specifies in advance the structure of the graphical model, and the subsets of parameters to be shared. Second, we describe a hierarchical extension of this framework based on Parameter Sharing Trees. Finally we present algorithms for using domain knowledge that specifies that certain groups of parameters share certain properties. In particular, we consider two kinds of constraints: first kind states certain groups of parameters share the same aggregate probability mass and second kind states the ratio of the parameters is preserved (shared) in several groups. As an example, we derive a novel form of parameter sharing for Bayesian Multinetworks.

1 Introduction

The task of learning models for many real-world problems requires researchers to incorporate problem Domain Knowledge into the learning algorithm – there is rarely enough training data to enable the learning of the structures and underlying relationships in the problem. Domain Knowledge comes in many forms: Domain Knowledge about relevance of variables, also called Fea-

ture Selection, can help us ignore certain variables when building our model. Domain Knowledge specifying conditional independencies among variables can guide our search over possible model structures. Both these forms have been extensively studied.

This paper presents a theoretical approach for incorporating a different kind of knowledge into learning: Domain Knowledge about relationships among parameters, specifically knowledge about sharing properties of groups of parameters across several conditional probability distributions (CPDs) in graphical models parameterized by conditional probability tables (CPTs). It is somewhat obvious that leveraging such information can greatly benefit learning, as proved by graphical models like HMMs, DBNs and Module Networks. For instance, if a parameter is shared across multiple subproblems, the variance in the estimates can be reduced by learning from all the relevant data from the subproblems, as opposed to estimating the parameter independently for each subproblem.

In Section 3, we describe a *General Parameter Sharing Framework* for making very fine-grained parameter sharing assumptions that go beyond the ones specified in the above mentioned models, and present sound algorithms for learning, in many different scenarios. In this framework, the CPDs in the model are partitioned and a parameter can either be shared among all CPDs in a subset or not shared at all. Some formal guarantees are provided about the derived estimators. We also discuss a scoring metric for comparing different sets of sharing assumptions and suggest how this metric can be used for automatically recovering relevant parameter sharing, provided that the expert does specify only a limited amount of domain knowledge. Section 4 describes a hierarchical extension of the previous framework, where we allow some parameters to be shared across all CPDs, then partition recursively the set of CPDs, allowing new parameters to be shared among subsets (of CPDs) of the

partition. In section 5 we discuss further extensions that investigate sharing properties of groups of more than one parameter across several CPDs. In particular, we consider two kinds of constraints: first kind states certain groups (each group belonging to a different CPD) of parameters share the same aggregate probability mass and second kind states the ratio of the parameters is preserved (shared) in several such groups. Section 6 shows an example that demonstrates that graphical models learned using parameter sharing, are preferred to the alternatives of sharing no parameters or using a global model that effectively shares all parameters on a task of email modelling using Bayesian Multinets. We plan to apply our work to other more complex real-world problems in the near future. The final section summarizes the main contributions of this paper and suggests some directions for future work. We begin by briefly reviewing some related research.

2 Related Work

The standard way of representing parameter related domain knowledge in graphical models is by using Dirichlet priors. It can be proved [7], that under certain conditions, choosing Dirichlet priors over the parameters of a Bayesian Network is inevitable. However, Dirichlet priors are specified over a CPD(Conditional Probability Distribution) in a CPT (Conditional Probability Table) and therefore do not represent constraints among parameters in different CPDs. An approach of relaxing the local independence assumption for binary variables is investigated in [8]. In [13], the authors compare several other smoothing methods for learning language models.

Parameter sharing has also been explored in a variety of graphical models. For example, HMMs [10] and Dynamic Bayes Nets [9] make the assumption that the CPTs (Conditional Probability Tables) corresponding to a given variable are the same at each point in time. In Probabilistic Relational Models [5], objects of a certain type class share the way they depend on related objects of other type classes. In Module Networks [12], variables in the same module exhibit similar behavior i.e. they have the same set of parents, same set of values and probabilistically they depend on their parents in the same way and therefore their corresponding CPTs can be learned together. Context Specific Independence (CSI) [1] states conditional independencies that hold only in certain contexts i.e. of the form $P[X|Y, Z, C = c] = P[X|Z, C = c]$ and therefore can specify that some CPDs in the CPT for a variable X are the same (they share all parameters in those CPDs). CSI can be exploited to efficiently encode and learn CPTs using default tables, decision trees and decision graphs as described in [3] and [4].

As an example, in section 6 we will illustrate the utility of our General Parameter Sharing Framework by showing how it allows a novel kind of parameter sharing in Bayesian Multinets [2],[6] for the task of modelling emails coming from several users.

3 A General Parameter Sharing Framework

We present a General Parameter Sharing Framework that describes learning in a broad category of graphical models. We show how to approach learning within this framework from both a Frequentist and a Bayesian point of view, from both complete and incomplete data, in the case where a domain expert specifies in advance the structure of the graphical model, and an arbitrary subset of parameters to be shared.

Each CPD (Conditional Probability Distribution) represents the constraint that its parameters should sum up to one. We denote by C the set of all CPDs in all CPT's in the graphical model. Below we will describe two assumptions that suffice to derive learning procedures that take advantage of Parameter Sharing:

Parameter Sharing. The CPD set C can be partitioned as $C = \cup_{k=1}^m C_k$, such that some parameters (denote their set by G_k) are shared (appear exactly once in each of the different CPDs) within the set C_k , but not shared within the same CPD or with other sets of CPDs of the partition. Let $L_k = C_k \setminus G_k$ be the set of local (not shared) parameters.

Decomposability of Log-Likelihood. For any complete dataset D , the log-likelihood can be decomposed as:

$$\log(P(D|\theta)) = \sum_{k=1}^m h_k(C_k)$$

$$\text{where } h_k(C_k) = \sum_{\theta_{gk} \in G_k} N_{gk} \cdot \log \theta_{gk} + \sum_{\theta_{lck} \in L_k, c \in C_k} N_{lck} \cdot \log \theta_{lck}$$

Above, N_{gk} represents the cumulative count corresponding to shared parameter θ_{gk} (appears in each CPD in C_k exactly once) and N_{lck} is the count corresponding to local parameter θ_{lck} (in CPD $c \in C_k$).

We now discuss several graphical models which all fit within our framework and satisfy the *Parameter Sharing* assumption. For instance, in HMMs and Dynamic Bayes Nets, the same variable has the same CPT at different time instants. Therefore, a subset C_k of the partition is made out of the CPDs in the CPTs which correspond to a given variable X and the same instantiation of the parents $PA(X) = pa$ across all time instants. In Module Networks, all variables in the same module share the same set of parents and have the same CPTs. Consequently, the subsets of the partition

contain the set of CPDs corresponding to all variables in a module for a given instantiation of the parents of those variables. Context Specific Independence is used to specify conditional independencies that hold in certain contexts and therefore is useful to specify which CPDs should be equal in a CPT for a fixed random variable. In this case, the subsets of the partition consist of CPDs in the same CPT which are assumed equal because of the CSI assumptions. However, note that our framework allows for much more flexibility in parameter sharing. We can share at the level of each parameter, not just at the whole CPD or table level. Also, the CPDs in the same sets of partition do not have to be the same size. Moreover, a shared parameter doesn't need to be in the same position in different CPDs within the same set of the partition.

Below we discuss different scenarios in our framework and provide learning algorithms.

3.1 Learning - Frequentist Approach, Full Data Observability, Known Structure. A Frequentist tries to learn one single model that "best" fits the data. When structure is known in advance, this often translates into finding the Maximum Likelihood Estimators (MLEs) for the parameters in the model. Subsequently, we are also going to discuss Maximum A Posteriori (MAP) Estimators. The MLEs in our General Parameter Sharing Framework, when structure and parameter sharing is specified by a domain expert, can be computed using the following theorem:

THEOREM 3.1. *Assume we are given a graphical model with known structure that satisfies the Parameter Sharing and Decomposability of Log-Likelihood assumptions. Then, the Maximum Likelihood Estimators (MLEs) for the parameters in our graphical model are given by:*

$$\hat{\theta}_{gk} = \frac{N_{gk}}{\sum_{\theta_{g'k} \in G_k} N_{g'k} + \sum_{\theta_{lck} \in L_k} N_{lck}}$$

$$\hat{\theta}_{lck} = \frac{\sum_{\theta_{lck'} \in L_k} N_{lck'}}{\sum_{\theta_{g'k} \in G_k} N_{g'k} + \sum_{\theta_{lck'} \in L_k} N_{lck'}} \cdot \frac{N_{lck}}{\sum_{\theta_{lck} \in L_k} N_{lck}}$$

The MLEs for shared parameters look similar to the ones in the case of standard Bayes Nets. However, the MLEs for local parameters are a product of two factors. First factor represents the probability mass that remains after subtracting the shared parameters. The second factor basically says that this remaining "local" probability mass in a CPD is split into values proportional to the counts corresponding to the local parameters in that CPD.

Proof of Theorem 3.1 (Sketch): Because of the *Parameter Sharing* and *Decomposability of Log-Likelihood* assumptions, the problem of maximizing the data likelihood can be broken down into a set of independent optimization subproblems:

$$P_k : \operatorname{argmax}_{C_k} \{h_k(C_k) \mid g_{ck}(C_k) = 0, \forall c \in C_k\}$$

where

$$g_{ck}(C_k) = (\sum_{\theta_{gk} \in G_k} \theta_{gk}) + (\sum_{\theta_{lck} \in L_k} \theta_{lck}) - 1 = 0$$

When all counts are positive, it can be easily proved that P_k has a global maximum which is achieved in the interior of the region determined by the constraints. In this case the solution of P_k can be found using Lagrange Multipliers. Introduce Lagrange Multipliers $\lambda_k = (\lambda_{ck})_{c \in C_k}$ for each CPD in C_k . Let $LM(C_k, \lambda_k) = h_k(C_k) - \sum_{c \in C_k} \lambda_{ck} \cdot g_{ck}(C_k)$. Then the point which maximizes P_k is among the solutions of the system $\nabla LM(C_k, \lambda_k) = 0$. It turns out that this system has a unique solution which is in fact the one in the statement of the theorem. Since the function achieves its maximum in the interior of the constraint region, it means that the solution of the system supplies the MLEs. In the case when some of the counts are zero, the corresponding parameters don't even appear in the likelihood function and the optimization problem then has inequality constraints but eventually the MLEs are given by the same formulas. This concludes the sketch of our proof. \square

Note that, if the expert makes a mistake about the structure or about the shared parameters of the model, then the learned distribution may be much different from the true distribution of the data. Below we will provide formal guarantees about our estimators.

In order to present these results, let us introduce the notion of *True Probabilistic Counts (TPC)*. Suppose P is the true distribution of which data is sampled. If θ_{lck} is the local parameter of the graphical model that is supposed to describe $P(X = x \mid PA(X) = pa)$, then let $TPC_{lck} = P(X = x, PA(X) = pa)$. If θ_{gk} is the global parameter of the graphical model that is supposed to describe the set $\{P(X_1 = x_1 \mid PA(X_1) = pa_1), \dots, P(X_s = x_s \mid PA(X_s) = pa_s)\}$, let $TPC_{gk} = \sum_{i=1}^s P(X_i = x_i, PA(X_i) = pa_i)$. Let P^* be the distribution that factorizes according to the structure provided by the expert and has parameters given by theorem 3.1 where the counts are replaced by the *True Probabilistic Counts*.

THEOREM 3.2. *P^* is the closest distribution to P (in terms of $KL(P, \cdot)$) that factorizes according to the given structure and obeys the expert's parameter sharing assumptions.*

Proof of Theorem 3.2 (Sketch): Let Q be such a distribution. Minimizing $K(P, Q)$ is equivalent to maximizing $\sum_d P(d) \cdot \log Q(d)$. Let θ be the set of parameters that describe this distribution Q . After breaking the logarithms into products of logarithms based on the factorization given by the provided structure, our optimization problem reduces to the maximization of $\sum TPC_{gk} \cdot \log \theta_{gk} + \sum TPC_{lck} \cdot \log \theta_{lck}$. The solution of this problem is obviously given by theorem 3.1. This is equivalent to the fact that P^* (see the definition above) minimizes $KL(P, \cdot)$ out of all the distributions that factorize according to the given structure and obey the expert's sharing assumptions. \square

THEOREM 3.3. *With infinite amount of data, the distribution \hat{P} given by the ML Estimators in Theorem 3.1 converges to P^* with probability 1.*

Proof of Theorem 3.3 (Sketch): Assume the number of data points in a dataset sampled from P is denoted by n . According to the Law of Large Numbers, we have $\lim_{n \rightarrow \infty} \frac{N_{lck}}{n} = TPC_{lck}$ and $\lim_{n \rightarrow \infty} \frac{N_{gk}}{n} = TPC_{gk}$ with probability 1. This is equivalent to the fact the \hat{P} converges to P^* with probability 1. \square

3.2 Learning - Frequentist Approach, Partial Data Observability, Known Structure. If some of the attributes of the data points are not observed, then the counts N_{gk} and N_{lck} should be treated as random variables (random counts). One still can do Maximum Likelihood parameter estimation using the 2-step EM algorithm. Given the formula of the log-likelihood for complete data and the parameter sharing assumptions, by completing the data in all possible ways, it is easy to see that the standard EM algorithm applied to our setting will repeat the following steps until convergence is reached:

E-Step: Use any inference algorithm to compute expected counts under the current parameter estimates $\hat{\theta}$. If just starting, assign $\hat{\theta}$ randomly or according to some domain knowledge.

M-Step: Reestimate the parameters by maximizing the data likelihood using Theorem 3.1, assuming that the observed counts are equal to the expected counts given by the E Step.

3.3 Learning - Bayesian Approach, Full Data Observability, Known Structure. From a Bayesian point of view, each choice of parameters is possible, but some choices have higher probability of occurring. Therefore, to do model averaging, we need to specify priors over the space of parameters. It can

be proved [7], that under certain conditions, choosing Dirichlet Priors over the parameters of a Bayesian Network is inevitable. Two of the assumptions that are usually made are local and global independence. Because of the *Parameter Sharing* assumption, the local and global independence assumptions may not hold. However, in our case we can make the following similar assumption:

Subset Independence Assumption: parameters in different subsets of the partition are independent of each other. In other words, $p(\theta) = p(C) = \prod_{k=1}^m p(C_k)$.

Under this assumption it is enough to define a prior over the parameters in each subset of the partition. A *Subset Specific Dirichlet Distribution* over C_k of parameters $\{\alpha_{gk} | gk \in G_k\} \cup \{\alpha_{lck} | lck \in L_k\}$ will have the following formula:

$$p(C_k) = \frac{1}{Z_k} \cdot \prod_{\theta_{gk} \in G_k} \theta_{gk}^{\alpha_{gk}-1} \cdot \prod_{\theta_{lck} \in L_k, c \in C_k} \theta_{lck}^{\alpha_{lck}-1}$$

$$SP_k = \left\{ \sum_{\theta_{gk} \in G_k} \theta_{gk} + \sum_{\theta_{lck} \in L_k} \theta_{lck} = 1 \quad \forall c \in C_k \right\}$$

Above, Z_k is a normalization constant which can be found by enforcing the condition that the integral of the pdf should be equal to one:

$$Z_k = \prod_{gk \in G_k} \Gamma(\alpha_{gk}) \cdot \prod_{c \in C_k} \frac{\prod_{lck \in L_k} \Gamma(\alpha_{lck})}{\Gamma(\sum_{lck \in L_k} \alpha_{lck})} \cdot \frac{\Gamma(\sum_{lck \in L_k, c \in C_k} \alpha_{lck} - |C_k| + 1)}{\Gamma(\sum_{lck \in L_k, c \in C_k} \alpha_{lck} + \sum_{gk \in G_k} \alpha_{gk} - |C_k| + 1)}$$

There are several interesting properties of this Subset Specific Dirichlet Distribution. First, the joint probability distribution over the shared parameters is a standard Dirichlet. Second, with no parameter sharing, this distribution is a product of independent standard Dirichlet distributions, one for each CPD in C_k . However, if there are both shared and local parameters, then the joint probability (obtained by marginalization) over a CPD $c \in C_k$ is not a standard Dirichlet. Finally, because of the *Decomposability of Log-Likelihood*, it is easy to see that the posterior $p(\theta|D) \propto p(D|\theta) \cdot p(\theta)$ is also a product of Subset Specific Dirichlet distributions and therefore the collection of multinomials with shared parameters (over C_k) and the Subset Specific Dirichlet distribution are conjugate distributions.

Now that we have defined priors over the space of parameters, it is trivial to compute MAP Estimators.

Because of *Decomposability of Log-Likelihood*, computing MAP Estimators is just a matter of adding corresponding Subset Specific Dirichlet exponents (not parameters) to the observed counts in the ML Estimators in (1) and (2).

From a Bayesian point of view, we are interested in predicting the next data point given previous data points. This can be written as follows:

$$p(D_{n+1}|D_1, \dots, D_n) = \frac{\int_{\bigcup_{SP_k} p(D_{n+1}, \dots, D_1|\theta) \cdot p(\theta) d\theta}}{\int_{\bigcup_{SP_k} p(D_n, \dots, D_1|\theta) \cdot p(\theta) d\theta}}$$

As stated above, $p(\theta|D) \propto p(D|\theta) \cdot p(\theta)$ is also a product of Subset Specific Dirichlet distributions. Therefore both integrals in (4) are products of normalization constants for some Subset Specific Dependent Dirichlet distributions. We already showed that we can compute these normalization constants and thus we can easily compute these integrals. Looking again at the formula for Z_k , it is worth mentioning that most of the factors will cancel out when computing the ratio of the two integrals. The only ones remaining would be the ones where the introduction of the new data point D_{n+1} would increment the counts.

3.4 Learning - Bayesian Approach, Partial Data Observability, Known Structure. When data is incomplete, we can no longer write $p(D|\theta)$ as a product of Subset Specific Dirichlet Distributions as we did in the complete data case discussed above. In this case, let U be the set of missing values such that $D \cup U$ is a complete dataset. Then, $\int p(D|\theta) \cdot p(\theta) d\theta = \sum_U \int p(D, U|\theta) \cdot p(\theta) d\theta$. Therefore, in the case of incomplete data, $\int p(D|\theta) \cdot p(\theta) d\theta$ is a sum of products of normalization constants for certain Subset Specific Dirichlet Distributions and can be used to compute the ratio in (4).

The above procedure for performing Bayesian Estimation is computationally expensive because the number of terms in the summation grows extremely fast. If only one binary value is missing in each of the n examples, there will be 2^n terms in the summation. Approximation techniques for $p(D|\theta)$ when data is incomplete are available for standard Bayes Nets, but investigating them in our parameter sharing framework is beyond the scope of this paper.

3.5 Comparing Parameter Sharing Schemes.

A *Parameter Sharing Scheme PShS* over a graphical model structure S is a set of valid parameter sharing assumptions (of the type specified in the general parameter sharing framework) on top of structure S . We

remind the reader that in all our work, the structure of the model is given by an expert and does not change. Learning structure in presence of parameter sharing is subject for future work.

A *PShS* helps reduce the variance in parameters' estimates. In section 6, we empirically show that this translates in estimated distributions closer in KL distance to the true underlying distribution than the ones estimated without taking advantage of parameter sharing. There we also show that the more valid parameter sharing assumptions we know, the better the estimates. Therefore it is important to recover these assumptions, even when the expert doesn't know the parameter sharing or can specify only a limited set of such assumptions.

Assume that a dataset D of examples is provided and our goal is to learn an *optimal PShS* over a given structure S . In order to do this, we need to be able to compare different PShSs. We propose a metric similar to the one used for structure search i.e. we try to find the *PShS* that maximizes $P(D|PShS)$ (*Sharing Score*). Averaging over all sets of parameters θ consistent with *PShS*, we obtain:

$$p(D|PShS) = \int_{\theta \in PShS} p(D|\theta, PShS) \cdot p(\theta|PShS) d\theta$$

It is easy to notice that the quantity inside the integral represents a product of Subset Specific Dirichlet Distributions and therefore $P(D|PShS)$ is a product of normalization constants for such distributions. We previously showed how such constants can be computed in the case of complete data. Therefore, in the case of complete data, it is straightforward how to obtain the *Sharing Score*.

What happens in the case of incomplete data is a little bit more complicated. In this situation we can write: $p(D|\theta) = \sum_{D' \text{ complete, consistent with } D} p(D'|\theta)$. Therefore, $P(D|PShS)$ will be a sum of products of normalization constants for Subset Specific Dirichlet Distributions. It is worth pointing out that the number of such products grows at least exponentially with the amount of missing data. However, one can think of techniques to approximate $P(D|\theta)$ with Subset Specific Dirichlet Distributions by processing data points one at a time and updating the current parameter prior accordingly. In this case, the order in which examples are processed matters. Investigating this is however beyond the scope of this paper.

We suggest that the above scoring metric can be used to uncover the underlying *PShS* via some hill climbing techniques i.e. one can think of deriving the current *PShS* candidate from previous one by using small modifications. Also, restricting the set

of potential *PShS* can be useful. For example, in module networks one can restrict the variables that can belong together in a module or in HMMs one would specify that one expects only transition tables to have sharing, but no sharing between transition tables and the tables describing the observed output. In addition, our suggested method can be useful when combined with an expert who knows a subset of sharing assumptions because this restricts further the superset of valid parameter sharing schemes.

4 Hierarchical Parameter Sharing Framework

Here we present a hierarchical extension of the framework in the previous section. This will address some of the limitations of the constraints that could be incorporated in the parameter sharing framework described before. In order to derive our main results in this section, we first need to make an important assumption about the graphical models we are working on:

Log-Likelihood Assumption. For any complete dataset of examples D , the log-likelihood can be written as: $\log(P(D|\theta)) = \sum_{\theta_i} N_{\theta_i} \cdot \log \theta_i$ where N_{θ_i} represents the cumulative observed count for parameter θ_i (which may appear in multiple places in the graphical model).

We are now ready to describe our learning framework. First of all, we present Parameter Sharing Trees as a way to encode hierarchical parameter sharing assumptions and second we show how one can take advantage of such a Parameter Sharing Tree in order to alleviate learning. We are going to discuss only the derivation of ML and MAP Estimates from complete data. Same as in the previous section, the EM adaptation for learning from incomplete data is just a matter of estimating the expected counts under current parameter estimates using any available inference algorithm, then reestimate the parameters using the expected counts as if they were observed counts. Also, the Bayesian approach for learning from complete and from incomplete data goes along the same lines with the discussion in previous section.

4.1 Parameter Sharing Trees. Again, let C represent the set of all CPDs in the graphical model. Each such CPD introduces a constraint on the possible values that the parameters θ can take. A *Parameter Sharing Tree (PST)* is a tree with the following properties:

- Each node v of the tree consists of a pair $(Scope(v), Shared(v))$, where $Scope(v)$ is a subset of C and $Shared(v)$ represents a non-empty set of parameters that are shared across these CPDs.

A parameter from $Shared(v)$ is a parameter that is known to appear exactly once in each of the CPDs in $Scope(v)$, but it is not shared multiple times within one CPD, nor with CPDs outside the $Scope$.

- By convention, $Scope(Root) = C$ (this amounts to the fact that we would like to allow for the extreme situation when a parameter is shared by all CPDs in the graphical model).
- The *Scopes* of the direct descendants of a node v form a partition of $Scope(v)$. Therefore, the PST will describe a recursive way of partitioning C , with the leaf level being the finest grain of such partition.
- A parameter cannot be shared in multiple places (different nodes of the tree). Because of the recursive partitioning of the CPDs, this amounts to the fact that $Shared(v)$ is disjoint with all *Shares* on the path from v to the root of *PST*.
- Each parameter θ of the graphical model is shared exactly once i.e. there exists a node v such that $\theta \in Shared(v)$. One may argue that there are parameters which are not shared at all, but for all nodes v that have CPDs in their *Scope* such that there remain unshared parameters, one can partition those nodes further in leaves that have only one CPD in their *Scope*, for which previously unshared parameters become shared at the level of that single CPD.

Let $Desc(v)$ be the set of descendants of node v (included) in and $Shared(Desc(v)) = \bigcup_{v' \in Desc(v)} Shared(v')$. Also, denote by $Ancestors(v)$ the set of nodes on the path from v to the root of the tree and $Shared(Ancestors(v)) = \bigcup_{v' \in Ancestors(v)} Shared(v')$.

4.2 Learning - Frequentist Approach, Full Data Observability, Known Structure. Assume we are given a graphical model with known structure that satisfies a set of parameter sharing assumptions given by a *PST* T and also satisfies the *Log-Likelihood Assumption*. In this section we are going to present a theorem that will justify an algorithm for finding the Maximum Likelihood Estimators for the parameters in such a graphical model. Denote by $\hat{\theta}$ the Maximum Likelihood Estimators.

THEOREM 4.1. *Let v be a node of T and $\theta_i \in Shared(v)$. The following equality holds:*

$$\hat{\theta}_i = \left(1 - \sum_{\theta_j \in Shared(Ancestors(v))} \hat{\theta}_j \right) \cdot$$

$$\frac{N_{\theta_i}}{\sum_{\theta_k \in \text{Shared}(\text{Desc}(v))} N_{\theta_k}}$$

Proof of Theorem 4.1 (Sketch): By definition,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \{ \sum_{\theta_i} N_{\theta_i} \cdot \log \theta_i \mid \theta \text{ satisfies } T \}$$

Each CPD represents a constraint on the space of parameters: $g_c(\theta) = (\sum_{\theta_j \in c} \theta_j) - 1 = 0$. Because of parameter sharing, these constraints can involve some common variables. It is easy to show that, if all the cumulative counts are positive, the likelihood function has a global maximum inside the region determined by the constraints in T . In the case when there exist counts equal to 0, it is also easy to show that the ML estimators for the corresponding parameters are also zero. When the maximum is reached in the interior of the domain, one can apply Lagrange Multipliers to optimize for P_{ik} . Therefore, let us introduce new variables λ_c for each constraint (CPD) $c \in C$. The new function to optimize will be: $LM(\theta, \lambda) = \sum_{\theta_i} N_{\theta_i} \cdot \log \theta_i - \sum_c \lambda_c \cdot g_c(\theta)$. According to Lagrange Multipliers theory, any point that is a local maximum or minimum for the initial optimization problem and it is NOT on the border of the region defined by the constraints will be obtained as a (partial) solution of the system of equations:

$$\nabla LM(\theta, \lambda) = 0$$

Therefore $\hat{\theta}$ verifies the above system for some values of λ . Because $\frac{\partial L(\theta|D)}{\partial \theta_i} = \frac{N_{\theta_i}}{\theta_i}$ and $\frac{\partial g_c}{\partial \theta_i} = \lambda_c$ if CPD c contains θ_i (otherwise the partial derivative is zero), we get:

$$(4.1) \quad \hat{\theta}_i = \frac{N_{\theta_i}}{\sum_{c \in \text{Scope}(v)} \lambda_c} \quad \forall \theta_i \in \text{Shared}(v)$$

Let $S(v) = \sum_{\theta_j \in \text{Shared}(\text{Desc}(v))} \hat{\theta}_j$. We will prove by induction the following stronger claim:

$$S(v) = \frac{\sum_{\theta_j \in \text{Shared}(\text{Desc}(v))} N_{\theta_j}}{\sum_{c \in \text{Scope}(v)} \lambda_c}$$

and

$$\hat{\theta}_i = \left(1 - \sum_{\theta_j \in \text{Shared}(\text{Ancestors}(v))} \hat{\theta}_j \right) \cdot \frac{N_{\theta_i}}{\sum_{\theta_k \in \text{Shared}(\text{Desc}(v))} N_{\theta_k}}$$

Base case: If v is a leaf, then the distributions in $\text{Scope}(v)$ are equal. First part of the claim is verified directly from 4.1 and the second part follows because of the fact that the probabilities that add up to $S(v)$ are proportional to their corresponding counts.

Induction Step: Assume v is not a leaf and has direct descendants d_1, \dots, d_k for which the claim holds. It is obvious that $S(d_1) = \dots = S(d_k)$. Now, using the induction hypothesis, we obtain $S(d_1) = \dots = S(d_k) = \frac{\sum_{l=1}^k \sum_{\theta_j \in \text{Shared}(\text{Desc}(d_l))} N_{\theta_j}}{\sum_{c \in \text{Scope}(v)} \lambda_c}$. This combined with 4.1 gives us the first part of the claim. The second part of the claim now follows from 4.1 and the fact that

$$\frac{1}{\sum_{c \in \text{Scope}(v)} \lambda_c} = \frac{S(v)}{\sum_{\theta_j \in \text{Shared}(\text{Desc}(v))} N_{\theta_j}}$$

The proof of our theorem is now complete. \square

The above theorem yields an obvious recursive top-down, breadth-first algorithm to compute the ML Estimates of the parameters. The correctness of the algorithm is justified by theorem 4.1 and the fact that a node v is processed sometime after all the nodes on the path from v to the root are processed. The algorithm uses a queue Q to perform breadth-first traversal of the tree.

Algorithm:

STEP 1. Enqueue the root of the tree in Q .

STEP 2. If $Q = \emptyset$, STOP. Else, $v \leftarrow \text{Dequeue}(Q)$.

STEP 3. Compute $\hat{\theta}_i$ for all $\theta_i \in \text{Shared}(v)$.

STEP 4. Enqueue all children of v . GO TO STEP 2.

4.3 Learning - Bayesian Approach, Full Data Observability, Known Structure.

In order to be able to compute MAP estimates and predict an example from the previous ones by bayesian averaging, we have to define priors over the space of parameters. Note however that in our case, when hierarchical parameter sharing is present, the local and global parameter assumptions are violated. Next we will show how to define priors over the space of parameters that obey the constraints given by a *Parameter Sharing Tree* T . First of all, it would be nice if $P(\theta)$ and $P(D|\theta)$ are conjugate distributions. This suggests we chose our priors of the following form:

$$P(\theta) = \frac{1}{Z(T)} \prod \theta_i^{\alpha_i - 1}$$

Note that these priors are defined over the whole space of parameters and that a parameter θ_i can appear in multiple places in the graphical model (according to

the given *Parameter Sharing Tree*). In addition, the normalization constant depends heavily on the structure of the parameter sharing tree since T describes the constraints among parameters (the sum of parameters shared on the path from the root to any leaf should sum up to 1). If for finding MAP estimates the normalization constant is not important, it becomes important when computing $P(D_{n+1}|D_1, \dots, D_n)$ (as we saw in the previous section). Before showing how to compute $Z(T)$, let us define the *Generalized Dirichlet Integral* to be:

$$\int_{\sum \theta_i = S} \prod \theta_i^{\alpha_i - 1} d\theta = (1-S)^{(\sum \alpha_i) - 1} \cdot \int_{\sum \theta_i = 1} \prod \theta_i^{\alpha_i - 1} d\theta$$

The last integral is a normalization constant for a standard Dirichlet distribution and this gives us a way for computing the *Generalized Dirichlet Integral*. Now, the constant $Z(T) = \int_{\theta \text{ obeys } T} \prod \theta_i^{\alpha_i - 1} d\theta$ can be recursively computed as follows. First, note that this integral can be evaluated starting with the parameters from the leaf level. For each leaf v , the integral over the parameters involved in $Shared(v)$ is a *Generalized Dirichlet Integral*. The effect of computing this integral is to get the constant given by the Standard Dirichlet and propagate upwards a single parameter $(1 - S(v))$ with cumulative Dirichlet parameter $(\sum_{\theta_i \in Shared(v)} \alpha_i) - 1$. Now, it is easy to see that this parameter is the same for all leaves that belong to the same parent p . This will make the integral over the parameters in $Shared(p)$ and the new parameter to be also a *Generalized Dirichlet Integral* and the procedure continues as described above until we end the computation at the root level. This concludes or sketch of showing how one can recursively compute $Z(T)$ using *Generalized Dirichlet Integrals*. Once the normalization constant is computed, bayesian prediction can be done in exactly the same way as in the previous section.

5 Sharing Properties of Subsets of Parameters

In this section we present other extensions to the framework presented in section 3. Again, the set C of CPDs in the graphical model is partitioned as: $C = \cup_{k=1}^m C_k$, where the distributions in a subset C_k are going to be constrained together. Moreover, assume there are no constraints tying CPDs in different subsets of the partition. If in section 3 the constraints were individual parameter equalities, in this section we are studying constraints that involve whole subsets of parameters within the same CPD.

5.1 Probability Mass Sharing. Here we are going to show how to do Maximum Likelihood learning in the case when the aggregate probability mass of a certain

parameter type is the same across all CPDs in a given subset C_k . For example, we would like to show how to take advantage of constraints like: "The frequency of nouns in English is the same as the frequency of nouns in Spanish", when modelling the word probability in each of the two languages. In these case, types would be: nouns, verbs, etc.

Before stating the main result of this subsection, let us introduce few notations. Assume for a specific k , the parameters in C_k may have the following types: T_1, \dots, T_s . Denote by θ_i^j the i^{th} parameter in j^{th} CPD in C_k . Each parameter has exactly one type. For example, in the above example, $P(Computer|English)$ has type Noun, while $P(Blue|English)$ has type Adjective. We would like to stress the fact that in our framework, C_k is an arbitrary subset of CPDs in the graphical model; these CPDs can have different number of parameters and they can belong to different CPTs. Formally, the constraints that we are dealing with are given by:

Probability Mass Sharing Assumption. For all types T_l and for any j_1^{th} and j_2^{th} distributions in C_k , the following holds:

$$\sum_{\theta_i^{j_1} \in T_l} \theta_i^{j_1} = \sum_{\theta_i^{j_2} \in T_l} \theta_i^{j_2}$$

Back to our example, this translates into: "The aggregate probability of Nouns is the same in all modelled languages and same holds for other grammatical categories/types." It might seem a little restrictive to have each parameter belong to one type because, for instance, one may argue that maybe only the probability of Nouns is being shared across languages. However, even if one specifies the Probability Mass Sharing Assumption only for Nouns, the rest of parameters (non-nouns) verify the same constraint and therefore that is equivalent to introducing a new dummy type that contains every other parameter in C_k .

Since there are no constraints between CPDs in different C_k s, one can break the optimization for finding the ML Estimates into a set of independent optimization problems. The function to optimize for each C_k will be $f(C_k) = \sum N_i^j \cdot \log \theta_i^j$, where N_i^j represents the observed count for parameter θ_i^j in the training set. With these considerations we are now ready to present the main result of this subsection:

THEOREM 5.1. *The maximum likelihood estimator $\hat{\theta}_i^j$*

for a parameter θ_i^j in C_k that has type T_l is given by:

$$\hat{\theta}_i^j = \frac{N_i^j}{\sum_{\theta_{i'}^j \in T_l} N_{i'}^j} \cdot \frac{\sum_{\theta_{i'}^j \in T_l} N_{i'}^j}{\sum_{\theta_{i'}^j} N_{i'}^j}$$

Proof of Theorem 5.1 (Sketch): We introduce new variables $A = (A_l)_{1 \dots s}$ that represent the probability mass associated with type T_l in any of the distributions in C_k . As stated above, the log-likelihood function decomposes nicely in components corresponding to different C_k . With the newly introduced variables, our optimization problem can be restated as maximizing $f(C_k, A) = \sum N_i^j \cdot \log \theta_i^j$ subject to the constraints that $\sum_{\theta_i^j \in T_l} \theta_i^j = A_l$ for all types T_l and for any j^{th} distribution in C_k . In addition to these constraints, we also have $\sum_i \theta_i^j = 1$. Similarly to the previous theorems, it is easy to show that, if all counts are positive, then the function reaches a maximum inside the region defined by the constraints (if any count is zero, then the specific parameter doesn't even show up in the log-likelihood function and its estimator will be zero too). In this case, we also apply Lagrange Multipliers theory, introducing a lagrange multiplier for each constraint: λ_l^j for the first type of constraints (probability mass equalities) and λ^j for the second type (distributions should sum up to 1). Therefore, differentiating with respect to θ and A , the point that maximizes f inside the region given by the constraints should also verify:

$$(5.2) \quad N_i^j = \hat{\theta}_i^j \cdot (\lambda^j + \lambda_l^j) \quad \forall \theta_i^j \in T_l$$

$$(5.3) \quad \sum_j \lambda_l^j = 0 \quad \forall l$$

For a fixed j and l , summing up 5.2 for all i such that $\theta_i^j \in T_l$ we get:

$$(5.4) \quad \sum_{\theta_i^j \in T_l} N_i^j = A_l \cdot (\lambda^j + \lambda_l^j)$$

For a fixed l , summing up 5.4 for all j and using 5.3 we obtain:

$$(5.5) \quad \sum_{\theta_i^j \in T_l} N_i^j = A_l \cdot \sum_j \lambda^j$$

Further, summing 5.5 over all values of l and using the fact that the distributions sum up to 1, we can compute:

$$(5.6) \quad \sum_{i,j} N_i^j = \sum_j \lambda^j$$

Now we can use 5.6 in 5.5 to get A_l which is further used in 5.4 to obtain $\lambda^j + \lambda_l^j$ which, substituted in 5.2 will yield the formulas in the statement of the theorem. Our sketch of the proof is now complete. \square

5.2 Probability Ratio Sharing. In the previous subsection, we showed how to do learning when certain parameter types share their aggregate probability mass across different distributions. Now assume we want instead to enforce the constraint that the relative proportions of parameters in a certain type are the same for all different CPDs within a C_k . Next we describe a setting where this constraints may arise naturally. We would like again to model the word probabilities in two languages i.e. model $P(\text{word}|\text{language})$ for both languages and their corresponding sets of words. In this case, types can be: "words about computers" ("computer", "mouse", "monitor", "keyboard" in both languages) or "words about business", etc. In some countries computer use is more extensive than in others and one would expect the aggregate probability of "words about computers" to be different. However, it would be natural to assume that the relative proportions of the "words about computers" are the same within the different languages.

We keep the same notations in the previous subsection. There are two major differences from the setting presented in the previous subsection. First, in this case, we must have the same number of parameters of type T_l in each of the distributions in C_k . For example, "words about computers" can be "mouse", "keyboard" and "monitor" in both Spanish and English (if we have synonyms for a certain word, they can all be modelled as one cumulative parameter). This allows us to permute the parameters in the distributions in C_k such that corresponding parameters in T_l are located on the same position in each of the distributions. For example, we can assume $p(\text{mouse}|\text{English})$ and $p(\text{mouse}|\text{Spanish})$ are both on the first position in the two language models. This allows us to write that a specific position $i \in T_l$. Second, now there may be parameters that do not belong to any type T_l . For example, the expert may specify that only the "words about computers" preserve their relative probability ratios across the languages of interest. With these considerations, let us formalize the constraints that we are looking at:

Probability Ratio Sharing Assumption. For any fixed type T_l and for fixed $i_1, i_2 \in T_l$, the following holds:

$$\frac{\theta_{i_1}^j}{\theta_{i_2}^j} = \text{constant} \quad \forall j$$

The problem of finding the maximum likelihood

estimators subject to the above constraints can again be broken down in independent subproblems over the different subsets of CPDs C_k and the function to optimize for each C_k will be $f(C_k) = \sum N_i^j \cdot \log \theta_i^j$.

THEOREM 5.2. *The maximum likelihood estimator $\hat{\theta}_i^j$ for a parameter θ_i^j in C_k is given by:*

- a) if $i \in T_l$: $\hat{\theta}_i^j = \frac{\sum_{j'} N_i^{j'}}{\sum_{i' \in T_l, j'} N_{i'}^{j'}} \cdot \frac{\sum_{i' \in T_l} N_{i'}^j}{\sum_{i'} N_{i'}^j}$
- b) if θ_i^j does not have a type: $\hat{\theta}_i^j = \frac{N_i^j}{\sum_{i'} N_{i'}^j}$

Proof of Theorem 5.2 (Sketch): Again, we use Lagrange Multipliers theory to derive our estimators. Each CPD should sum up to 1 and that translates in the constraint $(\sum_i \theta_i^j) - 1 = 0$. Let the corresponding lagrange multiplier be λ^j . The *Probability Ratio Sharing Assumption* implies that there exist A_l^j and τ_i such that $\theta_i^j - A_l^j \cdot \tau_i = 0$ for all $i \in T_l$. A_l^j represent proportionality constants for distribution j for parameters of type T_l and τ_i are reference constants that, when multiplied with the proportionality constants yield the parameters on position i in each distribution. Let λ_i^j be the lagrange multipliers corresponding to the last type of constraints. Our new objective function becomes $f(C_k, A, \tau) = \sum N_i^j \cdot \log \theta_i^j$. When applying Lagrange Multipliers theory to our optimization problem, differentiating with respect to θ and the newly introduced A_l^j and τ_i , we obtain:

$$(5.7) \quad N_i^j = \hat{\theta}_i^j \cdot (\lambda^j + \lambda_i^j) \quad \forall i \in T_l$$

$$(5.8) \quad N_i^j = \hat{\theta}_i^j \cdot \lambda^j \quad \forall i \notin \cup T_l$$

$$(5.9) \quad \sum_j \lambda_i^j \cdot A_l^j = 0 \text{ or } \sum_j \lambda_i^j \cdot \hat{\theta}_i^j = 0 \quad \forall i \in T_l$$

$$(5.10) \quad \sum_j \lambda_i^j \cdot \tau_i = 0 \text{ or } \sum_{i \in T_l} \lambda_i^j \cdot \hat{\theta}_i^j = 0 \quad \forall j, l$$

For fixed j and l , summing up 5.7 for all $i \in T_l$, then using 5.10 we get:

$$(5.11) \quad \sum_{i \in T_l} N_i^j = \lambda^j \sum_{i \in T_l} \hat{\theta}_i^j$$

If we further sum over all l and use 5.8 we obtain:

$$(5.12) \quad \lambda^j = \sum_i N_i^j$$

Because $\frac{\hat{\theta}_i^{j_1}}{\hat{\theta}_i^{j_2}} = \frac{A_l^{j_1}}{A_l^{j_2}}$ for all j_1, j_2, l and $i \in T_l$, we can write:

$$(5.13) \quad \frac{A_l^{j_1}}{A_l^{j_2}} = \frac{\sum_{i \in T_l} \hat{\theta}_i^{j_1}}{\sum_{i \in T_l} \hat{\theta}_i^{j_2}} = \frac{\lambda^{j_2}}{\lambda^{j_1}} \cdot \frac{\sum_{i \in T_l} N_i^{j_1}}{\sum_{i \in T_l} N_i^{j_2}}$$

For a fixed $i \in T_l$ summing up 5.7 over all j and using 5.9 we have:

$$(5.14) \quad \sum_j N_i^j = \hat{\theta}_i^j \cdot (\lambda^j + \sum_{j' \neq j} \lambda^{j'} \cdot \frac{\hat{\theta}_i^{j'}}{\hat{\theta}_i^j})$$

Using 5.12 and 5.13 in 5.14 proves part a) of the theorem. Part b) follows from 5.8 and 5.12. The sketch of the proof is now complete. \square

6 Example

Previously published experiments involving learning with Module Networks, HMMs, DBNs or Context Specific Independence all support the theory presented in this paper since they are particular cases of our General Parameter Sharing Framework. However, the Parameter Sharing assumptions in the above models are at the level of either entire CPT or entire CPD. In this section we present experimental results showing the benefits of incorporating more fine-grained parameter sharing assumptions when training Bayes Multinets.

6.1 Email Modelling and Bayesian Multinets.

Automatic modelling of email documents is a problem of considerable interest, and has been studied as a means of automatically sorting email into a user's email subfolders, or into other topical categories such as "email spam" [11]. Given a set (population) of email, one natural way to define Bayesian Multinet subpopulations is to consider each distinct email *author* to be a generator of email from a different distribution, forming a different subpopulation. This is reasonable because different people use somewhat different vocabularies and figures of speech. At the same time, there are many content words that are shared (used in a similar proportion) by all authors when emails address specific topics such as meetings. The conditional probabilities of these words given that the email is about a meeting should therefore be specified as globally shared parameters across the subpopulations in the Bayesian Multinet.

To study this "meeting" email modelling problem, we generated synthetic data that captures the characteristics of a real email data set: the PW CALO email corpus, produced by people in a role-playing game at

SRI. Based on this corpus, we generated several artificial datasets. Each data point consists of a triple: *Author*, *Email* and *Topic* (Class). The Topic says whether or not the email is about a meeting. In all experiments we generated simulated email from six authors, using the same prior probabilities of an email belonging to an author as in the PW CALO corpus, and generating emails from each author to match that author’s topic priors in this corpus. For each given author and topic, we generate emails according to a “Bag of Words” probability model, where each email contains between 0 and 150 words (to be consistent with the data observed in PW, even though we are not including stop words in our artificial dataset). The words are chosen from a vocabulary of 1070 words (same size as in PW). The word given topic probability models are generated randomly and differ from user to user, but also have some fraction of parameters in common (the so called shared parameters G_k in our framework). The fraction of shared parameters was varied in our experiments from 0 to 1. Creating this artificial data instead of using the real data allowed us to control and know which parameters are truly shared, allowed to assess the performance of our models in terms of KL Divergence from the true underlying distribution, and allowed us to control and study the effect of variations of different parameters in the true distribution (e.g. the fraction of parameters that are truly shared).

In our experiments we compare three models. First, a General Naive Bayes model (GNB) learned from all training examples. Second, a Bayesian Multinet (SSNB) in which each component network is a Naive Bayes model, and for which an oracle has indicated which parameters are shared when generating the data. Finally, a Bayesian Multinet (PSNB) identical to SSNB, but with no parameters shared among component networks. Note all three models are essentially Bayesian Multinets, conditioned on the email author which is observed in the header of the email. Each component network in each Bayes Multinet is a full naive Bayes model including both the Class/Topic variable and the word features in the email. One can think of GNB as a Bayesian Multinet where the component Bayes nets are copies of the GNB learned model. The only difference among the three is in the training procedure. They differ in their sharing of parameters (all shared in GNB, some shared in SSNB, not shared in PSNB). There is also a slight difference in the way we assign Dirichlet priors (we train all three models using MAP estimates, as is common when training Naive Bayes models from sparse data). In the case of GNB the effect is to increase each word count by one (equivalent to Dirichlet priors with all parameters equal to 2). In the case of PSNB and

SSNB, for each subpopulation the effect is to increase that subpopulation’s word counts by one. For PSNB, this is equivalent to Dirichlet priors with all parameters equal to 2 for each subpopulation, while for SSNB this is equivalent to assigning subpopulation-specific Dirichlet priors with parameters equal to 7 for shared parameters (7 is the number of subpopulations plus 1 in our experiments), and equal to 2 for local parameters.

6.2 Results and Discussion. We trained the three models while varying the number of training examples, and the fraction of word-given-class/topic model parameters that were shared (identical across authors). For each model, we measured the KL divergence $KL(T, M)$ of the learned model M to the true generating model T .

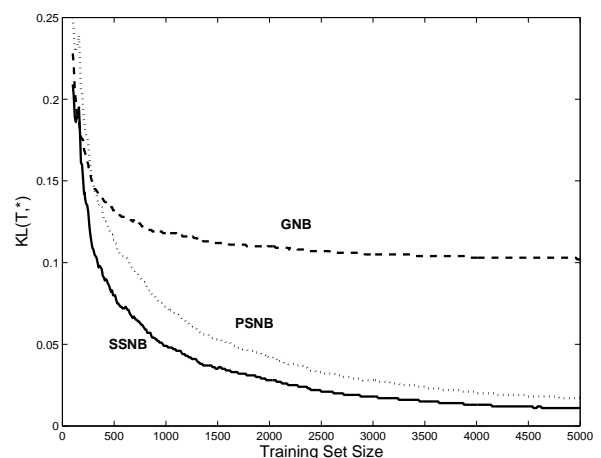


Figure 1: KL divergence of learned models with respect to correct model (T)

Figure 1 shows a plot of the KL divergence for each of the three models, as the number of training examples varies, keeping the fraction of general parameters constant at 0.5. As expected, $KL(T, *)$ decreases with increasing training set size for all three models (here $*$ stands for any possible model we are studying). However, SSNB outperforms the other two models across the entire range of training set size. It dominates PSNB especially at small training set sizes, because its shared global parameters allow it to produce lower-variance parameter estimates, especially when data is sparse. It dominates GNB especially with larger training sets, because it is capable of representing the correct model, whereas GNB considers a strictly smaller model class that does not contain the correct model. Note that asymptotically, as the training set size approaches infinity, the SSNB and PSNB models will both converge to the correct model, whereas GNB will not.

We also studied the impact of varying the fraction of global parameters in the true underlying probability model from 0 to 1, while holding the training set size constant at 1K. $KL(T, PSNB)$ is essentially constant as the fraction of true global parameters varies, because PSNB does not take advantage of parameter sharing. In contrast, both GNB and SSNB improve considerably with an increasing fraction of global parameters. Again, SSNB dominates the other two methods, as it can mix global and local parameters in its model.

7 Summary and Future Work

This paper presents a theoretical approach for incorporating several types of parameter related domain knowledge in learning procedures for graphical models parameterized by conditional probability tables. The main reason for taking advantage of such constraints is to alleviate learning from sparse data sets. First we describe in detail a General Parameter Sharing Framework that characterizes learning in a wide variety of graphical models like HMMs, DBNs or Module Networks. We develop sound procedures for learning in such models from both a Frequentist and a Bayesian point of view, from both complete and incomplete data, in the case when a domain expert can specify the structure of the graphical model and the parameters to be shared. Also, we prove some formal guarantees about our estimators and suggest ways for deciding among several potential Parameter Sharing Schemes. Second, we investigate a hierarchical extension of this framework based on Parameter Sharing Trees. Finally we present algorithms for using domain knowledge that specifies that certain groups of parameters share certain properties. In section 6, as an example, experimental results show that our shared parameter model achieved lower KL divergence to the true distribution when compared to two other classical methods: a global model and a standard Bayesian Multinet without parameter sharing.

This research suggests several directions for future work. First, we are developing proper conjugate prior distributions over the space of parameters that would allow us to develop MAP estimators in the setting presented in section 5. Second, we intend to explore the interaction between the different types of domain knowledge presented in this paper. Another interesting thing to investigate is how one can take advantage of parameter related domain knowledge to learn the structure of the graphical model.

Acknowledgements. This work was supported in part by DARPA research contract NBCD030010, and in part by a gift from Siemens Medical Solutions. We would like to thank the following people for

their useful comments during the development of this paper: William Cohen, Zoubin Ghahramani, Russ Greiner, John Lafferty, Andrew Moore and Sathyakama Sandilya.

References

- [1] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller, *Context-specific independence in Bayesian networks*, Proceedings of 12th UAI (1996), pp. 115–123.
- [2] J. Cheng and R. Greiner, *Learning bayesian belief network classifiers: algorithms and system*, Proceedings of the Canadian Conference on Artificial Intelligence (2001).
- [3] D. M. Chickering, D. Heckerman, and C. Meek, *A Bayesian Approach to Learning Bayesian Networks with Local Structure*, Technical Report, MSR-TR-97-07, 1997.
- [4] N. Friedman and M. Goldszmidt, *Learning Bayesian Networks with Local Structure*, Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, CA, 1996, pp. 252–262.
- [5] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, *Learning Probabilistic Relational Models*, Proceedings of 16th IJCAI (1999), pp. 1300–1307.
- [6] D. Geiger and D. Heckerman, *Knowledge representation and inference in similarity networks and Bayesian multinets*, Artificial Intelligence, 82 (1996), pp. 45–74.
- [7] D. Geiger and D. Heckerman, *A characterization of the Dirichlet distribution through global and local parameter independence*, The Annals of Statistics, 25 (1997), pp. 1344–1369.
- [8] D. Golinelli, D. Madigan, and G. Consonni, *Relaxing the local independence assumption for quantitative learning in acyclic directed graphical models through hierarchical partition models*, Proceedings of Artificial Intelligence and Statistics (1999), pp. 203–208.
- [9] K. P. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, PhD Thesis, UC Berkeley, Computer Science Division, 2002.
- [10] R. L. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech recognition*, Proceedings of the IEEE, 77 (2) (1989), pp. 257–286.
- [11] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, *A Bayesian Approach to Filtering Junk E-Mail*, AAAI Workshop on Learning for Text Categorization, Madison, WI, 1998.
- [12] E. Segal, D. Pe’er, A. Regev, D. Koller, and N. Friedman, *Learning Module Networks*, Proceedings of 19th UAI (2003), pp. 525–534.
- [13] C. Zhai and J. Lafferty, *A Study of Smoothing Methods for Language Models Applied to ad hoc Information Retrieval*, Proceedings of SIGIR (2001), pp. 334–342.

Exploiting Geometry for Support Vector Machine Indexing*

Navneet Panda[†]

Edward Y. Chang[‡]

Abstract

Support Vector Machines (SVMs) have been adopted by many data-mining and information-retrieval applications for learning a mining or query concept, and then retrieving the “top- k ” best matches to the concept. However, when the dataset is large, naively scanning the entire dataset to find the top matches is not scalable. In this work, we propose a kernel indexing strategy to substantially prune the search space and thus improve the performance of top- k queries. Our *kernel indexer* (KDX) takes advantage of the underlying geometric properties and quickly converges on an approximate set of top- k instances of interest. More importantly, once the kernel (e.g., Gaussian kernel) has been selected and the indexer has been constructed, the indexer can work with different kernel-parameter settings (e.g., γ and σ) without performance compromise. Through theoretical analysis, and empirical studies on a wide variety of datasets, we demonstrate KDX to be very effective.

1 Introduction

Support Vector Machines (SVMs) [6, 19] have become increasingly popular over the last decade because of their superlative performance and wide applicability. SVMs have been successfully used for many data-mining and information-retrieval tasks such as outlier detection [1], classification [5, 11, 14], and query-concept formulation [17, 18]. In these applications, SVMs learn a prediction function as a hyperplane to separate the training instances relevant to the target concept (representing a pattern or a query) from the others. The hyperplane is depicted by a subset of the training instances called *support vectors*. The unlabeled instances are then given a score based on their distances to the hyperplane. Many data-mining and information-retrieval tasks query for the “top- k ” best matches to a target concept. Yet it would be naive to require a linear scan of the entire unlabeled pool, which may contain thousands or millions of instances, to search for the top- k matches. To avoid a linear scan, we propose a kernel indexer (KDX) to work with SVMs. We demonstrate its scalable performance for top- k queries.

Traditional top- k query scenarios use a point in a vector space to depict the query, so the top- k matches are the k nearest instances to the query point in the vector space. A top- k query with SVMs differs from

that in the traditional scenarios in two aspects. First, a query concept learned by SVMs is represented by a hyperplane, not by a point. Second, a top- k query with SVMs can request the farthest instances from the hyperplane (the top- k matches for a concept), or those nearest to it (the top- k uncertain instances¹ for a concept). KDX supports top- k match as well as top- k uncertainty queries.

Intuitively, KDX works as follows. Given a kernel function and an unlabeled pool, KDX first finds the approximate center instance of the pool in the feature space. It then divides the feature space, to which the kernel function projects the unlabeled instances, into concentric hyper-rings (hereafter referred to as *rings* for brevity). Each ring contains about the same number of instances and is populated by instances according to their distances to the center instance in the feature space. Given a query concept, represented by a hyperplane, KDX limits the number of rings examined, and intelligently prunes out unfit instances from each ring. Finally, KDX returns the top- k results. Both the *inter-ring pruning* and *intra-ring pruning* are performed by exploiting the geometric properties of the feature space. (Details are presented in Section 4.)

KDX supports a couple of important properties. First, it can effectively support insertion and deletion operations. Second, given a kernel function, the indexer works independent of the settings of the kernel parameters (e.g., γ and σ). This parameter-invariant property is especially crucial, since varied query-concepts can best be learned under variable parameter settings. Through empirical studies on a wide variety of datasets, we demonstrate KDX to be very effective.

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 provides an overview on SVMs and introduces geometric properties useful to our work. We then propose KDX in Section 4, describing its key operations: index creation, top- k farthest instances lookup, and updates. Section 5 presents the results of our empirical studies. We offer our concluding remarks in Section 6, together with suggestions for future research directions.

*Supported by NSF grants IIS-0133802, and IIS-0219885.

[†]Department of Computer Science, UCSB.

[‡]Department of Electrical and Computer Engineering, UCSB.

¹In an active learning setting, the algorithm finds the most uncertain instances to query the user for labels. The most uncertain instances are the ones closest to the hyperplane.

2 Related Work

Indexing for SVMs to support top- k queries can be very challenging for three reasons. First, a kernel function K is the dot product of a basis function Φ , but we may not explicitly know the basis functions of most kernels. Second, even if the basis function is known, the dimension of the feature space F , to which the instances are projected, can be very high, possibly infinite. It is well known that traditional indexing methods do not work well with high-dimensional data for nearest-neighbor queries [20]. Third, a query represented by SVMs is a hyperplane, not a point.

Indexing has been intensively studied over the past few decades. We present some of the representative work in the field but our discussion is by no means exhaustive and for a detailed discussion please consult [13] or [10]. Existing indexers can be divided into two categories: *coordinate-based* and *distance-based*. The coordinate-based methods work on objects residing in a vector space by partitioning the space. A top- k query can be treated as a range query, and, ideally, only a small number of partitions need to be scanned for finding the best matches. Example coordinate-based methods are the X-tree [3], the R^* -tree [2], the TV-tree [16], and the SR-tree [12], to name a few. All these indexers need an explicit feature representation to be able to partition the space. As discussed above, the feature space onto which an SVM kernel projects data might not have an explicit representation. Even in cases where the projection function Φ is known, the dimension of the projected space could be too high to use the coordinate-based methods due to the curse of dimensionality [15]. Thus, the traditional coordinate-based methods are not suitable for kernel indexing.

Distance-based methods do not require an explicit vector space. The M-tree [8] is a representative scheme that uses the distances between instances to build an indexing structure. Given a query point, it prunes out instances based on their distances. SVMs use the distance from the hyperplane as a measure of the suitability of an instance. The farther the instance from the hyperplane in the positive half-space, the higher its “score” or confidence. The traditional distance-based methods require a query to be a *point*, whereas in this case we have a *hyperplane*. With infinite number of points on the query hyperplane, a top- k query using the points on the hyperplane may require scanning all buckets of the index.

When the data dimension is very high, the cost of supporting exact queries can be higher than that of a linear scan. The work of [9] proposes an approximate indexing strategy using latent semantic hashing. This approach hashes similar instances into the same bucket

with a high degree of accuracy. A top- k approximate query can be supported by retrieving the bucket into which the query point has been hashed. Unfortunately, this method requires the knowledge of the feature vector in the projected space, and cannot be used with SVMs. Another approximate approach is clustering for indexing [15] but this approach supports only point-based queries, not hyperplane queries.

We developed KDX to effectively tackle the three challenges specified in the beginning of this section.

3 Preliminaries

We briefly present SVMs, and then discuss the geometrical properties that are useful in the development of the proposed indexing structure.

3.1 Support Vector Machines

Let us consider SVMs in the binary classification setting. We are given a set of data $\{\mathbf{x}_1, \dots, \mathbf{x}_{m+n}\}$ that are vectors in some space $X \subseteq \mathbb{R}^d$. Among the $m+n$ instances, m of them, denoted as $\{\mathbf{x}_{l,1}, \dots, \mathbf{x}_{l,m}\}$ are assigned labels $\{y_1, \dots, y_m\}$, where $y_i \in \{-1, 1\}$. The rest are unlabeled data, denoted as $\{\mathbf{x}_{u,1}, \dots, \mathbf{x}_{u,n}\}$. The labeled instances are also called training data; and unlabeled are sometimes called testing data. In the remainder of this paper, we refer to a training instance simply as $\mathbf{x}_{l,i}$, and a testing instance as $\mathbf{x}_{u,i}$. When we just refer to an instance, either training or testing, we use \mathbf{x}_i .

In the simplest form, SVMs are hyperplanes that separate the training data by a maximal margin. The hyperplane is designed to separate the training data such that all vectors lying on one side of the hyperplane are labeled as -1 , and all vectors lying on the other side are labeled as 1 . The training instances that lie closest to the hyperplane are called *support vectors*. SVMs allow us to project the original training data in space X to a higher dimensional feature space F via a Mercer kernel operator K . Thus, by using K , we implicitly project the training data into a different (often higher dimensional) feature space F .

The SVM computes the α_i 's that correspond to the maximal margin hyperplane in F . By choosing various kernel functions (discussed shortly) we can implicitly project the training data from X into various feature spaces. (A hyperplane in F maps to a more complex non-linear decision boundary in the original space X .) Once the hyperplane has been learned based on the training data $\{\mathbf{x}_{l,1} \dots \mathbf{x}_{l,m}\}$, the class membership of an unlabeled instance $\mathbf{x}_{u,r}$ can be predicted using the α_i 's of the training instances and their labels $\{y_1, \dots, y_m\}$

by

$$(3.1) \quad f(\mathbf{x}_{u,r}) = \sum_{i=1}^m \alpha_i y_i K(\mathbf{x}_{l,i}, \mathbf{x}_{u,r}).$$

When $f(\mathbf{x}_{u,r}) \geq 0$ we classify $\mathbf{x}_{u,r}$ as +1; otherwise we classify $\mathbf{x}_{u,r}$ as -1.

SVMs rely on the values of inner products between pairs of instances to measure their similarity. The kernel function K computes the inner products between instances in the feature space. Mathematically, a kernel function can be written as,

$$(3.2) \quad K(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle$$

where ϕ is the implicit mapping used for projecting the instances, \mathbf{x}_1 and \mathbf{x}_2 . Essentially, the kernel function takes as input, a pair of instances, and returns the similarity between them in the feature space. Commonly used kernel functions are the Gaussian, the Laplacian kernels and the Polynomial. These are expressed as:

1. Gaussian : $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2}{2\sigma^2})$.
2. Laplacian : $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|_1)$.
3. Polynomial : $K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^p$.

The tunable parameters, σ for Gaussian, γ for the Laplacian kernel, and p for Polynomial, define different mappings. In each of the above, the mapping function ϕ is not defined explicitly. Yet, the inner product in the feature space can be evaluated in terms of the input space vectors and the corresponding parameter ($\sigma, \gamma, \text{or } p$) for the chosen kernel function.

3.2 Geometrical Properties of SVMs

We present three geometrical properties of kernel based methods used extensively throughout the rest of the paper.

1. *Similarity between any two instances measured by a kernel function is between zero and one.* Commonly used kernels like the Gaussian and the Laplacian are normalized kernels where the similarity between instances, as measured by the kernel function, takes on values between 0 and 1. A value of 1 indicates that the instances are identical while a value of 0 means they are completely dissimilar. The polynomial kernel, though not necessarily normalized, can easily be normalized by using

$$(3.3) \quad K_n(\mathbf{x}_1, \mathbf{x}_2) = \frac{K(\mathbf{x}_1, \mathbf{x}_2)}{K(\mathbf{x}_1, \mathbf{x}_1)K(\mathbf{x}_2, \mathbf{x}_2)},$$

where K_n is the normalized kernel function. Here, we have assumed that the features associated with each data instance are positive. If not, appropriate normalization needs to be performed.

2. *The projected instances lie on the surface of a unit hypersphere.* For a normalized kernel, the inner product of an instance with itself, $K_n(\mathbf{x}_i, \mathbf{x}_i)$, is equal to 1. This means that, after projection, all the instances lie on the surface of a hypersphere. Further, considering the fact that the kernel values are inner products, we see that the angle in feature space between any two instances is bounded above by $\frac{\pi}{2}$. This is so since the inner product is constrained to be always greater than or equal to 0 ($\cos^{-1}(0) = \frac{\pi}{2}$).

3. *Data instances exist on both sides of a query hyperplane.* The hyperplane needs to pass through the region on the hypersphere populated by the projected instances. Otherwise, it would be impossible to separate the positive from the negative training samples. This property is easily ensured since we have at least one training instance from the positive class and one from the negative class.

4 KDX

In this section, we present our indexing strategy, KDX, for finding the top- k relevant or the top- k uncertain instances (defined shortly) given a hyperplane. We discuss the construction of the index in Section 4.1, the approach for finding the top- k instances in Section 4.2, insertion and deletion operations in Section 4.3, and handling changes in kernel parameters in Section 4.4.

DEFINITION 4.1. Top- k Relevant Instances. *Given the set of instances $S = \{\mathbf{x}_r\}$, and the normal to the hyperplane, \mathbf{w} , represented in terms of the support vectors, the top- k relevant instances are the set of instances $(\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k) \subset S$ such that $\sum_{i=1, \mathbf{q}_i \in S}^k \mathbf{w} \cdot \phi(\mathbf{q}_i)$ is maximized over all possible choices of $\mathbf{q}_1, \dots, \mathbf{q}_k$ with $\mathbf{q}_i \neq \mathbf{q}_j$ if $i \neq j$. The subscripts do not represent the order of their membership in S . Ties are broken arbitrarily.*

DEFINITION 4.2. Top- k Uncertain Instances. *Given the set of instances $S = \{\mathbf{x}_r\}$, and the normal to the hyperplane, \mathbf{w} , represented in terms of the support vectors, the top- k uncertain instances are the set of instances $(\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k) \subset S$ such that $\sum_{i=1, \mathbf{q}_i \in S}^k |\mathbf{w} \cdot \phi(\mathbf{q}_i)|$ is nearest to zero over all possible choices of $\mathbf{q}_1, \dots, \mathbf{q}_k$ with $\mathbf{q}_i \neq \mathbf{q}_j$ if $i \neq j$. The subscripts do not represent the order of their membership in S . Ties are broken arbitrarily.*

4.1 KDX-create

The indexer is created in four steps.

1. Finding the instance $\phi(\mathbf{x}_c)$ that is approximately centrally located in the feature space F ,

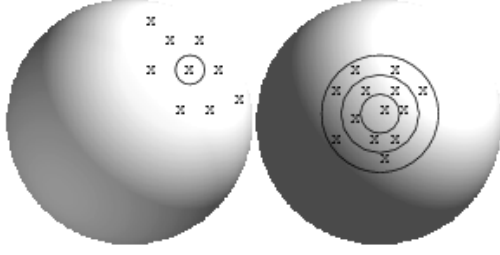


Figure 1: Approximate Central Instance and Rings.

2. Separating the instances into rings based on their angular distances from the central instance $\phi(\mathbf{x}_c)$,
3. Constructing a local indexing structure (*intra-ring indexer*) for each ring, and
4. Creating an *inter-ring* index.

4.1.1 Finding the central instance As shown in Figure 1, we attempt to find an approximate center $\phi(\mathbf{x}_c)$ after the implicit projection of the instances to the feature space F by kernel function K . The cosine of the angle between a pair of instances is given by the value of the kernel function K with the two instances as input (see Equation 3.2).

LEMMA 4.1. *The closest approximation of the central instance is the projection of the instance \mathbf{x}_c whose sum of distances from the other instances is the smallest.*

Proof. The point in F whose coordinates are the average of the coordinates of the projected instances in the dataset is at the center of the distribution of instances $\phi(\mathbf{x}_i)$, $i = 1 \dots n$. Choosing the instance which minimizes the variance gives us the closest approximation to the true center since it is closest to the point with average coordinates in F .

$$\begin{aligned}
\mathbf{x}_c &= \operatorname{argmin}_{\mathbf{x}_j} \sum_i (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^2 \\
&= \operatorname{argmin}_{\mathbf{x}_j} \sum_i (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i) + \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_j) \\
&\quad - 2\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)) \\
&= \operatorname{argmin}_{\mathbf{x}_j} \sum_i (2 - 2K(\mathbf{x}_i, \mathbf{x}_j)).
\end{aligned}$$

Given n instances in the dataset, each with d features, finding the central instance in the projected space takes $O(n^2d)$ time. However, since we are only interested in the approximate central instance, this cost can be easily lowered via a sampling method. This step can be achieved with $O(1)$ storage because at any point we need to store just the current known minimum, and the accumulated value of the sum of the angles of the rest of the instances with the current instance being evaluated.

4.1.2 Separating instances into rings In this step we compute the angles of the projected instances in F with the central instance, $\phi(\mathbf{x}_c)$, using K . The angles are stored in an array, which is then sorted. Here we have a choice of the number of instances that need to be included in a ring. The number of instances per ring can be based on the size of the L2 cache on the system to minimize cache misses. As we shall see later, only the instances in the same ring are processed together. Hence, at any given time during the processing of queries, we need only the amount of storage utilized by the instances in one ring.

Figure 1 shows the division of instances into different rings. To divide the instances into rings, we equally divide the sorted list. That is, if the number of instances per ring is g , then the first g elements in the sorted array are grouped together, and so on. This step requires $O(n \log n)$ time, and $O(n)$ space.

4.1.3 Constructing intra-ring index For each ring, KDX constructs a local index. We construct for each ring a $g \times g$ square matrix, where the i^{th} row of the matrix contains the angles between the i^{th} instance and the other $g - 1$ instances. Next, we sort each row such that the instances are arranged according to decreasing order of similarity (or increasing order of distance) with the instance associated with the row.

This step requires $O(g^2)$ storage and $O(g^2 d) + O(g^2 \log g)$ computational time for each ring.

4.1.4 Creating inter-ring index Finally, we construct the inter-ring index, which is the closest instance from the adjoining ring for each instance. This step requires $O(n)$ storage and $O(ng)$ time. All the steps above are essentially preprocessing of the data which needs to be done only once for the dataset.

4.2 KDX-top- k

In this section, we describe how KDX finds top- k instances relevant to a query (Definition 4.1) by just examining a fraction of the dataset. Details of the number of instances evaluated are presented in Section 5.

Let us revisit Definition 4.1 for top- k relevant queries. The most relevant instances to a query, represented by a hyperplane trained by SVMs, are the ones farthest from the hyperplane on the positive side. Without an indexer, finding the farthest instances involves computing the distances of all the instances in the dataset from the hyperplane, and then selecting the k instances with greatest distances. This linear-scan approach is clearly costly when the dataset is large. Further, the number of dimensions associated with each data instance has a multiplicative effect on this cost.

KDX performs inter-ring and intra-ring pruning to find the approximate set of top- k instances by:

1. Shifting the hyperplane to the origin parallel to itself, and then computing θ_c , the angular distance between the normal to the hyperplane and the central instance $\phi(\mathbf{x}_c)$.
2. Identifying the ring with the farthest coordinate from the hyperplane, and selecting a starting instance $\phi(\mathbf{x})$ in that ring.
3. Computing the angular separation between $\phi(\mathbf{x})$ and the farthest coordinate in the ring from the hyperplane, denoted as $\phi(\mathbf{x}^*)$.
4. Iteratively, replacing $\phi(\mathbf{x})$ with a closer instance to $\phi(\mathbf{x}^*)$ and updating the top- k list, until no “better” $\phi(\mathbf{x})$ in the ring can be found.
5. Identifying a good starting instance $\phi(\mathbf{x})$ for the next ring, followed by repeating steps 3 to 5, until the termination criterion is satisfied.

KDX achieves speedup over the naive linear scan method in two ways. First, KDX does not examine all rings for a query. KDX terminates its search for top- k when the constituents of the top- k set do not change over the evaluation of multiple rings, or the query time expires. Second, in the fourth step, KDX examines only a small fraction of the instances in a ring. The remainder of this section details these steps, explaining how KDX effectively approximates the top- k result for achieving significant speedup. The formal algorithm is presented in Figure 8.

4.2.1 Computing θ_c

Parameter θ_c is important for KDX to identify the ring containing the farthest coordinate from the hyperplane. To compute θ_c , we first shift the hyperplane to pass through the origin in the feature space. The SVM training phase learns the distance of the hyperplane from the origin in terms of variables b and \mathbf{w} [19]. The distance of the hyperplane from the origin is given by $-b/\|\mathbf{w}\|$. We shift the hyperplane to pass through the origin without changing its orientation by setting $b = 0$. This shift does not affect the set of instances farthest from the hyperplane because it has the same effect as adding a constant value to all distances. Next, we compute the angular distance θ_c of the central instance $\phi(\mathbf{x}_c)$ from the normal to the hyperplane.

Given training instances $\mathbf{x}_{l,1} \dots \mathbf{x}_{l,m}$ and their labels $y_1 \dots y_m$, SVMs solve for weights α_i for $\mathbf{x}_{l,i}$. The

normal of the hyperplane² can be written as

$$(4.4) \quad \mathbf{w} = \frac{\sum_i^m \alpha_i y_i \phi(\mathbf{x}_{l,i})}{\sqrt{\sum_{i,j}^m \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_{l,i}) \cdot \phi(\mathbf{x}_{l,j})}}.$$

The angular distance between the central instance and \mathbf{w} is essentially $\cos^{-1}(\mathbf{w} \cdot \phi(\mathbf{x}_c))$.

4.2.2 Identifying the starting ring

The most logical ring from which to start looking for the farthest instance is the one containing the coordinate on the hypersphere farthest from the hyperplane. Let $\phi(\mathbf{x}^\diamond)$ denote this farthest coordinate. Note that there may not exist a data instance at $\phi(\mathbf{x}^\diamond)$. However, finding an instance close to the farthest coordinate can help us find the farthest instance with high probability. The following lemma shows how we can identify the ring containing the farthest coordinate from the hyperplane.

LEMMA 4.2. *The point, $\phi(\mathbf{x}^\diamond)$, on the surface of the hypersphere, farthest from the hyperplane, is at the intersection of the hypersphere and the normal to the hyperplane passing through the origin.*

The proof follows from the fact that all the instances are constrained to lie on the surface of a hypersphere and the distance from the hyperplane decreases as we move away from the point of intersection of the normal with the hypersphere because of the curvature.

We do not need to explicitly compute the farthest coordinate, since we are only interested in the ring where it resides. To find the ring, we rely on the angular separation of $\phi(\mathbf{x}^\diamond)$ from $\phi(\mathbf{x}_c)$, which is the θ_c obtained in the previous section. We use Figure 2 to illustrate. The figure shows that $\phi(\mathbf{x}^\diamond)$ is at the intersection of the hypersphere and the normal to the hyperplane with θ_c angular separation from $\phi(\mathbf{x}_c)$. Given \mathbf{x}_c and the normal of the hyperplane, we can compute θ_c to locate the ring containing the farthest coordinate on the hypersphere from the hyperplane. The rings were formed from the sorted array of instances based on their angular separation from the central instance. Therefore, the first instance picked for every ring serves as a delimiter for that ring. To identify the ring, we therefore need to look only at these delimiters.

4.2.3 Intra-ring pruning

Our goal is to find the farthest instances in the ring from the hyperplane. In this section, we present our pruning algorithm, which aims to reduce the number of instances examined to find a list of *approximate* farthest

²Training instances with zero weights are not support vectors and do not affect the computation of the normal.

instances. In Section 5 we show that our pruning algorithm achieves high-quality top- k results, just by examining a small fraction of instances.

If the ring is the first one being evaluated, KDX randomly chooses an instance $\phi(\mathbf{x})$ in the ring as the anchor instance. (In Section 4.2.4 we show that if the ring is not the first to be inspected, we can take advantage of the inter-ring index to find a good $\phi(\mathbf{x})$.) Let $\phi(\mathbf{x}^*)$ be the farthest point from the hyperplane in the ring. We would like to find instances in the ring closest to $\phi(\mathbf{x}^*)$. Our goal is to find these instances by inspecting as few instances in the ring as possible.

Let us use a couple of figures to illustrate how this intra-ring pruning algorithm works. First, the circle in Figure 3 depicts the hyperdisc of the current ring. Please note that the hyperdisc can be inclined at an angle to the hyperplane as shown in Figure 4. Back to Figure 3. We would like to compute the distance s between $\phi(\mathbf{x})$ and $\phi(\mathbf{x}^*)$. Since both $\phi(\mathbf{x})$ and $\phi(\mathbf{x}^*)$ lie on the surface of a unit hypersphere, the angular separation between them can be obtained once s is known. Figure 3 shows that we need to determine h and v in order to use the Pythagorean theorem to obtain s . Determination of h and v , in turn, requires the knowledge of distances d_1 and d_2 . Distance d_1 denotes the distance from the center of the hyperdisc to the hyperplane, along the hyperdisc, and d_2 the distance of $\phi(\mathbf{x})$ to the hyperplane, along the hyperdisc. It is noteworthy that both these distances are measured along the surface of the hyperdisc as shown for d_2 in Figure 4. We discuss in detail how we derive s in the online version of the paper at <http://www.cs.ucsb.edu/~panda/sdm-complete.pdf>. To focus our presentation on the pruning algorithm, we assume that we have had s computed.

Given $\phi(\mathbf{x})$ and s , KDX at each step tries to find an instance farther than $\phi(\mathbf{x})$ from the hyperplane and closer to $\phi(\mathbf{x}^*)$. Such an instance would lie between $\phi(\mathbf{x}^*)$ and $\phi(\mathbf{x})$, or between $\phi(\mathbf{x}^*)$ and point C , as depicted in Figure 5. Once we find a “better” instance than $\phi(\mathbf{x})$, we replace $\phi(\mathbf{x})$ with the new instance, and search for yet another farther instance. Notice that as we find a farther $\phi(\mathbf{x})$ from the hyperplane, the search range between $\phi(\mathbf{x})$ and C is reduced. This pruning algorithm eventually converges when no instances reside in the search range. When the pruning algorithm converges, there is a high probability that we have found a point $\phi(\mathbf{x})$ in the ring that is the farthest from the hyperplane.

To understand the computational savings of this intra-ring pruning algorithm, let us move down to the next level of details. We use the example in Figure 6 to explain the pruning process. Starting at $\phi(\mathbf{x})$, we

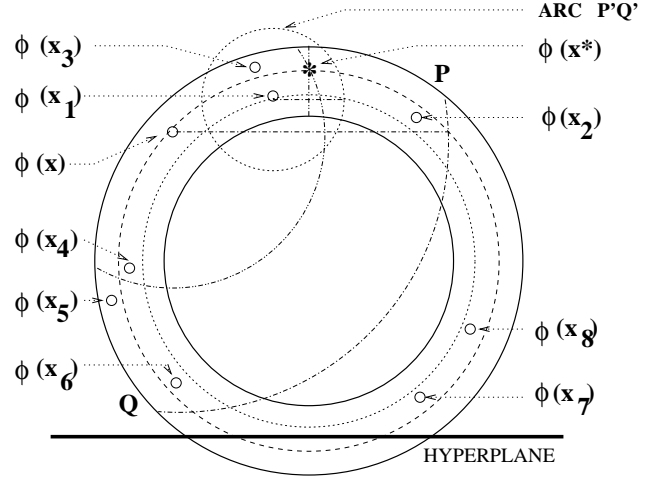


Figure 6: Arrangement of instances

seek to find an instance as close to $\phi(\mathbf{x}^*)$ as possible. The intra-ring index (Section 4.1.3) of $\phi(\mathbf{x})$ contains an ordered list of instances based on their distances from $\phi(\mathbf{x})$. Let τ denote the angular separation between $\phi(\mathbf{x})$ and $\phi(\mathbf{x}^*)$. To find an instance close to $\phi(\mathbf{x}^*)$, we search this list for instances with an angular separation of about τ from $\phi(\mathbf{x})$. For the example in Figure 6 the neighboring points of $\phi(\mathbf{x})$ appear in the order $\phi(\mathbf{x}_3)$, $\phi(\mathbf{x}_1)$, $\phi(\mathbf{x}_4)$, $\phi(\mathbf{x}_5)$, $\phi(\mathbf{x}_2)$, $\phi(\mathbf{x}_6)$, $\phi(\mathbf{x}_7)$, and $\phi(\mathbf{x}_8)$ in the sorted list of $\phi(\mathbf{x})$. First, we need only examine the instances lying within the arc PQ in the figure, since an instance outside this arc cannot be closer to $\phi(\mathbf{x}^*)$ than $\phi(\mathbf{x})$ itself. This step allows us to prune instances $\phi(\mathbf{x}_8)$ and $\phi(\mathbf{x}_7)$.

Next, we would like to re-sort the instances remaining on the list of $\phi(\mathbf{x})$ based on their likelihood of being close to $\phi(\mathbf{x}^*)$. To quantify this likelihood for instance $\phi(\mathbf{x}_i)$, we compute how close the angular distance between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x})$ is to the angular distance between $\phi(\mathbf{x}^*)$ and $\phi(\mathbf{x})$ (which is τ). The list does not need to be explicitly constructed since we have sorted and stored the distances between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x})$ in the intra-ring index. Once we find the instance closest to $\phi(\mathbf{x}^*)$ in the index, the rest of the instances on the re-sorted list can be obtained by looking up the adjacent instances of the closest instance in the intra-ring index. In our example, this re-sorted list is $\phi(\mathbf{x}_4)$, $\phi(\mathbf{x}_5)$, $\phi(\mathbf{x}_1)$, $\phi(\mathbf{x}_3)$, $\phi(\mathbf{x}_2)$ and $\phi(\mathbf{x}_6)$.

It may be surprising that $\phi(\mathbf{x}_5)$ and $\phi(\mathbf{x}_4)$ appear before $\phi(\mathbf{x}_1)$ on the re-sorted list. The reason is that we know only the angular distance between two instances, not their physical order on the ring. Fortunately, pruning out $\phi(\mathbf{x}_5)$ and $\phi(\mathbf{x}_4)$ from the list is simple—we need only remove instances that are closer to the hyperplane than $\phi(\mathbf{x})$. In this case, $\phi(\mathbf{x}_5)$ and $\phi(\mathbf{x}_4)$ are closer to the hyperplane than $\phi(\mathbf{x})$. After removing

ALGORITHM 4.1. KDX-top- k

Input: Support vectors \mathbf{z}_i
Dataset instances \mathbf{x}_i
Intra-ring index Arr
Inter-ring index $inter_ring$
Output: Top- k set top_k

```

1:  $counter = 0$ 
2:  $condition = \text{False}$ 
3:  $top\_k = \{\}$ 
4:  $\theta_c = \text{Find\_}\theta_c(\mathbf{z}_i, \mathbf{x}_c)$ 
5:  $R = \text{Find\_ring\_of\_interest}(\theta_c, ring)$ 
6:  $\psi = \text{Find\_}\psi(\theta_c)$ 
7:  $R' = R$ 
8:  $\mathbf{x} = \text{random instance in } R$ 
9: while  $counter < n/g$  and  $condition = \text{False}$  do
10:    $Converged = \text{False}$ 
11:    $S = \{\}$ 
12:   while  $\neg Converged$  do
13:      $(d_1, d_2) = \text{Find\_distances}(\mathbf{x}, \mathbf{w}, \psi)$ 
14:      $(h, v) = \text{Find\_h\_v}(d_1, d_2, \mathbf{x}, \mathbf{x}_c, \mathbf{w})$ 
15:      $(\tau, \xi) = \text{Find\_}\tau\_xi(h, v)$ 
16:      $index = \text{Bin\_search}(Arr[R'][\text{inverted\_index}[\mathbf{x}], \tau)$ 
17:     if  $ring[R'][index] == \mathbf{x}$  then
18:        $Converged = \text{True}$ 
19:     else
20:        $S_x = \text{Arrangement}(\mathbf{x}, \tau, \xi, R')$ 
21:        $\mathbf{x}_n = \cap S$  // Intersection chooses unevaluated in-
22:       stance only
23:        $\mathbf{x} = \mathbf{x}_n$ 
24:     end if
25:   end while
26:    $condition = \text{Ring\_termination\_condition}(top\_k, \mathbf{x})$ 
27:    $\mathbf{x} = inter\_ring(\mathbf{x})$ 
28:    $R' = \text{Adjacent}(R)$ 
29:    $counter = counter + 1$ 
30: end while

Procedure Find- $\theta_c(\mathbf{z}_i, \mathbf{x}_c)$ 
Section 4.2.1
1:  $\mathbf{z}_i \leftarrow$  Support vector  $i$ 
2:  $\mathbf{w} = \frac{\sum_i^{n_{sv}} \alpha_i y_i \phi(\mathbf{z}_i)}{\sqrt{\sum_{i,j}^{n_{sv}} \alpha_i \alpha_j y_i y_j \phi(\mathbf{z}_i) \cdot \phi(\mathbf{z}_j)}}$ 
3:  $\theta_c = \cos^{-1}(\mathbf{w} \cdot \phi(\mathbf{x}_c))$ 

Procedure Find-ring-of-interest( $\theta_c, ring$ )
Section 4.2.2
1: for  $i = 1$  to  $num\_rings$  do
2:    $temp\_array[i] = \cos^{-1}(K(ring[i][0], \mathbf{x}_c))$ 
3: end for
4:  $R = \text{Bin\_Search}(temp\_array, \theta_c)$ 

Procedure Find- $\psi(\theta_c)$ 
1: if  $\theta_c > \pi/2$  then
2:    $\psi = \pi - \theta_c$ 
3: else
4:    $\psi = \theta_c$ 
5: end if

Procedure Find-distances( $\mathbf{x}, \mathbf{w}, \psi$ )
1:  $d = \mathbf{w} \cdot \phi(\mathbf{x})$ 
2:  $d_2 = d / \sin(\psi)$ 
3:  $p = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_c)$ 
4:  $d_1 = p / \tan(\psi)$ 

Procedure Adjacent( $R$ )
1: static  $direction = 0$ 
2: static  $num_1 = 1, num_2 = 1$ 
3: if  $direction = 0$  and  $R + num_1 < n/g$  then
4:    $R' = R + num_1$ 
5:    $num_1 = num_1 + 1$ 
6: else if  $R - num_2 \geq 0$  then
7:    $R' = R - num_2$ 
8:    $num_2 = num_2 + 1$ 
9: end if
10:  $direction = 1 - direction$ 
11: return  $R'$ 

Procedure Find-h-v( $d_1, d_2, \mathbf{x}, \mathbf{x}_c, \mathbf{w}$ )
Section 4.2.3
1:  $r = \sin(\cos^{-1}(\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_c)))$ 
2: if  $d_1 \times d_2 \geq 0$  and  $d_1 \geq 0$  then
3:    $temp = d_2 - d_1$ 
4:    $v = \text{abs}(temp - r)$ 
5: else if  $d_1 \geq 0$  then
6:    $temp = d_1 - d_2$ 
7:    $v = r + temp$ 
8: else
9:    $temp = d_2 - d_1$ 
10:    $v = r - temp$ 
11: end if
12:  $h = \sqrt{r^2 - temp^2}$ 

Procedure Find- $\tau\_xi(h, v)$ 
Section 4.2.3
1:  $s = \sqrt{h^2 + v^2}$ 
2:  $\tau = \cos^{-1}(\frac{2-s^2}{2})$ 
3:  $\xi = \cos^{-1}(\frac{2-(2h)^2}{2})$ 

Procedure Arrangement( $\mathbf{x}, \tau, \xi, R'$ )
1:  $temp\_S = \{\}$ 
2:  $index1 = \text{Bin\_search}(Arr[R'][\text{inverted\_index}[\mathbf{x}], \tau)$ 
3:  $index2 = \text{Bin\_search}(Arr[R'][\text{inverted\_index}[\mathbf{x}], \xi)$ 
4:  $counter = 0$ 
5: while  $index1 + counter < index2$  or  $index1 - counter > 0$  do
6:   if  $index1 + counter < index2$  then
7:      $temp\_S = temp\_S \cup Arr[R'][\mathbf{x}][index1 + counter]$ 
8:   end if
9:   if  $index1 - counter > 0$  then
10:     $temp\_S = temp\_S \cup Arr[R'][\mathbf{x}][index1 - counter]$ 
11:   end if
12:    $counter = counter + 1$ 
13: end while
14: return  $temp\_S$ 

Procedure Ring-termination-condition( $top\_k, \mathbf{x}$ )
1: static  $flag = 0$ 
2:  $ring\_top\_k = k$  nearest neighbors of  $\phi(\mathbf{x})$ 
3: for  $i = 1$  to  $k$  do
4:   Merge  $ring\_top\_k$  and  $top\_k$ 
5:   if  $top\_k$  modified then
6:      $flag = 0$ 
7:   else
8:      $flag = flag + 1$ 
9:   end if
10: end for
11: if  $flag == num\_unproductive\_rings$  then
12:   return True
13: end if
14: return False

```

Figure 8: Algorithm for top- k retrieval

tification of the ring to which the new instance belongs and an update of the indexing structure of the ring. Identification of the ring requires $O(\log(|G|))$ time, $|G|$ being the number of rings. Updating the index struc-

ture within the selected ring requires $O(g)$ time, g being the number of instances in the ring. Insertion of instances does change the central instance. We are interested in an approximate central instance, which can

roughly ensure that the instances are evenly distributed in each ring. Addition of fresh instances does not disturb this situation, and hence the re-computation of the central instance is not mandatory. However, when the number of instances added is high compared to the existing dataset size, the possibility of a skewed distribution of the instances in the rings is higher. In such a case a re-computation of the central instance and the index would be beneficial. If we assume that the current set of instances in the dataset is representative of the distribution of instances, the approximate central instance represents a viable choice even after the insertion of new instances into the database. We discuss the details in the online version of the paper at http://www.cs.ucsb.edu/~panda/sdm_complete.pdf.

4.4 KDX-changing kernel parameters

In this section we discuss methods that allow us to perform indexing using the existing indexing structure when the kernel parameters can change. The form of the kernel function is assumed to remain the same. That is, if we had built the index using the Gaussian kernel, we would continue using the Gaussian kernel, but the parameter σ to the kernel would be allowed to change.

Suppose we wish to look at the ordering of the angles made by instances with a fixed instance say \mathbf{x}_f . We are interested in the values taken on by the function $K(\mathbf{x}_i, \mathbf{x}_f)$, where \mathbf{x}_i is any instance in the dataset. Consider the Gaussian kernel. The values of interest are given by $K(\mathbf{x}_i, \mathbf{x}_f) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_f\|^2}{2\sigma^2})$. Since the exponential function is monotonic in nature, the ordering of instances based on their angular separation from \mathbf{x}_f does not change with a change in parameter σ . The same follows for the Laplacian kernel. The polynomial kernel which has the form $(1 + \mathbf{x}_i \cdot \mathbf{x}_f)^p$ is also monotonic in nature if $p \geq 1$ and $\mathbf{x}_i \cdot \mathbf{x}_f \geq 0, \forall \mathbf{x}_i$.

Replacing \mathbf{x}_f by the central instance, we see that the ordering of instances based on their angular separation with the central instance does not change with change in the kernel parameter. Effectively, this means that the grouping of instances into rings, given a particular form of the kernel function, is invariant with change in the kernel parameter. Further, each row of the intra ring index is essentially the ordering of the instances in the ring based on their angular separation with the instance associated with that row in the ring. Again, these orderings are unaffected by changes in the value of the kernel parameter.

The functioning of the indexing approach outlined before locates a given angle in the sorted array of angles using binary search. Now, after changing the kernel parameter, we do not have the values of the angles which were used to construct the array. But, since the ordering

is unchanged, we can compute the values on the fly when we access an instance in the course of the binary search operation.

Finally, we turn our attention to the inter-group index. Since this index stores the closest instance from the adjacent group, the monotonic nature of the kernel functions implies that this index is completely unchanged. Thus, the old indexing structure can be used unchanged by computing only the required values when necessary. Since binary search in an array of size g takes $O(\log g)$ time, therefore the extra computations that need to be performed are of the order $O(\log g)$ for each binary search operation.

5 Experiments

Our experiments were designed to evaluate the effectiveness of KDX using a variety of datasets, both small and large. We wanted to answer the following questions:

- Are the top- k instances chosen by KDX of good quality?
- Quantitatively, how good are the results in terms of their distances from the hyperplane?
- How effective is KDX in choosing only a subset of the data to arrive at the results?
- How does the change in parameters (number of instances per ring and kernel parameter) affect the performance of KDX?

Our experiments were carried out on four UCI datasets [4], a 21k-image dataset, and a 300k-image dataset (obtained from Corbis). The four UCI datasets were selected because of their relatively large sizes; the two selected image-datasets have been used in several research prototypes [7]. The details of the datasets are presented in Table 1. In our experiments on top- k retrieval we obtained results for $k = 10, 20$ and 50 for the Corbis dataset, and $k = 20$ for the rest of smaller datasets. The experiments were carried out with the Gaussian kernel.

UCI Datasets We chose four UCI datasets—namely, Seg, Wine, Ecoli and Yeast.

Seg: The segmentation dataset was processed as a binary-class dataset by choosing its first class as the target class, and all other classes as the non-target classes. We then performed a top- k query on the first class.

Wine: The wine recognition dataset comes from the chemical analysis of wines grown in the same region of Italy but derived from three different cultivators. Each instance has 13 continuous features associated with it. The dataset has 180 instances. We performed three top- k queries on their three classes.

<i>Dataset</i>	<i># Classes</i>	<i># Training</i>	<i># Testing</i>
Seg	1	109	103
Wine	3	93	87
Yeast	10	747	737
Ecoli	8	165	171
21-k Image	116	4,321	16,983
Corbis	1,173	1,789	312,712

Table 1: Dataset description

Yeast: The yeast dataset is composed of predicted attributes of protein localization sites. The dataset contains 1,484 instances with eight predictive attributes and one name attribute. Only the predictive attributes were used for our experiments. This dataset has ten classes, but since the first three classes constitute nearly 77% of the data, we used only these three. **Ecoli:** This dataset also contains data about the localization pattern of proteins. It has 336 instances, each with seven predictive attributes and one name attribute. It has eight classes out of which the first three represent roughly 80% of the data and hence were used for our experiments.

21-k Image dataset The image dataset was collected from the Corel Image CDs. Corel images have been widely used by the computer vision and image-processing communities. This dataset contains 21-K representative images from 116 categories. Each image is represented by a vector of 144 features including color, texture and shape features [7].

Corbis dataset Corbis is a leading visual solutions provider (<http://pro.corbis.com/>). The Corbis dataset consists of over 300,000 images, each with 144 features. It includes content from museums, photographers, filmmakers, and cultural institutions. We selected a subset of its more than one thousand concepts.

The number of training and test instances vary slightly with the different classes in the same dataset because of differences in the number of positive samples in each class. The samples were randomly picked from both positive and negative classes. In the case of the smaller datasets (Seg, Wine, Yeast and Ecoli), the percentages of positive and negative samples picked were equal. We chose 50% of the entire dataset was chosen as training data. For the larger datasets (21-k image and the Corbis) the percentage of positive samples picked was higher (50%) than the percentage of negative samples chosen. This was done to ensure that the large volume of negative samples does not affect the SVM training algorithm, which is sensitive to imbalances in the sizes of the training and testing datasets. The details of the separation of the datasets are presented in Table 1.

5.1 Qualitative evaluation

Given a query, KDX performs a relevance search to return the k farthest instances from the query hyperplane. To measure the quality of the results, we first establish a benchmark by scanning the entire dataset to find the top- k instances for each query: this constitutes the “golden” set. The metric we use to measure the query result is *recall*. In other words, we are interested in the percentage of top- k golden results retrieved by KDX. Results for the qualitative evaluation are presented in the second column of Table 2. The results are averaged over three classes for all the datasets except for Seg. The average recall values for all datasets are above 80%. For the Corbis dataset, which has the largest number of instances, we have an average recall of 90% with less than 4% of data evaluated. (We report recall vs. fraction of data evaluated in Section 5.3.) The recall values are reasonably high for all the datasets.

5.2 Evaluation of discrepancy

This quantitative evaluation involved finding the discrepancy between the average distance to the hyperplane from the top- k instances found by KDX, and the average distance to the hyperplane from the top- k instances in the “golden” set. To obtain a percentage, we divide the average discrepancy by the difference of the distances of the most positive and least positive instances in the dataset. The results showing the percentage of average discrepancy for all the datasets are presented in the third column of Table 2. The low values of the percentage of average discrepancy indicate that even if the retrieved instances may not exactly match the golden set of top- k instances, they are comparable in their distances from the hyperplane. None of the datasets has more than 0.3% average discrepancy with the values being very low for the large datasets.

5.3 Percentage of data evaluated

This evaluation aimed to find the percentage of data evaluated before we obtained the best results using the indexing strategy. In other words, we were interested in finding approximately how quickly KDX converged on its set of best results. The results are reported in the fourth column of Table 2. These values are mostly very low (lower than 10%) except in the case of the smaller datasets where, because of the small size of the dataset, the percentage of evaluated samples, even with a small number of samples being evaluated, tends to be high. For the large datasets, we find that the results are impressive with less than 4% of the data being evaluated to reach 90% recall.

Figure 10 gives a detailed report of the percentage of average discrepancy, percentage of evaluated samples,

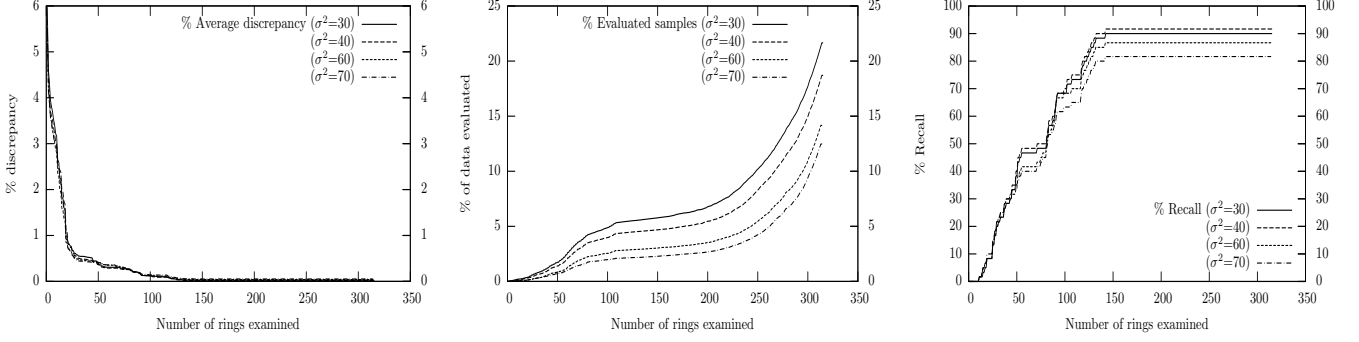


Figure 9: Corbis dataset: variation with change in σ^2 from 30 to 70

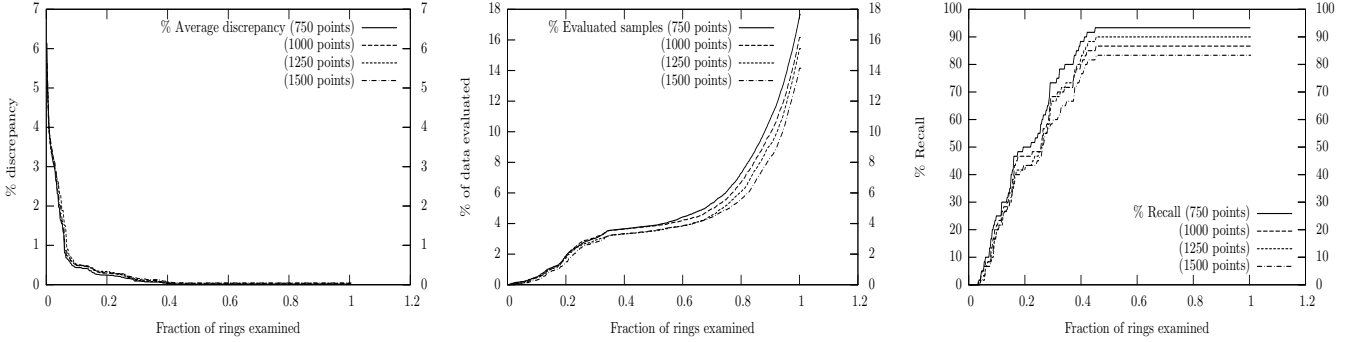


Figure 10: Corbis dataset: variation with change in number of points per ring from 750 to 1500 ($\sigma^2 = 50$)

and the change in recall as the number of rings increases. In each of the graphs, the x -axis depicts the fraction of the total number of rings processed, and the y -axis depicts the different quantities of interest. The recall (presented in the right-most graph in Figure 10) reaches a peak early in the evaluation with only a few instances being explicitly evaluated (presented in the middle graph). The discrepancy falls to its lowest level with roughly 4% of the data being evaluated (presented in the left-most graph).

5.4 Changes in parameters

This set of experiments focused on two different parameters. In the first set of experiments, we were interested in evaluating the performance of the indexing strategy when the kernel parameter (in this case σ of the Gaussian kernel) was changed after the index had been constructed. The second set of experiments evaluated the performance of the indexing strategy when the number of instances per ring was varied.

Figure 9 shows the results obtained by varying kernel parameter σ^2 between 30 and 70 for the Corbis dataset. Here the x -axis depicts the number of rings examined and the y -axis the quantities of interest (average discrepancy, percentage of data evaluated, and recall). As σ decreases, the angular separation between instances increases, and so does the width of each ring. This affects recall since with wider rings KDX

can miss instances as shown in Figure 7(a). However, the extremely low discrepancy values indicate the high quality of the selected instances. Figure 10 shows the results of changing the number of points in the rings for the Corbis dataset from 750 points to 1,500. Though recall generally improves when the number of instances per ring decreases, the percentage of evaluated instances increases. The above results indicate that changes in kernel parameters and number of points in the ring within reasonable limits do not significantly affect KDX’s performance.

We also experimented with different k values for the Corbis dataset. The results of $k = 10$ and $k = 50$ are reported in Table 3. When k is small, the recall tends to suffer slightly; when k is large, the recall can approximate 100%. In both cases, the distance discrepancy remains very small (less than 0.1%). Although KDX may occasionally miss a small fraction of the “golden” top- k instances, the quality of the top- k found is very good.

6 Conclusions

We have presented KDX, a novel indexing strategy for speeding up top- k queries for SVMs. Evaluations on a wide variety of datasets were carried out to confirm the effectiveness of KDX in converging on relevant instances quickly.

As future work we would like to pursue the goal

<i>Dataset</i>	<i>% Recall</i>	<i>% Discrepancy</i>	<i>% Evaluated till recall</i>
Seg	100	0	7.84314
Wine	93.3	0.27225	22.4806
Yeast	80.0	0.06603	3.547
Ecoli	100	0	17.2647
21K	85.0	0.0272883	2.8559
Corbis	90.0	0.03607813	2.94255

Table 2: Qualitative and quantitative comparison

<i>Dataset</i>	<i>Class</i>	<i>Recall</i>	<i>% Discrepancy</i>	<i>% Evaluated till recall</i>
Corbis	0	0.8	0.05241	3.7729
$(k = 10)$	1	1	0	1.82111
	2	0.7	0.119966	2.91755
Corbis	0	0.98	0.000324724	3.83965
$(k = 50)$	1	0.96	0.00851683	1.84253
	2	0.9	0.036358	3.06362

Table 3: Results with varying k

of further lowering the number of instances to be evaluated. We would also like to develop bounds on the number of instances that KDX evaluates. Another objective would be to lower the size of the index structure used by KDX. Currently, the index structure takes up $O(n g)$ space (g being the number of instances in each ring). Although the dataset itself takes up $O(n d)$ space (d being the dimensionality of each feature vector), the size of the index structure can quickly become very large. We would like to explore avenues restricting the size of the index.

References

- [1] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *SIGMOD Conference*, 2001.
- [2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R^* tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 322–331, 1990.
- [3] S. Berchtold, D. Keim, and H.P. Kriegel. The X-tree: An index structure for high-dimensional data. In *22nd Conference on Very Large Databases, Bombay, India*, pages 28–39, 1996.
- [4] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
- [5] M. Brown, W. Grundy, D. Lin, N. Christianini, C. Sugnet, M. Jr, and D. Haussler. Support vector machine classification of microarray gene expression data. 1999.
- [6] Christopher J.C. Burges. Geometry and invariance in kernel based methods. In Alex J. Smola Bernhard Schölkopf, Chris Burges, editor, *Advances in Kernel Methods*. MIT Press Cambridge, MA, 1998.
- [7] E. Chang, K. Goh, G. Sychay, and G. Wu. Content-based soft annotation for multimodal image retrieval using bayes point machines. *IEEE Trans. on Circuits and Systems for Video Technology Special Issue on Conceptual and Dynamical Aspects of Multimedia Content Description*, 13(1):26–38, 2003.
- [8] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *Proc. 23rd Int. Conf. on Very Large Databases*, pages 426–435, 1997.
- [9] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *The VLDB Journal*, pages 518–529, 1999.
- [10] Michael E. Houle and Jun Sakuma. Fast approximate similarity search in extremely high-dimensional data sets. In *ICDE*, 2004.
- [11] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag.
- [12] Norio Katayama and Shin’ichi Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 369–380, 1997.
- [13] D. A. Keim. Tutorial on high-dimensional index structures: Database support for next decades applications. In *Proceedings of the ICDE*, 2000.
- [14] Hyunsoo Kim, Peg Howland, and Haesun Park. Dimension reduction in text classification using support vector machines. *Journal of Machine Learning Research*, to appear.
- [15] Chen Li, Edward Chang, Hector Garcia-Molina, and Gio Wilderhold. Clindex: Approximate similarity queries in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(4), July 2002.
- [16] King-Ip Lin, H. V. Jagadish, and Christos Faloutsos. The TV-tree: An index structure for high-dimensional data. *VLDB Journal: Very Large Data Bases*, 3(4):517–542, 1994.
- [17] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. *ACM International Conference on Multimedia*, pages 107–118, 2001.
- [18] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In Pat Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 999–1006, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.
- [19] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.
- [20] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 194–205, 24–27 1998.

Parallel Computation of RBF Kernels for Support Vector Classifiers*

Shibin Qiu[†]

Terran Lane[‡]

Abstract

While kernel support vector machines are powerful classification algorithms, their computational overhead can be significant, especially for large and high-dimensional data sets. A recent biomedical dataset, for instance, could take as long as 3 weeks to compute its RBF kernel matrix on a modern, single-processor workstation. In this paper, we develop methods for high-performance parallel computation of kernel matrices. There are two key components to a parallel implementation: distribution of the computation across nodes and communication to combine the results. To address the first, we employ a dimension-wise data partition that yields efficient computation and low communication overhead during the initial phase. This partition provides dramatic speedups on large and high-dimensional data, applies to a wide variety of kernel functions, and is an exact computation, producing the same kernel matrix as its sequential implementation. To address communication needs during the second phase, we introduce an approximation specific to the Gaussian RBF kernel that yields sparse partial kernel matrices and, thus, efficient communication. We analyze the approximation error of this method, demonstrating that it falls off exponentially with N , the parameter of the approximation. We also examine the positive definiteness of the approximation with respect to Mercer's condition and show that (a) in the limit of N our approximation becomes positive definite for any data set and (b) for a fixed data set, there exists a finite N yielding a positive definite kernel matrix. We also give a simple iterative method for selecting N to yield a positive definite kernel matrix on any fixed data set. In practice, we find that positive definiteness is achieved on all of the data sets we examine with very small N (2–5). Finally, we test the empirical performance of our two methods on a variety of large, real-world data sets, demonstrating large computational speedups with little or no impact on accuracy.

Keywords: Parallel algorithms, Support vector machines, Kernel approximation, SVM performance.

*This work is supported by NIH Grant Number 1P20RR18754 from the Institutional Development Award (IDeA) Program of the National Center for Research Resources.

[†]Computer Science Department, University of New Mexico, sqiu@unm.edu.

[‡]Computer Science Department, University of New Mexico, terran@cs.unm.edu.

1 Introduction

Kernel support vector machines have gained prominence in recent years for their strong classification performance on a variety of linearly nonseparable and high-dimensional data. Unfortunately, particularly large and high-dimensional data sets, such as often arise in biomedical applications, may require substantial effort simply to compute the kernel matrix itself. For example, computing a Gaussian radial basis function (RBF) kernel matrix for a recent drug design database could require weeks on a current, high-end workstation. In this paper, we develop methods for a high-performance parallel implementation of the kernel computation.

There are two key components to a parallel implementation: distribution of the computation across nodes and communication to combine the results. To address the first component, we employ a dimension-wise data partition that yields efficient computation and low communication overhead during the initial, distributed computation, phase. This partition applies to a wide variety of kernel functions (e.g., those that are scalar functions of inner products) and is an exact computation, producing the same kernel matrix that would be computed on a uniprocessor system.

Even with efficient distributed computation, however, the communication overhead incurred in combining the partial results is substantial and significantly detracts from theoretical peak performance. To address communication overhead during this second, result combination, phase of the computation, we introduce an approximation specific to the Gaussian RBF kernel that yields extremely sparse partial kernel matrices and, therefore, low bandwidth requirements. This method employs a series of N rectangles of width τ to approximate the RBF kernel function. We analyze the approximation error of this method, demonstrating that it falls off exponentially with N . We also examine this approximation with respect to Mercer's positive definiteness condition. First we demonstrate that, in the limit of N , our approximation converges to the true RBF kernel and, therefore, obeys Mercer's condition for any data set. Unfortunately, for fixed τ and finite N , the approximate kernel matrix may not be positive definite for some data sets. However, we show that for any

fixed data set it is possible to choose N and τ so that the resulting approximate kernel matrix is positive definite. In practice, we employ a simple iterative procedure for selecting N for a specific data set, X , to yield a sparse, positive definite kernel matrix for that X . In practice, we find that positive definiteness is achieved on all of the data sets we examine with very small N (2–5), yielding extremely sparse matrices (fill factors of roughly 2%).

Finally, we examine the empirical performance of our two methods on a variety of large, real-world data sets including microarray data analysis, classification of functional magnetic resonance imaging (fMRI) neuroimaging data, and drug design. We find that our parallel implementation is able to achieve substantial speedups – up to 24 folds for the dimension-wise partition alone and 45 folds for the two approaches combined. With the two methods combined, the speedup is nearly linear in the number of processors dedicated to the computation up to 32 nodes in the cluster. Simultaneously, the approximate RBF kernel method produces small impact on classification accuracy – less than 1% in the cases we tested.

This paper is organized as follows. In Section 2, we analyze the necessity for parallel computing for large datasets. Section 3 discusses related works for enhancing SVM performance. Section 4 presents the dimension-wise partition method. Section 5 introduces the kernel approximation method and proves for its positive definiteness. Performance tests are conducted in Sections 4 and 5. Comprehensive experiments are given in Section 6, where more tests are conducted by combining the dimension-wise partition and the kernel approximation method. We conclude the paper in Section 7. Some related mathematical derivations are given in the appendices.

2 The Need for Parallel Computing

Higher dimensional datasets usually require more data points. Data for text mining has high dimensionality [12]. Prediction for biological sequences using string kernels also involves computation of high dimensional data [16]. Fortunately, kernels for these structured data can be implemented efficiently and direct computing in the high dimensional space can be avoided.

However, for some datasets, kernel computation must be conducted by direct matrix operations in the input space. The thrombin dataset [25], which was developed for the 2001 KDD cup, has 139,351 features and 2,543 data points. Computing an RBF kernel on this dataset requires a considerable amount of processing time. The CDK2 drug design dataset has 14,223 compounds, each of which has 35,926,557 descriptors [28]. This data set requires a few Tera bytes of memory

space, which is beyond the capacity of most uniprocessor computers. However, training time is an even significant barrier. The time complexity of computing an RBF kernel matrix is $O(m^2n)$ [4], where m is the number of data points and n is the dimension of input space. A computer with sufficient memory and the capacity of 4 GFlops per second, the common speed of today’s high end computer, would need 21 days to compute the kernel matrix for this data set (details in Appendix 1). And it takes much longer if memory is not enough. Modern high density microarrays have dimensions of more than 60 million [20]. If a dataset generated by this type of microarray has a few thousand data points, using the data representation as used by Brown et al. [1], then a few Tera bytes of space will be needed just for storage of the data. A uniprocessor computer would need a few weeks to compute the kernel matrix for this dataset (Appendix 1). Although the capacity of a uniprocessor computer may double in a few years to come, the kernel computing time will still be a matter of weeks. And the sizes of datasets are increasing constantly. Thus, even though the space requirement might be satisfied some time in the future by installing Tera byte memories in uniprocessor computers, the computing time is still inhibitive. Advanced feature selection can be used to reduce the dimension of the data and improve the training efficiency for some datasets. But the preprocessing in feature selection also takes considerable amount of time. In addition, direct use of the original features are preferred for some datasets. For instance, each element in a microarray represents a particular segment of sequence from the genomic data and is not intended to be replaced or mixed with any other element. For datasets whose dimensions can be reduced by feature selection, classification accuracy based on original features can be used as a reference for comparison with the feature selection approaches. Therefore, parallel computing is necessary for the improvement of computational performance for large and high dimensional data. A computer cluster of 512 nodes, with proper implementation of the algorithm, can reduce the training time for such large datasets roughly to a couple of hours, or less than an hour. Parallel computing can also use distributed memory in the system and satisfy larger space requirement.

Empirical results show that on a single computing node 85% to 98% of an SVM’s training time is spent on computing its kernel matrix. This phenomenon has also been observed by other authors such as in [18]. Therefore, improving the computational performance of an SVM depends on enhancing its kernel computing efficiency.

3 Related Work

Kernel computation and quadratic programming for training an SVM take substantial amount of time. To make the quadratic search faster, Joachims invented *SVM^{light}* [11], which selects a smaller working set than the training set at each iteration and reduces the amount of computation. Similar strategies were used to achieve faster performance in quadratic programming to solve the SVMs [9][13][21], while other authors have introduced different variants of SVMs, such as SMO [22] and SSVM [15]. To reduce space and time consumption, an approximate SVM, the RSVM was introduced [14], which randomly selected a subset from the training set and trained on this reduced subset. The proximal SVM (PSVM) was introduced as an alternative to quadratic programming [7], as it only depends on linear algebraic operations. Incremental training methods were proposed for PSVM to save space and to unlearn old data [8], where only linear classifiers without kernels were considered. An incremental learning algorithm was suggested for high dimensional data, but for only for the linear SVM [5]. This training method for high dimensional data used linear algebraic manipulations based on the Sherman-Morrison-Woodbury formula and is not applicable for nonlinear kernels. Leslie et al. [16] proposed an efficient implementation for their string kernels, which is, however, not applicable to general numerical data. Mahe et al. [18] introduced efficient algorithms for computing graph kernels, which are again, not suitable for general numerical data.

A parallel incremental learning algorithm was proposed for the linear PSVM [26]. A data parallel approach was proposed for Gaussian process regression using local learning method [2]. But the parallelism achieved in computing the compactly supported covariance function is based on the assumption that training data can be clustered into disjoint and spatially localized subsets. In practice, when this assumption is not met, a kernel matrix cannot be efficiently computed because each computing node needs to access data from all other nodes. A parallel mixture of SVMs was implemented and performance was improved on large data sets [3]. However, this work focused on reducing the impact of large number of training examples on computational efficiency (similar to RSVM [14]). It did not study the effect of high dimensionality and did not yield a unified kernel matrix over all training data. It is also noted that most related works use linear SVMs and avoid kernels for large and high dimensional data sets due to the cost of kernel computation. And parallel computation has only been implemented for linear PSVM without kernels [26] because of the difficulty in parallel implementation for kernel computation.

To parallelize the computation of a nonlinear kernel, we employ the dimension-wise partition to distribute the calculation across computing nodes with minimum data access overhead. To further improve parallel performance, we minimize communication cost by introducing a novel kernel approximation method. Our parallel algorithms are analyzed and proved theoretically, and tested for large data sets on a computer cluster.

4 The Dimension-wise Partition for Parallel Kernel Computation

In this section, we present the dimension-wise partition for parallel computation of RBF kernel matrices.

4.1 Kernel based support vector machines For the sake of notation, we first briefly summarize the optimization problem and the classifier function of a kernel based SVM. The nonlinear support vector machine can be formulated as follows,

$$(4.1) \quad \min \quad \nu \frac{1}{2} \|y\|^2 + \frac{1}{2} (u^\top u + \gamma^2) \\ \text{subject to} \quad D(K(X, X^\top)Du - e\gamma) + y \geq e,$$

where $X \in \mathbb{R}^{m \times n}$ is the training data matrix, m is the number of data points in the training set and n is the dimension of input space [19]. We use X_i to denote the i^{th} row of matrix X , representing the i^{th} data point. $D \in \mathbb{R}^{m \times m}$ is the diagonal label matrix, $D_{ii} = 1(-1)$, if the label at the i^{th} data point is $1(-1)$. $u \in \mathbb{R}^m$ and $\gamma \in \mathbb{R}$ are to be solved. ν is a weighting parameter for the separation error vector $y \in \mathbb{R}^m$. $e \in \mathbb{R}^m$ is a vector of all ones. $K = K(X, X^\top) \in \mathbb{R}^{m \times m}$ is the kernel matrix between X and X^\top . The optimization problem of (4.1) is generally solved through quadratic programming. When it is solved, the classifier function for an input data point x is,

$$(4.2) \quad f(x) = \text{sgn} \left(\sum_{i=1}^m \alpha_i D_{ii} k(x, x_i) + \gamma \right),$$

where the nonzero α_i determine the support vectors.

PSVM, on the other hand, solves (4.3) below, a variant optimization problem of (4.1) using linear algebra operations only and obtains equivalent quality on training and generalization [7].

$$(4.3) \quad \min \quad \nu \frac{1}{2} \|y\|^2 + \frac{1}{2} (u^\top u + \gamma^2) \\ \text{subject to} \quad D(K(X, X^\top)Du - e\gamma) + y = e.$$

And its classifier function is,

$$(4.4) \quad f(x) = \text{sgn} ((K(x^\top, X^\top)K(X, X^\top) + e^\top)Dv).$$

We implement the PSVM in this paper. But the kernel computation methods are equally applicable to other forms of SVMs.

4.2 The dimension-wise partition We formulate the dimension-wise partition method using the RBF kernel. In the RBF kernel family, a kernel function of two vectors x and x' only depends on the distance between them, and is of the form, $K(x, x') = K(x - x')$. A commonly used kernel function is the Gaussian kernel. The i^{th} row and j^{th} column entry in a Gaussian kernel matrix is evaluated as,

$$(4.5) \quad (K(X, X^T))_{ij} = \exp(-\|X_i - X_j\|^2 / 2\sigma^2).$$

We use $r = \|x - x'\|$ to denote the Euclidian distance between vectors x and x' , and write the kernel function as $k(r) = K(x, x') = e^{-r^2/2\sigma^2}$.

One common transform used for the calculation of (4.5) is to use the identity of $\exp(-\|x - x'\|^2) = \exp(2x^T x' - \|x\|^2 - \|x'\|^2)$. However, this formula does not eliminate the cross evaluation between the vectors x and x' and still has $O(m^2n)$ complexity.

Parallel computation of a kernel matrix involves two major parts: simultaneous calculation on all nodes and communication to combine the intermediate output into the final result. One way to compute the kernel matrix in parallel is to partition the data row-wise such that each computing node is assigned a subset of the training data. However, this row-wise partition has serious problems. First, communication cost is too high, as computing each element in the kernel matrix on a node requires data from all other nodes. Second, when combining the partial matrices to obtain the final kernel matrix, the computation is complicated and the time complexity increases. Therefore, the row-wise partition does not improve computational performance and only deteriorates performance. In fact any partitioning involving divisions among the rows has this problem of accessing data across nodes and has high communication overhead. Therefore, dividing the data matrix into blocks also results in low performance.

When m and n are substantially different, linear SVM can take advantage of exchanging rows and columns by using the Sherman-Morrison-Woodbury formula [5]. However, this technique is not applicable for SVMs with nonlinear kernels [14].

In the following, we show how to efficiently partition the data for parallel kernel computing and how to further reduce communication cost. Specifically, we propose the dimension-wise partition and show that it indeed speeds up computation. Communication performance is dealt with in Section 5.

Parallel calculation of the kernel function between

vector X_i and X_j using dimension-wise partition can be formulated as follows. We partition the data matrix X in the following way:

$$(4.6) \quad X = (X^1 | X^2 | \dots | X^p).$$

Each partition X^k is of dimension $m \times l$, where $l = n/p$ and p is the total number of processing nodes in the computing system. We assume l is an integer. The i^{th} row of X^k is $X_i^k = (X_{i1}^k, X_{i2}^k, \dots, X_{il}^k)$. For Gaussian kernel,

$$(4.7) \quad \begin{aligned} r_{ij}^2 &= \|X_i - X_j\|^2 \\ &= (X_{i1} - X_{j1})^2 + (X_{i2} - X_{j2})^2 + \dots + (X_{in} - X_{jn})^2 \\ &= (X_{i1}^1 - X_{j1}^1)^2 + \dots + (X_{il}^1 - X_{jl}^1)^2 \\ &\quad + (X_{i1}^2 - X_{j1}^2)^2 + \dots + (X_{il}^2 - X_{jl}^2)^2 + \\ &\quad \dots + (X_{i1}^p - X_{j1}^p)^2 + \dots + (X_{il}^p - X_{jl}^p)^2 \\ &= \sum_{k=1}^p N_{ij}^k, \end{aligned}$$

where N_{ij}^k is the partial squared distance computed on the k^{th} computing node,

$$(4.8) \quad N_{ij}^k = \sum_{d=1}^l (X_{id}^k - X_{jd}^k)^2.$$

Then the kernel function can be rewritten as,

$$(4.9) \quad \begin{aligned} (K(X, X^T))_{ij} &= \exp(-\sum_{k=1}^p N_{ij}^k / 2\sigma^2) \\ &= \prod_{k=1}^p \exp(-N_{ij}^k / 2\sigma^2) = \prod_{k=1}^p K_{ij}^{(k)}. \end{aligned}$$

where $K_{ij}^{(k)}$ is the element computed by node k and the multiplication is element by element (the Schur product). We can express the kernel matrix as,

$$(4.10) \quad K(X, X^T) = \prod_{j=1}^p K^{(j)}.$$

Thus, node j computes a partial kernel matrix $K^{(j)}$ based on its partition of the data without accessing data from other nodes, and a coordinator node combines these partial kernel matrices by multiplying them together to obtain the Schur product.

For an $m \times m$ kernel matrix, the partial matrices are also of dimension $m \times m$. The computing cost on each node is $O(m^2n/p)$. The communication cost for

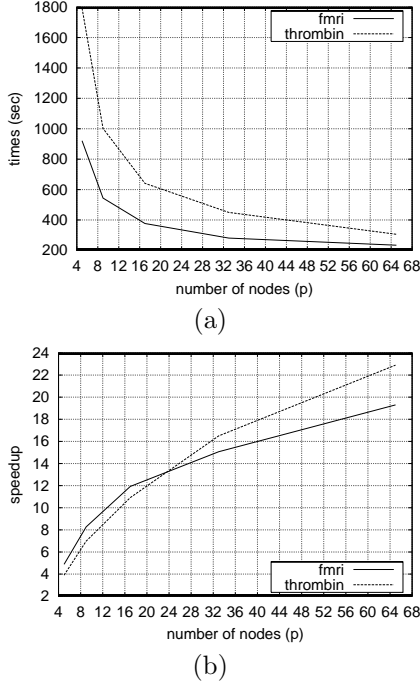


Figure 1: Parallel SVM classification times (a) and speedups (b) for fMRI and thrombin datasets using dimension-wise partition and Gaussian kernels.

sending these partial kernel matrices for combining is proportional to their sizes. If we neglect the startup cost, then the communication cost is approximately $T_{comm} = \eta m^2$, where η is a coefficient. Therefore the speedup can be represented by,

$$(4.11) \quad Sp(p) = \frac{m^2 n}{m^2 n/p + \eta m^2} = \frac{p}{1 + p\eta/n}.$$

If n is large and the communication cost is small, the speedup is close to linear in p , the number of computing nodes. Thus, dimension-wise partition can approximately achieve linear speedup. It is especially effective for high dimensional datasets and computing systems with fast interconnections.

Figures 1 shows the experimental results with the dementia fMRI dataset [6] and the thrombin dataset [25] on a cluster of 65 nodes. The dementia dataset contains $m = 2,500$ data points and has a dimension of $n = 65,536$. The purpose of this dataset is to classify a patient as demented or normal based on the patient's fMRI images. The task of the thrombin dataset is to classify a chemical compound into the active class or the inactive class. Figure 1 demonstrates speedups are achieved by using the dimension-wise partition. When p is less than 16, the speedups are roughly linear. But the speedups slow down as the number of

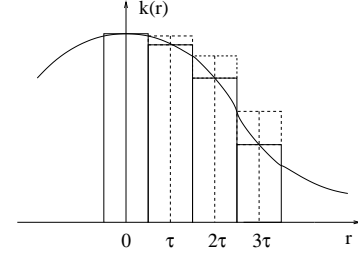


Figure 2: Approximating a Gaussian kernel with rectangle functions having widths τ , and heights of the kernel function evaluated in the center of the rectangles.

computing nodes increases. This decrease of speedup is because of the communication cost incurred during the process of combining the partial kernel matrices in (4.10). If communication cost is not reduced, the speedup will be saturated at some point. Since the thrombin dataset has larger dimension, it achieves more speedup when the cluster has more than 24 nodes. This is because the thrombin dataset has a larger computing task and each node is assigned larger load to compute and simultaneous calculation increased the speedup. In this case, computing cost dominates the overall performance. On the other hand, the computing task is relatively small for the fMRI dataset, and the communication cost becomes more evident and its speedup is decreased.

The communication cost for transferring the partial kernel matrices is proportional to their sizes. We use an approximation method in calculating the RBF kernel function, which makes the partial kernel matrices sparse. We then only need to send the nonzero elements by using sparse matrix technology. In the next section, we investigate the approximation method.

5 RBF Kernel Approximation for Communication Cost Reduction

In this section we introduce an approximation method for RBF kernel calculation and use it to improve communication performance.

5.1 RBF kernel approximation using rectangle functions We first show an RBF kernel can be approximated by an infinite series of rectangle functions and then truncate the series for implementation to obtain sparse matrices. We can approximate an RBF kernel function $k(r)$ by a series of rectangle functions,

$$(5.12) \quad k(r) \approx \hat{k}(r) = \sum_{j=-\infty}^{\infty} k(j\tau) a(r - j\tau),$$

where $\tau > 0$ is the width of the rectangles, $a(r - j\tau)$ is the shifted rectangle function of $a(r)$ [10], and $k(j\tau)$ is the value of $k(r)$ evaluated at points of $r = j\tau$, $j \in \mathbb{Z}$. The solid rectangles in Figure 2 denote the shifted rectangle functions.

According to the Mercer's theorem (finite case), if a symmetric function $k(x, x')$ satisfies the following positive definiteness condition, then it is a legitimate kernel function for SVM [24],

$$(5.13) \quad \int_{\chi \times \chi} k(x, x') f(x) f(x') dx dx' \geq 0 \quad \forall f \in L_2(\chi),$$

where $k : \chi \times \chi \rightarrow \mathbb{R}$ is any symmetric function and is square integrable in $\chi \times \chi$. For an RBF kernel, Mercer's condition can be expressed through its Fourier transform as,

$$(5.14) \quad (2\pi)^{\frac{n}{2}} \int_{\chi} |\tilde{f}(\omega)|^2 \tilde{k}(\omega) d\omega \geq 0,$$

where $\tilde{k}(\omega)$ is the Fourier transform of $k(r)$, and $\tilde{f}(\omega)$ is the Fourier transform of $f(x)$, and n is the dimension of $x \in \chi$ (details in Appendix 2). For $k(r)$ to satisfy (5.14), we need

$$(5.15) \quad \tilde{k}(\omega) \geq 0 \quad \forall \omega \in \chi.$$

A Gaussian kernel, for instance, has a Fourier transform in the form of Gaussian function [10], whose value is nonnegative in frequency domain, therefore satisfying Mercer's condition (5.15). The Fourier transform of the rectangle function $a(r)$ is $\tilde{a}(\omega) = \tau \text{sinc}(\frac{\tau\omega}{2})$ and has negative values for $\omega \in (-\infty, \infty)$ [10]. Therefore a single rectangle function as the approximation of an RBF function does not satisfy Mercer's condition. However, the sum of an infinite series of weighted rectangle functions, as in (5.12), satisfies the Mercer's condition when τ is infinitely small, as shown in the following theorem.

THEOREM 5.1. *Given a positive definite kernel $k(r)$, the approximate kernel $\hat{k}(r)$ in (5.12) is also positive definite when $\tau \rightarrow 0$.*

Proof: We first take the Fourier transform of $\hat{k}(r)$ in (5.12) as follows.

$$\begin{aligned} (5.16) \quad \tilde{\hat{k}}(\omega) &= \mathcal{F}\{\hat{k}(r)\} = \mathcal{F}\left\{\sum_{j=-\infty}^{\infty} k(j\tau) a(r - j\tau)\right\} \\ &= \sum_{j=-\infty}^{\infty} k(j\tau) \mathcal{F}\{a(r - j\tau)\} \\ &= \sum_{j=-\infty}^{\infty} k(j\tau) \tau \text{sinc}\left(\frac{\tau\omega}{2}\right) e^{-i\tau j\omega}, \end{aligned}$$

where $\mathcal{F}\{\cdot\}$ is the Fourier transform operator, and $i = \sqrt{-1}$. We then find the limit of the Fourier transform of (5.16), when τ approaches zero as follows.

$$\begin{aligned} (5.17) \quad \lim_{\tau \rightarrow 0} \tilde{\hat{k}}(\omega) &= \lim_{\tau \rightarrow 0} \sum_{j=-\infty}^{\infty} k(j\tau) \tau \text{sinc}\left(\frac{\tau\omega}{2}\right) e^{-i\tau j\omega} \\ &= \lim_{\tau \rightarrow 0} \sum_r k(r) e^{-ir\omega} \tau = \int_{-\infty}^{\infty} k(r) e^{-ir\omega} dr \\ &= \mathcal{F}\{k(r)\} = \tilde{k}(\omega). \end{aligned}$$

In evaluating the above limit, we use the fact that $\lim_{x \rightarrow 0} \text{sinc}(x) = 1$, and $\lim_{\tau \rightarrow 0} j\tau = r$. Thus, (5.17) indicates that if an RBF kernel function satisfies Mercer's condition, $\tilde{k}(\omega) \geq 0, \forall \omega \in \chi$, then the limit of the Fourier transform of its approximation also satisfies the Mercer's condition. \square

To use (5.12) to approximate the RBF kernel function, we truncate the series and use a fixed value for τ . Equation (5.12) becomes,

$$(5.18) \quad k(r) \approx \check{k}(r) = \sum_{j=-N}^N k(j\tau) a(r - j\tau).$$

Since the value of the approximate kernel function in (5.18) is zero for $|r| > N\tau$, the kernel matrix contains many zeros. For a given dataset X , we can pick τ and N so that the approximate kernel matrix is positive definite, as shown in the following theorem.

THEOREM 5.2. *For any fixed and finite data set X and a positive definite RBF kernel function $k(r)$, there exist a fixed $\tau > 0$ and a finite N for which $\check{k}(r)$ in (5.18) has zero error and yields a positive definite kernel matrix.*

Proof: We note that the Gaussian RBF kernel is purely a function of the Euclidean distances between data points in the input space. Since the distance is nonnegative, we ignore the rectangles positioned to the left of the origin in Figure 2. Let $D = \{d_1, d_2, \dots, d_{m^2}\}$ be the set of all distances derived from data set X . Our problem now reduces to showing that we can construct a set of rectangles by choosing τ and N , such that every d_i falls precisely at the midpoint of one rectangle where the height of the rectangle equals the value of the kernel function. On any real computer, all d_i must be of finite precision, i.e., be rational. There exists some integer $q > 0$, such that $qD = \{qd_1, \dots, qd_{m^2}\}$ are all nonnegative integers. Then let $\alpha = \text{GCD}(\{qd_i : qd_i \neq 0\})$ be the greatest common divisor of the nonzero elements of the scaled, integral distances. Now every qd_i can be written as $b_i\alpha$ for some integer b_i . We choose $\tau = \alpha/q$ and $N = \max_i \{b_i\} + 1$. Then every d_i falls at the

midpoint of some rectangle function, and $\check{k}(d_i) = k(d_i)$ for all i . Thus, the approximation error is zero and the approximate kernel matrix is positive definite. \square

In practice, of course, this choice of τ and N is hardly a useful approximation – the resulting kernel matrix will be the same (and just as dense) as the exact kernel matrix. Theorem 5.2, however, establishes the existence of finite approximations to the kernel function that are positive definite for *fixed data sets*. For any given data set, then, our task is to find a *sparse* positive definite approximation to the kernel function. We address this question in Section 5.2.

We next show the error bounds in frequency domain and in the distance domain when the series is truncated.

THEOREM 5.3. *The sum of squared errors of the approximate kernel given by (5.18) decreases exponentially with N in the frequency domain.*

Proof: We combine the conjugate terms in (5.16) and obtain,

$$\begin{aligned} \tilde{k}(\omega) &= \tilde{a}(\omega)k(0) + \tilde{a}(\omega) \sum_{j=1}^{\infty} (e^{-i\tau j\omega} + e^{i\tau j\omega})k(j\tau) \\ &= \tilde{a}(\omega)k(0) + \tilde{a}(\omega) \sum_{j=1}^{\infty} 2\cos(j\tau\omega)k(j\tau). \end{aligned} \quad (5.19)$$

If we only take the first N terms in the summation in (5.19), we get the Fourier transform of (5.18),

$$\tilde{\check{k}}(\omega) = \tilde{a}(\omega)k(0) + \tilde{a}(\omega) \sum_{j=1}^N 2\cos(j\tau\omega)k(j\tau). \quad (5.20)$$

The error generated by the series truncation is the difference between (5.19) and (5.20),

$$\begin{aligned} e(\omega) &= \tilde{k}(\omega) - \tilde{\check{k}}(\omega) \\ &= 2\tilde{a}(\omega) \sum_{j=N+1}^{\infty} \cos(j\tau\omega)k(j\tau) = 2\tilde{a}(\omega)D, \end{aligned} \quad (5.21)$$

where D is bounded as,

$$\begin{aligned} D &= \sum_{j=N+1}^{\infty} \cos(j\tau\omega)k(j\tau) \leq \sum_{j=N+1}^{\infty} |\cos(j\tau\omega)k(j\tau)| \\ &\leq \sum_{j=N+1}^{\infty} |k(j\tau)| = \sum_{j=N+1}^{\infty} e^{-\frac{(j\tau)^2}{2\sigma^2}} = \sum_{j=N+1}^{\infty} \left(e^{-\frac{\tau^2}{2\sigma^2}}\right)^{j^2} \\ &= \sum_{j=N+1}^{\infty} \left(\frac{1}{g}\right)^{j^2} \leq \sum_{j=N+1}^{\infty} \left(\frac{1}{g}\right)^j = \frac{1}{(g-1)g^N}, \end{aligned} \quad (5.22)$$

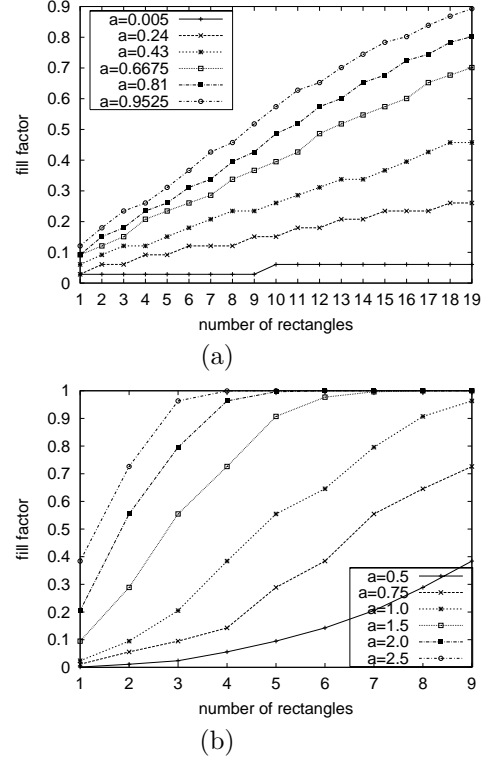


Figure 3: Fill factors of kernel matrices change with $\tau = a\sigma$, width of the rectangles, and N , the number of rectangles. (a) The ionosphere dataset. (b) The thrombin dataset.

where $g = e^{\tau^2/2\sigma^2} > 1$. Therefore $e(\omega)$ is bounded by,

$$e(\omega) \leq 2\tilde{a}(\omega) \frac{1}{(g-1)g^N}. \quad (5.23)$$

The total squared error in the frequency domain is,

$$\begin{aligned} sse(N) &= \int_{-\infty}^{\infty} e(\omega)^2 d\omega \\ &\leq \frac{4}{(g-1)^2 g^{2N}} \int_{-\infty}^{\infty} \tilde{a}(\omega)^2 d\omega \\ &= \frac{8\pi\tau}{(g-1)^2 g^{2N}}. \end{aligned} \quad (5.24)$$

In calculating the integral of (5.24), we used the fact that $\int_{-\infty}^{\infty} \text{sinc}^2(x) dx = \pi$. The actual error bound is tighter than that given in (5.24) because the geometric series used in (5.22) is a loose bound. Equation (5.24) shows that the total squared error in frequency domain decreases exponentially with N , the number of rectangles used in the approximation. \square

To evaluate the error bounds in the distance do-

Table 1: Sparseness of kernel matrices. ρ_1 and ρ_2 are the 10 fold cross validation accuracies using Gaussian kernel and the approximate kernel, respectively. ϕ is the fill factor of the kernel matrices. N is the number of rectangles used. e_ω and e_r are the error bounds in frequency domain and distance domain, respectively.

datasets	size ($m \times n$)	ρ_1	N	ϕ	ρ_2	e_ω	e_r
HD	297×14	0.99±0.032	2	0.01	0.99±0.031	0.07 σ	1.73 σ
mushroom	550×50	0.98±0.033	3	0.03	0.98±0.032	0.1 σ	1.78 σ
OC	253×15,154	0.99±0.032	5	0.018	0.99±0.033	962 σ	4.59 σ
TIS	13,375×927	0.99±0.033	2	0.01	0.99±0.034	0.01 σ	1.68 σ
thrombin	2,543×139K	0.97±0.034	2	0.14	0.96±0.035	1005 σ	45.2 σ

main, we show the total absolute error $sae(N)$ as below.

$$(5.25) \quad sae(N) \leq \left| \frac{\tau}{2} + \tau \frac{g}{g-1} \left(1 - \frac{1}{g^N}\right) - \sqrt{\frac{\pi}{2}}\sigma \right|.$$

The derivation of $sae(N)$ in (5.25) is given in Appendix 3. The behavior and performance of the approximate kernel is discussed next.

5.2 Sparseness of the approximate kernel matrix Before applying the kernel approximation method into parallel computing, we first investigate some properties of the approximate kernel. We study the relationship between the approximation parameters and the fill factors (the proportion of nonzero elements in a matrix) of the approximate kernel matrix. We show that high classification accuracies can be achieved with very small fill factors. We also describe our tuning procedure and analyze the communication reduction of the approximate kernel.

Algorithm 1 Setting Approximation Parameters

- 1: Pick $a < 1$, set $\tau = a\sigma$ and $N_0 = 2$;
 - 2: Compute approximate kernel matrix and solve SVM optimization program;
 - 3: **if** solution succeeds **then**
 - 4: return;
 - 5: **else**
 - 6: $N_{i+1} = N_i + 1$;
 - 7: Go to 2;
 - 8: **end if**
-

Since the value of the approximate kernel in (5.18) is zero for $r > N\tau$, the kernel matrix becomes sparse. Figure 3 shows how the fill factor changes with the rectangle width τ and the number of rectangles N . In Figure 3(a), we let the number of rectangles change from 1 to 19 for the ionosphere dataset [27], and observe the changes of the fill factor with different rectangle widths. In Figure 3(b), we investigate similarly on the thrombin dataset. It can be seen that the fill factor increases with larger τ and larger N . But the

relationships are not always linear. High classification accuracies can be achieved even for small τ and N . We therefore prefer lower fill factors. Very small fill factors may cause the kernel matrix to become non-positive definite. This situation can be detected when the matrix inversion operation used for solving the PSVM does not go through, or the Cholesky decomposition used for solving the PSVM encounters negative operand for the square root operation.

Table 1 shows the behavior of the approximate kernels for the five datasets. In Table 1, mushroom and HD (heart disease, also referred to as Cleveland heart) are datasets from the UCI machine learning repository [27]. Mushroom and HD are small datasets. We use them here to show the approximation accuracies. TIS (translation initiation site, based on biological sequences), OC (ovarian cancer, microarray data) are the same as in [5]. TIS, OC and thrombin [25] are large datasets and are time consuming in training. They can be sped up via our parallel computing algorithms. As shown in the table, the fill factors can be quite low while high cross validation accuracies are maintained.

Our procedure for setting the approximation parameters is summarized in Algorithm 1. First, we choose τ to be a fraction of σ . For the three datasets in Table 1, we set $\tau = 0.2\sigma$. Then starting with $N_0 = 2$, we increase N until the solution to the PSVM succeeds. If nothing goes wrong in the process of solving the PSVM, we obtain the smallest possible N . If the matrix inversion or the Cholesky decomposition when solving the PSVM does not go through, then increase N .

The reduction in communication cost by the kernel approximation method can be analyzed in the following. If we neglect the startup overhead of a data transfer, the communication cost is proportional the data size. The approximate kernel's reduction of communication time from the exact kernel can roughly be formulated as, $\beta_{comm} = \phi w_1/w_2$, where w_1 is the width of one encoded element of the sparse kernel matrix and w_2 is the element width of the exact kernel matrix, which for double precision data type is 8. By using proper

Table 2: Improvement by the approximate kernel. γ_{comm} and γ_{total} are the average reduction in communication time and total classification time, respectively.

datasets	OC	TIS	thrombin	average
γ_{comm}	90.2%	84.6%	76.4%	83.7%
γ_{total}	30.5	33.7%	29.6%	31.26%

encoding, e.g., differential encoding for matrix indices, w_1 is close to w_2 and β_{comm} is close to the fill factor ϕ given in Table 1. Therefore, the communication cost of the approximate kernel matrix is only a small percentage of that of the exact kernel matrix.

6 Experiments

In previous sections, we have independently tested the speedups of the dimension-wise partition, the sparseness of the approximate kernel matrix, the error bounds and the classification accuracies. In this section we test the combined performances of the dimension-wise partition and the approximate kernel methods in a single parallel implementation on a cluster of computers consisting of 65 nodes. We test these algorithms on the ovarian cancer, translation initiation site, and thrombin datasets. Figure 4 shows the improvement made by using the approximate kernel as compared to using the exact Gaussian kernel, for the thrombin dataset. Figure 4 demonstrates that on a cluster of size $p = 65$ nodes, a speedup of 45 folds has been achieved. From Figure 4(b) we can see that the approximate kernel achieved nearly linear speedups when p is less than 32. Figure 4 also indicates that the approximate kernel has gained more speedup when the cluster size is large. This is also evident in Figure 4(a), where for large p , the time used by the approximate kernel is reduced more compared with the situation when the number of nodes is small. This pattern of improvement is due to the effective reduction in communication time and has the potential to scale up to even larger clusters. Tests on other datasets (not shown) demonstrate similar speedups and patterns.

Table 2 shows the improvement contributed by the approximate kernel method by displaying the average reductions of communication time and total computing time that the approximate kernel has gained over the exact Gaussian kernel. In Table 2, γ_{comm} and γ_{total} are the average reduction in communication time and total classification time, respectively. They are defined as,

$$(6.26) \quad \gamma_{comm} = (T_{comm}^{exact} - T_{comm}^{app}) / T_{comm}^{exact},$$

$$(6.27) \quad \gamma_{total} = (T_{total}^{exact} - T_{total}^{app}) / T_{total}^{exact},$$

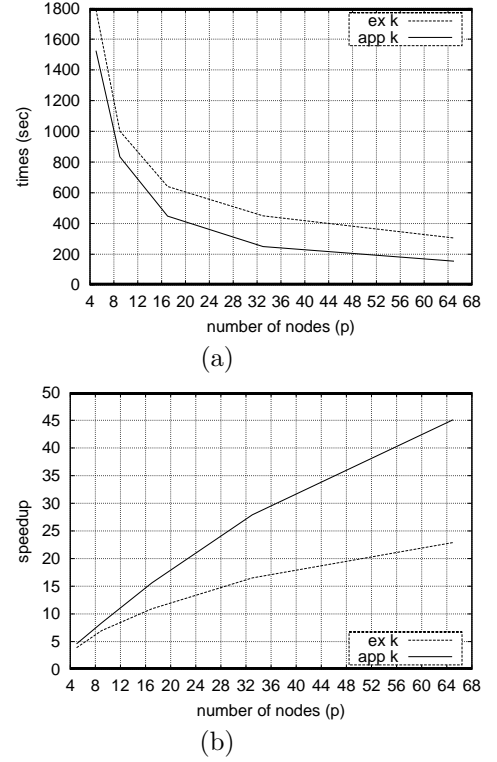


Figure 4: Comparison of parallel classification times (a) and speedups (b) between approximate kernel (“app k”) and the exact kernel (“ex k”) for the thrombin dataset.

where T_{comm}^{exact} and T_{comm}^{app} are the average communication times used by the exact kernel and the approximate kernel, respectively, and T_{total}^{exact} and T_{total}^{app} are the average total times used by the exact kernel and the approximate kernel, respectively. As seen in the table, the communication times have been reduced by 83.7% on average, and the overall computing times have been reduced by 31.26% on average.

7 Conclusion and Future Work

The computational performance for training an SVM depends on the efficiency of its kernel computation which can easily become intolerable on a single processor computer for large, high-dimensional data sets. We have proposed two methods for a high-performance parallel implementation of kernel computation. An effective parallel algorithm relies on a proper partition of the data and an efficient communication pattern in the computing system. Through analysis of the commonly used RBF kernel, we proposed the dimension-wise partition, which has a close to linear theoretical speedup and has achieved substantial speedups in real experiments. The dimension-wise partition alone has achieved a linear

speedup for clusters of up to 16 nodes. When 65 nodes are available, this method achieved a speedup of 23 folds. While we demonstrated the dimension-wise partition only for the Gaussian RBF kernel in this paper, it is also applicable to other kernels, such as string kernels [16, 17], where kernel inner products must be computed explicitly in the high dimensional feature space.

Speedup of the dimension-wise partition is limited because of the communication costs incurred when partial results are transferred to a coordinator node for combination. To overcome this barrier, we proposed an approximation method for RBF kernel functions that reduces a kernel matrix to a sparse matrix with a much smaller bandwidth requirement. We have proved that the approximated kernel matrix satisfies Mercer’s positive definiteness criterion in the limit for any dataset, and using only a finite approximation for a fixed data set. Further, we showed that the error bound decreases exponentially with the number of rectangles used. In practical implementations, we have found that we can achieve very low fill factors ($< 2\%$) while maintaining high classification accuracies and positive definiteness of the kernel matrix. Doing so has improved communication performance by 83.7% and overall parallel performance by 31% on average. With the approximation method and the dimension-wise partition, a linear speedup has been achieved when the cluster has fewer than 32 nodes. And with 65 nodes, a speedup of 45 folds has been observed.

The kernel approximation method will be more effective for even larger clusters, since communication overhead is more critical in larger systems and the approximate method can scale up well. For distributed computing environments connected through slow links, the approximation method is especially useful in reduction of communication cost. Since the approximation method makes the kernel matrix sparse, it can directly be used in circumstances where the space requirement is crucial.

We are currently working to refine our understanding of the RBF kernel approximation. Specifically, while we have a practical method for searching for the approximation parameters, N and τ , we would like an a priori method for designing these parameters for a fixed data set so as to minimize the fill factor while maintaining positive definiteness. Additionally, while the dimension-wise partition applies to a wide class of kernels, our kernel approximation method was developed specifically for the RBF kernel. We are developing similar approximation schemes for other classes of nonlinear kernels.

Acknowledgements We thank Lance Williams for his assistance with some of the formulations. We also thank

the anonymous reviewers for their invaluable feedback and suggestions.

Appendices

Appendix 1: Requirement Analysis for the CDK2 drug design and high density microarray datasets. The CDK2 drug design dataset has $m = 14,223$ compounds and a dimension of $n = 35,926,557$, representing descriptors of the compounds [28]. Using double precision data type, its space requirement for the data *alone* is $S = 14223 \times 35,926,557 \times 8\text{byte} = 4.087\text{TB}$. This amount of space is beyond the capacity of a uniprocessor computer and distributed memory in a parallel system must be used. The time complexity of computing an RBF kernel matrix is $T = O(m^2n)$ [4]. A computer of the capacity of 4 GFlops per second, which is the typical processing power of a high end Pentium 4 computer, would need 21 days to compute the kernel matrix,

$$(A-1) \quad T = \frac{m^2n}{4GFlop/second} = \frac{14223^2 \times 35.93 \times 10^6}{4 \times 10^9} \\ = 1.82 \times 10^6 \text{second} = 21 \quad \text{days}.$$

And this time is based on full CPU utilization, not accounting for cache misses and page faults.

A high density microarray has 60 million elements [20]. Due to cost considerations, today’s microarray datasets usually have a few hundred data points. However, with that high dimension, much more data points are necessary for effective classification. With the development of DNA technology, costs of microarrays are getting lower rapidly. Suppose, in the near future, the dataset has a few thousand data points, $m = 10,000$. Using the approach as in [1], $n = 60 \times 10^6$. The space needed to store the data alone is, $S = 10000 \times 60 \times 10^6 \times 8\text{byte} = 4800\text{GB} = 4.8\text{TB}$. A computer of the capacity of 4 GFlops per second would need 17 days to compute the kernel matrix,

$$(A-2) \quad T = \frac{m^2n}{4GFlop/second} = \frac{10000^2 \times 60 \times 10^6}{4 \times 10^9} \\ = 1.5 \times 10^6 \text{second} = 17.38 \quad \text{days}.$$

Appendix 2: Positive definiteness expressed in Fourier transform. We express the Mercer’s condition in terms of Fourier transforms as follows. Recall the Mercer’s condition from (5.13),

$$(A-3) \quad \int_{\chi \times \chi} k(x, x') f(x) f(x') dx dx' \geq 0 \quad \forall f \in L_2(\chi).$$

For an RBF kernel, the condition can be written as,

$$(A-4) \quad \int_{\chi \times \chi} k(x, x') f(x) f(x') dx dx' \\ = \int_{\chi \times \chi} k(x - x') f(x) f(x') dx dx'$$

$$(A-5) \quad = \int_{\chi} (f \circ k)(x) f(x) dx$$

$$(A-6) \quad = \langle (f \circ k), f \rangle = (2\pi)^{\frac{n}{2}} \langle \tilde{f} \cdot \tilde{k}, \tilde{f} \rangle$$

$$(A-7) \quad = (2\pi)^{\frac{n}{2}} \int_{\chi} |\tilde{f}(\omega)|^2 \tilde{k}(\omega) d\omega \geq 0,$$

where \tilde{f} and \tilde{k} are the Fourier transforms of functions f and k . In (A-5), $(f \circ k)(x) = \int_{\chi} k(x - x') f(x') dx'$, is the convolution between f and k . In (A-6), we used the fact that the Fourier transform of a convolution can be expressed as the dot product in the frequency domain,

$$(A-8) \quad \mathcal{F}[f \circ k] = (2\pi)^{\frac{n}{2}} \tilde{f} \cdot \tilde{k}.$$

For (A-7) to be true, we need $\tilde{k}(\omega) \geq 0, \quad \forall \omega \in \chi$. Expressing positive definiteness through Fourier transform is related to Bochner's theorem [23].

Appendix 3: Error bound in the distance domain. Since the distance between two vectors is non-negative, we only consider the case when $r \geq 0$. We first consider the sum of the error and then obtain the bound on the total absolute value. The error to approximate a kernel function $k(r)$ with a series of N rectangles functions $\hat{k}(r)$ is,

$$(A-9) \quad se(N) = \int_0^{\infty} [\hat{k}(r) - k(r)] dr \\ = \int_0^{(N-\frac{1}{2})\tau} [\hat{k}(r) - k(r)] dr + \int_{(N-\frac{1}{2})\tau}^{\infty} [\hat{k}(r) - k(r)] dr \\ = se_1(N) + se_2(N),$$

where,

$$(A-10) \quad se_1(N) = \int_0^{(N-\frac{1}{2})\tau} [\hat{k}(r) - k(r)] dr,$$

$$(A-11) \quad se_2(N) = \int_{(N-\frac{1}{2})\tau}^{\infty} [\hat{k}(r) - k(r)] dr.$$

$se_1(N)$ represents the sum of errors in the region where the series of N rectangle functions exist. It is the integral of the errors between the solid rectangles and the kernel function, as shown in Figure 2. It is smaller than the integral of the errors between the

dashed rectangles and the kernel function in Figure 2. Therefore,

$$(A-12) \quad se_1(N) \\ \leq \int_0^{\frac{1}{2}\tau} [k(0) - k(r)] dr + \int_{\frac{1}{2}\tau}^{1\frac{1}{2}\tau} [k(\frac{1}{2}\tau) - k(r)] dr \\ + \int_{1\frac{1}{2}\tau}^{2\frac{1}{2}\tau} [k(1\frac{1}{2}\tau) - k(r)] dr + \dots \\ + \int_{(N-1-\frac{1}{2})\tau}^{(N-\frac{1}{2})\tau} [k((N-1-\frac{1}{2})\tau) - k(r)] dr \\ = 1 \times \frac{\tau}{2} + k(\frac{1}{2}\tau)\tau + k(1\frac{1}{2}\tau)\tau + \dots \\ + k((N-1-\frac{1}{2})\tau)\tau - \int_0^{(N-\frac{1}{2})\tau} k(r) dr.$$

And $se_2(N)$ represents the total error in the region where the series of rectangle functions are absent, $se_2(N) = -\int_{(N-\frac{1}{2})\tau}^{\infty} k(r) dr$. Adding them together, we have,

$$(A-13) \quad se(N) = se_1(N) + se_2(N) \\ \leq 1 \times \frac{\tau}{2} + k(\frac{1}{2}\tau)\tau + k(1\frac{1}{2}\tau)\tau + \dots \\ + k((N-1-\frac{1}{2})\tau)\tau - \\ - \int_0^{(N-\frac{1}{2})\tau} k(r) dr - \int_{(N-\frac{1}{2})\tau}^{\infty} k(r) dr \\ = 1 \times \frac{\tau}{2} + k(\frac{1}{2}\tau)\tau + k(1\frac{1}{2}\tau)\tau + \dots \\ + k((N-1-\frac{1}{2})\tau)\tau - \int_0^{\infty} k(r) dr \\ = T - \int_0^{\infty} k(r) dr,$$

where $\int_0^{\infty} k(r) dr = \sqrt{\frac{\pi}{2}}\sigma$, and,

$$(A-14) \quad T = 1 \times \frac{\tau}{2} + k(\frac{1}{2}\tau)\tau + k(1\frac{1}{2}\tau)\tau + \dots \\ + k((N-1-\frac{1}{2})\tau)\tau = \frac{\tau}{2} + \tau \sum_{j=0}^{N-2} k(j + \frac{1}{2}\tau) \\ = \frac{\tau}{2} + \tau \sum_{j=0}^{N-2} e^{-\frac{((j+\frac{1}{2})\tau)^2}{2\sigma^2}} \leq \frac{\tau}{2} + \tau \sum_{j=0}^{N-2} e^{-\frac{(j\tau)^2}{2\sigma^2}} \\ \leq \frac{\tau}{2} + \tau \sum_{j=0}^{N-2} (\frac{1}{g})^j \leq \frac{\tau}{2} + \tau \frac{g}{g-1} (1 - \frac{1}{g^N}),$$

where $g = e^{\frac{\tau^2}{2\sigma^2}} > 1$. Consequently,

$$(A-15) \quad se(N) \leq \frac{\tau}{2} + \tau \frac{g}{g-1} \left(1 - \frac{1}{g^N}\right) - \sqrt{\frac{\pi}{2}} \sigma.$$

It is usually the case that $se(N) \geq 0$, then the absolute total error is bounded by,

$$(A-16) \quad sae(N) \leq \left| \frac{\tau}{2} + \tau \frac{g}{g-1} \left(1 - \frac{1}{g^N}\right) - \sqrt{\frac{\pi}{2}} \sigma \right|.$$

In the worst case where $N = 1$ and $\tau \rightarrow 0$, the absolute total error is bounded by, $sae(N) \leq \sqrt{\frac{\pi}{2}} \sigma$.

References

- [1] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, Jr. M. Ares and D. Hausler, *Knowledge-based analysis of microarray gene expression data by using support vector machines*, Proc. Natl. Acad. Sci. USA, 97(2000), pp. 262-267.
- [2] A. Choudhury, P. B. Nair and A. J. Keane, *A data parallel approach for large-scale Gaussian process modeling*, In R. L. Grossman, J. Han, V. Kumar, H. Mannila and R. Motwani, Eds., Proc. 2nd SIAM Int'l Conf. Data Mining, Arlington, VA, USA, SIAM 2002.
- [3] R. Collobert, S. Bengio and Y. Bengio, *A parallel mixture of SVMs for very large scale problems*, Neural Computation, 14(5)(2002), pp. 1105-1114.
- [4] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines*, Cambridge University Press, 2000.
- [5] Thanh-Nghi Do and F. Poulet, *Incremental SVM and visualization tools for bio-medical data mining*, In Proc. European Workshop on Data Mining and Text Mining for Bioinformatics, Dubrovnik, Croatia, Sep., 2003.
- [6] fMRI data center, <http://www.fmridc.org>.
- [7] G. Fung and O. L. Mangasarian, *Proximal support vector machine classifiers*, Proc. 7th ACM Intl. Conf. KDD, San Francisco, USA, pp. 77-86, ACM, 2001.
- [8] G. Fung and O. L. Mangasarian, *Incremental support vector machine classification*, In R. Grossman, H. Mannila, R. Motwani, et al., Eds., Proc. 2nd SIAM Int'l Conf. Data Mining, pp. 247-260, SIAM 2002.
- [9] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1981.
- [10] R. C. Houts and O. Alkin, *Signal Analysis in Linear Systems*, Saundser College Publishing, ISBN 0-03-028744-8.
- [11] T. Joachims, *Making large scale SVM learning practical*, In B. Scholkopf, C. Burges and A. J. Smola, Eds., Advances in Kernel Methods-Support Vector Learning, pp. 169-184, MIT Press, 1999.
- [12] T. Joachims, *Text categorization with support vector machines: Learning with many relevant features*, In C. Nedellec and C. Rouveirol, Eds., Proc. European Conf. Machine Learning, pp. 137-142, Berlin, Springer, 1998.
- [13] L. Kaufman, *Solving the quadratic programming problem arising in support vector classification*, In B. Scholkopf, C. Burges and A. Smola, Eds., Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, USA, 1998.
- [14] Y.-L. Lee and O. L. Mangasarian, *RSVM: Reduced support vector machines*, In Proc. 1st SIAM Int'l Conf. Data Mining, Chicago, IL, April, SIAM, 2001.
- [15] Y.-L. Lee and O. L. Mangasarian, *SSVM: A smooth support vector machine*, Computational Optimization and Applications, 20(2001), pp. 5-22.
- [16] C. Leslie, E. Eskin and W. S. Noble, *The spectrum kernel: a string kernel for SVM protein classification*, In Proc. Pacific Symp. Biocomputing, pp. 566-575, PSB, 2002.
- [17] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins, *Text classification using string kernels*, In T. K. Leen, T. G. Dietterich and V. Tresp, Eds., NIPS 2000, pp. 563-569, MIT Press, 2001.
- [18] P. Mahe, N. Ueda, T. Akutsu, Jean-Luc Perret and Jean-Philippe Vert, *Extensions of marginalized graph kernels*, In Proc. 21st Intl. Conf. Machine Learning, Banff, Canada, ICML, 2004.
- [19] O. L. Mangasarian, *Generalized support vector machines*, In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, Eds., Advances in Large Margin Classifiers, pp. 135-146, Cambridge, MA, MIT Press, 2000.
- [20] L. Melton, *Pharmacogenetics and Genotyping: On the trail of SNPs*, Nature, 422(2003), pp. 917-923.
- [21] E. Osuna, R. Freund and F. Girosi, *Support vector machines: training and applications*, Tech. Report, AIM-1602, MIT, 1997.
- [22] J. Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, In B. Scholkopf, C. Burges, and A. Smola, Eds., Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, USA, 1998.
- [23] M. Reed and B. Simon, *Methods of Modern Mathematical Physics I: Functional Analysis*, 2nd edition, Academic Press, San Diego, CA, 1980.
- [24] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.
- [25] The thrombin dataset, <http://www.cs.wisc.edu/~dpage/kddcup2001/>.
- [26] A. Tveit and H. Engum, *Parallelization of the incremental proximal support vector machine classifier using a heap-based tree topology*, Tech. Report, IDI, NTNU, Trondheim, Norway, Aug. 2003.
- [27] UCI machine learning data datasets, <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [28] M. K. Warmuth, J. Liao, G. Ratsch, M. Mathieson, San-tosh Putta and C. Lemmen, *Support vector machines for active learning in drug discovery process*, Journal of Chemical Information Science, 43(2)(2003), pp. 667-673.

Loadstar: A Load Shedding Scheme for Classifying Data Streams

Yun Chi*, Philip S. Yu†, Haixun Wang†, Richard R. Muntz*

*Department of Computer Science, University of California, Los Angeles, CA 90095

†IBM Thomas J. Watson Research Center, Hawthorne, NY 10532

ychi@cs.ucla.edu, {psyu,haixun}@us.ibm.com, muntz@cs.ucla.edu

Abstract

We consider the problem of resource allocation in mining multiple data streams. Due to the large volume and the high speed of streaming data, mining algorithms must cope with the effects of system overload. How to realize maximum mining benefits under resource constraints becomes a challenging task. In this paper, we propose a load shedding scheme for classifying multiple data streams. We focus on the following problems: i) how to classify data that are dropped by the load shedding scheme? and ii) how to decide when to drop data from a stream? We introduce a quality of decision (QoD) metric to measure the level of uncertainty in classification when exact feature values of the data are not available because of load shedding. A Markov model is used to predict the distribution of feature values and we make classification decisions using the predicted values and the QoD metric. Thus, resources are allocated among multiple data streams to maximize the quality of classification decisions. Furthermore, our load shedding scheme is able to learn and adapt to changing data characteristics in the data streams. Experiments on both synthetic data and real-life data show that our load shedding scheme is effective in improving the overall accuracy of classification under resource constraints.

keywords: data mining, data streams, load shedding, classification, quality of decision, feature prediction, Markov model.

1 Introduction

Many new applications process multiple data streams simultaneously. For instance, in a sensor network, data flows from a large number of embedded sensors; and in the stock market, each security generates a stream of quotes and trades. However, applications that handle these unbounded, high speed incoming data are constrained by limited resources (e.g., CPU cycles, bandwidth, and memory).

Resource allocation for distributed stream applications is often formulated as an optimization problem [12, 8]. Much work focuses on allocating resources in a best-effort way so that performance degrades gracefully. For instance, if the data characteristics from a sensor exhibit a predictable trend, then the precision constraints might be satisfied by transmitting only a fraction of the sensor data to the remote server.

Other approaches assume that a set of Quality-of-Service (QoS) specifications are available [1, 3, 14]. A load shedding scheme decides when and where to discard data and how much data to discard according to the QoS specification. In other words, these approaches assume that the impact of load shedding on performance is known *a priori* through the QoS specifications.

Goal In this paper, we argue that for many data mining tasks a more intelligent load shedding scheme for streaming data is required. The goal of mining is to maximize certain benefits (e.g., to detect as many credit card transaction frauds as possible). It is more complex than achieving high precision of simple computations such as aggregation (e.g., AVG, SUM, and COUNT). Because in those cases, high precision can usually be secured as long as the percentage of data sampled from the stream reaches a certain threshold.

The benefits of mining does not depend on the sampling rate. For instance, assume the percentage of fraudulent credit card transactions is p and our fraud detector has 100% accuracy. Then, under random sampling, the expected number of frauds we can detect per investigated transaction is fixed at p . To increase the efficiency of fraud detection, we need to know how benefits of mining will be affected by the next incoming data before resources are committed to investigate it.

Thus, our goal is to design a load shedding scheme to maximize the benefits per resource used in mining.

Challenges Load shedding in *mining* data streams is a new topic and it raises many challenges. Although load shedding has been studied for *managing* data streams, many assumptions in these studies are not appropriate

—*The work of these two authors was partly supported by NSF under Grant Nos. 0086116, 0085773, and 9817773.

for data mining tasks.

First, for many simple queries (e.g., aggregation) considered for managing data streams, it is often safe to assume that the quality of the query result depends only on the sample size. Some approaches assume simple (e.g., monotonic, or even concave or piecewise linear) QoS curves, which depict the relationship between the quality and the sample size, are available to the load shedding mechanism. In contrast, in mining data streams, sample size itself cannot guarantee good mining result, because the quality of mining often depends on specific feature values in a non-monotonic way. For example, in certain regions of the feature space, a classifier may have very high confidence in its classification decision, even if the feature value is only known approximately. But in other regions, the classifier may not be very sure about its classification decision because in these regions, a small variation in a feature value may change the decision. In this case, resources (i.e., computing the exact feature values) should be allocated to a stream if the decision is more sensitive to the feature value of the data in this stream. Thus, the challenge lies in determining how to make the resource allocation in a best effort way to minimize classification errors.

Second, data mining applications are often more sensitive to changes in data characteristics. For instance, a small move in the feature space may totally change the classification results, and more often than not, it is such changes that we care about the most. Thus, feature value prediction is important to load shedding design for mining data streams. Fortunately, many feature values (e.g., the reading of sensors that measure the temperature or the water level of a river, or the feature values extracted from consecutive satellite images) have strong time-correlation and we can build models to take advantage of such correlation. Thus, the challenge lies in building a feature predictor that is able to capture the time-correlation and adapt to the time-variance of the feature values.

Our Contributions To the best knowledge of the authors, this is the first work on load shedding for mining data streams. We make the following contributions. (1) We define two *quality of decision* (QoD) measures for classification based on the predicted distribution of the feature values in the next time unit. (2) We develop a prediction model for feature values using Markov models whose parameters can be updated in real time to reflect parameter changes. (3) We combine the first two to obtain a load shedding scheme, Loadstar¹, for classifying multiple data streams. Experiments on both synthetic

data and real-life data show that our load shedding scheme is effective in improving the accuracy of data stream classification in the presence of system overload.

Paper Organization The rest of the paper is organized as follows. We formally define the problem in Section 2. In Section 3, we introduce two QoD (quality of decision) measures. In Section 4, we present our Markov-chain model for predicting feature values. In Section 5, we describe Loadstar, the overall load shedding scheme. In Section 6, we summarize experiment results which demonstrate Loadstar's effectiveness. In Section 7, we review related work and we conclude in Section 8 with future directions.

2 Problem Definition

The major system components are illustrated in Figure 1. Raw data flows in via multiple streams and are fed to the data preparation and analysis block through a communication channel. The data preparation and analysis block is responsible for data cleaning, feature extraction and composition, etc. The derived features enter the data classification block, which outputs mining results.

In this paper, we assume that data preparation and analysis is CPU intensive. In comparison, the CPU consumption for classification is negligible. This is true in many real applications especially those that handle multimedia data, for which feature extraction is usually a CPU intensive procedure. For example, if the raw data are text documents, the data preparation and analysis may involve removing stop words, counting the frequency of important words, projecting the vector of word frequencies to some pre-defined conceptual space, filtering the projected values in each dimension using thresholds, etc [2]; if the raw data are images from satellites, computing the features, such as luminance, shape descriptor, amplitude histogram, color histogram and spatial frequency spectra, will usually take a lot of CPU time [13]. As a result, when the system is overloaded, the data preparation and analysis block cannot process all of the data and load shedding is needed. (Another equivalent scenario is when the bandwidth of the communication channel is limited and therefore not all raw data can be transmitted to the data preparation and analysis block.)

The input to the system consists of multiple streams of raw data. When the system is overloaded, data from some of the streams are dropped. For those streams whose data is dropped, their feature values can be predicted by the feature predictor block, based on historic feature values. Therefore, the classifier will handle both the real feature values generated by

¹A Load Shedding Scheme for Streaming Data Classifiers

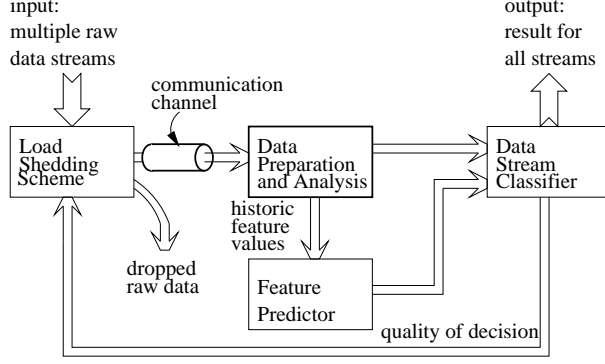


Figure 1: The System Setup

the data preparation and analysis block, and predicted feature values for those streams whose data has been dropped.

We assume that the classifier handles data streams consisting of a d -dimensional feature vector $\mathbf{x} \in \mathbf{X}^d$ (x_i can be either continuous or categorical) and produces a class label $c_i \in \{c_1, \dots, c_K\}$. The classifier performs classification for each incoming \mathbf{x} no matter whether \mathbf{x} is real or predicted feature values. The objective is to design a load shedding scheme that minimizes the overall error rate of the data mining task when the system is overloaded.

In this paper, we restrict the data mining task to be the classification problem, although the technique can be extended to other data mining tasks, such as clustering, on data streams.

3 Quality of Decision

Load shedding takes place when data from multiple streams exceeds the processing capacity. We are interested in load shedding schemes that ensure shed load has minimal impact on the benefits of mining.

In order to do this, we need a measure of benefit loss if data \mathbf{x} from a certain stream is discarded. However, we must be able to do that without seeing \mathbf{x} 's real feature values. In this section, we propose two QoD metrics, and in Section 4, we present a method to predict feature values such that we can make load-shedding decisions before seeing the real data.

3.1 The Quality of Classification One way to view a classifier is to consider it as a set of *discriminant functions* $f_i(\mathbf{x}), i = 1 \dots K$. The classifier assigns class label c_k to \mathbf{x} if $f_k(\mathbf{x}) \geq f_i(\mathbf{x}), \forall i$ ([6]). For traditional classification, only the ranks of the discriminant functions are important in decision making, i.e., we only care if we are right or wrong, and do not care how far off we are.

Consider an example where there are two classes

and the data is one dimensional (i.e., there is a single feature x). Figure 2(a) shows the two discriminant functions and Figure 2(b) their log ratio. We use logarithmic values because first, logarithm is a monotonically increasing function which preserves the original ranks of the discriminant functions; second, the ratio is invariant with the respect to the scale; third, as we will see shortly, it makes computations simpler.

For a given feature value x , if $f_2(x)$ is greater than $f_1(x)$, we assign class label c_2 to x ; we do not care how much $f_2(x)$ is greater than $f_1(x)$. For example, in two data streams, incoming elements with feature values $x = 2$ and $x = 1.5$ are both classified as c_2 . However, when the feature values are not exact, the two classification decisions will have different levels of certainty. For example, assume that $x = 2$ and $x = 1.5$ are *current* feature values and we believe x will not change dramatically in the *next* step. In such a case, if the classifier has to make a classification decision for the next step without updated feature values, it may still assign class label c_2 to both data streams; however, in this case, for the data stream with $x = 2$, the classifier is much more *certain* about its decision than for the data stream with $x = 1.5$. Intuitively, the *quality* of the classification decision for the first data stream is higher than that of the second data stream. If we have to shed load in the next step, we should shed load from the data stream whose current feature value is $x = 2$, because by allocating the available resource to the data stream with less quality of decision (i.e., the data stream with current feature value $x = 1.5$), we expect to gain more benefits in term of the improvement in the classification accuracy.

The question is, how to quantify this quality of decision?

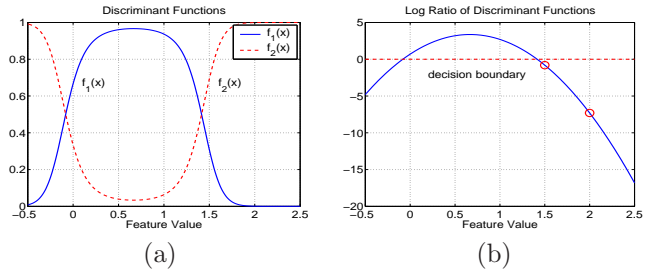


Figure 2: Certainty of a Classifying Decision

3.2 Quantifying the Quality of Decision Assume we have derived a probability density function for \mathbf{X} , the feature value in the next time unit:

$$(3.1) \quad \mathbf{X} \sim p(\mathbf{x})$$

It is worth mentioning that $p(\mathbf{x})$ is different from the estimated prior distribution $p(\mathbf{x}|\mathcal{D})$ that can be obtained from the training data \mathcal{D} . When we build the classifier based on \mathcal{D} , we consider each observation in \mathcal{D} as an independent sample from an unknown distribution. Here by $p(\mathbf{x})$, we mean that through some mechanism (e.g., by taking advantage of the temporal locality of the data), we have obtained an estimation of the feature value of the next time unit, and it is in the form of a density $p(\mathbf{x})$.

Quality Defined on Log Ratio We assume that the discriminant functions have positive values. Using Eq (3.1), the distribution of feature values in the next time unit, we can compute the expected logarithmic value of the discriminant functions:

$$(3.2) \quad E_{\mathbf{X}}(\log f_i(\mathbf{x})) = \int_{\mathbf{X}} (\log f_i(\mathbf{x})) p(\mathbf{x}) d\mathbf{x}$$

We use δ_1 to represent the decision which chooses the class label that maximizes the expected value:

$$(3.3) \quad \delta_1 : k = \arg \max_i E_{\mathbf{X}}(\log f_i(\mathbf{x}))$$

Eq (3.3) only gives the classifying decision; to perform load shedding, we need to give a quantitative measure about the certainty of the decision. We introduce our first *quality of decision* (QoD) measure:

$$(3.4) \quad Q_1 = E_{\mathbf{X}} \log \left(\frac{f_k(\mathbf{x})}{f_{\tilde{k}}(\mathbf{x})} \right) \\ = E_{\mathbf{X}}(\log f_k(\mathbf{x})) - E_{\mathbf{X}}(\log f_{\tilde{k}}(\mathbf{x}))$$

where \tilde{k} is the second best decision according to Eq (3.2).

From the definition, we have $Q_1 \geq 0$. Intuitively, the higher the Q_1 , the more we are confident in our best-effort decision, that is, we compare our decision with the second-best choice, and if the expected performance of our decision is much better than that of the second-best choice, we believe that our decision has high quality.

Quality Defined on Overall Risk We introduce another quality of decision measure based on Bayesian decision theory. We use the *posterior* distribution of the classes given the feature vectors as the discriminant functions.

At point \mathbf{x} in feature space \mathbf{X} , if we decide the class is c_i , then the conditional risk of our decision is

$$R(c_i|\mathbf{x}) = \sum_{j=1}^K \sigma(c_i|c_j) P(c_j|\mathbf{x})$$

where $\sigma(c_i|c_j)$ is the loss function, i.e., the penalty incurred when the real class is c_j and our decision is c_i . For example, for zero-one loss, we have:

$$\sigma(c_i|c_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases}$$

in which case we can simplify the conditional risk as

$$R(c_i|\mathbf{x}) = 1 - P(c_i|\mathbf{x})$$

Because we have the distribution of the feature value \mathbf{x} at the next time unit, we can compute the expected risk for a decision for next time unit as

$$E_{\mathbf{X}}[R(c_i|\mathbf{x})] = \int_{\mathbf{X}} R(c_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

We use δ_2 to represent the best-effort decision rule which minimizes this expected risk:

$$(3.5) \quad \delta_2 : k = \arg \min_i E_{\mathbf{X}}[R(c_i|\mathbf{x})]$$

Assume for $\mathbf{x} \in \mathbf{X}$, the optimal decision is c^* . (More rigorously, c^* should be written as $c^*(\mathbf{x})$.) Because the real value of \mathbf{x} is unknown, c^* is infeasible to realize in our load shedding environment. The risk associated with c^* is

$$E_{\mathbf{X}}[R(c^*|\mathbf{x})] = \int_{\mathbf{X}} R(c^*|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

This risk is the Bayesian lower bound based on distribution $p(\mathbf{x})$. We then define the QoD based on the difference between the expected risk and the lower bound:

$$(3.6) \quad Q_2 = 1 - (E_{\mathbf{X}}[R(c_k|\mathbf{x})] - E_{\mathbf{X}}[R(c^*|\mathbf{x})]) \\ = 1 - \int_{\mathbf{X}} [P(c^*|\mathbf{x}) - P(c_k|\mathbf{x})] p(\mathbf{x}) d\mathbf{x}$$

From the definition, we have $0 \leq Q_2 \leq 1$. Also, $Q_2 = 1$ if and only if c_k is the optimal decision for every $\mathbf{x} \in \mathbf{X}$ where $p(\mathbf{x}) > 0$. Intuitively, the larger the Q_2 , the higher quality the decision.

A Comparison of the Two QoDs The quality of classification depends on two factors. The first factor is the feature distribution $p(\mathbf{x})$. Both Q_1 and Q_2 have taken $p(\mathbf{x})$ into consideration. For example, as shown in Figure 2, we are quite confident that if $x = 1$ then the class is c_1 , and if $x = 2$ then the class is c_2 . If $p(\mathbf{x})$ is given by $P(x = 1) = P(x = 2) = 0.5$, then both Q_1 and Q_2 will give low values. Thus, resources allocated to the stream (which helps to reveal the real feature values) will improve the quality of decision.

The second factor is the discriminant functions. In this case, although both Q_1 and Q_2 reflect the quality of decision, Q_2 is a better metric, because it indicates the benefit of allocating resources to the data stream. For example, consider an extreme case where $f_1(x) = f_2(x) = 0.5$ for all x . Then, Q_1 is 0, which (correctly) indicates that the classification result is very unreliable. Q_2 is 1, which (also correctly) indicates that allocating more resources to the data stream will not improve the accuracy of the classification.

Base on the above discussion, we expect Q_2 to perform better than Q_1 . This is verified by the experiments that will be given in a later section.

Naive Bayesian Classifier The QoDs defined above are mathematically appealing but computationally challenging, especially when the dimension of the feature space d is large. We can, however, simplify the QoD defined on log ratio (δ_1 and Q_1) by assuming that each feature is conditionally independent given the class labels. With this assumption, a very simple classifier, the naive Bayesian classifier, can be applied. In spite of its naivety, it has been shown in many studies that the performance of naive Bayesian classifiers are competitive with other sophisticated classifiers (such as decision trees, nearest-neighbor methods, etc.) for a large range of data sets [5, 9]. Because of the “Bayesian” assumption, we restrict the discriminant function to be the posterior distribution of each class. (Without the “Bayesian” restriction, for δ_1 and Q_1 , the discriminant functions could be any positive functions on the feature space.)

With the assumption of a naive Bayesian classifier, we have

$$\begin{aligned} E_{\mathbf{X}}(\log f_i(\mathbf{x})) &= E_{\mathbf{X}}(\log P(c_i|\mathbf{x})) \\ &= E_{\mathbf{X}} \left(\log \frac{P(\mathbf{x}|c_i)P(c_i)}{\sum_j P(\mathbf{x}|c_j)P(c_j)} \right) \end{aligned}$$

The classifying decision δ_1 and the QoD Q_1 only depend on the relative value. So we ignore the denominator and derive the following.

$$\begin{aligned} E_{\mathbf{X}} [\log(P(\mathbf{x}|c_i)P(c_i))] &= E_{\mathbf{X}} \log P(\mathbf{x}|c_i) + E_{\mathbf{X}} \log P(c_i) \\ &= E_{\mathbf{X}} \sum_j \log P(x_j|c_i) + \log P(c_i) \\ &= \sum_j E_{\mathbf{X}} \log P(x_j|c_i) + \log P(c_i) \\ &= \sum_j E_{X_j} \log P(x_j|c_i) + \log P(c_i) \end{aligned}$$

Thus, we only need the distribution of each feature $X_j \sim p(x_j)$ instead of the joint density function $\mathbf{X} \sim p(\mathbf{x})$ to compute δ_1 and Q_1 .

4 Prediction in the Feature Space

In Section 3, we assume that we know the distribution of the feature values when their real values are not available. The computation of the QoD and the choice of load shedding are based on the distribution. In this section, we study how to obtain the feature distribution.

If the feature values of the current time is independent of those in the next time unit, the best we can do is to use the prior distribution of the feature values². This is commonly assumed in data stream management systems, where data-value histograms are often created to assist in query answering.

However, in many real life applications, feature values often have short-term temporal correlation. For example, temperatures of a region and water levels of a river usually do not change dramatically over a short period of time. Feature values extracted from consecutive satellite images also have strong temporal correlation. On the other hand, data characteristics of a stream usually change with time. Thus, our task is to capture short-term time correlation in a time-varying environment.

In this section, we propose a finite-memory Markov model and introduce an algorithm to incrementally update the Markov model so that it reflects the characteristics of the most recent data.

4.1 The Markov Model Markov models have been used extensively in many fields to model stochastic processes [11]. In this study, we use discrete-time Markov-chains with a finite number of states. A discrete-time Markov-chain is defined over a set of M states s_1, \dots, s_M , and an $M \times M$ state transition probability matrix P , where P_{ij} is the probability of transition from state s_i to s_j . We use one Markov-chain for each feature in each data stream. The Markov-chains are used to model both categorical and numerical features. For continuous values, we discretize them into finite number of bins.

Consider any feature x and its corresponding Markov-chain. Assume the feature value at time t_0 is known to us, and we have $x = s_i$, $1 \leq i \leq M$. Thus, the distribution of the feature value at t_0 is $p_0(x) = e_i$, where e_i is a $1 \times M$ unit row vector with 1 at position i and 0's at other positions. The distribution of the feature value in the next time unit t_1 is $p_1(x) = p_0(x)P = e_iP$, where P is the state transition probability matrix. In the next time unit t_2 , the distribution of the feature value becomes $p_2(x) = p_1(x)P = e_iP^2$.

If we shed load at time t_1 , $p_1(x)$ will give us a distribution of the value of x at t_1 . At time t_i ,

²In such a case, the quality of decision for each classifier will not change with time.

the distribution is $p_i(x)$. When i becomes large, the distribution will converge to $p(x) = \pi$, where π is the steady-state solution of the Markov-chain, i.e., π is the solution to

$$\begin{cases} \pi = \pi P, \\ \sum_j \pi_j = 1. \end{cases}$$

It is clear that π is the prior distribution (among the historic data based on which we have built the Markov-chain) of the feature values. In other words, the probability of a certain feature value in the next time unit is approximately the fraction of its occurrence in the historic data. This makes sense, because as the gap between the current time and the time when we last investigated the feature values becomes larger, the temporal correlation will disappear.

In this study, we assume that for a given data stream, the Markov-chains for the features are independent. In other words, we assume that given the feature values of the data stream at current time, the distribution of each feature of next time unit is independent of the distributions of other features. This assumption makes it easier for us to solve the problem (e.g., to compute δ_2 and Q_2) numerically by using, e.g., Monte Carlo methods. Without this assumption, we have to use some special sampling technique (e.g., for the Gibbs sampler, we need the marginal distribution for each feature given all other features [10]), with the independence assumption, sampling is easier, i.e., we can draw samples for each feature following its own distribution, independent of other features. To study the cases in which the feature distributions are dependent is among our future work.

4.2 Finite-Memory Markov-Chains An important issue in data stream is time-variation, i.e., the data characteristics may drift with time. To handle this phenomena, we adopt a finite-memory Markov-chain model and incrementally update its parameters so that they reflect the characteristics of the most recent data. The main idea is to maintain the Markov-chains using a sliding window of the most recent W transitions and update the parameters of the Markov-chains when new observations are available.

First, we consider a simple case in which there is no load shedding. At time t , the most recent $W + 1$ states are $s(t - W), \dots, s(t - 1), s(t)$, and these $W + 1$ states contain W transitions, i.e., from $s(t')$ to $s(t' + 1)$ for $t - W \leq t' < t$. Assume $s(t - W), \dots, s(t - 1), s(t)$ are generated by a Markov-chain P , it can be shown that the maximum-likelihood estimation (MLE) for P_{ij} is

$$(4.7) \quad \hat{P}_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$$

where n_{ij} is the number of observed transitions from state s_i to s_j among the W transitions. To obtain the MLE, we only need to maintain a matrix \bar{P} of $M \times M$ counters and update the entries using the most recent observations. For example, assume that $s(t - W) = s_p$, $s(t - W + 1) = s_q$, $s(t) = s_i$, and at time $t + 1$, a new observation becomes available and $s(t + 1) = s_j$. To update \bar{P} , we increase \bar{P}_{ij} by 1 as we insert s_j into the sliding window, and decrease \bar{P}_{pq} by 1 as we remove s_p from the sliding window. To get the MLE, we multiply each row of \bar{P} by a normalizing factor to make the row sum to 1.

However, when load shedding takes place, we may not have consecutive observations. When load shedding is frequent, the observations could be very sparse. To obtain the maximum-likelihood estimation of the parameters based on observations with missing values, we can use, for example, the EM algorithm to compute the unobserved feature values. However, iterative algorithms such as EM are time-consuming, which makes them unacceptable for data stream applications. In addition, such algorithms very often only converge to local maximums.

To solve this problem, we use an approximate approach to update the parameters of the Markov-chains: for each data stream, we maintain a flag to indicate if it has been observed in the previous time unit (we say that a data stream is *observed* or it gets an *observation* at time t if we do not shed load from the data stream at time t); if at time t , a data stream is observed, and it was not observed at time $t - 1$, then we will observe the data stream in two consecutive time units (i.e., t and $t + 1$), whenever possible. In such a case, we say that the data stream has a *consecutive observation request (COR)* at time t . If all CORs are satisfied, the observations from a data stream will be in pairs of consecutive states, with possible gaps among the pairs. Therefore, instead of maintaining $W + 1$ most recent states, we maintain in the sliding window the most recent W transitions, where each transition consists of a pair of states (s_{from}, s_{to}). The method to compute and update \bar{P} is similar to the one introduced above, and we still use Eq (4.7) to estimate the P matrix for a Markov-chain, knowing that it is just an approximation.

Furthermore, because the memory of the Markov-chains is finite, it is possible that some rows of \bar{P} are zero vectors. To handle this case and to represent certain prior knowledge about the models, in our implementation we added some pseudo-counts to \bar{P} , that is, instead of all zeros, some counters in the \bar{P} matrix (e.g., those on diagonal) are initialized with some small positive integers.

5 The Load Shedding Scheme

Our load shedding scheme, Loadstar, is based on the two components introduced in the previous two sections, i.e., the quality of decision and the predicted distribution in the feature space. Pseudo-code for the Loadstar algorithm is given in Figure 3. The inputs to the algorithm are i) N' data streams that contain data at time t ($N' \leq N$), and ii) the capacity C of the system, in terms of the number of data streams that the system can handle, at time t . When $N' > C$, load shedding is needed. The outputs of the algorithm are the decision of the classifier for each data stream at time t .

Figure 3 actually contains two versions of our load shedding scheme: the basic Loadstar algorithm (without lines 8–11), in which the parameters of Markov-chains are fixed, and the extended version (with lines 8–11), which we call Loadstar*, in which the parameters of Markov-chains are updated in real time. For the basic version Loadstar, we assume that the parameters of Markov-chains do not drift with time so they are learned from training data; for the extended version Loadstar*, we assume that the parameters of Markov-chains drift with time and they are updated using the most recent observations.

Algorithm Loadstar (N', C)	
inputs: data from N' streams, and system capacity C ;	
outputs: decisions $(\delta_1, \dots, \delta_N)$;	
static variables: feature distributions $p(\mathbf{x})$'s, Markov-chains MC's, COR flags (f_1, \dots, f_N) ;	
1:	update $p(\mathbf{x})$ for each feature \mathbf{x} using its MC;
2:	compute decisions $(\delta_1, \dots, \delta_N)$ and QoDs (q_1, \dots, q_N) using $p(\mathbf{x})$'s;
3:	select C streams from N' data streams based on (q_1, \dots, q_N) and (f_1, \dots, f_N) ;
4:	for each selected stream i do
5:	observe the feature value for stream i ;
6:	revise δ_i for stream i ;
7:	revise $p_i(\mathbf{x})$ for stream i ;
8:*	if stream i has had load in the previous time unit then
9:*	update MC's for stream i ;
10:*	$f_i \leftarrow false$;
11:*	else $f_i \leftarrow true$;
12:	return $(\delta_1, \dots, \delta_N)$;

Figure 3: The Loadstar and Loadstar* Algorithms

Some internal variables are maintained as static by the algorithm. Among them, $p(\mathbf{x})$'s are the distributions of the features in the current time unit; MC's represent the Markov-chains learned from training data for Loadstar or the Markov-chains in the current time unit for

Loadstar*; (f_1, \dots, f_N) are a vector of COR flags for the data streams in Loadstar*, and in Loadstar, they are all set to *false*.

At time $t - 1$, the feature distributions at time t are predicted by updating the $p(\mathbf{x})$'s using the Markov-chains (line 1). Each stream first assumes that it will not be observed at time t ; it computes the decisions using Eq (3.3) or Eq (3.5) and the qualities of decision using Eq (3.4) or Eq (3.6), both based on the predicted feature distributions (line 2). Then when N' and C are available at time t , if $N' > C$, load shedding is applied. C streams are selected to be observed based on the COR flags and the QoDs: if among the N' data streams, the number of streams with *true* COR flags is less than C , then their requests are fulfilled first and the remaining resources are assigned to other streams based on their QoDs; otherwise, the C streams will be only selected from the data streams whose COR flags are *true*, based on their QoDs (line 3). When deciding which streams to be observed based on QoDs, we use a weighted randomized algorithms where the chance for a stream to be observed is inversely proportional to its QoD value. We choose to use a randomized scheme in order to avoid starvation of a data stream. For the data streams that are observed, because they obtain the real feature values, their feature distributions are changed to unit vectors, and their classification decisions are updated using the new feature distributions (lines 5–7). For Loadstar*, after the data streams to be observed are selected, their COR flags are updated, and if necessarily, their MC's are updated (lines 8–11). Finally, the classification decisions are returned (line 12).

Time Complexity We study the overhead introduced by the load shedding algorithm shown in Figure 3. For each feature of each data stream, updating $p(\mathbf{x})$ (line 1) takes $O(M^2)$ time, where M is the number of states in the Markov-chain. In the Loadstar* version, updating the Markov-chains (lines 8–11) takes $O(1)$ time, because it only involves updating a counter and a flag. The most time-consuming step is to compute the decision and the QoD for each data stream (line 2), because it requires integrations over the whole (possibly multi-dimensional) feature space. This situation, however, can be alleviated by making some assumptions. For example, we have shown that the conditional independence assumption on the feature values makes Q_1 easy to compute; furthermore, we have argued that the conditional independence assumption on the feature distributions makes the numerical integration easier. In the section of experimental studies, we will show that a numerical integration using only a few samples can give us a very accurate approximation and save us from computing the exact integration over the whole feature space.

6 Experimental Results

In this section, we use both synthetic and real-life data sets to study the performance of the Loadstar algorithm. We compare Loadstar with a naive algorithm, in which loads are shed from each data stream equally likely. Both algorithms are implemented in C++. In the experiment setup, for easy of study, instead of varying loads, we fix the load (to be 100 data streams for both the synthetic and the real-life data sets) and change the number of data streams that the system can handle at each time unit. In other words, we study the system under different levels of overloads. In addition, because of the random nature of the algorithms, for all the experiments we run 10 times with different random seeds and report the average values.

6.1 The Synthetic Data Set By using synthetic data, we sought to answer the following experimental questions about our load shedding algorithm:

- (1) Does Loadstar improve the performance over the naive algorithm? If so, how is the improvement achieved?
- (2) Do the Markov-chains capture the models of feature space accurately? Do they adapt to drifts of data characteristics?

We generate data for 100 data streams, and for each data stream, we set the number of features d to be 3. Among the three features, x_1 and x_2 are numerical and x_3 is categorical. The two numerical features are generated using the following random walk model:

$$(6.8) \quad x_t = x_{t-1} + \varepsilon, \text{ where } \varepsilon \sim N(0, \sigma^2)$$

where $N(\mu, \sigma^2)$ is a Normal distribution with mean μ and variance σ^2 . In addition, we add boundaries at 0 and 1 in the random walk model, i.e., at a given time unit t , if $x_t > 1$ or $x_t < 0$, we switch the sign of the corresponding ε and make x_t between 0 and 1. We partition the 100 streams into two families: for the first family, which consists of 10 data streams, the σ in Eq (6.8) is set to be 0.1; for the second family, which consists of 90 data streams, $\sigma = 0.01$. For obvious reasons, we call the first family the *volatile* streams and the second family the *non-volatile* streams. As can be seen soon, such a setup reveals the mechanics that Loadstar uses to obtain good performance. For the categorical feature x_3 , which consists of 4 distinct values s_1, \dots, s_4 , all data streams have the same characteristics: the feature values are generated as time series using a Markov-chain whose P matrix has the following form: the element on diagonal is 0.91 and all other elements have value 0.03.

To generate the model for the classification problem, we use two class labels, + and -, and we assume the features to be independent given the class label. For the two numerical features, their likelihood functions are given as $p(x_1|+) \sim N(0.2, 0.1^2)$, $p(x_1|-) \sim N(0.8, 0.1^2)$, $p(x_2|+) \sim N(0.8, 0.1^2)$, and $p(x_2|-) \sim N(0.2, 0.1^2)$. For the categorical feature x_3 , its likelihood functions are given as $p(s_1|+) = p(s_3|+) = 0.4$, $p(s_2|+) = p(s_4|+) = 0.1$, $p(s_1|-) = p(s_3|-) = 0.1$, and $p(s_2|-) = p(s_4|-) = 0.4$. Because of the symmetry of the model, we assume that the prior distribution for the two classes to be equally likely. Finally, the real class label for each feature triplet is assigned to be the class that has higher joint posterior distribution value.

We generate data for 11,000 time units, where data for each time unit consists of 100 observations for the 100 data streams. Data in the first 6,000 time units are used as training data to build a naive Bayesian classifier. For the naive Bayesian classifier, we use 10 bins with equal width to discretize the two features with numerical values. Although our algorithm allows each data stream to have its own classifier, for simplicity, in the experiments we use a single naive Bayesian classifier for all data streams. Data in the last 5,000 time units are used as test data. We set the window size W for Markov-chain learning in Loadstar* to be 100.

Performance Comparison In this study, we compare Loadstar (and its extension, Loadstar*) with the naive algorithm in terms of error rates under different levels of overload. For this, we fix the load to be 100 data streams, and increase the number of data streams to have loads shed at each time unit from 0 to 80. Figure 4(a) and Figure 4(b) show the error rates of the classifier under different levels of overload, using δ_1, Q_1 and δ_2, Q_2 , respectively.

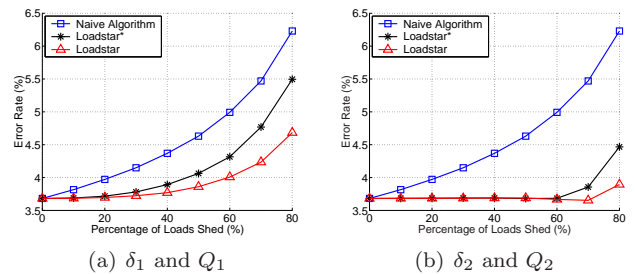


Figure 4: Performance Comparison

From the figures we can see that in both cases Loadstar has lower error rates than the naive algorithm under different levels of overload. Loadstar that uses δ_2 and Q_2 has better performance than that uses δ_1 and Q_1 . In particular, for the former one, when the percentage of loads shed is under 70%, the error rate remains the same as that of the case with no load

shedding. Because of this, in the remaining discussion, we focus on δ_2 and Q_2 . Also can be seen from the figures, the error rates of Loadstar* are higher than those of Loadstar. This result is not unexpected, because for learning Markov-chains, Loadstar* requires consecutive observations. That is, with 80% loads shed, for Loadstar, on average each data stream is observed every 5 time units; for Loadstar*, each stream is observed consecutively every 10 time units. As we know, because of the temporal locality, consecutive observations every 10 time units does not provide as much information as two separate observations with distance of 5 time units.

To shed light on the reasons for Loadstar's good performance, in Figure 5(a) we plot the percentage of observations that are assigned to the volatile data streams under different levels of load shedding. As can be seen from the figure, the naive algorithm always assigns 10% observations to the volatile streams because there are 10 out of 100 data streams that are volatile. In contrast, for Loadstar, as the number of available observations becomes smaller, a higher fraction of them are assigned to the volatile streams. For example, when there are only 20 observations available, on average, at each time unit the naive algorithm assigns 2 of them to the volatile streams, but Loadstar assigns more than 5 observations to the volatile streams.

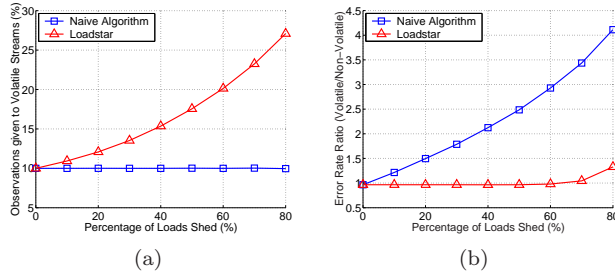


Figure 5: (a) Percentage of Observations Assigned to Volatile Data Streams, and (b) Error Rate Ratio between Volatile and Non-Volatile Data Streams

In addition, we compute the error rates for the volatile and non-volatile families separately. Figure 5(b) shows the error rate ratio between the volatile family and the non-volatile family, under different levels of load shedding. As can be seen, for the naive algorithm, because it sheds loads from all data streams equally likely without considering their data characteristics, as the percentage of load shedding increases, the error rate of the volatile family suffers more and more comparing to that of the non-volatile family; in contrast, for Loadstar, because the quality of decision automatically includes the characteristics of data into consideration, the error rate ratio between the two families remains around 1 until the percentage of load shedding increases

to 60%, and does not go beyond 1.5 even when the percentage of load shedding increases to 80%.

In summary, when different data streams have different characteristics, Loadstar is more fair in that it gives more available resources to the data streams that are less certain, and as a result, it balances the error rates among the data streams with different characteristics and achieves better overall performance.

Markov-Chain Learning In this experiment, we study the Markov-chain learning part of our load shedding scheme. We generate the data streams such that x_3 has time-varying characteristics, using the following two Markov-chains:

$$P_A = \begin{bmatrix} .91 & .03 & .03 & .03 \\ .03 & .91 & .03 & .03 \\ .03 & .03 & .91 & .03 \\ .03 & .03 & .03 & .91 \end{bmatrix}, P_B = \begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{bmatrix}$$

For the test data, for the first 1,000 time unit, we generate x_3 using P_A (P_A is also used to generate the training data); then at time unit 1,000, we switch to P_B ; finally, at time unit 3,000, we switch back to P_A .

To quantify the performance of Markov-chain learning, we use the Kullback-Leibler divergence as the measure of error. Notice that each row P_i of the P matrix is a distribution; in our algorithm, we have a estimation matrix \hat{P} and each row \hat{P}_i of \hat{P} is also a distribution. To see if the two distributions are near to each other, we compute their Kullback-Leibler divergence $d(P_i, \hat{P}_i) = \sum_j P_{ij} \log(\frac{P_{ij}}{\hat{P}_{ij}})$. And finally, we sum the Kullback-Leibler divergences over all the rows and all the data streams. Figure 6 shows the results over time units 500 to 5,000 for Loadstar and Loadstar*. For Loadstar*, we report the results for two cases: the case in which there is no load shedding and the case in which there is 50% load shedding.

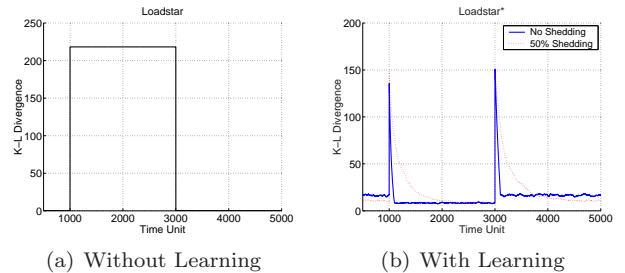


Figure 6: Learning the Markov-Chains

As can be seen from Figure 6(a), because Loadstar learns the parameters of Markov-chains from the training data and because P_A is used to generate the training data, before time 1,000, the error is very small; the error increases sharply when the parameters are changed at

time 1,000, and remains high until at time 3,000, when the original parameters are restored. In contrast, as can be seen from Figure 6(b), Loadstar* can learn the new parameters of Markov-chains in real time: when the parameter change happens, the error of Loadstar* also increases sharply; however, when there is no load shedding, as we expected, after 100 time units (which is the sliding window size W), the new parameters are learned and the error drops back; this learning takes longer time for the case of 50% load shedding.

It is interesting to observe from Figure 6(b) that when the Markov-chain has parameter P_A , Loadstar* has more accurate estimation for the parameters when there is 50% load shedding than when there is no load shedding. To explain this, we have to see the difference between the two cases: in the case of no load shedding, data from the most recent 100 time units are used to learn the parameter; in the case of 50% load shedding, on average, samples from the most recent 200 time units are used. When the distributions are skewed (e.g., P_A), the temporal locality prevents us from learning the parameter very accurately using only 100 time units; when there is 50% load shedding, samples are drawn from longer history (on average 200 time units) and therefore the parameters can be learned more accurately. To verify this, we look at Figure 6(b) between time units 2,000 and 3,000. During this period, P_B is used and from the parameters we can see that when P_B is used, there is no temporal locality at all. Therefore, as expected, during this period both cases learned the parameters equally accurately.

Monte Carlo From Eq (3.5) and Eq (3.6) we can see that to compute δ_2 and Q_2 , we need to do an integration (or weighted sum) over all the feature space. We now show that a sampling method can help us reduce the computation. We use a Monte Carlo method that instead of integrating over the whole feature space, just samples some points from the feature space, and compute unweighted average of δ_2 and Q_2 over these points. In our implementation, because of the conditional independence assumption on the feature distributions, to draw a sample point (x_1, x_2, x_3) , we can draw x_1 following $p_1(x)$, x_2 following $p_2(x)$, x_3 following $p_3(x)$ (all with replacement) and then put them together. Figure 7(a) and Figure 7(b) show the results for the Monte Carlo method with 5 sample points and 10 sample points, respectively. As can be seen from the figures, with only 5 sample points, the Monte Carlo method has already clearly outperformed the naive method, and with 10 sample points, the performance of the Monte Carlo method becomes very close to that of the original Loadstar algorithm in which integration is taken over the whole feature space. This experiment demonstrates

that our load shedding scheme is particularly suitable for data stream applications, in which quick response time is crucial.

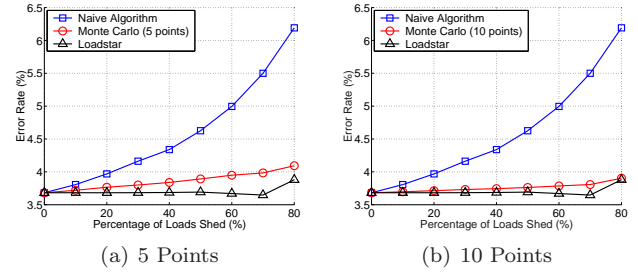


Figure 7: The Performance for Monte Carlo

6.2 The NASDAQ Data Set For real-life data, we use a data set of stock streaming price data. We recorded 2 weeks of price data for 100 stocks (NASDAQ-100) as well as the NASDAQ index. For each stock, the close price at the end of each minute is recorded. The streaming price for each stock is considered as a data stream. Therefore, there are 100 data streams and for each data stream, there are observations for 3,900 time units (10 business days \times $6\frac{1}{2}$ hours per day \times 60 minutes per hour) with a time unit of 1 minute. The price for each stock is normalized with respect to the stock's open price on the first day. In other words, after normalization, the price of a stock at time unit t will be $\frac{v_t}{v_1}$, where v_i is the stock's real price at time unit i .

We define the classification problem as the following. At a given time t , a stock is called *outperform* if its normalized price is higher than that of the NASDAQ index, *underperform* otherwise. The classification problem is defined as at each time unit t , predicting the class (*outperform* or *underperform*) of each stock at time $t+1$.

Here is the way how we build our classifier. We assume that the NASDAQ index follows the random walk model given in Eq (6.8). (Stock price is one of the best-known examples of time series that behave like random walks [4].) We assume the noises at different time units are independent. Because we do not have the noise variance σ^2 , at any given time t we use the sample variance $\hat{\sigma}^2$ of the NASDAQ index in the hour before t as an estimation for σ^2 . If we have y_t , the NASDAQ index value at time t , then according to our model, the NASDAQ index value at time $t+1$ follows a Normal distribution:

$$y_{t+1} \sim N(y_t, \hat{\sigma}^2)$$

For our Bayesian classifier, we choose the posterior probability, as shown in Figure 8, as our discriminant function (here we assume an equal prior distribution for *outperform* and *underperform*). For example, assume

$y_t = 1.2$ and we know the value of a stock at time $t+1$ to be $\tilde{x}_{t+1} = 1.3$, then if we decide the class to be *outperform*, the probability for the decision to be correct is the area under the curve of the distribution of y_{t+1} for which y_{t+1} is less than 1.3; if we decide the class to be *underperform*, the probability for this decision to be correct is the area under the curve where y_{t+1} is greater than 1.3. Obviously, conditioning on the value of a stock \tilde{x}_{t+1} at time $t+1$, the decision will be *outperform* if $\tilde{x}_{t+1} > y_t$, and *underperform* otherwise (here we use \tilde{x}_{t+1} because we do not know the real value x_{t+1} , i.e., we are making decisions about time $t+1$ at time t).

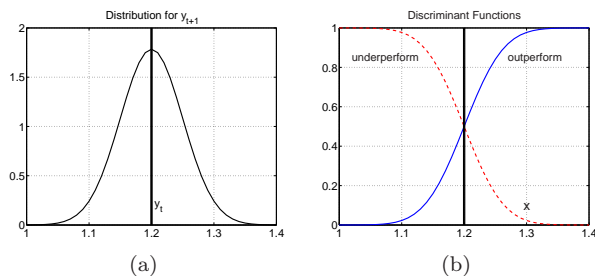


Figure 8: Bayesian Classifier for the Stock Data Set

For our load shedding scheme, we choose δ_2 and Q_2 as they are defined before. For the Markov-chains, because the feature values (i.e., normalized stock price at time t) are continuous, we discretize them into 20 bins with equal width where each bin corresponds to 1 percentile. In this experiment, because the prices for all stocks behave similarly, for simplicity we use a single Markov-chain for all data streams, where the parameters of the Markov-chain are learned using the first hour of data. Again, as a base case, we defined a naive load shedding algorithm, which chooses data streams to have observations shed equally likely.

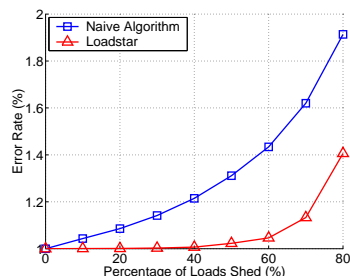


Figure 9: The Performance for the Stock Data

The experimental results are shown in Figure 9. As can be seen from the figure, because the stock prices do not change very dramatically in a time interval of 1 minute, the error rate for this classification problem is not very high. However, as load shedding becomes more severe, the error rate for the naive algorithm grows

continuously. In contrast, when the load shedding level is between 0% and 40%, there is no obvious change in error rates for our load shedding algorithm. In the whole load shedding range, Loadstar always outperforms the naive algorithm.

7 Related Work

In [8], Jain et al. proposed using Kalman filters to adaptively manage resources in data stream management systems, where the main goal is to minimize bandwidth usage under a given precision requirement. In [12], Olston et al. proposed an adaptive-filter scheme for continuous queries over distributed data sources, where the main concern is also the tradeoff between the precision of the answers to queries and the communication cost. These studies are similar to ours in that they use mathematical models to model the data sources and adaptively allocate resources accordingly. The main difference of these methods with ours is they assume that the data sources have processors to do complicated computation (to filter data based on thresholds or to solve linear equations). This assumption may be acceptable for simple numerical values; however, for complex data types, such as multimedia data, whose feature values must be derived using specialized software or hardware, such an assumption become invalid. In addition, these studies all assume numerical data; in our algorithm the features can have either numerical values or categorical values.

In [3], Babcock et al. studied the load shedding problem in systems that process continuous monitoring queries over data streams. The main idea of the study is that when overload happens, inserting load shedders in various locations of the query plan, such that the maximum relative error among all queries is minimized (with high probability). However, this study was restricted to sliding window aggregate queries and did not consider queries that involve the join operation among multiple streams. In [14], Tatbul et al. described a scheme for load shedding in the Aurora Data Stream Management System. In the study, the load shedding is based on the QoS specifications on latency, values, and loss-tolerance. This work is similar to ours in that it adjusts load shedding according to the status of each sub system of the whole system. However, it assumed static QoS curves (e.g., concave or piece linear curves) are available to guide load shedding. In contrast, how to defined the quality measure for data mining tasks is a major part of our work.

Another topic that is closely related to our work is concept drifts in mining data streams. In [15], Wang et al. proposed an algorithm for mining concept-drifting data streams using weighted ensemble classifiers. In

[7], Fan et al. proposed an active mining method that detects potential changes in data streams. These studies are similar to ours in that statistics are defined to measure the characteristics of current data streams. However, they assume that the correct class labels are readily available for newly arrived testing data and therefore, as the concepts in data streams change, it is possible to revise the decision models correspondingly in real-time. In our study, we do not assume the availability of the correct class labels for test data, and therefore our classifier is fixed beforehand based on training data. Instead, we assume that at different time, the feature values are moving around different regions of the feature space. In other words, it is the region of the concept we are currently in, not the concept itself, that changes with time.

8 Conclusion and Future Directions

In this paper, we studied the resource allocation problem in mining data streams and in particular, we developed a load shedding algorithm, Loadstar, for classifying data streams. The Loadstar algorithm consists of two main components: i) the quality of decision (QoD) measures that are defined based on the classifier, the feature space, and the predicted feature distribution of the next time unit, and ii) the feature predictor which is based on finite-memory Markov-chains, whose parameters can be updated in real time. Extensive experimental results on both synthetic and real-life data sets showed that Loadstar has better performance than a naive algorithm in term of classification accuracy, where its superior performance is achieved by automatically focusing on data streams that are more uncertain while shedding data streams whose class labels in the next time unit are more certain. In addition, experiments showed that the Loadstar algorithm can efficiently learn parameters of its Markov-chains and computation in Loadstar can be reduced by using Monte Carlo methods.

For future work, we plan to extend our study in the following directions. First, in this paper we assume that the streams are independent; however, in many real-life applications, one mining task may need multiple data streams and each data stream can be involved in multiple data mining tasks. To take these relationships into consideration in our algorithm is one of our future directions. Second, in this paper we assume the data mining task (the classification) is the last stage of the system. In the future, we plan to consider systems in which data mining is just an intermediate computation, e.g., as a filter to decide which data streams to be sent for more detailed analysis. Third, in this paper we consider a simple case that at each given time, we either

apply load shedding to a data stream or not; in the future, we plan to extend our load shedding algorithm to control the communication rates of the data streams, e.g., given many video streams, the frame rate of each stream is proportional to its importance.

References

- [1] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.
- [2] C. C. Aggarwal and P. S. Yu. On effective conceptual indexing and similarity search in text data. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, 2001.
- [3] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *20th International Conference on Data Engineering*, 2004.
- [4] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC, 2004.
- [5] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, 1997.
- [6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2001.
- [7] W. Fan, Y-A Huang, H. Wang, and P. S. Yu. Active mining of data streams. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.
- [8] A. Jain, E. Y. Chang, and Y-F Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004.
- [9] P. Langley, W. Iba, and K. Thompson. Analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992.
- [10] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, 2001.
- [11] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [12] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.
- [13] W. K. Pratt. *Digital Image Processing*. John Wiley & Sons, 1991.
- [14] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *Proc. of the 29th Intl. Conf. on Very Large Databases (VLDB'03)*, 2003.
- [15] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

Topic-driven Clustering for Document Datasets *

Ying Zhao [†]

George Karypis [‡]

Abstract

In this paper, we define the problem of *topic-driven clustering*, which organizes a document collection according to a given set of topics. We propose three topic-driven schemes that consider the similarity between documents and topics and the relationship among documents themselves simultaneously. We present a comprehensive experimental evaluation of the proposed topic-driven schemes on five datasets. Our experimental results show that the proposed topic-driven schemes are efficient and effective with topic prototypes of different levels of specificity.

1 Introduction

Fast and high-quality document clustering algorithms play an important role in providing intuitive navigation and browsing mechanisms by organizing large amounts of information into a small number of meaningful clusters. As unsupervised learning methods, clustering algorithms do not require any prior knowledge of the datasets in general. However, when such prior knowledge is available, clustering algorithms should also be able to benefit from it to produce more desired clustering solutions. In particular, we focus on the type of the prior knowledge that reflects the cognition of the natural clusters by domain experts. For example, in many knowledge management applications, even though the complete taxonomy of the document collection is not available, often times domain experts can describe the major topics (clusters) that the collection covers. Moreover, they would like the clustering algorithms produce

the clustering solutions that are consistent with their cognition models. Hence it is important to be able to organize a document collection according to a given set of topics (either from domain experts, or as a requirement satisfying users' needs). We refer to this problem as *topic-driven clustering*.

For example, in a typical environment of law firms, a large amount of letters, memoranda, e-mail messages, and contracts are generated on a daily basis. Thus, organizing legal documents into meaningful clusters to leverage browsing and searching is very important. Even though a complete law firm taxonomy may not be available, law librarians in a law firm can provide some information on the major topics of the document collection based on their knowledge on the practice areas of the law firm, related law categories, and custom base. For example, a law firm may focus on the areas of banking, bankruptcy, insurance, and debt. This information is not only helpful for organizing the documents, but also serves as a requirement of desired clustering solutions. That is, the resultant clusters should correspond to the identified topics (for example, the four practice areas). Also note that acquiring prior knowledge of labeled documents associated with each topic, even a small amount, can be very time-consuming and costly.

The traditional classification algorithms cannot solve the topic-driven clustering problem because of the insufficient information about each class (topic). Since sometimes the available descriptions may only contain a few words, in order to produce good organization, the information of unlabeled documents must be taken into account to leverage classification technology. Semi-supervised classification [13, 3] and active learning [6, 23] are two of such approaches. However, these approaches either need sufficient labeled data to start with, or need to have access to a nontrivial amount of labeled data during the process.

The current approaches of semi-supervised clustering [25, 1, 26, 11], which can use class labels or pairwise constraints on some documents during the clustering process, fail to satisfy the requirements of the topic-driven clustering problem as well, mainly because the prior knowledge of the topic-driven clustering problem is not in the format of labeled objects, but of the de-

*This work was supported in part by NSF CCR-9972519, EIA-9986042, ACI-9982274, ACI-0133464, and ACI-0312828; the Digital Technology Center at the University of Minnesota; and by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

[†]Dept. of Computer Science and Engineering, University of Minnesota, MN

[‡]Dept. of Computer Science and Engineering, University of Minnesota, MN

scriptions of possible topics.

We propose the topic-driven clustering schemes to solve this problem by defining two properties that a good clustering solution must have. First, the documents clustered to a certain topic should contain the content of the topic (*i.e.*, the documents are similar or relevant to the topic). Second, the documents within one cluster should be more similar to each other than the documents from two different clusters.

The contribution of this paper is two-fold. First, to our knowledge, we introduce this novel problem of topic-driven clustering. Second, we propose effective and efficient topic-driven clustering methods that emphasize the relationship between documents and topics and relationship among documents themselves simultaneously. In addition, we present a comprehensive experimental evaluation using various datasets and our experimental results show that the proposed topic-driven clustering schemes are effective with topic prototypes of different levels of specificity.

The rest of this paper is organized as follows. Section 2 discusses some related research. Section 3 provides some information on how documents are represented and how the similarity or distance between documents is computed. Section 4 describes the criterion functions that are the focus of this paper and describes the algorithm that optimizes the various topic-driven criterion functions and the clustering algorithm itself. Section 5 provides the detailed experimental evaluation of the various topic-driven criterion functions. Finally, Section 6 provides some concluding remarks.

2 Related Research

Active learning [6, 23] acknowledges the fact that acquiring labeled data is very time-consuming and costly, and tries to minimize the number of labeled data required to build a successful classifier. The active learning approaches start with a very small number of labeled data and request unlabeled objects to be labeled based on whether the unlabeled objects are “more informative” point. The active learning approaches utilize the information provided by unlabeled data. However, they still need to have access to sufficient labeled data.

Incorporating prior knowledge into the clustering process has drawn people’s attention recently. The focus of this research has been on semi-supervised clustering, which assumes the prior knowledge (background knowledge) is given by a limited set of labeled data, from which the knowledge of two objects should belong to the same cluster (must-link) or should not belong to the same cluster (cannot-link) can be derived. Previous semi-supervised approaches fall into three categories: constraint-based, metric-based and the combined ap-

proaches. Constraint-based approaches explicitly modify the objective function or make certain constraints during the clustering process[25]. Whereas, metric-based approaches parametrize distance metric and learn the metric parameters in a manner, so that the distance between objects connected by must-links is smaller and the distance between objects connected by cannot-links is larger in general [1, 26]. Finally the combined approaches integrate both of these techniques in the clustering process [2]. Another recent approach of incorporating prior knowledge tackles the problem differently [11]. They defined the non-redundant data clustering as a problem of discovering alternative clustering solutions given a known clustering solution, where the prior knowledge is an entire clustering solution.

3 Preliminaries

Through-out this paper we will use the symbols n , m , and k to denote the number of documents, the number of terms, and the number of clusters, respectively. We will use the symbol S to denote the set of N documents that we want to cluster, S_1, S_2, \dots, S_k to denote each one of the k clusters, n_1, n_2, \dots, n_k to denote the sizes of the corresponding clusters, and T_1, T_2, \dots, T_k to denote the topic prototype vectors given as prior knowledge.

The various clustering algorithms that are described in this paper use the vector-space model [20] to represent each document. In this model, each document d is considered to be a vector in the term-space. In particular, we employed the $tf-idf$ term weighting model, in which each document can be represented as

$$(tf_1 \log(n/df_1), tf_2 \log(n/df_2), \dots, tf_m \log(n/df_m)).$$

where tf_i is the frequency of the i th term in the document and df_i is the number of documents that contain the i th term. To account for documents of different lengths, the length of each document vector is normalized so that it is of unit length ($\|d_{tfidf}\| = 1$), that is each document is a vector on the unit hypersphere. In the rest of the paper, we will assume that the vector representation for each document and for each topic has been weighted using $tf-idf$ and it has been normalized so that it is of unit length. Given a set A of documents and their corresponding vector representations, we define the **composite** vector D_A to be $D_A = \sum_{d \in A} d$, and the **centroid** vector C_A to be $C_A = D_A/|A|$. We also define the composite vector of the entire dataset to be $D = \sum_{i=1}^N d_i$, and the composite vector of the entire topics to be $T = \sum_{i=1}^k T_i$.

In the vector-space model, the cosine similarity is the most commonly used method to compute the similarity between two documents d_i and d_j , which is defined to be $\cos(d_i, d_j) = d_i^t d_j / (\|d_i\| \|d_j\|)$. The cosine

formula can be simplified to $\cos(d_i, d_j) = d_i^t d_j$, when the document vectors are of unit length. This measure becomes one if the documents are identical, and zero if there is nothing in common between them (*i.e.*, the vectors are orthogonal to each other).

4 Topic-driven Clustering Algorithms

At a high-level the problem of topic-driven clustering is defined as follows. Given a set S of n documents and a set T of k topics, we would like to partition the documents into k subsets S_1, S_2, \dots, S_k , each corresponding to one of the topics, such that (i) the documents assigned to each subset are more similar to each other than the documents assigned to different subsets, and (ii) the documents of each subset are more similar to its corresponding topic than the rest of the topics.

Even though there are a number of different ways that can be used to convert the above high-level problem definition into a precise clustering algorithm, in this paper, we will limit our focus to the class of algorithms that use a global criterion function \mathcal{C} to capture the properties and quality of the desired clustering solution and formulate the clustering problem as that of an optimization problem that tries to compute a k -way clustering solution such that the value of \mathcal{C} is optimized [10].

In the rest of this section we first present a number of different criterion functions that can represent the requirements of the topic-driven clustering problem, followed by a description of the algorithms that were used to perform their optimization.

4.1 Criterion Functions

Since the requirements of the topic-driven clustering contain two components, we first look at how to model them separately. The first component emphasizes the relationship between documents and tries to guide the clustering process to produce clustering solutions in which documents from the same cluster are more similar to each other than the documents assigned to different clusters. This component does not consider the topics and we will refer to the criterion functions that fall into this category as *unsupervised criterion functions*. On the other hand, the second component emphasizes whether the documents in each cluster are relevant to the topic associated with the cluster. We will refer to the criterion functions in this category as *supervised criterion functions*. In the rest of this section we will first discuss several criterion functions from each category and then propose two schemes to combine them together. At the end, we propose the third scheme, which is a hybrid approach that incorporates the two aspects into a single criterion function.

4.1.1 Unsupervised Criterion Functions

People have proposed a great number of criterion functions in this category over the past few years [7, 15, 8, 9]. Recently, we [31, 30] studied eight different partitionial clustering criterion functions in the context of document clustering, which optimize various aspects of intra-cluster similarity, inter-cluster dissimilarity, and their combinations. Our experiments revealed that different criterion functions lead to substantially different results, whereas our analysis showed that their performance depends on the degree to which they can correctly operate when the dataset contains clusters of different densities (*i.e.*, they contain documents whose pairwise similarities are different) and the degree to which they can produce balanced clusters.

In this study, we focus on two internal criterion functions (\mathcal{I}_1 and \mathcal{I}_2) and one external criterion function (\mathcal{E}_1) [29, 31]. This subset represents some of the most widely-used criterion functions for document clustering, and includes some of the best- and worst-performing schemes.

The \mathcal{I}_1 criterion function (Equation 4.1) maximizes the sum of the average pairwise similarities (as measured by the cosine function) between the documents assigned to each cluster weighted according to the size of each cluster and has been used successfully for clustering document datasets [19].

$$\mathcal{I}_1 = \sum_{r=1}^k n_r \left(\frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} \cos(d_i, d_j) \right) \quad (4.1)$$

$$\mathcal{I}_1 = \sum_{r=1}^k \frac{\|D_r\|^2}{n_r}.$$

The \mathcal{I}_2 criterion function (Equation 4.2) is used by the popular vector-space variant of the K -means algorithm [7, 15, 8, 22]. In this algorithm each cluster is represented by its centroid vector and the goal is to find the solution that maximizes the similarity between each document and the centroid of the cluster that is assigned to.

$$\mathcal{I}_2 = \sum_{r=1}^k \sum_{d_i \in S_r} \cos(d_i, C_r) = \sum_{r=1}^k \|D_r\|. \quad (4.2)$$

The \mathcal{E}_1 criterion function (Equation 4.3) computes the clustering by finding a solution that separates the documents of each cluster from the entire collection. Specifically, it tries to minimize the cosine between the centroid vector of each cluster and the centroid vector of the entire collection. The contribution of each cluster is weighted proportionally to its size so that larger clusters

will be weighted higher in the overall clustering solution. \mathcal{E}_1 was motivated by multiple discriminant analysis and is similar to minimizing the trace of the between-cluster scatter matrix [10].

$$(4.3) \quad \mathcal{E}_1 = \sum_{r=1}^k n_r \cos(C_r, C) \Leftrightarrow \sum_{r=1}^k n_r \frac{D_r^t D}{\|D_r\|}.$$

4.1.2 Supervised Criterion Functions

For the topic-driven clustering problem, we assume that the description of each cluster is available as prior knowledge and can be represented as a vector. Given these cluster prototype vectors, the similarity between each document and its topic can be defined as the cosine similarity between the vector of the document d and the prototype vector of the topic T_r . Hence, we can define internal and external supervised criterion functions similar to the unsupervised criterion functions.

The internal supervised criterion function, denoted by $\mathcal{S}_{\mathcal{I}}$, tries to maximize the similarity between the documents in a cluster to the topic associated with the cluster. The formal definition can be written as

$$(4.4) \quad \mathcal{S}_{\mathcal{I}} = \sum_{r=1}^k \sum_{d_i \in S_r} \cos(d_i, T_r) = \sum_{r=1}^k D_r^t T_r.$$

The external supervised criterion function, denoted by $\mathcal{S}_{\mathcal{E}}$, tries to minimize the similarity between each document to the topics that are not associated with its cluster. Let $\bar{T}_r = T - T_r$, then the external supervised criterion function ($\mathcal{S}_{\mathcal{E}}$) can be written as

$$(4.5) \quad \mathcal{S}_{\mathcal{E}} = \sum_{r=1}^k \sum_{d_i \in S_r} \cos(d_i, \bar{T}_r) = \sum_{r=1}^k D_r^t \bar{T}_r / \|\bar{T}_r\|.$$

Note that since maximizing $\sum_{r=1}^k D_r^t T_r$ is the same as minimizing $\sum_{r=1}^k D_r^t \bar{T}_r$, the difference between $\mathcal{S}_{\mathcal{I}}$ and $\mathcal{S}_{\mathcal{E}}$ is that in $\mathcal{S}_{\mathcal{E}}$ the dissimilarities between documents to the other topics is scaled by the norm of the composite of the other topics.

4.1.3 Combined Criterion Functions

Combining unsupervised and supervised criterion functions can be treated as a multi-objective optimization problem, which has been studied in many different domains [14, 28, 21]. One of the real difficulties in this problem is that no single optimal solution exists. Instead, an optimal solution exists for each objective in

the solution space. The result is that the definition of a good solution becomes ambiguous. Thus, we need to develop a scheme that can disambiguate the definition of a good solution. A good scheme should allow fine-tuned control of the tradeoffs among the objectives and be able to handle objectives that correspond to quantities that are both of similar as well as of different types.

One straightforward means of disambiguating the definition of a good multi-objective solution is to assign the objectives different weights before combining them together, which we refer to as the *weighted* scheme.

Given two criterion functions X and Y , the weighted scheme can be written as

$$(4.6) \quad M_1(X, Y) = \alpha X + (1 - \alpha)Y,$$

where α is the preference factor.

The weighted scheme allows a fine-tuned control of the tradeoffs among the objectives by varying the preference factor α . However, this formulation cannot handle dissimilar criterion functions or the criterion functions that change in different scales, because a weighted sum of them can be meaningless.

Deriving topic-driven criterion functions based on the weighted scheme can be done easily for \mathcal{I}_2 and \mathcal{E}_1 , since both \mathcal{I}_2 and \mathcal{E}_1 have N terms in their formulas. However, for \mathcal{I}_1 , to make it also contain N terms as in $\mathcal{S}_{\mathcal{I}}$, we need to multiply the \mathcal{I}_1 function by N before combining it with $\mathcal{S}_{\mathcal{I}}$.

Motivated by the method of combining multiple objective functions in graph partitioning [21], we propose the second scheme, the *normalized* scheme. Our formulation is based on the intuitive notion of what constitutes a good multi-objective solution. Quite often, a natural way of evaluating the quality of a multi-objective solution is to look at how close it is to the optimal solution of each individual objective. Hence, before combining two criterion functions, we normalize them with the optimal values that can be achieved by optimizing the two criterion functions separately.

Given two criterion functions X and Y , let X^* and Y^* denote the criterion function values of the optimal solutions with respect to X and Y , respectively, then the normalized scheme of combining two criterion functions can be defined as

$$(4.7) \quad M_2(X, Y) = \alpha \frac{X}{X^*} + (1 - \alpha) \frac{Y}{Y^*},$$

where α is the preference factor.

In essence, optimizing Equation 4.7 attempts to compute a k -way clustering such that the criterion function values with respect to each criterion are not far away from the optimal values. This scheme works with both similar and dissimilar objectives. This is

because it makes all quantities similar before combining them. Each criterion function value is divided by the optimal value of its corresponding criterion, and so, represents a certain fraction of the optimal value. Since all components now represent a fraction of the optimal value, they can be combined meaningfully.

Finally, the various topic-driven criterion functions derived by the two combined schemes are shown in Table 1, the clustering problem becomes that of maximizing $M_1(\mathcal{I}_1)$, $M_1(\mathcal{I}_2)$, $M_2(\mathcal{I}_1)$, and $M_2(\mathcal{I}_2)$, and minimizing $M_1(\mathcal{E}_1)$ and $M_2(\mathcal{E}_1)$ accordingly.

4.1.4 Hybrid Criterion Functions

For \mathcal{I}_1 and \mathcal{I}_2 , we propose the third scheme that incorporates the two aspects into a single criterion function. The motivation behind this hybrid scheme is that we noticed the relationship between the unsupervised criterion function \mathcal{I}_2 and the internal supervised criterion function $\mathcal{S}_{\mathcal{I}}$. The former maximizes the summation of the similarity of each document to its cluster centroid and the latter maximizes the summation of the similarity of each document to its cluster topic. If we could define a new center that represents both the cluster centroid and the topic, then we could maximize the summation of the similarity of each document to this new center and address the two requirements of the topic-driven clustering problem at the same time. To this end, we define the *topic-weighted* composite vector of the r th cluster C'_r as $D'_r = \sum_{d \in S_r} \cos(d, T_r) d_i$ and the weighted size $n'_r = \sum_{d \in S_r} \cos(d, T_r)$. The *topic-weighted* centroid can be defined as

$$C'_r = \frac{\sum_{d \in S_r} \cos(d, T_r) d_i}{n'_r},$$

which takes into account the similarity of each document to its cluster topic. Using the above definition, the hybrid \mathcal{I}_2 criterion function, denoted by $H(\mathcal{I}_2)$, can be obtained by requiring the clustering solution to maximize the similarity between the documents assigned to a cluster and its topic-weighted centroid. This is formally defined as follows:

$$(4.8) \quad H(\mathcal{I}_2) = \sum_{r=1}^k \sum_{d_i \in S_r} \cos(d_i, C'_r) = \sum_{r=1}^k D_r^t C'_r$$

Similarly, the \mathcal{I}_1 can be rewritten as

$$\mathcal{I}_1 = \sum_{r=1}^k \sum_{d_i \in S_r} \frac{d_i^t D_r}{n_r}.$$

If we use the topic-weighted composite and size of the r cluster to replace the composite and size, we can get

the hybrid \mathcal{I}_1 criterion function, denoted by $H(\mathcal{I}_1)$, as follows:

$$(4.9) \quad H(\mathcal{I}_1) = \sum_{r=1}^k \sum_{d_i \in S_r} \frac{d_i^t D'_r}{n'_r}.$$

Given the formulation of $H(\mathcal{I}_1)$ and $H(\mathcal{I}_2)$, the clustering problem becomes that of finding the clustering solutions that maximize $H(\mathcal{I}_1)$ and $H(\mathcal{I}_2)$, respectively.

4.2 Partitional Clustering Algorithm

The partitional method we used to optimize the various criterion functions is very similar and also similar to that used in [22, 29]. Our optimizer computes the clustering solution by first obtaining an initial k -way clustering and then applying an iterative refinement algorithm to further improve it. The algorithms that optimize unsupervised, supervised and topic-driven criterion functions differ in two ways: whether topic vectors are used as initial seeds; and whether topic vectors are allowed to move to another cluster.

4.2.1 Initialization

We employed two different ways of producing the initial clustering. For optimizing unsupervised criterion functions, during initial clustering, k documents are randomly selected to form the *seeds* of the clusters and each document is assigned to the cluster corresponding to its most similar seed.

For the various supervised and topic-driven criterion functions, the k topic vectors are used as the initial seeds for the k clusters and each document is assigned to the cluster corresponding to its most similar seed.

4.2.2 Optimization Methods

The refinement strategy that we used consists of a number of iterations. During each iteration, the documents are visited in a random order. For each document, d_i , we compute the change in the value of the criterion function obtained by moving d_i to one of the other $k-1$ clusters. If there exist some moves that lead to an improvement in the overall value of the criterion function, then d_i is moved to the cluster that leads to the highest improvement. If no such cluster exists, d_i remains in the cluster that it already belongs to. The refinement phase ends, as soon as we perform an iteration in which no documents moved between clusters. Note that unlike the traditional refinement approach used by K -means type of algorithms, the above algorithm moves a document as soon as it is determined that it will lead to an improvement in the value of the criterion function. This type of refinement algorithms are often called *incremental* [10]. Since each move directly optimizes the particular crite-

Table 1: Clustering Criterion Functions.

Weighted Scheme	
$M_1(\mathcal{I}_1) = M_1(N\mathcal{I}_1, \mathcal{S}_\mathcal{I}) = \alpha N \sum_{r=1}^k \frac{\ D_r\ ^2}{n_r} + (1 - \alpha) \sum_{r=1}^k D_r^t T_r$	
$M_1(\mathcal{I}_2) = M_1(\mathcal{I}_2, \mathcal{S}_\mathcal{I}) = \alpha \sum_{r=1}^k \ D_r\ + (1 - \alpha) \sum_{r=1}^k D_r^t T_r$	
$M_1(\mathcal{E}_1) = M_1(\mathcal{E}_1, \mathcal{S}_\mathcal{E}) = \alpha \sum_{r=1}^k \frac{D_r^t D}{\ D_r\ \ D\ } + (1 - \alpha) \sum_{r=1}^k D_r^t \bar{T}_r$	
Normalized Scheme	
$M_2(\mathcal{I}_1) = M_2(N\mathcal{I}_1, \mathcal{S}_\mathcal{I}) = \alpha \frac{\sum_{r=1}^k \frac{\ D_r\ ^2}{n_r}}{\mathcal{I}_1^*} + (1 - \alpha) \frac{\sum_{r=1}^k D_r^t T_r}{\mathcal{S}_\mathcal{I}^*}$	
$M_2(\mathcal{I}_2) = M_2(\mathcal{I}_2, \mathcal{S}_\mathcal{I}) = \alpha \frac{\sum_{r=1}^k \ D_r\ }{\mathcal{I}_2^*} + (1 - \alpha) \frac{\sum_{r=1}^k D_r^t T_r}{\mathcal{S}_\mathcal{I}^*}$	
$M_2(\mathcal{E}_1) = M_2(\mathcal{E}_1, \mathcal{S}_\mathcal{E}) = \alpha \frac{\sum_{r=1}^k \frac{D_r^t D}{\ D_r\ \ D\ }}{\mathcal{E}_1^*} + (1 - \alpha) \frac{\sum_{r=1}^k D_r^t \bar{T}_r}{\mathcal{S}_\mathcal{E}^*}$	

tion function, this refinement strategy always converges to a local minima.

Note that for the various supervised and topic-driven criterion functions, it is important to keep the topic vector always associate with its own cluster. Hence we do not allow the topic vector move to other clusters, and the clustering problem becomes that of forming clusters around the topic vectors.

The optimization method for the normalized scheme is different from the others, because it requires the optimal criterion function values obtained by optimizing the two criterion functions separately before performing the optimization of the combined criterion functions. Hence, the optimization method for the normalized scheme contains three rounds of refinement. The first two rounds optimize the two individual criterion functions, and the third round starts from the same initial clustering and uses the optimal criterion function values achieved in the first two rounds as the normalization factors.

The greedy nature of the refinement algorithm does not guarantee that it will converge to a global optima, and the local optima solution it obtains depends on the particular set of seed documents that were selected during the initial clustering. To eliminate some of this sensitivity, the overall process is repeated a number of times. That is, we compute N different clustering solutions (*i.e.*, initial clustering followed by cluster refinement), and the one that achieves the best value for the particular criterion function is kept. In all of our experiments, we used $N = 10$. For the rest of this discussion when we refer to the clustering solution we will mean the solution that was obtained by selecting the best out of these N potentially different solutions.

4.2.3 Computational Complexity

One of the advantages of our partitional algorithm and that of other similar partitional algorithms, is that it has relatively low computational requirements. A k -way clustering of a set of documents can be computed in time linear on the number of documents and the number of clusters k , as in most cases the number of iterations required by the greedy refinement algorithm is small (less than 20), and is to a large extent independent on the number of documents. The evaluation of all the various criterion functions presented in this paper at each refinement step can be implemented efficiently and bounded by a constant determined by the document that contains the maximum number of terms, thus the overall amount of time required to compute a k -way clustering solution is $O(kN)$.

5 Experimental Results

We experimentally evaluated the performance of the various topic-driven clustering schemes, compared with the corresponding unsupervised and supervised clustering schemes on five datasets, and studied various issues associated with our topic-driven clustering schemes. In the rest of this section we first describe the various datasets and our experimental methodology, followed by a description of the experimental results.

5.1 Document Collections

In our experiments, we used a total of five different datasets, whose general characteristics are summarized in Table 2. The smallest of these datasets contained 1,560 documents and the largest contained 2,838 documents. To ensure diversity in the datasets, we obtained them from different sources. For all datasets, we used a

stop-list to remove common words, and the words were stemmed using Porter’s suffix-stripping algorithm [18]. Moreover, any term that occurs in fewer than two documents was eliminated.

The datasets *trec6*, *trec7* and *trec8* were derived from the Financial Times Limited (FT) and the Los Angeles Times (LATimes) articles that are distributed as part of the TREC collection [24]. We used the queries of the ad hoc test from TREC-6 [24], TREC-7 [24] and TREC-8 [24] as the topic prototypes, and derived the datasets by including all the relevant document in FT and LATimes to particular queries. The queries that have fewer than 10 relevant documents were eliminated from the datasets. Each TREC query contains a title, a description and a narrative. The title usually contains 2-3 words as the key words. The description describes what are the contents of the relevant document briefly, and the narrative provides more detailed descriptions. Thus, we could use the titles, descriptions and narratives to form the topic prototypes of different levels of specificity. In particular, we used the titles to form short topics, titles and descriptions to form medium topics, and all three parts to form long topics.

The dataset *rel* is from Reuters-21578 text categorization test collection Distribution 1.0 [16], and contains the documents from 25 categories. For *rel*, we selected documents that have a single class label. Finally, the dataset *wap* are from the WebACE project [17, 12, 4, 5]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo! [27]. The original sources for the *rel* and *wap* datasets do not contain sufficient information that we can derive as topics. Thus, we selected the median document of each class as the representer of the content and treated it as the topic prototype.

Table 2: Summary of datasets used to evaluate the various clustering criterion functions.

Dataset	Topic #	Source	# of Docs	# of terms	# of classes
trec6	301-350	FT & LATimes	2619	32790	38
trec7	351-400	FT & LATimes	2838	33963	45
trec8	401-450	FT & LATimes	2804	36347	43
rel		Reuters-21578 [16]	1657	3758	25
wap		WebACE [12]	1560	8460	20

5.2 Experimental Methodology and Metrics

For each one of the different datasets we obtained a k -way clustering solution that optimized the various topic-driven criterion functions shown in Table 1 and Equations 4.9 and 4.8, where k is the number of topics (classes) present in the dataset. In addition, we compared the three topic-driven schemes with the cor-

responding unsupervised and supervised schemes. In particular, for supervised schemes, we compared the various \mathcal{I}_1 and \mathcal{I}_2 topic-driven criterion functions with $\mathcal{S}_{\mathcal{I}}$, and the various \mathcal{E}_1 topic-driven criterion functions with $\mathcal{S}_{\mathcal{E}}$. To illustrate whether the performance improvements gained by the various topic-driven schemes are due to the initialization with topic prototypes, we also performed another set of experiments using topics as the initial seeds but optimizing the various unsupervised criterion functions, which we will refer to as the *seed-based* scheme.

In addition, for each TREC dataset, we also compared the various topic-driven clustering schemes using long, medium, and short topics as the topic prototypes to evaluate the effectiveness of the proposed methods with topic prototypes of different levels of specificity.

The quality of a clustering solution was evaluated using the *entropy* measure that is based on how the various classes of documents are distributed within each cluster.

Given a particular cluster S_r of size n_r , the entropy of this cluster is defined to be

$$E(S_r) = -\frac{1}{\log q} \sum_{i=1}^q \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r},$$

where q is the number of classes in the dataset and n_r^i is the number of documents of the i th class that were assigned to the r th cluster. The entropy of the entire solution is defined to be the sum of the individual cluster entropies weighted according to the cluster size, *i.e.*,

$$Entropy = \sum_{r=1}^k \frac{n_r}{n} E(S_r).$$

A perfect clustering solution will be the one that leads to clusters that contain documents from only a single class, in which case the entropy will be zero. In general, the smaller the entropy values, the better the clustering solution is.

To eliminate any instances that a particular clustering solution for a particular criterion function got trapped into a bad local optimum, in all of our experiments we found ten different clustering solutions. As discussed in Section 4 each of these ten clustering solutions correspond to the best solution (in terms of the respective criterion function) out of ten different initial partitioning and refinement phases.

5.3 Comparison of the Various Clustering Schemes

The first set of experiments was focused on evaluating the quality of the clustering solutions produced by the three topic-driven schemes and the corresponding

unsupervised, supervised, and seed-based schemes for the \mathcal{I}_1 , \mathcal{I}_2 and \mathcal{E}_1 criterion functions. The long topics (including titles, descriptions, and narratives) were used for the *trec6*, *trec7*, and *trec8* datasets in this set of experiments. The α values used in the various schemes were fixed for all the datasets. We will discuss how those values were determined in Section 5.5.

Table 3 shows the relative improvements of the various topic-driven schemes and the seed-based scheme over the corresponding unsupervised and supervised schemes averaged over all the five datasets.

The results in Table 3 show several trends. First, all the topic-driven schemes outperform the corresponding unsupervised and supervised schemes, and the overall best topic-driven scheme is the normalized scheme (combining unsupervised and supervised with normalization), which achieved the most improvements for all the three criterion functions. Second, in addition to the improvements made by initializing using topics as seeds, all the topic-driven schemes made further improvements for \mathcal{I}_1 and \mathcal{I}_2 , which showed that the observed improvements are not only because of good initializations, but also because of the good properties of the proposed topic-driven criterion functions. For \mathcal{E}_1 , the normalized scheme made additional improvements but to a less extent. Finally, \mathcal{I}_1 achieved the most improvements by applying topic-driven schemes.

Table 4 shows the more detailed results of this set of experiments on each dataset. All the entries in Table 4 are entropy values, except for the two columns under the unsupervised and seed-based methods labeled “CrFun”, where the entries are the criterion function values of the clustering solutions. Note that the entropy values in the supervised method column were achieved by the supervised criterion functions compared against the other schemes. The supervised criterion function is $\mathcal{S}_{\mathcal{I}}$ for \mathcal{I}_1 and \mathcal{I}_2 , and $\mathcal{S}_{\mathcal{E}}$ for \mathcal{E}_1 . The entries that are bold-faced correspond to the methods that perform the best for a particular dataset and criterion function.

A number of observations can be made by analyzing the results in Table 4. First, for most of the cases topic-driven schemes perform the best. The exception is the *wap* dataset, for which the seed-based scheme performed the best and the topic-driven schemes sometimes even performed worse than the unsupervised scheme. One of the differences between the *wap* dataset and the rest of the datasets is that the median documents used as topics are significant longer than the topics used in other datasets. Since we did not perform any pruning on the median documents, they contain the terms that are not specific to the topic. As a result, the performances of the supervised schemes were much worse than other schemes and topic-driven schemes did not benefit from

incorporating these topics. Second, the two supervised schemes $\mathcal{S}_{\mathcal{I}}$ and $\mathcal{S}_{\mathcal{E}}$ perform similarly, which is not surprising as we have discussed the relationship between them in Section 4.1.2. Finally, \mathcal{E}_1 benefits the most by using the topics as initial seeds. By looking at the criterion function values achieved by the unsupervised and seed-based schemes for \mathcal{E}_1 , we can see that unlike \mathcal{I}_1 and \mathcal{I}_2 , using topics in the initialization process helped the optimization process to find a clustering solution with a better criterion function value for \mathcal{E}_1 .

5.4 Topic Prototypes of Different Levels of Specificity

The second set of experiments was focused on how the various topic-driven schemes perform with the topic prototypes of different levels of specificity. For each TREC dataset, we performed the same set experiments as in Section 5.3 with long, medium, and short topics. The results are shown in Table 5, in which all the entries are the entropy values of the clustering solutions obtained by the various schemes. Again, the entropy results for the weighted and normalized schemes were obtained with a fixed α value for all the datasets. The entries that are bold-faced correspond to the methods that perform the best for a particular dataset and criterion function.

A number of observations can be made by analyzing the results in Table 4. First, the supervised scheme performs better as the topics become more specific for all the datasets. Second, for most of the cases topic-driven schemes perform the best. Finally, overall the various topic-driven schemes performed similarly with the topic prototypes of different levels of specificity, despite the fact that the short and medium topics alone (used in the supervised scheme) perform much worse than the long topics, which shows that the proposed topic-driven schemes are effective with the topic prototypes of different levels of specificity.

5.5 Parameter Sensitivity

In this section, we present the results of the parameter study on α for the weighted and normalized schemes, and show how we determined the α values that were used to produce clustering solutions shown in Table 4. The purpose of this study is two-fold: (1) to see whether there is a range of α values that can perform well for most of the datasets; (2) to see which scheme is better by comparing the dynamic range and how sensitive the two schemes are to the change of α values. In particularly, we tested the two schemes with $\alpha = 0.1$ to 0.9 with an increment of 0.1 on the five datasets for all three criterion functions. Note that the two combined schemes emphasize more on the supervised component

Table 3: Average relative improvements of the various topic-driven schemes over the unsupervised and supervised schemes.

	Seed-based Scheme			Topic-driven Schemes							
	\mathcal{E}_1	\mathcal{I}_1	\mathcal{I}_2	Hybrid		Weighted			Normalized		
CrFun	\mathcal{E}_1	\mathcal{I}_1	\mathcal{I}_2	$H(\mathcal{I}_1)$	$H(\mathcal{I}_2)$	$M_1(\mathcal{E}_1)$	$M_1(\mathcal{I}_1)$	$M_1(\mathcal{I}_2)$	$M_2(\mathcal{E}_1)$	$M_2(\mathcal{I}_1)$	$M_2(\mathcal{I}_2)$
Unsupervised	6%	3%	5%	15%	9%	5%	26%	11%	9%	26%	12%
Supervised	23%	4%	26%	16%	30%	22%	27%	31%	25%	27%	32%

Table 4: Comparison of the clustering solutions obtained by the various clustering methods.

trec6								
CrFun	Unsupervised Method		Supervised Methods	Seed-based Method		Topic-driven Methods		
	CrFun	Entropy		CrFun	Entropy	$H()$	$M_1()$	$M_2()$
\mathcal{E}_1	3.98	0.238	0.281	3.96	0.210		0.215	0.193
\mathcal{I}_1	4.37	0.275	0.283	4.32	0.268	0.238	0.177	0.180
\mathcal{I}_2	1.01	0.208	0.283	1.01	0.196	0.147	0.178	0.162

trec7								
CrFun	Unsupervised Method		Supervised Methods	Seed-based Method		Topic-driven Methods		
	CrFun	Entropy		CrFun	Entropy	$H()$	$M_1()$	$M_2()$
\mathcal{E}_1	4.76	0.261	0.305	4.76	0.239		0.249	0.228
\mathcal{I}_1	4.67	0.322	0.307	4.59	0.334	0.264	0.217	0.219
\mathcal{I}_2	1.09	0.227	0.307	1.09	0.235	0.215	0.207	0.211

trec8								
CrFun	Unsupervised Method		Supervised Methods	Seed-based Method		Topic-driven Methods		
	CrFun	Entropy		CrFun	Entropy	$H()$	$M_1()$	$M_2()$
\mathcal{E}_1	4.51	0.201	0.277	4.50	0.215		0.196	0.185
\mathcal{I}_1	4.19	0.270	0.275	4.16	0.278	0.252	0.186	0.175
\mathcal{I}_2	1.04	0.208	0.275	1.03	0.194	0.188	0.170	0.160

rel								
CrFun	Unsupervised Method		Supervised Methods	Seed-based Method		Topic-driven Methods		
	CrFun	Entropy		CrFun	Entropy	$H()$	$M_1()$	$M_2()$
\mathcal{E}_1	1.94	0.297	0.315	1.93	0.273		0.277	0.282
\mathcal{I}_1	2.66	0.365	0.311	2.59	0.308	0.288	0.270	0.252
\mathcal{I}_2	6.31	0.290	0.311	6.29	0.265	0.268	0.252	0.247

wap								
CrFun	Unsupervised Method		Supervised Methods	Seed-based Method		Topic-driven Methods		
	CrFun	Entropy		CrFun	Entropy	$H()$	$M_1()$	$M_2()$
\mathcal{E}_1	1.75	0.331	0.442	1.76	0.307		0.327	0.337
\mathcal{I}_1	1.65	0.372	0.454	1.65	0.355	0.306	0.359	0.377
\mathcal{I}_2	4.69	0.327	0.454	4.66	0.306	0.342	0.332	0.347

with a smaller α value, and emphasize more on the unsupervised component with a larger α value. The two combined schemes become the supervised scheme and the unsupervised scheme with $\alpha = 0$ and 1, respectively.

The results for \mathcal{I}_1 and \mathcal{I}_2 are similar and we only show the results for \mathcal{I}_2 . In Figure 1, we plot the entropy values obtained by the weighted and normalized schemes against the α values for all the datasets in a) and b), respectively. We can see that at $\alpha = 0.4$ and $\alpha = 0.6$, most of the datasets reached the best entropy value (or close to the best entropy value) for the weighted and normalized schemes, respectively. We also tested the same α values on the same dataset with long, medium, and short topics. Figure 2 shows such a plot of the entropy values obtained by the weighted and normalized

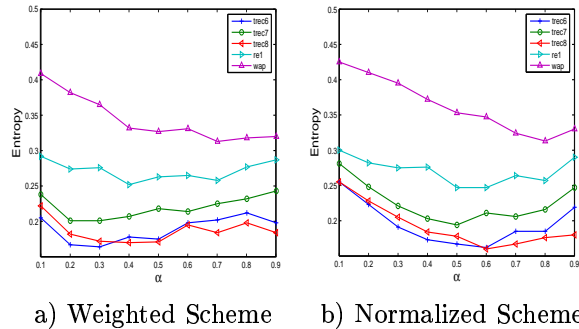


Figure 1: Entropy values obtained by the weighted and normalized schemes with \mathcal{I}_2 for all the datasets.

Table 5: Comparison of the clustering solutions obtained by the various clustering methods with long, medium, and short topics.

trec6							
Topic Type	CrFun	Unsupervised Method	Supervised Methods	Seed-based Method	Topic-driven Methods		
					$H()$	$M_1()$	$M_2()$
long	\mathcal{E}_1	0.238	0.281	0.210		0.215	0.193
medium	\mathcal{E}_1		0.298	0.206		0.220	0.191
short	\mathcal{E}_1		0.323	0.211		0.215	0.216
long	\mathcal{I}_1	0.275	0.283	0.268	0.238	0.177	0.180
medium	\mathcal{I}_1		0.299	0.269	0.246	0.199	0.198
short	\mathcal{I}_1		0.323	0.279	0.314	0.222	0.205
long	\mathcal{I}_2	0.208	0.283	0.196	0.147	0.178	0.162
medium	\mathcal{I}_2		0.299	0.205	0.157	0.161	0.181
short	\mathcal{I}_2		0.323	0.208	0.186	0.174	0.190

trec7							
Topic Type	CrFun	Unsupervised Method	Supervised Methods	Seed-based Method	Topic-driven Methods		
					$H()$	$M_1()$	$M_2()$
long	\mathcal{E}_1	0.261	0.305	0.239		0.249	0.228
medium	\mathcal{E}_1		0.334	0.244		0.239	0.227
short	\mathcal{E}_1		0.371	0.242		0.241	0.247
long	\mathcal{I}_1	0.322	0.307	0.334	0.264	0.217	0.219
medium	\mathcal{I}_1		0.332	0.318	0.279	0.219	0.221
short	\mathcal{I}_1		0.370	0.331	0.295	0.247	0.248
long	\mathcal{I}_2	0.227	0.307	0.235	0.215	0.207	0.211
medium	\mathcal{I}_2		0.332	0.246	0.202	0.210	0.204
short	\mathcal{I}_2		0.371	0.233	0.208	0.224	0.228

trec8							
Topic Type	CrFun	Unsupervised Method	Supervised Methods	Seed-based Method	Topic-driven Methods		
					$H()$	$M_1()$	$M_2()$
long	\mathcal{E}_1	0.201	0.277	0.215		0.196	0.185
medium	\mathcal{E}_1		0.292	0.200		0.201	0.193
short	\mathcal{E}_1		0.303	0.194		0.217	0.189
long	\mathcal{I}_1	0.270	0.275	0.278	0.252	0.186	0.175
medium	\mathcal{I}_1		0.291	0.265	0.260	0.190	0.191
short	\mathcal{I}_1		0.295	0.282	0.255	0.185	0.185
long	\mathcal{I}_2	0.208	0.275	0.194	0.188	0.170	0.160
medium	\mathcal{I}_2		0.291	0.196	0.182	0.169	0.172
short	\mathcal{I}_2		0.300	0.202	0.176	0.171	0.174

schemes against the α values for the *trec8* dataset in a) and b), respectively. The results are similar for *trec6* and *trec7* as well. As shown in Figure 2, the results of long, medium, and short topics are very similar to one another. In Figures 1 and 2, the curves produced by the normalized scheme are smoother than those by the weighted scheme. However, the dynamic range of the weighted scheme is narrower around the α value that achieved the best entropy value than that of the normalized scheme, which suggests that the weighted scheme can achieve relative good performance with a broader choice of α values for \mathcal{I}_2 .

Figure 3 shows the same plot generated by the weighted and normalized schemes with \mathcal{E}_1 for all the datasets. Unlike \mathcal{I}_1 and \mathcal{I}_2 , the trend and the best α value differ from dataset to dataset for \mathcal{E}_1 , especially for the weighted scheme, which suggests that the problem of different scales has greater impacts on \mathcal{E}_1 than \mathcal{I}_1 and \mathcal{I}_2 . Another difference between \mathcal{I}_2 and \mathcal{E}_1 is that for short topics, the normalized scheme tends to achieve

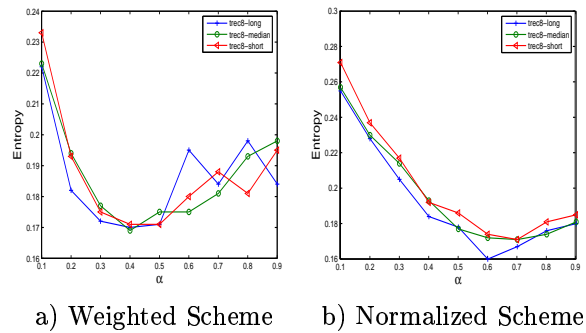


Figure 2: Entropy values obtained by the weighted and normalized schemes with \mathcal{I}_2 for *trec8* with long, medium and short topics.

the best performance with a larger α value as shown in Figure 4. Since there is no consistent trend for the weighted scheme with \mathcal{E}_1 , we selected the α value

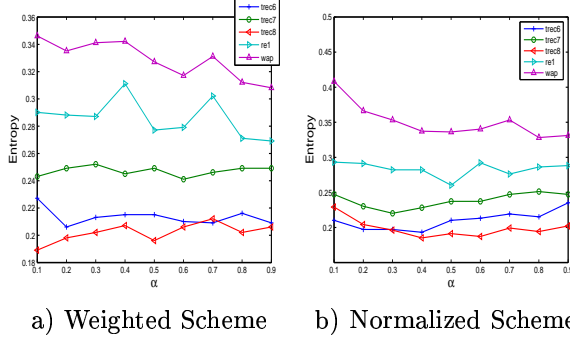


Figure 3: Entropy values obtained by the weighted and normalized schemes with \mathcal{E}_1 for all the datasets.

Table 6: Selected α values for the various combined schemes and criterion functions.

	\mathcal{I}_1	\mathcal{I}_2	\mathcal{E}_1	\mathcal{E}_1 (short)
Weighted Scheme	0.4	0.4	0.5	0.5
Normalized Scheme	0.4	0.6	0.5	0.7

that achieved the best average entropy value. For the normalized scheme with \mathcal{E}_1 , we determined the α value in a similar fashion. The only difference is that we selected one α value for short topics and another α value for the rest of the cases.

In summary, the selected α values that were used to produce the clustering solutions in Sections 5.3 and 5.4 are shown in Table 6.

Furthermore, we compared the performance of the two combined schemes with the fixed α values with the best performance among all the tested α values, and calculated the relative degradation on all the datasets. We show the box plots of the relative degradations for all the combined criterion functions in Figure 5.

A number of observations can be made by analyzing the results in Figure 5. First, for all the cases, the median relative degradation is lower than 2% except for $M_1(\mathcal{E}_1)$, which suggests that the fixed α can perform well for most of the datasets. The poor performance of the fixed α for $M_1(\mathcal{E}_1)$ is consistent with that fact that the weighted scheme does not perform well with \mathcal{E}_1 as shown in Figure 3. Second, the variance of the relative degradation of the normalized scheme is larger than that of the weighted scheme for \mathcal{I}_1 and \mathcal{I}_2 , which is consistent with the fact that the weighted scheme has a narrower dynamic range than the normalized scheme for \mathcal{I}_2 as shown in Figure 1.

6 Concluding Remarks

In this paper, we defined the problem of *topic-driven clustering*, which organizes a document collection ac-

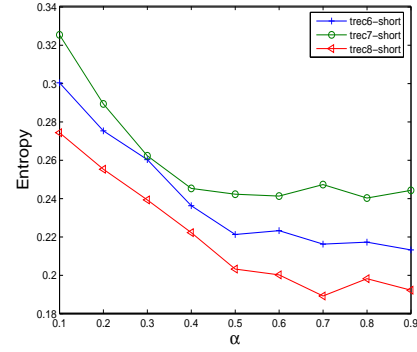


Figure 4: Entropy values obtained by the normalized scheme with \mathcal{E}_1 and short topics.

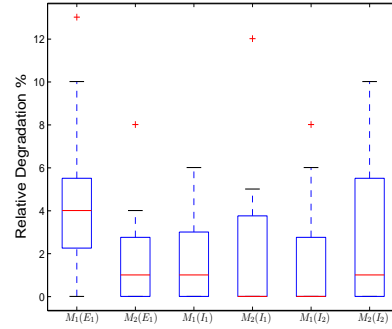


Figure 5: Relative performance of the selected α for all the combined criterion functions.

cording to a given set of topics, such that the resultant clusters correspond to the given topics and the documents in the same cluster are similar to the cluster topic. We proposed three efficient topic-driven schemes that consider the similarity between the document to its topic and the relationship between the documents themselves simultaneously. Our experimental results showed that the proposed topic-driven schemes outperform the unsupervised and supervised schemes, which suggests that the proposed topic-driven schemes take advantages of both the unsupervised and supervised components. We also showed that the proposed topic-driven schemes perform well with topic prototypes of different levels of specificity.

7 Acknowledgments

We will like to thank Jack G. Conrad, Senior Research Scientist in Research & Development at Thomson Legal & Regulatory, for introducing us to the typical knowledge management environment of law firms and valuable discussions on the clustering requirements in law firms.

References

- [1] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. of the 10th Int'l Conference on Knowledge Discovery and Data Mining*, 2004.
- [2] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. of 21th International Conference on Machine Learning (ICML-2004)*, 2004.
- [3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Computational Learning Theory 11*, 1998.
- [4] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using WebACE. *AI Review*, 11:365–391, 1999.
- [5] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.
- [6] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.
- [7] D.R. Cutting, J.O. Pedersen, D.R. Karger, and J.W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the ACM SIGIR*, pages pages 318–329, Copenhagen, 1992.
- [8] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001.
- [9] Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst Simon. Spectral min-max cut for graph partitioning and data clustering. Technical Report LBNL-47937, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, 2001.
- [10] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.
- [11] David Gondek and Thomas Hofmann. Non-redundant data clustering. In *Proc. of the fourth IEEE International Conference on Data Mining*, 2004.
- [12] E.H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. WebACE: A web agent for document categorization and exploitation. In *Proc. of the 2nd International Conference on Autonomous Agents*, May 1998.
- [13] T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press, 1999.
- [14] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. J. Wiley & Sons, New York, 1976.
- [15] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.
- [16] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. <http://www.research.att.com/~lewis>, 1999.
- [17] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering. In *7th Workshop on Information Technologies and Systems*, Dec. 1997.
- [18] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [19] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *PATREC: Pattern Recognition*, Pergamon Press, 33:617–634, 2000.
- [20] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
- [21] K. Schloegel, G. Karypis, and V. Kumar. A new algorithm for multi-objective graph partitioning. In *Proceedings of Europar 1999*, September 1999.
- [22] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
- [23] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.
- [24] TREC. Text REtrieval conference. <http://trec.nist.gov>, 1999.
- [25] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. of 18th International Conference on Machine Learning (ICML-2001)*, pages 577–584, 2001.
- [26] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems 15*, pages 505–512, 2003.
- [27] Yahoo! Yahoo! <http://www.yahoo.com>.
- [28] P. Yu. *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*. Plenum Press, New York, 1965.
- [29] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report TR #01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001. Available on the WWW at <http://cs.umn.edu/~karypis/publications>.
- [30] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc. of Int'l. Conf. on Information and Knowledge Management*, pages 515–524, 2002.
- [31] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

Variational Learning for Noisy-OR Component Analysis

Tomas Singliar and Milos Hauskrecht
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
{*tomas, milos*}@cs.pitt.edu

Abstract

Latent factor models offer a very useful framework for modeling dependencies in high-dimensional multivariate data. In this work we investigate a class of latent factor models with hidden noisy-or units that let us decouple high dimensional vectors of observable binary random variables using a ‘small’ number of hidden binary factors. Since the problem of learning of such models from data is intractable, we develop its variational approximation. We analyze special properties of the optimization problem, in particular its “built-in” regularization effect and discuss its importance for model recovery. We test the noisy-or model on an image deconvolution problem and illustrate the ability of the variational method to successfully learn the underlying image components. Finally, we apply the latent noisy-or model to analyze citations in a large collection of Statistical Machine Learning papers and show the benefit of the model and algorithms by discovering useful and semantically sound components characterizing the dataset.

Keywords: Learning, Variational methods, Bayesian networks.

1 Introduction

Latent variable models [14, 2] provide a very useful framework for modeling dependencies in high dimensional data. The models are often used in the component analysis where we want to identify characteristics of a small number of underlying components (factors, sources, or signals) that combine into the expression of observed high dimensional data. Examples of latent factor models include probabilistic principal component analysis [18, 3], mixtures of factor analysers [1], multinomial PCA (or aspect) models [5, 10, 4], multi-cause model [9, 16], or other independent component analysis frameworks [1, 15]. In addition to their role in modeling and understanding the structure of high-

dimensional data, latent factor models used in the component analysis can be applied in the dimensionality reduction where the vector of hidden factors represents a low-dimensional representation of the data sample.

In this work we investigate a special class of latent factor models that let us represent high-dimensional multivariate distributions of binary attributes and their local dependencies. The dependencies are modeled in terms of a small number of hidden binary factors that are combined through noisy-or units. Intuitively, noisy-or units let us model local dependencies (couplings) among observable components in the data indirectly – in terms of hidden factors and their values. Such a framework is especially useful if we want to model random binary variables with confounded stochastic fluctuations. However, it can be also applied in more general settings to approximate local dependencies among random variables. We believe that models with such characteristics can be very useful and applied to represent stochastic dependencies among components of large distributed systems, such as failures or congestions in transportation networks, spread of disease in epidemiology, and others.

The key step of component analysis corresponds to the learning of the parameters of the latent factor model from the data. Once the model is learned it can be used to make inferences on hidden factors, such as to identify the document topics in the aspect model [10, 4] or regulatory signals in the microarray DNA data [13]. In the statistical sense the learning corresponds to the estimation of parameters of the model. The limitation of latent factor models is the complexity of the learning problem; the standard EM formulation (decomposition) becomes exponential in the number of hidden factors. Variational approximations offer one possible solution to make the learning task more efficient, but at some loss of accuracy. To address this problem, we develop and test a variational learning algorithm for optimizing the parameters of the noisy-or network with hidden factors. Our algorithm builds upon and extends the work of

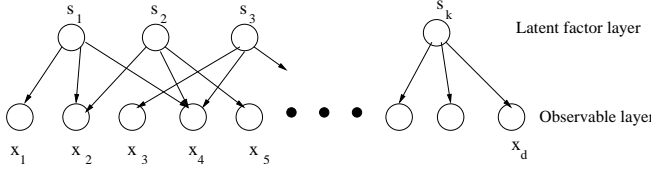


Figure 1: A bipartite belief network structure of the latent factor models with noisy-or units. \mathbf{x} is a vector of binary random variables that are observable and \mathbf{s} is a vector of hidden binary factors. Latent factors and noisy-or units model local interactions between components of x_j .

Jaakkola and Jordan who focused on and developed methods of variational inference for noisy-or networks [11]. The methods for learning a noisy-or network model with hidden components have not yet been investigated, to our knowledge. A very restricted model was explored by Kearns and Mansour [12] but their algorithm is exponential in the maximum number of hidden factors contributing to any observable variable. Our algorithm does not make any structural assumption and it is able to recover very well the active (nonzero) structure of a noisy-or network.

In the following text, we first describe the noisy-or network model with hidden units and its limitation in efficient inferences. Next we analyze the problem of learning the parameters of the latent factor network from data and point out the shortcomings of the exact Expectation-Maximization (EM) technique associated with its computational complexity. To alleviate the problems with the exact EM we develop and present its variational approximation. We test the model and the approximation algorithm on a synthetic image deconvolution problem. We investigate two aspects of the approach: recovery of complex multivariate distributions and dimensionality reduction, and show a very good performance of the algorithm on both of these tasks. Finally, we apply the model to analyze citations in a large collection of Machine Learning papers. We show the benefit of the new model and the algorithm by discovering useful and semantically sound subcommunities characterizing the dataset.

2 Latent Factor Model with Noisy-or Units

Consider a latent variable model with bipartite belief network structure illustrated in Figure 1. Nodes in the top layer represent a vector of latent factors $\mathbf{s} = \{s_1, s_2, \dots, s_K\}$ with binary $\{0, 1\}$ values and nodes in the bottom layer an observable vector of binary variables $\mathbf{x} = \{x_1, x_2, \dots, x_d\}$. We assume that \mathbf{x} is a

high-dimensional vector and that $d > K$. The connections between two layers of the bipartite graph represent dependencies among the components of the observable variables: the nodes coupled by one of the latent factor nodes are assumed to exhibit a local dependency pattern. The probabilistic dependency between nodes in the two layers is modeled via the noisy-or conditional distribution, which gives us a compact (low-complexity) parameterization of the relation among configurations of hidden factors and observable variables. The parameters Θ of the model consist of:

- a set of prior probabilities π_i parameterizing the (Bernoulli) prior distributions $P(s_i)$ for every hidden factor i ;
- a set of probabilities p_{ij} representing parameters of noisy-or conditional probability tables, one for each pair of hidden factor i and observed component j .

The structure of the model is similar to the QMR-DT model used for diagnosis in internal medicine [17]; the difference is that the top layer variables in our model are hidden. Their sole purpose is to model stochastic interaction patterns among observable variables in \mathbf{x} .

2.1 The joint distribution over observables

Given the bipartite model, the joint probability of an observation vector \mathbf{x} is defined as:

$$P(\mathbf{x}) = \sum_{\{\mathbf{s}\}} \left(\prod_{j=1}^d p(x_j|\mathbf{s}) \right) \left(\prod_{i=1}^K p(s_i) \right), \quad (2.1)$$

where $\{\mathbf{s}\}$ denotes the sum over all configurations of \mathbf{s} , and $P(s_i)$ is the prior probability of a hidden factor s_i . Given a vector of hidden binary factors \mathbf{s} , the conditional probability $p(x_j|\mathbf{s})$ for an observable random component $x_j \in \{0, 1\}$ is obtained through a noisy-or model as:

$$P(x_j|\mathbf{s}) = \left[1 - (1 - p_{0j}) \prod_{i=1}^K (1 - p_{ij})^{s_i} \right]^{x_j} \cdot \left[(1 - p_{0j}) \prod_{i=1}^K (1 - p_{ij})^{s_i} \right]^{(1-x_j)} \quad (2.2)$$

where p_{0j} is the leak probability that models “all other” causes. The Equation 2.2 can be reparameterized with $\theta_{ij} = -\log(1 - p_{ij})$ to obtain:

$$\begin{aligned} P(x_j|\mathbf{s}) &= \exp \left[x_j \log \left(1 - \exp \left\{ -\theta_{0j} - \sum_{i=1}^K \theta_{ij} s_i \right\} \right) \right] \\ &\quad + (1 - x_j) \left(-\theta_{0j} - \sum_{i=1}^K \theta_{ij} s_i \right). \end{aligned} \quad (2.3)$$

2.2 Factorization The bottleneck in computing the joint probability over observables $P(\mathbf{x})$ in Equation 2.1 is the sum that ranges over all possible latent factor configurations, and thus, it is exponential in K . It is easy to see that if $P(x_j|\mathbf{s})$ for both $x_j = 0$ and $x_j = 1$ can be expressed as:

$$P(x_j|\mathbf{s}) = \prod_{i=1}^K h(x_j|s_i), \text{ such that } \forall i, j : h(x_j|s_i) \geq 0 \quad (2.4)$$

then the full joint and the joint over the observables $P(\mathbf{x})$ decompose as:

$$P(\mathbf{x}, \mathbf{s}) = \prod_{j=1}^d P(x_j|\mathbf{s}) \prod_{i=1}^K P(s_i) = \prod_{i=1}^K \left(P(s_i) \prod_{j=1}^d h(x_j|s_i) \right), \quad (2.5)$$

$$\begin{aligned} P(\mathbf{x}) &= \sum_{\{\mathbf{s}\}} \prod_{i=1}^K \left(P(s_i) \prod_{j=1}^d h(x_j|s_i) \right) \\ &= \prod_{i=1}^K \left(\sum_{\{s_i\}} \left[\prod_{j=1}^d h(x_j|s_i) \right] P(s_i) \right). \end{aligned} \quad (2.6)$$

But this means that the summation in Equation 2.1 can be performed efficiently. We note that Condition 2.4 is sufficient to ensure the efficiency of other probabilistic inferences, such as the posterior of a hidden factor s_i :

$$P(s_i|\mathbf{x}) \sim P(s_i) \prod_{j=1}^d h(x_j|s_i). \quad (2.7)$$

2.3 Factorization via variational approximation

The Equation 2.3 for $P(x_j|\mathbf{s})$ does not factorize for $x_j = 1$. Thus, in general, it is impossible to compute $P(\mathbf{x})$ efficiently. To address this problem we approximate $P(x_j|\mathbf{s})$ for $x_j = 1$ with a factored variational lower bound used by Jaakkola and Jordan [11] in fully observable settings:

$$\begin{aligned} P(x_j = 1|\mathbf{s}) & \quad (2.8) \\ & \geq \prod_{i=1}^K \exp \left\{ q_j(i) s_i \left[\log(1 - e^{-\theta_{0j} - \frac{\theta_{ij}}{q_j(i)}}) \right. \right. \\ & \quad \left. \left. - \log(1 - e^{-\theta_{0j}}) \right] + q_j(i) \log(1 - e^{-\theta_{0j}}) \right\}, \end{aligned}$$

where q_j s represent sets of variational parameters defining a multinomial distribution. Each component $q_j(i)$ of the distribution can be viewed as a responsibility of a latent factor s_i for observing $x_j = 1$.

Incorporating the variational bound in the first part of Equation 2.3 we can obtain approximations $\tilde{P}(\mathbf{x}|\mathbf{s}, \Theta, \mathbf{q}) \leq P(\mathbf{x}|\mathbf{s}, \Theta)$, $\tilde{P}(\mathbf{x}, \mathbf{s}|\Theta, \mathbf{q}) \leq P(\mathbf{x}, \mathbf{s}|\Theta)$ and $\tilde{P}(\mathbf{x}|\Theta, \mathbf{q}) \leq P(\mathbf{x}|\Theta)$ that factorize along latent factors s_i .

3 Learning of Noisy-or Networks with Hidden Units

The problem of learning of noisy-or bipartite networks has been addressed only in fully observable settings, that is, when both sources and observations are known. The learning methods take advantage of the decomposition of the model created by the introduction of special hidden variables. EM algorithm is then used to estimate the parameters of the modified network, which translate directly into the parameters of the original model. A reader interested in these transformations may consult papers by Heckerman [6], Vomlel [19] or Diez and Galan [8].

3.1 Standard EM learning Learning of the latent factor version of the Noisy-or network is much harder. Let $D = \{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N\}$ be a set of independent identically distributed samples of observable variables. Our objective is to find parameters Θ that maximize the likelihood of the data, $P(D|\Theta)$. A standard approach to learn the parameters of the model in the presence of hidden variables is to use the Expectation-Maximization (EM) algorithm [7]. The EM computes the parameters iteratively by taking the following parameter update step:

$$\Theta^* = \arg \max_{\Theta} \sum_{n=1}^N \langle \log P(\mathbf{x}^n, \mathbf{s}^n|\Theta) \rangle_{P(\mathbf{s}^n|\mathbf{x}^n, \Theta')}$$

where Θ' denotes previous-step parameters.

The main problem in applying the EM to our noisy-or model is that the joint distribution over every “completed” sample $P(\mathbf{x}^n, \mathbf{s}^n|\Theta)$ does not decompose along hidden factors s_i and its expectation $\langle \log P(\mathbf{x}^n, \mathbf{s}^n|\Theta) \rangle_{P(\mathbf{s}^n|\mathbf{x}^n, \Theta')}$ requires to iterate over all possible factor configurations. This is unfeasible since the configuration space grows exponentially in the number of factors. To alleviate this problem we optimize the parameters using the variational learning framework.

3.2 EM for variational learning The idea is to approximate the likelihood terms with their imprecise, but structurally more convenient surrogates. Additional set of free variational parameters is introduced to offer more flexibility and tune the approximation to specific settings. In our model, we substitute true conditional probabilities $P(\mathbf{x}^n|\mathbf{s}^n, \Theta)$ that do not factorize, with

their factored lower-bound variational approximation $\tilde{P}(\mathbf{x}^n|\mathbf{s}^n, \Theta, \mathbf{q}^n)$ as described in Section 2.3. Note that every datapoint \mathbf{x}^n comes with a different set of variational parameters \mathbf{q}^n .

In maximum likelihood learning we optimize the loglikelihood $\log P(D|\Theta)$. In our variational approach we optimize its lower bound:

$$\log \tilde{P}(D|\Theta, \mathbf{q}) = \sum_{n=1}^N \log \tilde{P}(\mathbf{x}^n|\Theta, \mathbf{q}^n)$$

The new quantity $\log \tilde{P}(D|\Theta, \mathbf{q})$ depends on both parameters of the noisy-or model Θ as well as on the variational parameters \mathbf{q} . Although we are ultimately interested in optimizing Θ and variational parameters only play an auxiliary role, from the viewpoint of optimization of $\log \tilde{P}(D|\Theta, \mathbf{q})$ there is no difference between the two and they must be treated the same way. Such an optimization can be carried out within the EM framework. In particular, the quantity can be maximized by iteratively reoptimizing (Θ, \mathbf{q}) pairs:

$$(\Theta, \mathbf{q})^* = \arg \max_{\Theta, \mathbf{q}} \sum_{n=1}^N \left\langle \log \tilde{P}(\mathbf{x}^n, \mathbf{s}^n|\Theta, \mathbf{q}^n) \right\rangle, \quad (3.9)$$

where $\langle \cdot \rangle$ denotes the expectation, in this case taken over $P(\mathbf{s}^n|\mathbf{x}^n, \Theta', \mathbf{q}'^n)$ and

$$\tilde{P}(\mathbf{x}^n, \mathbf{s}^n|\Theta, \mathbf{q}^n) = \tilde{P}(\mathbf{x}^n|\mathbf{s}^n, \Theta, \mathbf{q}^n)P(\mathbf{s}^n|\Theta) \quad (3.10)$$

$$P(\mathbf{s}^n|\mathbf{x}^n, \Theta', \mathbf{q}'^n) = Q'(\mathbf{s}^n) = \frac{\tilde{P}(\mathbf{x}^n, \mathbf{s}^n|\Theta, \mathbf{q}^n)}{\tilde{P}(\mathbf{s}^n|\Theta', \mathbf{q}'^n)},$$

and Θ', \mathbf{q}'^n represent values of the parameters in the previous step. To simplify the notation in the rest of the paper, we use $Q'(\mathbf{s}^n)$ to denote the posterior on hidden factors given the previous-step parameter values. Note that even if the \tilde{P} quantities are not probabilities, the posterior $Q'(\mathbf{s}^n)$ is.

3.3 Factorization of Expectations Thanks to the factored form of $\tilde{P}(\mathbf{x}^n|\mathbf{s}^n, \Theta, \mathbf{q}^n)$, optimization steps in Equation 3.9 do not require us to iterate explicitly over all possible hidden factor configurations. More specifically, by substituting the expressions for $\tilde{P}(\mathbf{x}^n, \mathbf{s}^n|\Theta, \mathbf{q}^n)$ and by taking the expectation in terms of the posterior $Q'(\mathbf{s}^n)$ we obtain:

$$\begin{aligned} & \left\langle \log \tilde{P}(\mathbf{x}^n, \mathbf{s}^n|\mathbf{q}) \right\rangle_{Q'(\mathbf{s}^n)} \\ &= \left[\sum_{i=1}^K \langle s_i^n \rangle_{Q'(\mathbf{s}^n)} \log \frac{\pi_i}{(1 - \pi_i)} + \log(1 - \pi_i) \right] + \\ &+ \left[\sum_{j=1}^d \left(\sum_{i=1}^K -\langle s_i^n \rangle_{Q'(\mathbf{s}^n)} - \theta_{ij}(1 - x_j^n) \right) - \theta_{0j}(1 - x_j^n) \right] \end{aligned} \quad (3.11)$$

$$\begin{aligned} &+ \sum_{j=1}^d \sum_{i=1}^K \left[\langle s_i^n \rangle_{Q'(\mathbf{s}^n)} q_j^n(i) x_j^n \log \left(1 - e^{-\theta_{0j} - \frac{\theta_{ij}}{q_j^n(i)}} \right) \right. \\ &+ \left. (1 - \langle s_i^n \rangle_{Q'(\mathbf{s}^n)}) q_j^n(i) x_j^n \log(1 - e^{-\theta_{0j}}) \right] \end{aligned}$$

We see that for our factored approximation, the expectations are easy and the computations boil down to taking expectations over individual factors. Since the hidden factors take on binary values 0 and 1, the expectations for individual factors are just their probabilities of assuming value 1 and can be calculated using Equation 2.7.

3.4 Parameter optimizations in EM In every cycle of the EM algorithm we must reoptimize both the parameters Θ and all variational parameters \mathbf{q}^n , one set per every data point. Unfortunately, no closed form solution for this task exists. We resort to iterative solutions, where parameters \mathbf{q}^n and Θ are updated (optimized) until convergence.

We apply numerical and iterative optimization techniques to obtain partial solutions. However, we note that the dependencies among parameters to be optimized are relatively sparse and optimizations can be often performed quite efficiently. In particular, the iterative formula for a variational parameter $q_j^n(i)$ only involves $q_j^n(i)$ itself. We are dealing with DK instances of one-dimensional optimization for each datapoint, rather than with optimization in a higher-dimensional space.

Complete parameter update formulas we derived and use in our procedure are summarized in Figure 2. The updates were derived by calculating partial derivatives of the objective function and setting them to 0.

The precise analysis of algorithm's time complexity would be a tedious undertaking as it involves considerations of the convergence rates of nested iterative procedures. We demonstrate experimentally that the approximation yields a tractable algorithm.

3.5 Regularization effect While testing our variational learning algorithm we noticed its ability to automatically shut off “unused” noisy-or links. This suggests the presence of an inherent regularization correction. Examining the objective function (Equation 3.9) and optimization updates (in Figure 2) we can see that there is indeed a “natural” tendency of the method to drive unused parameters to 0, due to the presence of the term: $-\langle s_i^n \rangle_{Q'(\mathbf{s}^n)} \theta_{ij}(1 - x_j^n)$ in the objective function in Equation 3.9. The term can be viewed as a regularization penalty assigned to large values of θ_{ij} if these are not supported by data. Intuitively, the link with a poor

Updates of variational parameters $q_j^n(i)$ (one per sample). Iterate until fixpoint:

$$q_j^n(i) \leftarrow \langle s_i^n \rangle_{Q'(s)} \frac{1}{\log(1 - e^{-\theta_{0j}})} \left[q_j^n(i) \log(1 - A^n(i, j)) - \theta_{ij} \frac{A^n(i, j)}{1 - A^n(i, j)} - q_j^n(i) \log(1 - e^{-\theta_{0j}}) \right] \quad (3.12)$$

subject to condition $\sum_{i=1}^K q_j^n(i) = 1$ assured through the normalization step. $A^n(i, j)$ stands for $e^{-\theta_{0j} - \frac{\theta_{ij}}{q_j^n(i)}}$.

Updates of θ_{ij} s. Search for the root of $\partial \mathcal{F} / \partial \theta_{ij}$ via a numerical method:

$$\sum_{n=1}^N \langle s_i^n \rangle_{Q'(s)} \left[-1 + x_j^n \frac{1}{1 - A^n(i, j)} \right] = 0 \quad (3.13)$$

Updates θ_{0j} s. Search for the root of $\partial \mathcal{F} / \partial \theta_{0j}$ via a numerical method:

$$\sum_{n=1}^N \left[\sum_{i=1}^K \langle s_i^n \rangle_{Q'(s)} q_j^n(i) x_j^n \left(\frac{A^n(i, j)}{1 - A^n(i, j)} - \frac{e^{-\theta_{0j}}}{1 - e^{-\theta_{0j}}} \right) \right] + \left[-(1 - x_j^n) + \sum_{i=1}^K x_j^n q_j^n(i) \frac{e^{-\theta_{0j}}}{1 - e^{-\theta_{0j}}} \right] = 0 \quad (3.14)$$

Updates of π_i s:

$$\pi_i = \frac{1}{N} \sum_{n=1}^N \langle s_i^n \rangle_{Q'(s)} \quad (3.15)$$

Figure 2: A summary of iterative optimization steps for the variational learning method

support in the data is shut down to avoid the penalty.

4 Evaluation of the variational learning algorithm

To analyze the performance of our variational algorithm, we applied it first to an image deconvolution problem. In this problem, we use a bipartite noisy-or network with 8 hidden sources. Each source is associated with an 8×8 image pattern. The patterns are shown in Figure 3. If the source is active (set to 1) its noise-corrupted pattern is projected to the output. The patterns from multiple sources (if they are active) and the leak pattern are combined using noisy-or units to generate the output image. The image patterns and their associated noise components are defined fully by the parameters of the noisy-or model.

We used the above noisy-or model to generate a set of training images. Figure 4 shows examples of 16 convoluted images generated by the model. The learning objective was to estimate and recover the distribution of the original model purely from the observational data – the noise-corrupted convoluted images. In order to assess the characteristics of our variational algorithm we run two sets of experiments, observing the quality of the solution and its running time while varying (1) the number of samples and (2) the number of assumed latent sources.

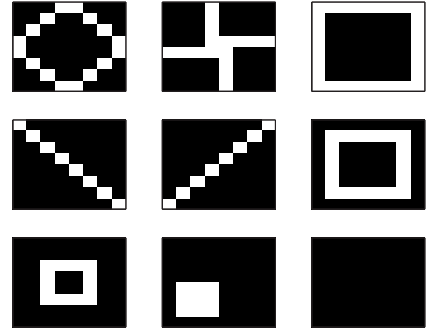


Figure 3: Image patterns associated with hidden sources used in the image deconvolution problem. The ninth (bottom-right) pattern corresponds to the leak.

4.1 Effect of the sample size We used the noisy-or network with 8 hidden sources and image patterns from Figure 3 to generate datasets with 50 - 5000 examples. These samples were then given to the learning algorithm. The learning process always starts from the complete network, no structure relating the sources and observables is given. The new (learned) model was evaluated in terms of: (1) Comparison of learned source images to original images (2) Data

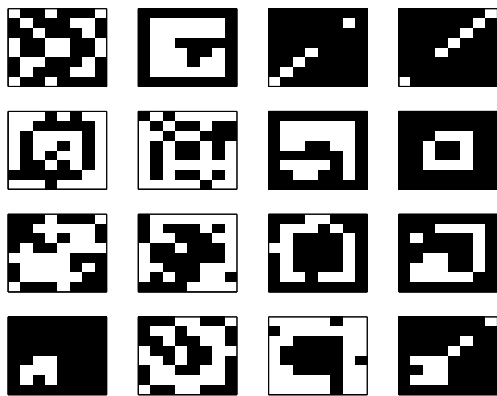


Figure 4: Example images generated by the latent noisy-or model with parameters corresponding to patterns in Figure 3.

reconstruction error.¹

Figure 5 shows the parameters of three noisy-or models recovered by the learning algorithm for varied sample sizes. It is apparent from the figure that larger number of samples lead to progressively improving models that are closer to the original model and approximates its patterns better. The model learned from 50 samples is cumbered with high variance brought about by the low number of training examples, but nevertheless it begins to capture some of the original source patterns. Sample sizes of 500 and 1000 improve the pattern recovery. For 1000 samples we were able to recover almost all sources used to generate data with relatively small distortion. Naturally, inherent stochasticity will cause the sources to differ slightly in each run of the algorithm.

Latent variable models are very useful in dimensionality reduction. Given the learned noisy-or model and an image observed on the output, one can compute the posterior of each hidden source and pick the value (0 or 1) that comes with the higher posterior probability. Hidden sources and their 0/1 values then act as a low-dimensional representation of the data. High-dimensional data can be recovered back by sampling the output according to hidden source values and the pa-

¹Note that it is very difficult to apply standard distance measures for distributions, such as KL-divergence or Hellinger’s distance, to evaluate and compare two high-dimensional multivariate distributions. In our case, it would require to compute and compare probabilities of 2^{64} possible image configurations. Approximate distance measures based on corresponding empirical distributions obtained via sampling suffer from a similar problem: it is extremely difficult to achieve an overlap carrying a significant probability mass between the supports of the respective empirical distributions.

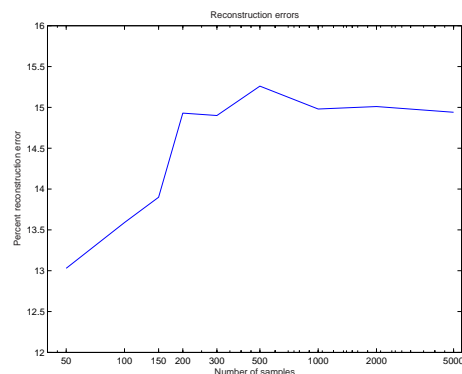


Figure 6: Reconstruction errors obtained from the learning algorithm for varied sample-sizes, averaged over 50 trials.

rameters of the noisy-or model. The difference between the original data point and its reconstruction after the initial reduction defines the reconstruction error. In our case, the reconstruction error is computed as portion of bits in which the original data differs from the reconstructed data.²

Figure 6 illustrates the reconstruction error of the model learned by the variational algorithm for different sample sizes. We clearly see the reconstruction error is smaller for very small sample sizes and stabilizes for sample sizes over 200. This can be explained by overfitting of the model for small sample sizes, and the saturation of the model to its stochastic limit for larger sample sizes.

The running time of the variational algorithm for different sample sizes is shown in Figure 7. The nearly straight line plotted indicates that the complexity of the algorithm grows polynomially with the number of samples. Indeed, we have observed that the time complexity scales approximately linearly with the number of samples. There appears to be no statistically significant effect of sample size on the number of EM iterations the algorithm performs.

4.2 Model selection In real-world data, the correct number of hidden sources is only rarely known in advance. Then the important question is whether the correct number of sources can be determined automatically by the learning algorithm. To analyze this aspect of the problem we run a series of learning experiments on models with different number of latent sources. To assure

²To assess the significance of the learning error, consider that the training sets used contain on average approximately 32% of 1s. Therefore, the trivial majority-class reconstruction baseline would achieve that error.

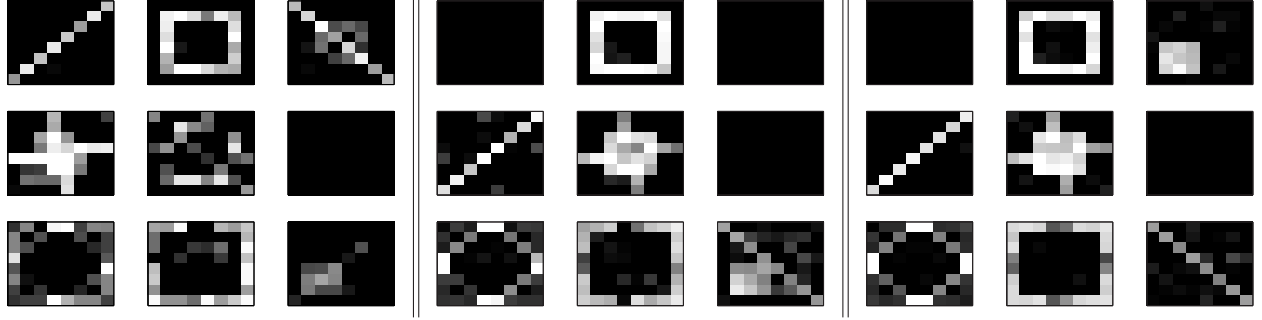


Figure 5: Examples of models learned from 50, 200 and 1000 samples (from left to right). The differences among models illustrate the improvement in the model recovery for increasing sample size. Although some source images are visibly identified with as few as 50 samples, the noise in many images is apparent. Models learned from 200 and 1000 samples are improved. Contrasting 200-sample model to 1000-sample model, a source image stepped out of the leak factor (top row, right column). Additionally, the sources have stabilized, “shadows” were cleaned (compare the source in left column, second row). The only flaw to the 1000-sample model is the source in the center which captured two of the original sources.

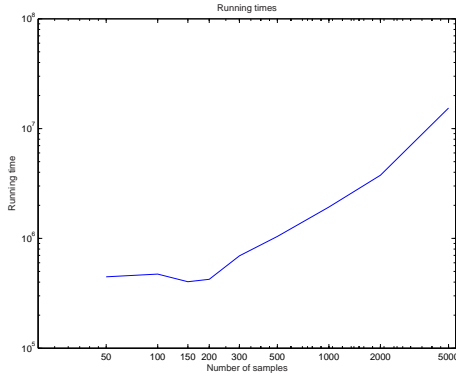


Figure 7: Runtimes of the algorithm, averaged over 50 trials. Considering the wide range of sample sizes tested, we plot the runtimes on a log-log plot.

a fair comparison, the dataset used to train the models was fixed over the course of the experiment.

The results are summarized in Figure 8. The reconstruction error plot demonstrates that as we increase the number of latent sources the learner takes advantage of all sources available to it at 6 sources or fewer, then starts to plateau at 8 sources. This agrees well with the number of latent sources used to generate the data.

To assess the recovery performance, we looked at patterns learned by the algorithm, much like those in Figure 5 and counted the number of identified sources. The inspection of the learned models showed that the number recovered sources levels out at around 7, other sources were shut down via regularization

effects (Section 3.5). Taking into account the existence of the leak node (which effectively adds one source), this matches or is very close to the true number of sources. Taking advantage of these phenomena one does not have to identify the number of hidden sources in advance, the algorithm finds a reasonable estimate of the correct number on its own at only minor additional computational cost.

The analysis of running times for different number of sources in Figure 8 shows that the runtimes scale roughly linearly with the number of assumed latent sources. This gives an empirical support for the efficiency of variational EM approximation as compared to the exponential complexity of the exact EM algorithm with respect to the number of sources.

5 Noisy-or component analysis of citation data

To show the benefit of our model in a real-world application we applied the model to perform *component analysis* of a citation dataset derived from online publications in the area of Machine learning. The dataset was built from approximately 17,000 hypertext documents from the CiteSeer service. We chose 40 prominent authors in the field of Statistical Machine Learning, to limit ourselves to a domain where we can confidently assess the soundness of the obtained results. The data were then processed into a binary matrix. This matrix contains 1 at position (i, j) if the document i cites author j .

The noisy-or model fits well the structure this dataset exhibits. A contemporary paper in this field is likely to touch upon several topics and combine or improve on them. We would expect the hidden factors to roughly match the paper keywords, each topic factor

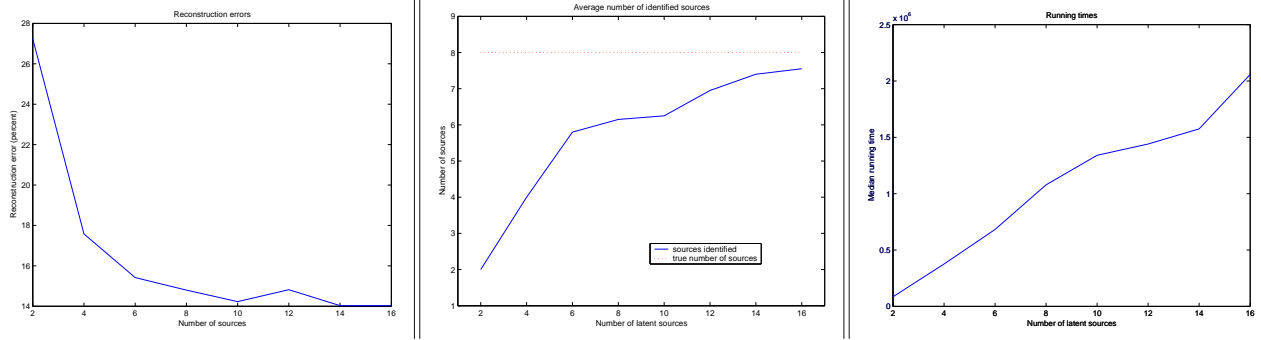


Figure 8: Average reconstruction error (left panel), average number of identified sources (middle) and median running times (right) plotted against the number of assumed latent sources. The red dotted line in the middle plot represents the true number of sources (8). The statistics in this figure were obtained from 25 experimental runs.



Figure 9: The result of the noisy-OR component analysis on the citation dataset. The columns visualize the parameters of the noisy-or loading matrix after they are rescaled by the prior of the source. Black fields correspond to 0s in the loading matrix, while white would correspond to 1s.

having its seminal papers whose authors thereby become likely to be cited.

We ran our noisy-or model on the CiteSeer dataset using 5 hidden sources. Figure 9 illustrates the outcome of the analysis. The obtained results indicate the presence of the following components:

- The authors dominating the first component are: J. Pearl, M. Jordan, S. Lauritzen and D. Spiegelhalter. Weaker ties are to W. Buntine, N. Friedman and D. Koller. This component discovered many respected authors of basic references and tutorials on Bayesian belief networks.
- The second source was shut down as the algorithm did not reveal any other interesting group in this run.
- C. Burges, B. Schölkopf, A. Smola and V. Vapnik form the core of the third component. Without any doubt, this component represents the kernel and SVM research community.
- The authors prominent in the fourth factor are Z. Ghahramani, M. Jordan, G. Hinton, R. Neal, L. Saul, C. Bishop and M. Tipping. This source captures the variational approximation community.
- The last component consists of the following authors: B. Frey, W. Freeman, K. Murphy, S. Lauritzen, J. Pearl, Y. Weiss and J. Yedidia. All authors published extensively on loopy belief propagation, using J. Pearl's BP algorithm. The presence of an outlier in this set, S. Lauritzen, can be attributed to the fact that he is among the most frequently cited authors in the general context of Bayesian networks. Conclusively, we can say our algorithm found the LBP community.

The results obtained for the citation data show the potential benefit of the noisy-or model and its ability to uncover semantically sound component structure in the binary data. We note there is a conceptual difference between the noisy-or model and mixture models, such as the aspect model [10], used frequently in the analysis of documents. The key difference is that the aspect model assigns each document a convex combination of topic factors, while our model computes a vector of binary indicators, each corresponding to one topic. Each model stresses a different type of the structure and both analyses can complement each other to improve the understanding of the data at hand.

6 Conclusions

We have devised and presented an EM-based variational algorithm for learning latent factor models with noisy-or units. The algorithm alleviates the key limitation of exact learning algorithms – their exponential dependency on the number hidden factors. The proposed variational algorithm makes no assumption about the structure of the underlying noisy-or network, the structure is fully recovered during the learning process.

We tested the algorithm on two problems: (1) image deconvolution problem and (2) analysis of citation data. The algorithm showed a good scale-up potential with a very good model recovery and error reconstruction performances on the image problem. On citation data it successfully discovered components that represent established communities. We demonstrated how the noisy-or latent variable model offers itself as a tool of inquiry of social networks and internet communities.

An in-depth comparison of the noisy-or component analyzer to alternative component analysis frameworks, most importantly Probabilistic Latent Semantic Analysis, remains an interesting open problem and a focus of our continued research interest.

7 Acknowledgements

This research was supported in part by the Research Development Fund Award 36851 from the University of Pittsburgh and by National Science Foundation grants CMS-0416754 and ITR-0325353.

References

- [1] Hagai Attias. Independent Factor Analysis. *Neural Computation*, 11(4):803–851, 1999.
- [2] Christopher M. Bishop. Latent variable models. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 371–403. MIT Press, 1999.
- [3] Christopher M. Bishop. Variational principal components. In *Proceedings of Ninth International Conference on Artificial Neural Networks*, volume 1, pages 509–514. ICANN, 1999.
- [4] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, JAN 2003.
- [5] W. Buntine. Variational extensions to EM and multinomial PCA. In *ECML 2002*, 2002.
- [6] Heckerman David. Causal independence for knowledge acquisition and inference. In *Proc. of 9th Conf. on UAI93*, San Francisco, CA, 1993. Morgan Kaufmann Publishers.
- [7] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood for incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–38, 1977.
- [8] Francisco Diez and Severino Gallan. Efficient computation for the noisy max. *International Journal of Intelligent Systems*, 2003.
- [9] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proceedings of Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995.
- [10] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.
- [11] Tommi Jaakkola and Michael I. Jordan. Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, 10:291–322, 1999.
- [12] Michael Kearns and Yishay Mansour. Exact inference of hidden structure from sample data in noisy-OR networks. In *Proc. 14th Conf. on UAI98*, pages 304–310, 1998.
- [13] Xinghua Lu, Milos Hauskrecht, and Roger S. Day. Modeling cellular processes with variational bayesian cooperative vector quantizer. In *Pacific Symposium on Biocomputing (PSB)*, page to appear, 2004.
- [14] David MacKay. Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.
- [15] James W. Miskin. *Ensemble Learning for Independent Component Analysis*. PhD thesis, Selwyn College, University of Cambridge, 2000.
- [16] D. Ross and R. Zemel. Multiple cause vector quantization. In *Advances in Neural Information Processing Systems*, 2002.
- [17] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255, 1991.
- [18] Michael Tipping and Christopher Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group,

Aston University, September 1997.

- [19] Jiří Vomlel. Noisy-or classifier. In *Proceedings of the 6th Workshop on Uncertainty Processing (WUPES 2003)*, pages 291–302, September 2003.

Summarizing Sequential Data with Closed Partial Orders *

Gemma Casas-Garriga[†]

Abstract

In this paper we address the task of summarizing a set of input sequences by means of local ordering relationships on items occurring in the sequences. Our goal is not mining these structures directly from the data, but going beyond the idea of closed sequential patterns and generalize it into a novel notion of closed partial order. We will show that just a simple (but not trivial) post-processing of the closed sequences found in the data leads to a compact set of informative closed partial orders. We analyze our proposal not only algorithmically but also theoretically, by showing the connection with Galois lattices. Finally, we illustrate the approach by applying it to real data.

General Terms. Closed partial orders, sequence analysis, post-processing closed sequential patterns.

1 Introduction

Mining sequences of events is an important data mining task with broad applications in business, web mining, computer intrusion detection, DNA sequence analysis and so on. The problem was first introduced in [1] as a problem of mining frequent sequential patterns in a set of sequences, and since then, it has been extensively studied (e.g., algorithms like SPADE [19] or PrefixSpan [13] among others). Unfortunately, one problem of this sequential pattern mining task arises when considering a very low support in the algorithms or when mining very long sequences; in these cases, the number of frequent patterns is usually too large for a thorough examination and the algorithms face several computational problems. A proper solution to this problem is recently proposed in some papers, such as [15, 16, 17], and it consists on mining just a compact and more significative set of patterns called the *closed sequential patterns* (or closed sequences). These closed sequential patterns are defined to be “stable” in terms of support, that is, they are maximal sequences among those others having the same support in the database.

The idea of mining just closed sequential patterns instead of all frequent patterns stems from the parallel case of mining closed itemsets in a binary database ([12, 18]). The foundations of closed itemsets are based on the mathematical model of concept lattices ([7, 8]):

a closure operator is defined by using the properties of the Galois connection, and from there, one can draw a lattice of formal concepts. Then, it can be proven that the set of closed itemsets is necessary and sufficient to capture all the information about frequent itemsets and association rules in the unordered context. Moving to the sequential case again, a recent work in [4] proves that the set of closed sequential patterns mined by existing algorithms [15, 16, 17] can be formalized in terms of a closure operator as well.

In general, dealing with closed patterns is currently an interesting topic in data mining since it provides a more compact set of patterns. However, we consider that there are still some criticisms to be done about the closed sequences: mainly, the number of those patterns can be still quite large due to the combinatorial nature of the problem, and it is not clear how they can be useful to the final user once we have mined them.

1.1 Goals of this Work In this paper we propose a way to handle these resulting closed sequences so that they provide useful information of our data. We are not focusing here on algorithmic solutions for finding closed sequential patterns, and we rely on current proposals such as TSP [15], BIDE [16] or CloSpan [17]; our intention is not contributing to the efficiency of existing algorithms, but to the post-processing of closed sequences once we have mined them. Our goal is to outcome with a new notion of partial orders that can be obtained out of the closed sequences, in such a way that (1) it advances in the summarization of sequential data; (2) it has a sound theory supporting it; and (3) it can be implemented with efficient algorithms without accessing the input data, just the set of closed sequences. Finally, we will show that these partial orders represent indeed the closure of hybrid episodes introduced in [11], and they can be seen also as complementary to other works of mining episodes.

1.2 Paper Overview The rest of the paper is organized as follows. In section 2 we present some basic definitions of the frequent closed sequence mining. Section 3 motivates our intention of going beyond closed sequences and defines our post-processing approach for generating partial orders out of the set of closed se-

*Supported by MCYT TIC 2002-04019-C03-01 (MOISES)

[†]Universitat Politècnica de Catalunya, Barcelona, Spain

quences. Sections 4 and 5 develop algorithmically and theoretically the two steps of the proposal; section 6 discusses other algorithmic issues to complete the current proposal. Finally, section 7 evaluates this work with experiments, and section 8 discusses the relation of our final partial orders with the hybrid episodes of [11]. In section 9 we conclude the study.

2 Preliminaries

Let $\mathcal{I} = \{i_1, \dots, i_n\}$ be a finite set of items. In the classical formalization given by [1], *sequences* are ordered lists of itemsets. The input data we are considering consists of a database of ordered transactions $\mathcal{D} = \{t_1, t_2, \dots, t_n\}$ that we model as a set of sequences, i.e. each transaction t_i is in fact a sequence, also called *input sequence*. Our notation for the component itemsets of a given sequence will be $s = \langle (I_1)(I_2) \dots (I_n) \rangle$, where each $I_i \subseteq \mathcal{I}$ and I_i occurs before itemset I_j if $i < j$. Note that each I_i may contain several items that occur simultaneously; e.g. $\langle (AC)(B) \rangle$, meaning that items A and C come at the same time but always before item B . The universe of all sequences is noted with \mathcal{S} . An example of such data \mathcal{D} is presented in figure 1, where each itemset I_i contains only one single item. We choose this simplification to make this first example more clear, and because we consider that these single-item sequences model popular types of data such as DNA, Web click streams, command histories of Unix users and so on. However, our proposals will work also when having sequences of subsets of items, and in section 6 of this paper we will follow up an example with simultaneity.

Seq id	Input sequences
t_1	$\langle (C)(B)(C)(A)(C) \rangle$
t_2	$\langle (C)(B)(A)(C)(C)(C)(A) \rangle$
t_3	$\langle (A)(C)(A)(C)(C)(A)(A)(A) \rangle$
t_4	$\langle (C)(A)(C) \rangle$

Figure 1: Collection of data \mathcal{D}

Some basic operations on sequences can be defined in a general way as follows. Sequence $s = \langle (I_1) \dots (I_n) \rangle$ is a *subsequence* of another sequence $s' = \langle (I'_1) \dots (I'_m) \rangle$ if there exist integers $j_1 < j_2 < \dots < j_n$ such that $I_1 \subseteq I'_{j_1}, \dots, I_n \subseteq I'_{j_n}$. We note it by $s \subseteq s'$. For example, $s = \langle (C)(B)(A) \rangle$ is a subsequence of the first and second input sequences from figure 1, $s \subseteq t_1$ and $s \subseteq t_2$. For later formalization purposes it will be necessary to keep track of the identifiers of those input sequences where a certain s is contained, named its tid list, $tid(s)$; e.g., $tid(\langle (C)(B)(A) \rangle) = \{t_1, t_2\} = \{1, 2\}$ in figure 1. Then, the *support* of a sequence s is the number of input sequences where s is contained, $support(s) = |tid(s)|$; e.g., the support of $\langle (C)(B)(A) \rangle$ is 2.

The *intersection* of a set of sequences $s_1, \dots, s_n \in \mathcal{S}$ is the set of *maximal* subsequences contained in all the s_i . Note that the intersection of a set of sequences, or even the intersection of two sequences, is not necessarily a single sequence. For example, the intersection of the two sequences $s = \langle (A)(C)(B) \rangle$ and $s' = \langle (A)(B)(C) \rangle$ is the set of sequences $\{\langle (A)(C) \rangle, \langle (A)(B) \rangle\}$: both are contained in s and s' , and among those having this property they are maximal; all other common subsequences are not maximal since they can be extended to one of these. The maximality condition discards redundant information since the presence of, e.g., $\langle (A)(B) \rangle$ in the intersection already informs of the presence of each of the itemsets (A) and (B) .

The *head* of a sequence $s = \langle (I_1) \dots (I_n) \rangle$ up to a position j s.t. $1 \leq j \leq n$, is noted by $head(s, j) = \langle (I_1) \dots (I_j) \rangle$. Similarly, the *tail* of a sequence from position j s.t. $1 \leq j \leq n$, is noted by $tail(s, j) = \langle (I_j) \dots (I_n) \rangle$. The *concatenation* of two sequences will be noted by $s \diamond s'$.

2.1 Mining Closed Sequential Patterns Typically, associated to this discrete sequential data there is the problem of mining frequent sequences, that is, those subsequences in \mathcal{D} whose support is over a user-specified threshold. Unfortunately, the performance of the algorithm degrades when using a very low support or having a dense database. Some recent works, such as [15, 16, 17], propose to mine frequent closed sequences instead.

A sequence s is *closed* in input data \mathcal{D} if s is maximal in the set of transactions where it is contained, that is, it cannot be extended. More formally:

DEFINITION 2.1. (CLOSED SEQUENCE) A sequence $s \in \mathcal{S}$ is *closed* for data \mathcal{D} if there exists no sequence s' with $s \subset s'$ s.t. $support(s) = support(s')$.

For instance, taking data from figure 1, sequence $\langle (A)(C) \rangle$ is not closed since it can be extended to $\langle (C)(A)(C) \rangle$ in all the input sequences where it belongs. However, $\langle (C)(B)(A)(C) \rangle$ or $\langle (C)(C)(C) \rangle$ are closed sequences in \mathcal{D} . In other words, closed sequences are “stable” in terms of support since they are maximal among those having the same support. The set of all closed sequential patterns of data from figure 1 is shown in figure 2. Usually, a minimum support condition is provided by the user to mine only those closed sequences up to the threshold; here, given that data in figure 1 is small enough, we will suppose that the threshold is set to zero for this ongoing example.

Tid list	Closed Sequential Patterns
{1, 2, 3, 4}	$\langle\langle C \rangle(A)(C)\rangle$
{1, 2, 3}	$\langle\langle C \rangle(C)(A)\rangle$
{1, 2, 3}	$\langle\langle C \rangle(C)(C)\rangle$
{2, 3}	$\langle\langle A \rangle(C)(C)(C)(A)\rangle$
{2, 3}	$\langle\langle C \rangle(A)(C)(C)(A)\rangle$
{1, 2}	$\langle\langle C \rangle(B)(A)(C)\rangle$
{1, 2}	$\langle\langle C \rangle(B)(C)(A)\rangle$
{1, 2}	$\langle\langle C \rangle(B)(C)(C)\rangle$
{1}	$\langle\langle C \rangle(B)(C)(A)(C)\rangle$
{2}	$\langle\langle C \rangle(B)(A)(C)(C)(C)(A)\rangle$
{3}	$\langle\langle A \rangle(C)(A)(C)(C)(A)(A)(A)\rangle$

Figure 2: Set of all closed sequences and their tid lists

3 Discussion and Motivation

Frequent closed sequential patterns represent the most informative total orders in \mathcal{D} with respect to support. So, apart from reducing the algorithmic overhead, this final set of closed patterns is useful in many ways: (1) the user needs to examine fewer patterns obtained as an output of the mining algorithms; (2) hitting with the right minimum support threshold is not so important; for example, mining all the subsequences in \mathcal{D} with a threshold close to zero is unrealistic and it does not provide useful information about the data, but the set of all closed sequences is not so dramatic and still gives an overall idea of the whole database; and (3) closed patterns have a sound theoretical background based on formal concept analysis (see [4]), which provides several important properties and formalizations.

Therefore, the set of closed sequences provides a more compact set of patterns that keeps the same information as frequent sequences. However, many questions arise: what do these sequences say about our data \mathcal{D} ? what to do with these closed sequential patterns once we have mined them? how to go beyond closed sequential patterns?

We consider that the set of closed sequential patterns does not represent all the particularities hidden in the sequential data. Formally, it can exist two closed sequences s and s' such that they occur in the same transactions, so that $\text{support}(s) = \text{support}(s')$, but $s \not\subseteq s'$ and $s' \not\subseteq s$. In other words, contrary to the case of closed itemsets in binary data ([12, 18]), here there is no unique representative closed pattern for a given set of ordered transactions. By way of example, the closed sequences $\langle\langle A \rangle(C)(C)(C)(A)\rangle$ and $\langle\langle C \rangle(A)(C)(C)(A)\rangle$ from figure 2 occur exactly in the same transactions but none of them can be considered “better” than the other, they simply *coexist* together. This fact cannot be captured by the unidimensional representation of the set of closed sequences.

We want to go beyond the notion of closed sequen-

tial patterns. Our goal is to generalize this idea of compacting the information as much as possible so that we can produce not just fewer patterns, but also more informative ones. Next, we will show how these closed sequences coexisting together lead to a novel notion of closed partial order that summarize the data in a compact way. We formalize our proposal:

1. First, grouping closed sequential patterns occurring together in a maximal set of transactions. We will see that this task has not a direct solution since some closed sequences can coexist in several groups and also, we want to make these groups nonredundant and maximal.
2. Second, constructing a new notion of closed partial orders out of those groups, without the need of accessing again the data \mathcal{D} .

We will see that this process is supported by the mathematical theory of formal concept analysis since the set of maximal sequences in a maximal set of transactions is a closure system. This ensures the good properties of the obtained results. In next two sections, we describe with detail the goals of our approach and we provide efficient algorithms to implement each one of the steps.

4 Grouping Closed Sequential Patterns

The first step of our proposal is to make nonredundant groups of sequences coexisting together in the same maximal set of transactions. Formally, we state this problem as: given the set of frequent closed sequential patterns $CS = \{s_1, s_2, \dots, s_n\}$ mined by any of the existing algorithms, we want to output a list of *valid* pairs (S, T) .

DEFINITION 4.1. A *valid pair* (S, T) is one where: $S \subseteq CS$ is a nonredundant set of closed sequences, whose tid lists are at least T ; and $T \subseteq \mathcal{D}$ is the maximal set of transactions where all $s \in S$ are contained.

We say that a set S of closed sequences is *nonredundant* when for all $s, s' \in S$ s.t. $s \neq s'$ we have that $s \not\subseteq s'$ and $s' \not\subseteq s$. By computing the list of valid pairs (S, T) from CS , we get nonredundant groups S of closed sequences where all $s \in S$ is contained in all $t \in T$, and there is no other set of transactions T' with $T \subset T'$ where sequences in S still coexist together. An obvious observation then is that the set of transactions T of a valid pair (S, T) will correspond to the maximal tid list of one of the sequences $s \in S$, that is, $T = \max\{\text{tid}(s) | s \in S\}$. Otherwise T would not be maximal as we want, since all the sequences in S must have tid lists at least T by definition.

E.g., sequences $\langle(A)(C)(C)(C)(A)\rangle$ and $\langle(C)(A)(C)(C)(A)\rangle$ from figure 2, will be fitted into the same set S since they appear exactly in the same input sequences $T = \{2, 3\}$. Any other closed sequence whose tid list is at least T (i.e. coexisting still in transactions T) would not fit in S , since it would turn it redundant. Moreover, the set $T = \{2, 3\}$ is a maximal set of transactions for this S as well, that is, for any other larger set of transactions we have that the sequences in S coexist together. A complete example of the desired groups for the closed sequences in 2, is shown in figure 3. Note that there is no valid pair involving the set of transactions $T = \{1, 3\}$: this T is not maximal for the set of closed sequences whose tid list is at least $T = \{1, 3\}$ (and no closed sequential pattern has a tid list coinciding with this T).

T	S
$\{1, 2, 3, 4\}$	$\{\langle(C)(A)(C)\rangle\}$
$\{1, 2, 3\}$	$\{\langle(C)(A)(C)\rangle, \langle(C)(C)(A)\rangle, \langle(C)(C)(C)\rangle\}$
$\{2, 3\}$	$\{\langle(A)(C)(C)(C)(A)\rangle, \langle(C)(A)(C)(C)(A)\rangle\}$
$\{1, 2\}$	$\{\langle(C)(B)(A)(C)\rangle, \langle(C)(B)(C)(A)\rangle, \langle(C)(B)(C)(C)\rangle\}$
$\{1\}$	$\{\langle(C)(B)(C)(A)(C)\rangle\}$
$\{2\}$	$\{\langle(C)(B)(A)(C)(C)(C)(A)\rangle\}$
$\{3\}$	$\{\langle(A)(C)(A)(C)(C)(A)(A)\rangle\}$

Figure 3: Groups of closed sequences occurring together

Obviously, an initial algorithmic naive approach is to group all those closed sequences in CS with the same tid list, that is, create each pair (S, T) so that all $s \in S$ has the same tid list and $T = \text{tid}(s)$. In this way we ensure that the set T will be maximal. However, the set S created in the naive way may be incomplete, with some missing element. The following property formalizes this idea.

PROPOSITION 4.1. *Let (S, T) be a valid pair, then we have that $S = \bigcap_{t \in T} t$.*

Proof. If T is a set of transactions from a given valid pair, then the intersection of transactions $t \in T$ returns a set of sequences S s.t. for all $s \in S$ we have that $\text{tid}(s)$ is at least T . Moreover, all sequences $s \in S$ will be closed and nonredundant by definition of the intersection operation, that always keeps only maximal subsequences.

Proposition 4.1 simply states that the nonredundant group of closed sequences coexisting in a set of transactions T must necessarily coincide with the intersection of those transactions. Note that the reverse implication of the proposition does not hold, because we cannot ensure the maximality of the set of transactions T . Then, a naive grouping such as $S_1 = \{\langle(A)(C)(C)(C)(A)\rangle, \langle(C)(A)(C)(C)(A)\rangle\}$ and

$T = \{2, 3\}$ of our example, is valid because it coincides with the intersection of second and third input sequences of \mathcal{D} . However, another naive group such as $\{\langle(C)(C)(A)\rangle, \langle(C)(C)(C)\rangle\}$ and $T = \{1, 2, 3\}$, does not form a valid pair because the intersection of first, second and third input sequences contains one more closed sequence; the proper family of sequences of this valid pair should be $S_2 = \{\langle(C)(A)(C)\rangle, \langle(C)(C)(A)\rangle, \langle(C)(C)(C)\rangle\}$. Indeed, the closed sequence $\langle(C)(A)(C)\rangle$ is contained in all the transactions of the database; then, it always occurs simultaneously with any other sequence, and it should be included in any S as long as it does not make it redundant. Because of this, $\langle(C)(A)(C)\rangle$ belongs to S_2 as an element, but not to S_1 , where it is already included as a subsequence of $\langle(C)(A)(C)(C)(A)\rangle$.

Of course, we want to output the valid list of pairs (S, T) without directly intersecting the transactions of the database, just by grouping the sequences in CS . But there can be closed sequences that belong to several sets S as long as S is not redundant. The general property that follows from this reasoning is expressed by the following lemma.

LEMMA 4.1. *Given two valid pairs (S', T') and (S, T) , if $T \subseteq T'$ then for all $s' \in S'$ there exists $s \in S$ s.t. $s' \subseteq s$.*

Proof. For all $s' \in S'$ we have that s' is subsequence of t' , for all $t' \in T'$ (proposition 4.1); in particular, if $T \subseteq T'$, then we have that s' is subsequence of t for all $t \in T$, and thus there exists $s \in S$ s.t. $s' \subseteq s$.

4.1 Algorithmic Analysis Conceptually, lemma 4.1 provides a way to organize the list of valid pairs into a graph where each node is a pair (S, T) , and there is an edge between each node and its predecessors. The predecessors of a node (S, T) are those pairs (S', T') s.t. $T \subseteq T'$. The property stated in lemma 4.1 must hold between each node and their predecessors.

In figure 4 we graphically depict the mentioned conceptual graph for the valid pairs of figure 3. Each node represents a valid pair (S, T) , and here we depict only those edges connecting immediate predecessors: (S', T') is an immediate predecessor of (S, T) if $T \subseteq T'$ and there is no other (S'', T'') s.t. $T \subset T''$ and $T'' \subset T'$. E.g. the node $\{\langle(C)(B)(C)(A)(C)\rangle\}$ has one immediate predecessor, which is $\{\langle(C)(B)(A)(C)\rangle, \langle(C)(B)(C)(A)\rangle, \langle(C)(B)(C)(C)\rangle\}$. We will note immediate predecessors by relation \preceq : $(S', T') \preceq (S, T)$. Notice that if lemma 4.1 holds for immediate predecessors of a given node, then it will hold for the rest of predecessors as well (see figure 4).

This conceptual graph will be used as a tool to

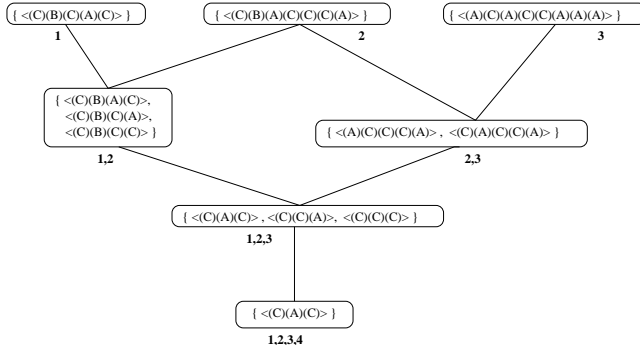


Figure 4: Conceptual graph of pairs (S, T)

analyze our algorithmic proposal: it simplifies the pseudocode and it eases the complexity analysis. However, the final implementation does not necessarily need to recreate such structure in memory, and a wise use of lists and indexes turns out to be enough to come up with the right groupings. The pseudocode to obtain valid pairs is presented in algorithm 1.

Algorithm 1 Grouping Closed Sequential Patterns

Input: List CS of closed sequential Patterns

Output: List of pairs (S, T)

```

1: Sort  $CS$  in descending order by tid list;
2: while  $CS \neq \emptyset$  do
3:    $S \leftarrow$  Next sequences  $s \in CS$  with same tid list;
4:    $T \leftarrow \text{tid}(s)$ , for some  $s \in S$ ;
5:   for each  $(S', T') \preceq (S, T)$  do
6:     for each  $s' \in S'$  do
7:       if  $s' \not\subseteq s, \forall s \in S$  then  $S \leftarrow S \cup \{s'\}$  end if
8:       output  $(S, T)$ ;
9:     end for
10:  end for
11: end while

```

The idea of algorithm 1 is simple: lines 3-4 perform naively by getting groups of closed sequences in CS with same tid list. Then, lines 5-11 complete the set S with sequences already belonging to immediate predecessors that should be also in S . Considering that CS is ordered in descending order by the tid list of its elements, then the algorithm traverses the conceptual graph bottom-up in a breadth-first fashion.

Notice again the fact that it is only necessary to look for immediate predecessors of a set S to make lemma 4.1 hold. The algorithmic proposal is bounded by $O(n \cdot m \cdot k^2)$, where we consider n to be the number of closed sequences in CS , m is the maximum number

of immediate predecessors for a node (S, T) , and k is the maximum number of closed sequences belonging to immediate predecessors of S .

4.2 Theoretical Analysis An important theoretical consideration to be done at this point is that the set of closed sequential patterns in S of valid pairs (S, T) can be formalized in terms of a closure operator named Δ , presented in [4]. Basically, a closure operator of any fixed universe is one that satisfies the three basic closure axioms: monotonicity, extensivity and idempotency. The formal operator Δ developed in [4] works with *sets of sequences*.

Broadly speaking, this resulting Δ stems from the composition of two derivation operators forming a Galois connection and it works as follows: given \mathcal{D} , the closure $\Delta(S)$ of a set of sequences S , includes all the maximal sequences that are present in all transactions having all sequences in S ; for example, in data from figure 1 we have $\Delta(\{\langle(A)(C)(C)\rangle\}) = \{\langle(A)(C)(C)(C)(A)\rangle, \langle(C)(A)(C)(C)(A)\rangle\}$ because all the maximal subsequences contained in those transactions where $\{\langle(A)(C)(C)\rangle\}$ belongs are $\langle(A)(C)(C)(C)(A)\rangle$ and $\langle(C)(A)(C)(C)(A)\rangle$. Then, we say that closed sets of sequences are those coinciding with their closure, that is, $\Delta(S) = S$.

Given proposition 4.1 and lemma 4.1, it is possible to prove that the set of closed sequences S from each pair valid (S, T) indeed fulfills $\Delta(S) = S$. So, sets S from valid pairs are also a closed set of sequences according to Δ . This characterization carries several consequences. The first important consequence is that the conceptual graph described in subsection 4.1 can be considered a Galois lattice ([7, 8]) under these certain conditions: (1) when nodes of the graph are closed under intersection; and (2) there exists a supremum and infimum for any pair of elements. Broadly, first condition means that when intersecting two sets of sequences S_1 and S_2 of two different nodes, we get another set of sequences, $S_3 = S_1 \cap S_2$, belonging to another node of the same graph. That is, the intersection of two sets of sequences from valid pairs returns another set of sequences from another valid pair. Here, the intersection of two sets of sequences $S_1 \cap S_2$ is defined as the cross intersection of all $s_1 \in S_1$ with all $s_2 \in S_2$. Similarly, the set of transactions T of valid pairs must be also close under intersection to be considered a Galois lattice. For example, the graph depicted in figure 4 is closed under intersection, so, if we added an artificial top connected to the upper vertices of the graph, then we would get a Galois lattice. The conceptual graph of valid pairs may not be always closed under intersection, then, the missing nodes must be

added to form a closure system. More details can be followed from [4].

The second consequence of having this characterization in terms of Δ , is that it is possible to derive a notion of deterministic association rules from valid pairs (S, T) and the generators of S . We say that a set S' is a generator of S when $\Delta(S') = S$ and $S \neq S'$. Then, the theory of associations for this ordered context can be completely formulated by considering implications of the form $S' \rightarrow S$. Actually, a recent work in [3] shows that these implications axiomatize the Horn theory for ordered data.

5 Obtaining Closed Partial Orders

The second step of our proposal is to obtain a compact representation from each valid pair (S, T) . Our starting point is to realize that each $s \in S$ is in fact a total order compatible with all transactions $t \in T$. Since sequences in S coexist together in T , it should be possible to derive a partial order describing the set of transactions T by properly combining all $s \in S$. To get an initial intuition of this idea we show in figure 5 the desired transformation that will turn the sets S from figure 3 into partial orders; e.g., the set $S = \{\langle(A)(C)(C)(C)(A)\rangle, \langle(C)(A)(C)(C)(A)\rangle\}$ is converted into a partial order compatible with transactions $T = \{2, 3\}$. Basically, we are creating a partial order out of S , without adding new restrictions among the different items and still respecting all the restrictions given by each $s \in S$. We may think that this is an easy task, but the way of overlapping the positions of a set of sequences S to get a partial order still compatible with a set of transactions T is not direct, specially when having repeated items.

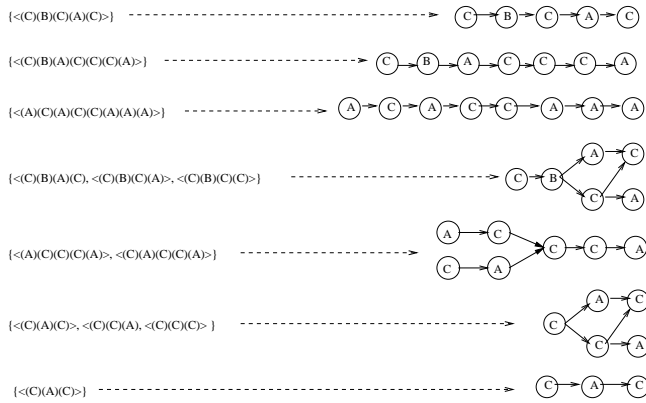


Figure 5: Partial orders from (S, T)

To make this goal more formal we introduce some basic definitions. A *partial order* can be modelled as

a triple $p = (V, E, l)$ where V is the set of vertices; $E \subseteq V \times V$ is the set of edges such that the relation on V established by edges in E is reflexive, antisymmetric and transitive; and l is the labelling function mapping each vertex into a set of items, i.e. $l : V \rightarrow 2^I$. In our proposed example of figure 5, this labelling function simply maps each vertex to a single item, but indeed a node can be labelled with a set of items when considering sequences of itemsets. The *transitive reduction* of $p = (V, E, l)$ is the smallest relation resulting from deleting those edges in E that come from transitivity. Partial orders will be graphically depicted here by means of its transitive reduction to make them more understandable, but of course, all edges of the transitive closure are present in E . The graphical representation of partial orders is particularly useful for displaying results: we display a poset by using arrows between the connected labelled vertices, and the symbol \parallel (parallel) to indicate trivial order among the different components of a partial order.

We say that a partial order $p = (V, E, l)$ is *compatible* with an (input) sequence s if: $\forall u \in V$ we have that $l(u)$ is in s ; and, $\forall (u, v) \in E$ we have that $\langle l(u)l(v) \rangle \subseteq s$. The support of a partial order is the number of input transactions that is compatible with.

Given (S, T) our goal is to generate a partial order $p = (V, E, l)$ s.t. for all $s' \subset s \in S$, we have that s' is included in p ; and p is still compatible with all transactions in T . The first condition forces the partial order p to respect all the restrictions given by each $s \in S$, and the second condition ensures that no extra edges will be added between vertices (next lemma 5.1 will formalize this idea). This partial order will summarize the set of transactions T in the most specific way. In subsection 5.1 we will provide more clear formalizations about this notion. We want the partial order generated from a pair (S, T) to be exactly compatible with transactions in T .

One way to get such structure out of (S, T) is by considering a partial order whose *maximal paths* are *exactly* defined by sequences $s \in S$. We define a path from a partial order $p = (V, E, l)$ as a sequence $\langle (I_1) \dots (I_n) \rangle$ such that there is an equivalent list of different nodes u_1, \dots, u_n in V that $(u_j, u_{j+1}) \in E$ and $l(u_j) = I_j$ and $l(u_{j+1}) = I_{j+1}$. E.g. in figure 5, groups of sequences S define the maximal paths of the new partial orders, and the maximal paths of these partial orders coincide exactly with the sequences in the set S . Moreover, if we considered different paths not included in S , then p would not be compatible with all transactions in T , as it shows the following lemma.

LEMMA 5.1. *Given a valid pair (S, T) and a partial order p , we have that:*

- (1) if p has maximal paths exactly defined by sequences in S , then p is only compatible with transactions in T .
- (2) if p has any paths not included in S , then p is not compatible with transactions T .

Proof. Proving part (1) is easy: if p has maximal paths defined by sequences in S of a valid pair (S, T) , then p will be compatible with transactions in T , since all $s \in S$ is a subsequence of all $t \in T$ (prop. 4.1). Moreover, p cannot be compatible with more transactions not considered in T , because by definition of valid pair we have that T is a maximal set of transactions for S . So, it does not exist an larger set of transactions T' where sequences in S are all of them included at a time.

Part (2) of the lemma can be proved as follows. If there exists a sequence s' representing path of p s.t. $s' \notin S$, then we can rewrite it as $s' \notin \bigcap t, \forall t \in T$ (prop. 4.1). This implies that s' is not a subsequence of some $t \in T$, and so, the considered p cannot be compatible with all transactions T , as stated by the lemma.

According to lemma 5.1, given (S, T) it is not possible to create a partial order which is still compatible with T and includes more paths not considered in S . Following this idea, we will generate partial orders whose maximal paths are exactly defined by sequences in S . Then, we have that (1) p will be compatible with all transactions $t \in T$, since each $s \in S$ appears in all $t \in T$; (2) p is not having new edges representing restrictions not considered in S ; and (3) p obviously respects each $s \in S$, since for all $s' \subset s$ we have that s' will be included in p . In subsection 5.2 we will detail theoretical results making this partial order closed for data \mathcal{D} .

5.1 Algorithmic Analysis The point is now how to match positions of sequences in S so that they form the maximal paths of a partial order. This is not a direct task since sequences can be overlapped in several ways so that the resulting partial order still has maximal paths in S . For example, from the set of closed sequences $S = \{\langle(A)(C)(C)(C)(A)\rangle, \langle(C)(A)(C)(C)(A)\rangle\}$ there are several partial orders whose maximal paths are exactly sequences in S : starting from the fully independent parallelization of both sequences, or just matching the last item (A) of both sequences, or just matching the two last items $(C)(A)$ of both sequences, and so on; but we are interested only in the partial order generated as in figure 6, which is matching the last three positions $(C)(C)(A)$ from both sequences.

In other words, from all the partial orders whose maximal paths are exactly sequences in S , we are interested in the most specific one. We define that



Figure 6: Transformation into a partial order

a partial order $p' = (V', E', l')$ is more *specific* than another partial order $p = (V, E, l)$, noted by $p \leq p'$, if there exists an injective mapping $h : V \rightarrow V'$ such that preserves labels (that is, $l \subseteq l' \circ h$), and $(u, v) \in E \Rightarrow (h(u), h(v)) \in E'$.

To construct the most specific partial order p we will match as many positions as possible from sequences $s \in S$. Yet there is still the problem of identifying which positions must be matched. Note that not all positions having same label are good to be overlapped: e.g. taking again the set $S = \{\langle(A)(C)(C)(C)(A)\rangle, \langle(C)(A)(C)(C)(A)\rangle\}$, the first (C) of $\langle(A)(C)(C)(C)(A)\rangle$ will not be overlapped with any (C) of the other sequence, otherwise we would get a partial order having more paths than the ones included in S . So, when matching positions of sequences in S , the algorithm has to take care of not adding new paths to p that are not in S (and so, keep the properties of lemma 5.1). The idea is that, given two sequences $s, s' \in S$, two positions can be matched as long as they are path preserving with respect to all sequences in S .

DEFINITION 5.1. (PATH PRESERVING POSITIONS)

Given a set of sequences S and let $s, s' \in S$ be two sequences s.t. $s = \langle(I_1) \dots (I_i) \dots (I_n)\rangle$ and $s' = \langle(I'_1) \dots (I'_j) \dots (I'_m)\rangle$, then positions i of s and j of s' are path preserving if,

- $I_i = I'_j$; and,
- $head(s, i) \diamond tail(s', j + 1) \subseteq s''$, for some $s'' \in S$; and,
- $head(s', j) \diamond tail(s, i + 1) \subseteq s''$, for some $s'' \in S$.

Then, we say that position i of s matches with position j of s' ; we note it by $p[i] \sim q[j]$.

This definition ensures that when matching different positions of sequences in S , any new possible paths that can be created by this overlapping will be always included in S . Note that this operation is symmetric, i.e. if $p[i] \sim q[j]$ then $q[j] \sim p[i]$; and also transitive as it is showed in the following proposition.

PROPOSITION 5.1. (TRANSITIVITY) Given a valid pair (S, T) and let $s, s', s'' \in S$, if $s[i] \sim s'[j]$ and $s'[j] \sim s''[k]$, then $s[i] \sim s''[k]$.

Proof. For a given valid pair (S, T) we have that $S = \bigcap t, \forall t \in T$ (prop. 4.1), which makes this set S consistent for transitivity property. In particular, if $s[i] \sim$

$s'[j]$ and $s'[j] \sim s''[k]$, then we have that S contains at least these following three sequences among others: $head(s, i) \diamond tail(s, i + 1)$ (i.e. sequence s), and $head(s', j) \diamond tail(s, i + 1)$, and $head(s', j) \diamond tail(s'', k + 1)$. The presence of these three sequences in S forces the sequence $head(s, i) \diamond tail(s'', k + 1)$ to be also present in the intersection of all $t \in T$ as well. Similarly, we can justify that S also contains the other necessary path $head(s'', k) \diamond tail(s, i + 1)$; thus, reaching the conclusion that $s[i] \sim s''[k]$.

Transitivity also ensures that a position from a sequence $s \in S$ cannot be matched with two different positions from $s' \in S$; otherwise, we would get a cycle and sequences in S would not be valid. The algorithmic solution for this problem finds the matching positions of a group of sequences S , of a valid pair (S, T) . The idea is checking definition 5.1 for all positions of the sequences in S . It is possible to improve the algorithm by using the transitivity property, and we rely on the fact that the number of sequences in S and the length of these sequences is quite reasonable.

Algorithm 2 Matching Positions of a Set of Sequences

Input: A group of sequences S

Output: A list of positions to be joined

```

1: for all  $s, s' \in S$  s.t.  $s \neq s'$  do
2:   for all positions  $i$  of  $s$  and  $j$  of  $s'$  do
3:     output whether  $p[i] \sim q[j]$ ;
4:   end for
5: end for

```

Algorithm 2 is the pseudocode that decides which positions must be matched of a group of closed sequences S . To get the most specific partial order, all the path preserving positions must be overlapped. Of course, this algorithm must be used over each set of closed sequences S obtained after algorithm 1, leading to different partial orders for each (S, T) . For example, in figure 5, each poset is the result of matching all the path preserving positions of sequences in S .

5.2 Theoretical Analysis In this section we want to show that the partial orders obtained by the proposed approach are indeed closed partial orders in \mathcal{D} .

DEFINITION 5.2. (CLOSED PARTIAL ORDER) *We say that a partial order p is closed if there exists no other partial order p' with $p \sqsubseteq p'$ s.t. $support(p) = support(p')$.*

A closed partial order is the most specific one among all those posets compatible with the same set

of input sequences, so they are the most informative ones. Actually, we can prove the following result.

LEMMA 5.2. *Given a valid pair (S, T) , a partial order obtained by matching all the path preserving positions of sequences in S is a closed partial order.*

Proof. It follows from lemma 5.1: part (1) ensures that the generated partial order whose maximal paths are in S is compatible with all transactions T ; part (2) of the same lemma ensures that this partial order cannot contain other paths not included in S , otherwise it is not compatible with T . Then, by matching all the path preserving positions from sequences in S we get a partial order p such that for no other partial order p' we have that $p \sqsubseteq p'$ and p' compatible with T . Thus, p is closed.

Note that closed partial orders can be obtained only by means of a set of closed sequences, that is, from the set S of each valid pair (S, T) ; otherwise the partial orders would not be closed. The set of closed partial orders describe the input sequential data \mathcal{D} in the most specific way. In terms of category theory, this process can be formalized as a coproduct transformation when not considering repetition of items in the input sequences (already proved in [5])

In the practice, this equivalence between groups of closed sequences and closed partial orders represents an important algorithmic simplification, since algorithms such as BIDE or CloSpan or TSP are now able to efficiently transform their patterns into closed partial orders and we do not need to mine them directly from the data. Note that if a minimum support condition is specified over the closed sequences mined by those algorithms, then the generated closed partial orders will be also over that minimum support.

6 Further Discussions

In this section we would like to note first that the contribution presented here also works when considering simultaneity condition of input sequences, that is, when having input sequences of itemsets, as in example of figure 7. Notice that algorithms such as CloSpan, BIDE or TSP are already able to mine closed sequential patterns when having sequences of itemsets, and grouping those sequences in valid pairs (S, T) can be done with the described algorithm.

Seq id	Input sequences
t_1	$\langle (AF)(D)(E)(A) \rangle$
t_2	$\langle (E)(ABF)(G)(BDE) \rangle$
t_3	$\langle (E)(A)(B)(G) \rangle$

Figure 7: Collection of data \mathcal{D}

The way of matching positions of sequences $s \in S$ for each (S, T) will be done as usual, that is, according to definition 5.1. This definition takes care of not adding new paths in the generated closed partial order: because of that, the set of items of two different positions must coincide exactly to be matched. The results obtained from the example of figure 7 are shown in figure 8 and figure 9.

T	S
$\{1, 2, 3\}$	$\{\langle(E)(A)\rangle\}$
$\{2, 3\}$	$\{\langle(E)(A)(B)\rangle, \langle(E)(A)(G)\rangle, \langle(E)(B)(G)\rangle\}$
$\{1, 2\}$	$\{\langle(AF)(D)\rangle, \langle(AF)(E)\rangle, \langle(E)(A)\rangle\}$
$\{1\}$	$\{\langle(AF)(D)(E)(A)\rangle\}$
$\{2\}$	$\{\langle(E)(ABF)(G)(BDE)\rangle\}$
$\{3\}$	$\{\langle(E)(A)(B)(G)\rangle\}$

Figure 8: Groups of closed sequences occurring together

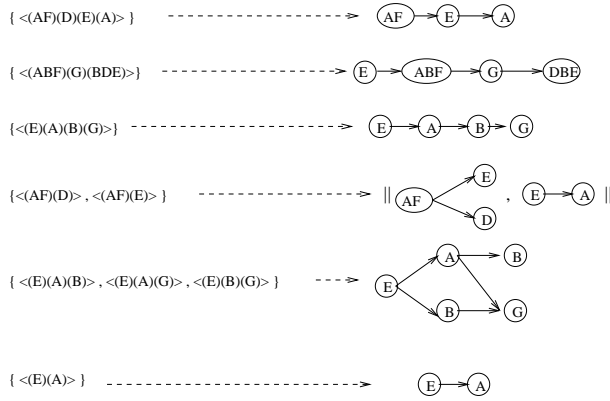


Figure 9: Partial orders from (S, T)

A different observation is that instead of generating all the closed partial orders out of the closed sequences, we may want to generate only those partial orders whose support is over a maximum threshold. To perform this transformation is also easy since we just need to select those valid pairs (S, T) s.t. the number of transactions in T is over this maximum value. We can play with other parameters, such as the length or the structure of the partial order.

Finally, we want to raise the discussion of how to represent the obtained partial orders. A first approach is to use adjacency matrices and consider that each different node is an entry of the matrix. It is easy to transform a group of sequences S and the list of overlapping positions given by algorithm 2 into an adjacency matrix. Other more visual solutions would involve graphical representations of directed acyclic graphs, which can certainly ease the interpretation of

the partial order made by the user. We are currently working towards a visualization of the final partial orders with GraphML.

7 Experimental Results

We evaluate our approach by performing experiments on different discrete sequential databases: a first database of 1000 transactions of synthetic data (we used a context-free grammar as generative model for sequences of words); a second database of 607 transactions corresponding to the command history of a unix computer user (downloaded from the UCI repository¹); and a third database corresponding to the first chapter of the book “1984” by George Orwell, where each different word is considered a different item and each sentence an input sequence. We are aware that these two real databases are quite small; but due to our lack of a large real dataset, we decided to perform here just a preliminary experimentation as an overview of the first results.

Our first goal is to evaluate the number of total closed partial orders in comparison to closed sequential patterns, and also, to analyze the quality of those partial orders. The process is first mining the closed sequential patterns over a certain minimum threshold named σ (we can use any existing algorithm), and then, organizing the closed sequences into valid pairs, as described in section 4. We also want to compare the performance of these two phases.

A first set of numerical comparisons obtained with the synthetic data are shown in table 1. We observe that the number of closed partial orders is always less than or equal to the number of closed sequential patterns; moreover, as we decrease the minimum support and we get more frequent sequences, the number of closed partial orders gets considerably smaller. Actually, we generated a first set of only 1000 synthetic transactions, to show that, even when the number of frequent sets and closed sequences is larger than the number of transactions, the number of closed posets never beats this limit.

σ	Frequent Seqs.	Closed Seq.	Closed Posets
200	31	26	26
100	114	92	92
50	520	401	92
40	850	645	224
30	1467	1108	389

Table 1: Counting of patterns for the synthetic data

The same results are shown in figure 2 using the unix command data. Here, we forced the minimum sup-

¹<http://kdd.ics.uci.edu/summary.data.type.html>

port to very low values, in order to evaluate the number of closed sequences with respect to the total posets. We still have that the number of closed partial orders is less than the number of closed sequences; however, we see this number is sometimes larger than the 607 original transactions, making the final patterns less manageable. Some real closed partial orders extracted from this database are shown in next figure 10.

σ	Frequent Seqs.	Closed Seq.	Closed Posets
30	277	259	259
20	1111	913	885
10	23460	1890	1565
5	37898	4778	3779

Table 2: Counting of patterns for the Unix command database

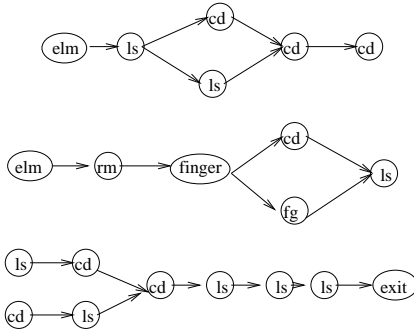


Figure 10: Some partial orders obtained from the Unix command database

In case we are dealing with unix user data, the frequent closed partial orders may be later used as the normal user profile for the intrusion detection systems (such as proposed in [9]). However, these closed posets may have other utilities in the field of knowledge discovery depending on the context. For example, in case of having a text (such as our second database with a text written by George Orwell), the closed partial orders can be used to classify subsequent texts according to a list of authors. In the first chapter of “1984” we have a total of 340 input sequences. We experimented with different minimum supports, and some of the obtained closed posets are shown in figure 11.

As mentioned, we can divide our discovery process in two phases: a first one of finding the closed sequences, and a second phase of obtaining valid pairs and closed partial orders from there. We observed that the first phase is the most I/O intensive, since it requires to examine a combinatorial number of patterns. On the other hand, the second phase is not an intensive step: the input to be examined is only the set of frequent closed

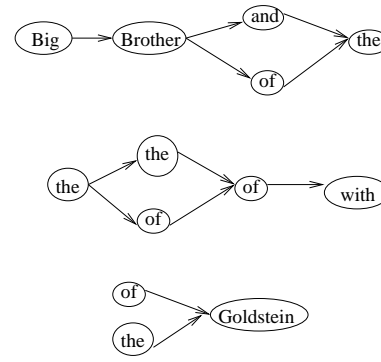


Figure 11: Some partial orders obtained from the “1984” database

sequences, and we do not require to combine those patterns, just organizing them in the conceptual structure; moreover, the operations performed to get such lattice are simply standard operations of sets, such as inclusion, intersection and so on. With small datasets, as the ones described here, this organization takes only a few seconds. We must point out though, that this second phase can be more costly when considering a very large database: in this case the tid lists are larger and the comparisons between these lists can be more expensive; however, we evaluated with synthetic data that, regardless the length of the database, the cost of organizing closed sequences is still insignificant compared to the first phase.

With these preliminary experiments we wanted to show that the main profit of our proposal is the possibility of generating classical partial orders out of closed sequences, without dealing directly with the input data.

8 Related Work

The importance of mining partial order structures from sequential data was first introduced in [11]. There, the authors start with a slightly different data model described as a long sequence of events; in the practice, this long sequence can be divided in several sliding windows, so, coinciding with the transactional model presented here.

The basic problem defined in [11] is to find frequent episodes, i.e. collections of events occurring frequently together in the input sequence. Episodes are formalized as acyclic directed graphs, so, it is equivalent to the one presented in section 5 of this paper. Episodes can be classified into: serial episodes (total orders), parallel episodes (trivial orders), and finally, hybrid episodes (indeed, general partial orders).

The work in [11] discusses different algorithmic approaches for the discovery of serial, parallel and hybrid episodes. In particular, the popular approach called Winepi is intended to look for frequent episodes in a Apriori fashion by sliding a window of fixed width along the event sequence: i.e., a complete pass along the data is used to compute the support of current episode candidates and, after each pass, new larger episodes are generated as long as the antimonotonicity property of support keeps them alive. So, at the end of the process Winepi has discovered all the frequent episodes of any kind fitting in the window.

This mentioned algorithmic approach performs two complex operations: first, generating new candidate episodes out of the smaller ones, and second, recognizing episodes in the sequence to update the support. In case of mining dense data, and specially, when dealing with hybrid episodes (i.e., general partial orders) or with non-injective episodes (those having repeated items), the algorithm incurs in a substantial runtime overhead (see [11] for more details). Apart from this algorithmic overhead, the number of the final discovered episodes is quite large and many of these episodes are not the most specific ones and they could be considered redundant: e.g. many of the final parallel episodes may be less informative than some of the serial episodes, and also, many serial episodes may be less informative than the hybrid episodes.

The proposals suggested in this paper reduce the discovery of episodes to the most specific ones, i.e. those which are closed. In this sense, it represents a semantic advance of hybrid episodes, since we just consider the most informative partial orders to summarize the database. Moreover, we show that it is not necessary to mine these structures directly from the data (as it is done by the Winepi approach), but just post-processing the closed sequential patterns. Operations done by algorithms such as CloSpan or BIDE are less expensive than Winepi since they just compare plain sequences, so, the final process is less costly.

Alternatively, the work in [11] proposes also another algorithmic approach called Minepi. In this case the mined episodes are unbounded in length, and for the moment, it is not clear how our proposal can improve semantically these episodes. Another work worth mentioning is [10]: it presents a method based on viewing a partial order as a generative model for a set of sequences and it applies different mixture model techniques. The final partial orders are not necessarily closed and so, they could be redundant; besides, they restrict the attention to a subset of partial orders called series-parallel partial orders (such as series-parallel digraphs) to avoid computational problems. Note that here we do not re-

strict in any sense the form of the final closed posets or the repetition of items.

Other works worth mentioning are [2] or [6, 14], but they deal with serial episodes more than hybrid structures as we manage here. Moreover, the difference with respect the current proposal, is that we are considering the discovery of episodes as a post-processing step of closed sequences, more than a direct discovery from the original data.

9 Conclusions

We have presented the notion of closed partial orders compatible with input sequences of itemsets. We show that this is a general data model that can be adapted to long sequences of events as well. By definition, these orders are the most informative ones for a set of maximal transactions and they provide a more compact overview of the input data. As a main contribution, we show that these closed partial orders can be derived simply from the closed set of sequences mined by existing algorithms.

In the practice, this transformation implies that going beyond closed partial orders once we have the closed sequential patterns is not costly. So, current algorithms can efficiently transform the discovered patterns into closed partial orders; thus, avoiding the complexity of mining these structures directly from the data. First experiments show that postprocessing closed sequences is not a costly phase, so that the real profit of the proposal is in the ability to generate closed partial orders without accessing the input data, more than the reduction in the number of patterns.

Yet there is still the work of formalizing the matching process of closed sequences into a partial order. We mentioned that in case of not having repetition of items in the input sequences, this process can be formalized by means of coproduct operations of category theory (as it shows [5], where also the necessary closure system of partial orders is characterized). The next step is to do this formalization for the general data model, that is, when having repetition of items and simultaneity. We are currently working towards this characterization by means of colimit operations of category theory.

Other further work includes the study of implications or association rules of partial orders. Given that implications among closed sequential patterns are already formalized, it is reasonable to go further and find a similar characterization for these closed partial orders.

Acknowledgements: The author thanks José L. Balcázar for his useful discussions and comments, and also Pablo Díaz-López for helping with the implementations and experiments.

References

- [1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, pages 3–14. IEEE Computer Society Press, 1995.
- [2] M.J. Atallah, R. Gwadera, and W. Szpankowski. Detection of significant sets of episodes in event sequences. In *Proceedings of the 4th International Conference on Data Mining*, pages 3–10, 2004.
- [3] J.L. Balcázar and G. Casas-Garriga. On Horn axiomatizations for sequential data. In *Proceedings of the 10th Int. Conference on Database Theory*, pages 215–229, 2005.
- [4] G. Casas-Garriga. Towards a formal framework for mining general patterns from structured data. In *Workshop Multi-relational Datamining, in KDD Int. Conf*, pages 215–229, 2003.
- [5] G. Casas-Garriga and J.L. Balcázar. Coproduct transformations on lattices of closed partial orders. In *Proceedings of 2nd. Int. Conference on Graph Transformation*, pages 336–351, 2004.
- [6] G. Das, R. Fleischer, L. Gasieniec, D. Gunopulos, and J. Kärkkäinen. Episode matching. In *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching*, pages 12–27, 1997.
- [7] B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order*. Cambridge, 2002.
- [8] B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations*. Springer, 1998.
- [9] W. Lee, S.J. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 120–132, 1999.
- [10] H. Mannila and C. Meek. Global partial orders from sequential data. In *Proceedings of the 6th Int. Conference on Knowledge Discovery in Databases*, pages 161–168, 2000.
- [11] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovering frequent episodes in sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
- [12] N. Pasquier, Y. Bastide, R. Taouil L., and Lakhal. Closed set based discovery of small covers for association rules. In *Proceedings of the 15th Int. Conference on Advanced Databases*, pages 361–381, 1999.
- [13] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: mining sequential patterns by prefixprojected growth. In *Proceedings of the 17th Int. Conference on Data Engineering*, pages 215–224, 2001.
- [14] Z. Tronicek. Episode matching. In *Combinatorial Pattern Matching*, pages 143–146, 2001.
- [15] P. Tzvetkov, X. Yan, and J. Han. TSP: Mining top-k closed sequential patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 347–358, 2003.
- [16] J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the 19th Int. Conference on Data Engineering*, pages 79–90, 2003.
- [17] X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of the Int. Conference SIAM Data Mining*, pages 166–177, 2003.
- [18] M. Zaki. Generating non-redundant association rules. In *Proceedings of the 6th Int. Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2000.
- [19] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal, special issue on Unsupervised Learning*, 42(1/2):31–60, 2001.

SUMSRM: A New Statistic for the Structural Break Detection in Time Series

Kwok Pan PANG and Kai Ming Ting

Gippsland School of Computing and Information Technology
Monash University, Victoria 3842, Australia
Email : {ben.pang, kaiming.ting}@infotech.monash.edu.au,

Abstract. Structural break is one of the important concerns in non-stationary time series prediction. The cumulative sum of square (CUSUMS) statistic proposed by Brown et al (1975) has been developed as a general method for detecting a structural break. To better understand CUSUMS, this paper analyses the relationship among the bias of the break location estimation, pre-break data size and the decay rate of square residual. Our analysis reveals that small pre-break data size or low decay rate will greatly increase the bias of the break location estimation when there is a change of the mean. Based on the analysis, the paper proposes a new statistic SUMSRM to improve the performance of structural break detection and to reduce the bias of break location estimation. Our empirical evidence confirms that our intended design of the new statistic performs better than the CUSUMS statistic when there is a change of mean in the time series.

1. Introduction

In forecasting time series, ignoring structural breaks which often occur in the time series significantly reduces the accuracy of the forecast (Pesaran and Timmermann, 2003). Since the classical Chow test (1960) was developed, the past decade has seen considerable empirical and theoretical research on structural break detection in time series. Cumulative Sum of Recursive Residual (CUSUM) and Cumulative Sums of Square (CUSUMS) statistics (Brown et al. 1975) have been developed as general methods for single structural break detection. Kramer and Schotman (1992) proposed a modified statistic from CUSUM; the structural change is detected based on the range of the CUSUM rather than the maximum point of the absolute value of CUSUM. Chu et al (1995) proposed a test for the structural change based on the moving sums (MOSUMS) of the recursive residual. Inclan and Tiao (1994) used the CUSUMS for multiple structural breaks detection. Pesaran and Timmermann (2002) proposed Reverse CUSUM for detecting the most recent break. They showed that the accuracy of the forecast can be improved only if the data after the most recent break is selected as the training set, instead of using all available data for training in the time series which contains structural breaks. Pang and Ting (2003) further extended the idea of Reverse CUSUM and proposed a data selection method for time series prediction: all segments that have the same structure as the most recent segment will be grouped together to form an accumulated segment to be used as the new training set. Further, Pang and Ting (2004)

provided an analysis about the centered version of CUSUMS. Their analysis reveals that the structural break detection performance can be improved if we can increase the pre-break data size or decrease the post-break data size, resulting a modified Centered CUSUMS that overcome the existing weakness.

This paper first analyses Centered CUSUMS by examining the relationship among the pre-break data size, decay rate of square residual and the bias of the structural break estimation. The analysis shows that small pre-break data size or low decay rate will increase the bias of the break location estimation when there is the structural change of mean or trend. The decay rate of square residual is defined as the change rate of the Centered CUSUMS, decay rate = (square residual at time b - square residual at time a)/($b-a$), where $b > a$. Then, it proposes a new statistic SUMSRM by using square deviation about the median and the sliding window prediction residual. Our analysis shows that the square deviation about the median has higher sensitivity to the structural change compared with the square deviation about the mean used in Centered CUSUMS. The analysis also finds that the sliding window prediction residual can provide a higher decay rate than recursive residual. The empirical evidence shows that the proposed statistic can effectively improve the break detection, and eliminate the bias of the break location estimation, especially when there is a mean change. In the paper, we evaluate the performance of the proposed statistic when there are structural changes with single or multiple breaks.

Section 3 briefly describes the background of the Centered CUSUMS. We present the bias analysis for the Centered CUSUMS in section 4, and the new statistic SUMSRM is proposed in section 5. The experiments and results are reported in section 6.

2. Structural change

The parameters of the predictive model are assumed to be consistent and constant over time. If these conditions cannot be met, it is said that the structural change has occurred in the time series.

Let us take a linear regression as an example. Suppose a time series can be explained by:

$$y_t = x_t \beta_t + \sigma_t \varepsilon_t, \text{ where } \varepsilon_t \sim N(0,1)$$

The time series is regarded to have “no structural change” if the parameters β_t and σ_t are constant and consistent over time.

3. Background of CUSUMS

Let y_1, y_2, \dots, y_n be the time series under consideration. We first convert the series into input and output pairs to be used for ordinary linear regression.

The basic linear regression model we used is having the output y_t with k input variables:

$$y_t = \lambda_0 + \lambda_1 y_{t-1} + \lambda_2 y_{t-2} + \dots + \lambda_k y_{t-k}.$$

We use the following notation to denote the observation matrices $Y_{m,n}$ and $X_{m,n}$ which consist of n observations in the time series.

$$Y_{m,n} = \begin{bmatrix} y_m \\ y_{m+1} \\ \dots \\ y_n \end{bmatrix}, \quad \beta_n = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \dots \\ \lambda_k \end{bmatrix} \text{ and}$$

$$X_{m,n} = \begin{bmatrix} x_m \\ x_{m+1} \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 & y_{m-1} & y_{m-2} & \dots & y_{m-k-2} & y_{m-k} \\ 1 & y_{m-2} & y_{m-3} & \dots & \dots & y_{m-k-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & y_{n-1} & y_{n-2} & \dots & \dots & y_{n-k} \end{bmatrix}$$

where $m < n$, and the x_m, x_{m+1}, \dots, x_n are the row vectors.

Using the observations as the training data, the least square coefficients β_n can be estimated by

$$\hat{\beta}_n = (X'_{1,n} X_{1,n})^{-1} X'_{1,n} Y_{1,n}.$$

The CUSUM and CUSUMS statistics (Brown et al 1975) are defined as follows. The CUSUM statistic is based on the standardized recursive residual w_r :

$$w_r = (y_r - x_r \hat{\beta}_{r-1}) / d_r, \quad r = 2k+1, \dots, n-1, n \quad (1)$$

where

$$\hat{\beta}_r = (X'_{1,r} X_{1,r})^{-1} X'_{1,r} Y_{1,r} \quad (2)$$

$$d_r = 1 + x_r (X'_{1,r} X_{1,r})^{-1} x'_r$$

The CUSUMS is defined in terms of w_r :

$$s_r = \frac{\sum_{i=1}^r w_i^2}{\sum_{i=1}^n w_i^2}, \quad r = 2k+1, \dots, n-1, n \quad (3)$$

The Centered CUSUMS is defined as:

$$s_{r,n}^* = \frac{\sum_{i=1}^r w_i^2}{\sum_{i=1}^n w_i^2} - \frac{r}{n}, \quad r = 2k+1, \dots, n-1, n \quad (4)$$

Note that $s_{r,n}^*$ has zero mean.

The test statistic for structural break detection is:

$$T = \sqrt{\frac{n}{2}} \max_r |s_{r,n}^*|$$

The estimated break location, if T is above a critical value (for a specific confidence level), is defined as:

$$\hat{r} = \arg \max_r |s_{r,n}^*|$$

Under variance homogeneity, Inclan and Tiao (1994) show that $s_{r,n}^*$ behaves like a Brownian Bridge asymptotically.

Pang and Ting (2004) show that small post-break data size or large pre-break data size can improve the structural break detection performance, and they propose the modified Centered CUSUMS statistic as follows:

$$MT_{n_1} = \max_{0 \leq r \leq n_1} \sqrt{\frac{n_1}{2}} \left| \frac{\sum_{i=1}^r w_i^2}{\sum_{i=1}^{n_1} w_i^2} - \frac{r}{n_1} \right|, \quad \text{where } r \leq n_1 \leq n$$

The estimated break location will be obtained by:

$$\hat{r} = \arg \max_{0 \leq r \leq n_1} \sqrt{\frac{n_1}{2}} \left| \frac{\sum_{i=1}^r w_i^2}{\sum_{i=1}^{n_1} w_i^2} - \frac{r}{n_1} \right|, \quad \text{where } r \leq n_1 \leq n$$

In the modified Centered CUSUMS statistic, we will use an adjustable data size n_1 instead of data size n . n_1 is always within the range $r \leq n_1 \leq n$, and it is not fixed. MT_{n_1} will be optimized by selecting an appropriate n_1 . As s_{r,n_1}^* behaves like the Brownian Bridge asymptotically, we can use the same critical value for the specified n_1 that are tabulated in Inclan and Tiao (1994).

4. Bias analysis for the Centered CUSUMS

In this section, we concentrate on analyzing the bias of the break location estimation when changing the parameter β or σ . We will split the analysis into two parts: (a) change of β_t and (b) change of σ_t , after we show the effect of a step change of the Centered CUSUMS below.

The Centered CUSUMS in (4) can be rewritten as

$$C(r) = \frac{1}{nV} \left(\sum_{i=1}^r w_i^2 - rV \right)$$

where $V = (\sum_{i=1}^n w_i^2)/n$ can be interpreted as the average of the square residual.

The step change of the Centered CUSUMS can be written as:

$$C(r) - C(r-1) = \frac{1}{nV} (w_r^2 - V) \quad (5)$$

It shows that the change of the Centered CUSUMS can be explained using the difference between the value of the square residual and the average square residual value. If $w_r^2 > V$, the Centered CUSUMS will increase at time r . However, if $w_r^2 < V$, the Centered CUSUMS will decrease at time r .

(a) The bias of the break location estimation when there is a change of β_t

We suppose the time series is composed of two segments. We assume the variance σ_t^2 is constant over the time, and only the mean or the trend β_t changes.

$$\beta_t = \begin{cases} \beta_A & t = 1, 2, \dots, r \\ \beta_B & t = r+1, r+2, \dots, n \end{cases}$$

where $\beta_A \neq \beta_B$

Based on the above structural change, we generate the square residual plot (the square of the recursive residual vs. time) in figure 1. It produces the peak at $t = r+1$, and a downward slope after the peak. Then we take the average of the square of the recursive residual (i.e. the horizontal dotted line in figure 1).

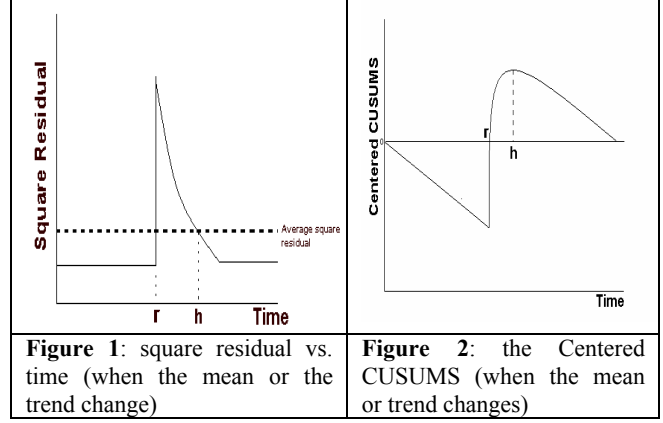


Figure 1: square residual vs. time (when the mean or the trend change)

Figure 2: the Centered CUSUMS (when the mean or trend changes)

Based on the equations (4) and (5), we convert the square residual to the Centered CUSUMS, and produce the plot of the Centered CUSUMS in figure 2. The value of the Centered CUSUMS starts to decrease until it reaches the minimum point at $t = r$. Then the value of Centered CUSUMS starts increasing until $t = h$, and the value decreases after $t = h$.

After taking the absolute value to the Centered CUSUMS, the estimated break location can be determined by :

$$\text{Break point} = \begin{cases} r & \text{when } |C(r)| \geq |C(h)| \\ h & \text{when } |C(r)| < |C(h)| \end{cases}$$

Let us consider the first condition $|C(r)| \geq |C(h)|$

$$\begin{aligned} |C(r)| \geq |C(h)| &\Rightarrow \left| \frac{1}{nV} \left(\sum_{i=1}^r w_i^2 - rV \right) \right| \geq \left| \frac{1}{nV} \left(\sum_{i=1}^h w_i^2 - hV \right) \right| \\ &\Rightarrow \left| \sum_{i=1}^r w_i^2 - rV \right| \geq \left| \sum_{i=1}^h w_i^2 - hV \right| \end{aligned}$$

As $C(h) > 0$ and $C(r) < 0$, the above formula can be rewritten as :

$$\begin{aligned} |C(r)| \geq |C(h)| &\Rightarrow \left(rV - \sum_{i=1}^r w_i^2 \right) \geq \left(\sum_{i=1}^h w_i^2 - hV \right) \\ &\Rightarrow rV - \sum_{i=1}^r w_i^2 \geq \sum_{i=1}^r w_i^2 + \sum_{i=r+1}^h w_i^2 - hV \\ &\Rightarrow 2rV - 2 \sum_{i=1}^r w_i^2 \geq \sum_{i=r+1}^h w_i^2 - (h-r)V \\ &\Rightarrow 2rV - 2 \sum_{i=1}^r w_i^2 \geq nV(C(h) - C(r)) \end{aligned}$$

Using the similar approach for the second condition, we can write it as:

$$|C(r)| < |C(h)| \Rightarrow 2rV - 2 \sum_{i=1}^r w_i^2 < nV(C(h) - C(r))$$

Based on the above analysis, a bias estimate of the break location, instead of r , will be introduced when $\left(2rV - 2\sum_{i=1}^r w_i^2\right) < nV(C(h) - C(r))$

The bias of the break location estimation can be reduced by the following two factors:

- (i) Pre-break data size
- (ii) Decay rate of square residual after the break.

If the pre-break data size is large enough, then $2rV - 2\sum_{i=1}^r w_i^2 \geq nV(C(h) - C(r))$, and the estimated break location is expected to be r . Unfortunately, it is not possible to increase the pre-break data size in many situations. Increasing the decay rate is the other way to eliminate the bias of the break location estimation by forming a steeper slope after the break, which makes the value h closer to the value r .

In order to increase the decay rate, we propose to use sliding window prediction residual instead of the recursive residual. More details will be discussed in section 5.2.

b) Break location estimation when there is a change of σ_t

We suppose the time series are composed of two segments. We assume the mean or trend β_t is constant over the time and only the variance σ_t^2 changes.

$$\sigma_t = \begin{cases} \sigma_A & t = 1, 2, \dots, r \\ \sigma_B & t = r+1, r+2, \dots, n \end{cases}$$

Where: $\sigma_A \neq \sigma_B$

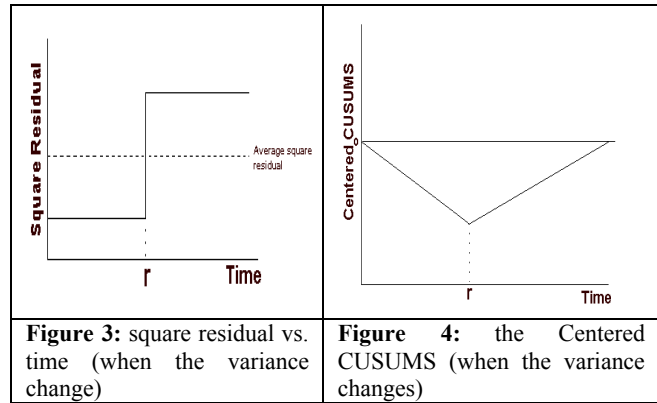
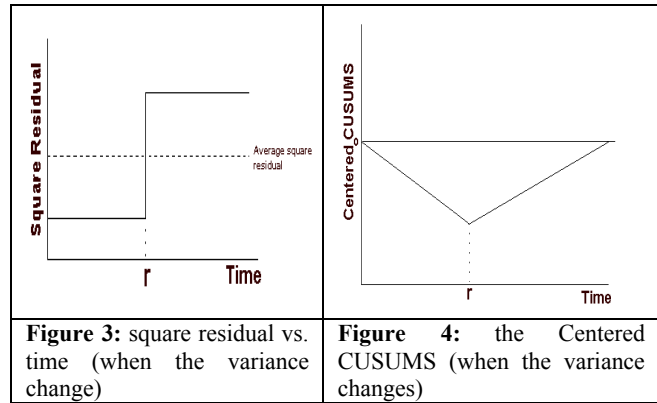
Based on the above structural change, let us consider the situation: $\sigma_A < \sigma_B$. Its square residual plot (the square of the recursive residual vs. time) is indicated in figure 3. The square residual plot consists of two horizontal lines. The lower horizontal line represents the square residual of the first segment, and the upper horizontal line represents the square residual of the second segment. The average of the square of the recursive residual is shown as the horizontal dotted line in figure 3.

Based on the equations (4) and (5), we convert the square residual to the Centered CUSUMS, and produce a plot of the Centered CUSUMS in figure 4. The value of the Centered CUSUMS starts to decrease until it reaches the minimum

point at $t = r$. Then the value starts increasing until $t = n$. The estimated break location = r will be determined after taking the maximum absolute value of the Centered CUSUMS.

It is interesting to note that the square residual after the break point ($t = r$) keeps constant and all stay above the average of the square residual. The estimated break location is thus expected to be at $t = r$. We have provided the evidence to show that changing the pre-break data size or the decay rate won't make any alternation on the bias of the break location estimation.

Same approach can be applied into the situation $\sigma_A > \sigma_B$. The bias of break location estimation won't be affected by the change of the variance.



5. A new statistic

Based on the above analysis, the paper proposes a new statistic using the square deviation about the median and sliding window prediction residual. The details of discussion are listed in the followings.

5.1 Square deviation about the median

Let us consider the mean of square deviation about the parameter θ

$$\begin{aligned} E[(x - \theta)^2] &= E[(x - E(x) + E(x) - \theta)^2] \\ &= E\{[x - E(x)]^2 + 2[x - E(x)][E(x) - \theta] + [E(x) - \theta]^2\} \\ &= E[(x - E(x))^2] + (E(x) - \theta)^2 \\ &= Var(x) + (Bias)^2 \end{aligned}$$

Note that the Bias here refers to the bias the residual estimation, which is different from the bias of the break location estimation discussed in the previous sections.

If we let $\theta = \text{mean}$, square deviation about the mean will be exactly the same as the $var(x)$. If we let $\theta = \text{Median}$, the Median will be approximately equal to the $E(x)$ when x is symmetrically distributed and when the data size is large enough. However, when the structure changes, the median

starts to deviate from $E(x)$. It is found that the structural change intensifies the bias, which leads to larger value of square deviation about the median. There is an implication that the square deviation about the median is more sensitive to the structural change (i.e. change of the mean or trend) comparing with the square deviation about the mean. For instance, we let x be the residual. The mean of residual is always assumed to be zero in the ordinary linear regression. The square deviation for the residual will be the same as the sum of square residual no matter the structural change occurs or not. However, if we use the square deviation about the median, we expect the difference between the value of square deviation about the median and square deviation about the mean will be small if no structural change occurs, and the difference will be enlarged if the structure changes

5.2 Sliding window prediction residual

We propose to use the sliding window prediction residual instead of the recursive residual because the sliding window prediction can increase the decay rate. When using the sliding window approach, only the windows that cover the break point will be affected by the structural change. Suppose the data size of the time series is n , the size of slide window is m and the break is located at c . If we use the recursive residual, all prediction residual after point c will be affected by the structural change. However, if we use the sliding window prediction residual instead, only the prediction residuals $\{w_i, c \leq i \leq c + m\}$ will be affected by the structural change. That means less data or residual will be affected by the structural change when using sliding window prediction residual. It also implies that higher decay rate (i.e. the slope after the peak become steeper in the square residual plot) is obtained.

The sliding window prediction residual can be described as figure 5. For each window, the size of the each window (p) is fixed. The prediction residuals are obtained by using sliding window. All data in the window will be used as training set to train the model, and then we use the trained model to make a one step ahead prediction. After obtaining the prediction residual, we slide the window one step ahead to make the next prediction residual. This approach is different from the Centered CUSUMS which adopts the recursive residual.

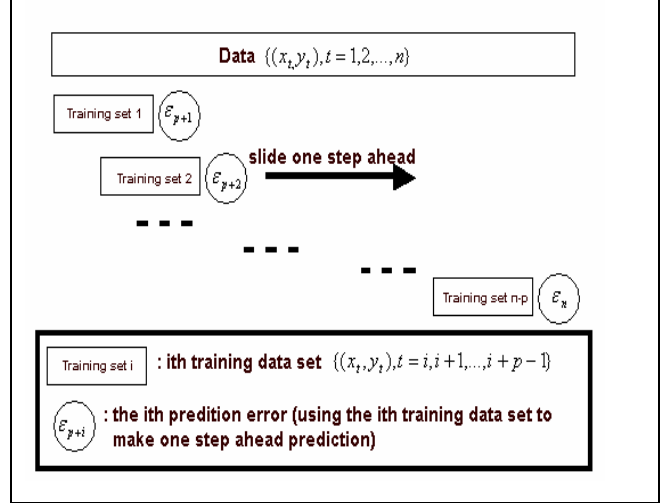


Figure 5 : Sliding window prediction residual

5.3. SUMSRM statistic

As mentioned in section 5.1, when the structure changes, the square deviation about the median increases as the bias of the expected residual estimation increases. Also, using the slide window prediction residual, instead of the recursive prediction residual, will increase the decay rate of square residual after the peak caused by the structural change. We propose a new statistic by combining these two ideas and the main idea from the modified Centered CUSUMS, searching for an appropriate post-break data size for structural break detection. Empirical evidence from Pang and Ting (2004) has shown that large post-break data size weakens the structural break detection performance, so it is important to select appropriate post break data size. We call the new proposed statistic “Sum of Square Sliding Residual about the Median” (SUMSRM).

Suppose the time series $\{(x_t, y_t), t = 1, 2, \dots, n\}$, where x_t is the row vector, $x_t = (y_{t-1}, y_{t-2}, \dots, y_{t-p})$ and p is the size of each sliding window.

The SUMSRM test statistic for the break detection is:

$$S_{n_1} = \max_{0 \leq r \leq n_1 - p} |D_{r, n_1}|, \quad r \leq n_1 - p \leq n - p$$

where

$$D_{r, n_1} = \sqrt{\frac{n_1 - p}{2}} \left(\frac{\sum_{i=1}^r \gamma_i^2}{\sum_{i=1}^{n_1 - p} \gamma_i^2} - \frac{r}{n_1 - p} \right), \quad r \leq n_1 - p \leq n - p$$

$$\gamma_i = \frac{(\varepsilon_i - \text{Med}_i)}{d_i},$$

$$d_i = 1 + x_i (X'_{i-p, i-1} X_{i-p, i-1})^{-1} x_i', \quad i = p+1, p+2, \dots, n$$

ε_i is the prediction residual of the i^{th} slide window:

$$\varepsilon_i = y_i - x_i \hat{\beta}_i, \quad i = p+1, p+2, \dots, n$$

Med_i is the median of the i^{th} window training set residual:

$$Med_i = \text{Median}(Y_{i-p,i-1} - X_{i-p,i-1} \hat{\beta}_i), \quad i = p+1, p+2, \dots, n$$

If S_{n_1} is above the critical value for the adjusted data size n_1 , the estimated break location is defined as

$$\hat{k} = p + \arg \max_{0 \leq r \leq n_1 - p} \sqrt{\frac{n_1 - p}{2} \left(\frac{\sum_{i=1}^r \gamma_i^2}{\sum_{i=1}^{n_1 - p} \gamma_i^2} - \frac{r}{n_1 - p} \right)}, \quad r \leq n_1 - p \leq n - p$$

The parameter β of each sliding windows can be obtained by:

$$\hat{\beta}_i = (X'_{i-p,i-1} X_{i-p,i-1})^{-1} X'_{i-p,i-1} Y_{i-p,i-1}, \quad i = p+1, p+2, \dots, n$$

We assume that the residual is normally distributed. Based on this assumption, D_{r,n_1} behaves like the Brownian Bridge Asymptote, and its proof is shown in the appendix A. According to the above statistic, the critical value at different data size and window size are estimated through the simulation. The critical value tables are shown in the appendix B. For the simulation, we use a Gaussian random walk with varied data size and varied window size to approximate the wiener process and replicate 10000 times with different seeds.

6. Experiment

We divide the experiments into two sections. The first section of our experiments is designed to verify the analysis conducted in section 4, showing the relationship between the pre-break data size and the bias of the break location estimation. The second section aims to evaluate the performance of the SUMSRM test statistic when single break or multiple breaks exist. In the experiments, we use the significant level α at 0.01 for the statistical test. The performance will be measured in terms of (a) accuracy, (b) mean of square deviation, (c) estimated bias and (d) mode of the estimated break location. We define the first three measures below.

(a) Accuracy is the percentage of correct classification. Every predicted break location falls into one of the following categories:

(i) **Correct Classification:** If the estimated break location is within the boundary (i.e. actual break

location ± 10), we classify it to be correct classification.

(ii) **Incorrect Classification:** If the estimated break location is outside the boundary (i.e. actual break location ± 10), we classify it to be incorrect classification.

(b) Mean of Square Deviation (MSD) is defined as:

$$MSD = \frac{\sum_{i=1}^v (\hat{b}_i - Actb_i)^2}{v}$$

where

\hat{b} is the estimated break location, $Actb$ is the actual break location which is the nearest break point to the estimated break location, and v is the number of simulation used in the experiment.

(c) The bias of the estimated break location (EB) is defined as:

$$\text{Estimated Bias (EB)} = \frac{\sum_{i=1}^v (\hat{b}_i - Actb_i)}{v}$$

As some simulated time series data with the structural change cannot be detected in some cases (i.e. the estimated break location will be defined as zero when the series with the structural change cannot be detected), these undetected observations may make EB generate the misleading measurement. Therefore, we mainly use the mode of the estimated location for measuring the bias of estimated break location, and EB is used for the reference.

For each experiment, we simulate the series 3000 times with different seeds, and we report the average performance over 3000 runs. In the experiments, we specify the number of input variables to be 3, and the size of each sliding window to be 40.

6.1 Validating the analysis result for the Centered CUSUMS

This experiment aims to validate the relationship between the pre-break data size and the bias of the break point estimation when the mean or variance changes. The procedure is described in the following.

Procedure

1. Two single break time series are designed to evaluate the break point estimation, as shown in the

table 1. The first time series is designed for the mean change, and the second one is for the variance change. Both series have the pre-break data size 300, and post break data size 300. We specify σ to be 0.2 for both segments in the first series, and in the second series, σ is specified to be 0.5 and 2.0 respectively for the first and second segment of the second series. We record their break detection performance and break location estimation.

- We use the same time series as in (1), but the pre-break data size is reduced to 100 (the post-break data size remains the same). We record their break detection performance and break point estimation, and then compare the result with (1).

Table 1. Description of the two single-break series (when mean or variance changes): each series is composed of two segments. The whole sequence of the series is $\{y_t, t = 1, 2, \dots, 600\}$

Category of the structural change	Segment 1 $\{y_t, t = 1, 2, \dots, 300\}$	Segment 2 $\{y_t, t = 301, 302, \dots, 600\}$
Mean change	$y_t = 20 + 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_1 \varepsilon_t$ $\sigma_1 = 0.2$	$y_t = 30 + 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_1 \varepsilon_t$ $\sigma_1 = 0.2$
Variance change	$y_t = 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_2 \varepsilon_t$ $\sigma_2 = 0.5$	$y_t = 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_3 \varepsilon_t$ $\sigma_3 = 2.0$

Experiment Results

The results are summarized as follows:

The effect of Pre-break data size to the estimation of the break location (when the mean changes).

As shown in figures 6a and 6b, the result shows the bias of the break location estimation is affected by the pre-break data size. When the pre-break data is 300, the mode of the estimated break location is 300 that is exactly the same as the actual break location. However, when we reduce the pre-break data size to 100, we find the mode of estimated break location is 122 and the estimated bias of break location is 21.7 where the actual break location is 100. The bias of estimate break location is increased to 22 after reducing the pre-break data size.

The effect of the pre-break data size to the estimation of the break location (when the variance changes)

As shown in figures 7a and 7b, the result shows that pre-break data size has a small impact on the bias of the break

location estimation. When the pre-break data size is 300, the mode of the estimated break location is 300 that is exactly same as the actual break location. When we reduce the data size to 100, the mode of the estimated break location is 100 that is also the same as the actual break location.

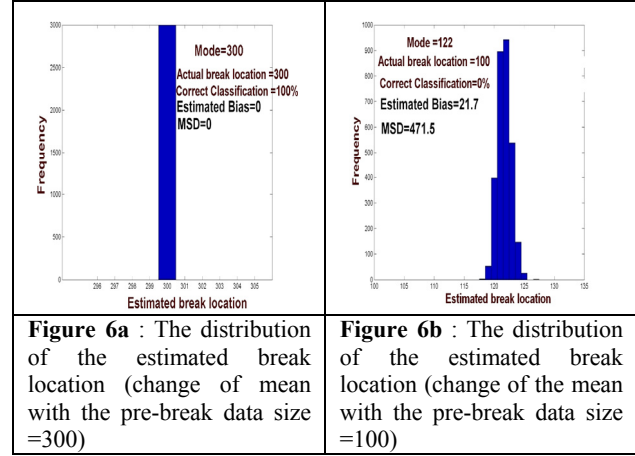


Figure 6a : The distribution of the estimated break location (change of mean with the pre-break data size =300)

Figure 6b : The distribution of the estimated break location (change of the mean with the pre-break data size =100)

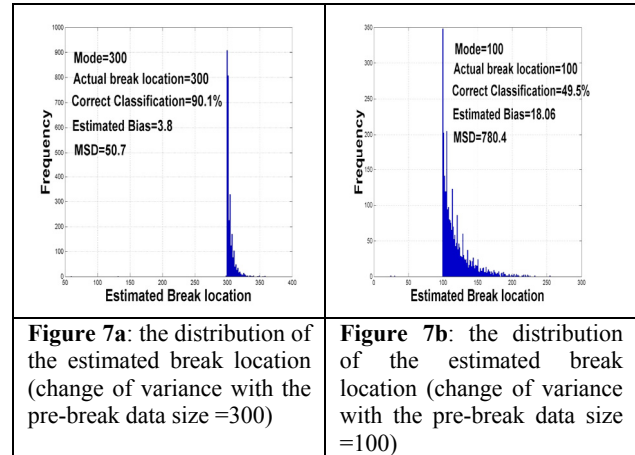


Figure 7a: the distribution of the estimated break location (change of variance with the pre-break data size =300)

Figure 7b: the distribution of the estimated break location (change of variance with the pre-break data size =100)

6.2 Evaluating the effectiveness of the proposed statistic

The following experiment is designed to evaluate the performance of the proposed statistic and compare it with the traditional Centered CUSUMS and the modified Centered CUSUMS statistics when single break or multiple breaks exist.

6.2.1 Time series with a single break

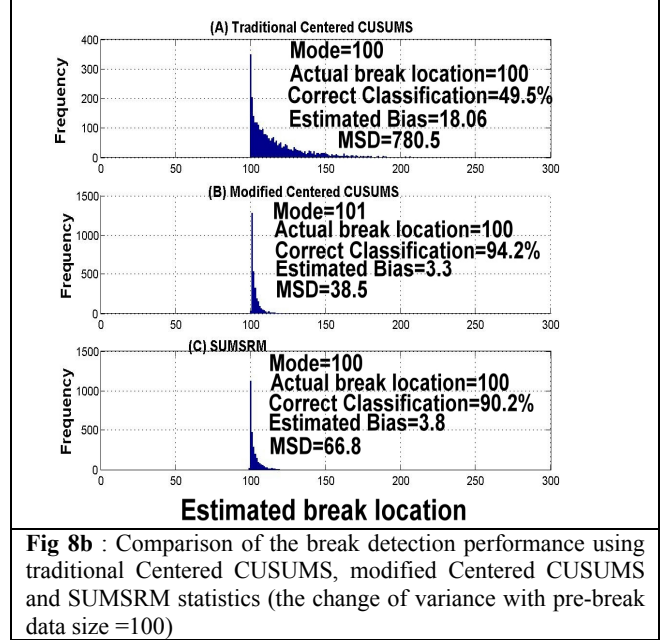
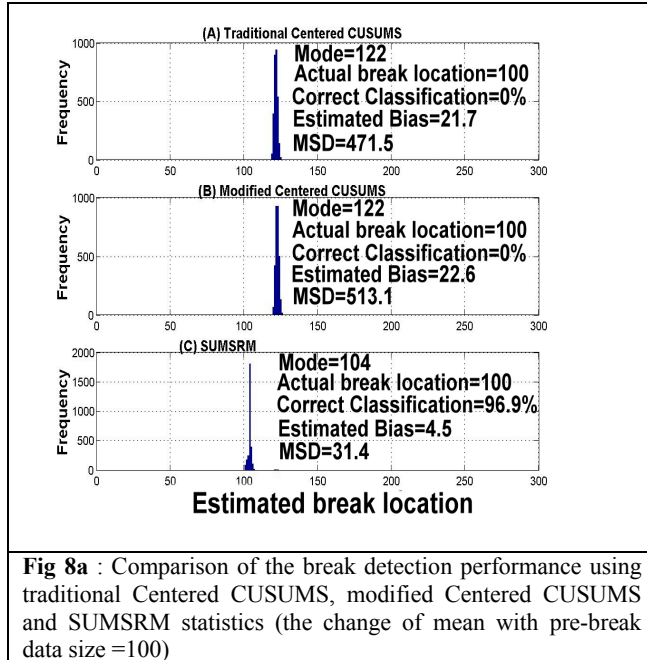
The experiment is designed to evaluate the performance of the proposed SUMSRM statistic for the time series in which

a single break exists. In the experiment, we will use the same time series as in the experiment in section 6.1 with the pre-break data size 100 and post break data size 300. We record the break detection performance and break point estimation using the traditional centered CUSUMS, modified centered CUSUMS and SUMSRM statistic.

Experiment Results

The results are summarized as follows:

- As shown in figure 8a, when the mean changes, the SUMRM significantly reduces the bias of the break location estimation. The result clearly shows that the SUMSRM statistic outperforms the traditional Centered CUSUMS and the modified Centered CUSUMS, and is significantly better in all four performance measures. The difference between the mode of estimated break location and actual break location is reduced from 22 to 4 after switching to SUMSRM from the traditional Centered CUSUMS or modified Centered CUSUMS.
- As shown in figure 8b, when variance changes, the SUMSRM statistic performs better than the traditional Centered CUSUMS, but is slightly worse than the modified Centered CUSUMS, in terms of the percentage of correct classification and MSD. Also, the modes of the estimated break location derived by those three selected statistics are almost identical, and they are the same or almost the same as the actual break location.



6.2.2 Time series with multiple breaks

This part of the experiments is to evaluate the performance of the SUMSRM statistic for the time series with multiple breaks when the mean or the variance changes. It also examines whether the SUMSRM can still detect any one of the structural changes and whether the estimated break location still falls into the acceptable range of break location. The time series is composed of 7 segments $\{seg_1, seg_2, \dots, seg_7\}$. The characteristics of the series with the change of mean and variance are described in tables 2 and 3 respectively.

Table 2: Descriptions of 7 segment series with the change of mean: the sequence of the series is $\{seg_1, seg_2, seg_3, seg_4, seg_5, seg_6, seg_7\}$. The length of each segment is specified to be 100, and the initial value of the series are specified as: $y_1 = y_2 = y_3 = 1$. The distribution of ε_t is $\varepsilon_t \sim N(0,1)$.

$seg_1, seg_3, seg_5, seg_7$	seg_2, seg_4, seg_6
$y_t = 20 + 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_1\varepsilon_t,$ $\sigma_1 = 0.2$	$y_t = 30 + 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_1\varepsilon_t,$ $\sigma_1 = 0.2$

Table 3: Description of 7 segment series with the variance change: the sequence of series is {seg₁, seg₂, seg₃, seg₄, seg₅, seg₆, seg₇}. The length of each segment is specified to be 100, and the initial value of the series are specified as: $y_1 = y_2 = y_3 = 1$. The distribution of ε_t is $\varepsilon_t \sim N(0,1)$

seg1, seg3, seg5, seg7	seg2, seg4, seg6
$y_t = 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_2\varepsilon_{1t},$ $\sigma_2 = 0.5$	$y_t = 0.6y_{t-1} + 0.3y_{t-2} + 0.1y_{t-3} + \sigma_3\varepsilon_t,$ $\sigma_3 = 2.0$

Experiment Result

The results are summarized as follows:

The performance of the three statistics in the multiple-break time series when the mean changes.

As shown in table 4a, the SUMSRM statistic outperforms the traditional Centered CUSUMS and the modified Centered CUSUMS. The SUMSRM obtains 100% correct classification rate and has the smallest bias of the estimated break location in terms of the estimated bias and the mode among three statistics. A large bias of the estimated break location has been observed in the experiment using the traditional Centered CUSUMS.

The performance of the three statistics in the multiple-break time series when the variance changes.

As shown in table 4b, all three statistics perform comparably in this experiment. While Modified Centered CUSUMS performs the best in terms of the correct classification; Centered CUSUMS is the best in terms of estimated bias; SUMSRM is the best in terms of MSD. Nevertheless, the differences are small.

Table 4a: Comparison of the break detection performance among the Centered CUSUMS, the modified Centered CUSUMS and the SUMSRM statistic when the mean changes in the time series with the multiple breaks. The actual break location are : 100, 200, 300, 400, 500 and 600.

	Traditional Centered CUSUMS	Modified Centered CUSUMS	SUMSRM
mode	213	101	100
Correct Classification	13.5%	100%	100%
Estimated Bias	11.8	1	0
MSD	145.5	1	0

Table 4b: Comparison of the break detection performance among the Centered CUSUMS, the modified Centered CUSUMS and the SUMSRM statistic when the variance changes in the time series with the multiple breaks points. The actual break location are: 100, 200, 300, 400, 500 and 600.

	Traditional Centered CUSUMS	Modified Centered CUSUMS	SUMSRM
mode	600	100	100
Correct Classification	92.7%	95.2%	93.7%
Estimated Bias	-0.4	2	0.8
MSD	36	35	33

8. Conclusions

This paper makes the following contributions in the structural break detection and break location estimation.

- It provides a better understanding on how the pre-break data size and the decay rate of square residual affect the bias of the break location estimation in CUSUMS. Large pre-break data size and high decay rate of the square residual can effectively reduce the bias of the structural break location estimation.
- We identify a key weakness of the CUSUMS statistic: high bias of the break location estimation. We proposed a new statistic SUMSRM which has a low bias.
- Our experimental results show that the proposed SUMSRM statistic can effectively minimize the bias of the break location estimation and provide better structural break detection performance when there is a change of mean in the time series with single or multiple breaks. This result confirms our claim about the SUMSRM statistic - using the square deviation about the median and sliding window prediction residual can improve the break detection performance and eliminate the bias of the break location estimation.
- SUMSRM significantly outperforms the Centered CUSUMS and modified Centered CUSUMS, when there is a structural change of mean with a single break or multiple breaks in all four performance measures.

- Empirical experiments have demonstrated that when there is a structural change of variance in the time series with single or multiple breaks, the SUMSRM and modified Centered CUSUMS outperform the Centered CUSUMS. Both SUMSRM and modified Centered CUSUMS can improve the structural break detection by selecting appropriate post break data size.
- When there is a variance change in the time series with single or multiple breaks, SUMSRM performs a bit less satisfactory than the modified Centered CUSUMS. Sliding window approach cannot help SUMSRM to improve the structural break detection in this situation. As mentioned in sections 4b and 5.2, sliding window approach can help increase the decay rate. However, empirical evidence has shown that the decay rate has no relationship with the bias of the break location estimation when a variance change occurred in the time series. Besides, sliding window approach adopted by SUMSRM generates fewer pre-break prediction residuals and thus resulted in less satisfactory performance. This result can be supported and explained by our previous research analysis (Pang and Ting 2004) which clearly demonstrates that fewer pre-break data size (number of pre-break residual) will weaken the structural break detection performance, especially when the data set with small pre-break data size is used.

Appendix A: Proof of the proposed statistics SUMSRM having the Brownian asymptotes.

The notion below will be used in the following proof.

$$\begin{aligned} \xrightarrow{P} & : \text{Probability Convergence} \\ \xrightarrow{D} & : \text{Distribution Convergence} \\ W & : \text{Brownian Motion} \\ W^o & : \text{Brownian Bridge} \end{aligned}$$

Let $\xi_t = (w_t - M_t)^2 - E[(w_t - M_t)^2]$, $t = p+1, p+2, \dots, n$

Let $i = t-p$, then

$$\xi_i = (w_i - M_i)^2 - E[(w_i - M_i)^2], \quad i = 1, 2, 3, \dots, n-p$$

$$\begin{aligned} E(\xi) &= E[(w_i - M_i)^2 - E[(w_i - M_i)^2]] \\ &= E[(w_i - M_i)^2] - E[(w_i - M_i)^2] \\ &= 0 \end{aligned}$$

$$Var(\xi_i) = Var\{(w_i - M_i)^2 - E[(w_i - M_i)^2]\}$$

$$\begin{aligned} &= Var[(w_i - M_i)^2] - 0 \\ &= Var[(e_i - M_i)^2] \end{aligned}$$

when data size for median is large enough, $\xi \sim \chi^2$ distribution.

$$\text{Let } E[(e_i - M_i)^2] = \sigma_w^2$$

$$Var(\xi_i) = Var[(e_i - M_i)^2] = \sigma = 2\sigma_w^2 = 2E[(e_i - M_i)^2]$$

where $E[(e_i - M_i)^2]$ is the mean of square deviation from the median; e_i is the residual, $e \sim N(0, \sigma^2)$

$$\text{Let } X_{n-p}(t) =$$

$$\frac{1}{\sigma\sqrt{(n-p)}} \sum_{i=0}^{[(n-p)t]} \xi_i + ((n-p)t - [(n-p)t]) \frac{1}{\sigma\sqrt{(n-p)}} \xi_{[(n-p)t]+1}$$

where t , $0 < t \leq 1$

Based on the Donsker's Theorem (Billingsley 1968 Theorem 10.1), $X_n \xrightarrow{d} W$,

and based on the theorem 5.1 of Billingsley (1968), so $\{X_n(t) - tX_n(1)\} \xrightarrow{D} W^o$

then

$$\begin{aligned} X_{n-p}(t) - tX_{n-p}(1) &= \\ &= \frac{1}{\sigma\sqrt{(n-p)}} \sum_{i=0}^{[(n-p)t]} \xi_i + ((n-p)t - [(n-p)t]) \frac{1}{\sigma\sqrt{n-p}} \xi_{[(n-p)t]+1} - \\ &= \frac{1}{\sigma\sqrt{n-p}} \left(\sum_{i=0}^{[(n-p)t]} \xi_i - t \sum_{i=0}^{n-p} \xi_i \right) + \\ &= \frac{1}{\sigma\sqrt{n-p}} \left(\sum_{i=0}^{[(n-p)t]} \xi_i - \frac{(n-p)t}{n-p} \sum_{i=0}^{n-p} \xi_i \right) + \\ &= \frac{1}{\sigma\sqrt{n-p}} \left(\sum_{i=0}^{[(n-p)t]} \xi_i - \frac{(n-p)t}{n-p} \sum_{i=0}^{n-p} \xi_i \right) + \\ &= \frac{1}{\sigma\sqrt{n-p}} \left(\sum_{i=0}^{[(n-p)t]} \xi_i - \frac{(n-p)t}{n-p} \sum_{i=0}^{n-p} \xi_i \right) + \end{aligned}$$

Let the integer $r = (n-p)t$, then

$$\begin{aligned} X_{n-p}(t) - tX_{n-p}(1) &= \frac{1}{\sigma\sqrt{n-p}} \left(\sum_{i=0}^r \xi_i - \frac{r}{n-p} \sum_{i=0}^{n-p} \xi_i \right) + \\ &= \frac{1}{\sigma\sqrt{n-p}} \left(\sum_{i=0}^r \xi_i - \frac{r}{n-p} \sum_{i=0}^{n-p} \xi_i \right) + \end{aligned}$$

When $n \rightarrow \infty$,

$$\lim_{n \rightarrow \infty} ((n-p)t - [(n-p)t]) \frac{1}{\sigma \sqrt{n-p}} \xi_{n-p+1}^p \rightarrow 0$$

and

$$X_{n-p}(t) - tX_{n-p}(1) \rightarrow W^0$$

According to the theorem (Billingsley theorem 4.1, p.25), it shows that:

If $X_n \xrightarrow{D} X$ and distance $\rho(X_n, Y_n) \xrightarrow{p} 0$ then $Y_n \xrightarrow{D} X$

Therefore,

$$X_{n-p}(t) - tX_{n-p}(1) = \frac{1}{\sigma \sqrt{n-p}} \left(\sum_{i=0}^r \xi_i - \frac{r}{n-p} \sum_{i=0}^{n-p} \xi_i \right)$$

substitute $\xi_i = (e_i - \text{Median}_i)^2 - E[(e_i - \text{Median}_i)^2]$ and

$\sigma^2 = 2\sigma_\psi^2$ into above equation.

$$X_{n-p}(t) - tX_{n-p}(1) =$$

$$\begin{aligned} & \frac{1}{\sigma_\psi \sqrt{2(n-p)}} \left(\sum_{i=0}^r \{(e_i - \text{Median}_i)^2 - E[(e_i - \text{Median}_i)^2]\} - \right. \\ & \left. \frac{r}{n-p} \sum_{i=0}^{n-p} \{(e_i - \text{Median}_i)^2 - E[(e_i - \text{Median}_i)^2]\} \right) \\ & = \frac{1}{\sigma_\psi \sqrt{2(n-p)}} \left(\sum_{i=1}^r (e_i - \text{Median}_i)^2 - \frac{r}{n-p} \sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2 \right) \\ & = \frac{\sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2}{\sigma_\psi \sqrt{2(n-p)}} \left(\frac{\sum_{i=1}^r (e_i - \text{Median}_i)^2}{\sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2} - \frac{r}{n-p} \right) \\ & = \frac{\sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2}{\sigma_\psi \sqrt{2(n-p)}} \left(\frac{\sum_{i=1}^r (e_i - \text{Median}_i)^2}{\sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2} - \frac{r}{n-p} \right) \end{aligned}$$

When $n \rightarrow \infty$, $\frac{1}{n-p} \sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2 \rightarrow \sigma_\psi^2$

Therefore,

$$\sqrt{\frac{n-p}{2}} \left(\frac{\sum_{i=1}^r (e_i - \text{Median}_i)^2}{\sum_{i=1}^{n-p} (e_i - \text{Median}_i)^2} - \frac{r}{n-p} \right) \xrightarrow{D} W^0$$

And the distribution of $\sup_t |W_t^o|$ was given in equation (11.39) of Billingsley (1968)

$$P\{\sup_t |W_t^o| \leq b\} = 1 + 2 \sum_{k=1}^{\infty} (-1)^k e^{-2r^2 b^2}, \quad b > 0$$

Appendix B. Critical Value

signifcant level :0.1

data size \ win width	200	300	400	500	600	700	800	900	1000	2000	4000
20	1.21	1.23	1.24	1.25	1.25	1.26	1.26	1.26	1.26	1.28	1.27
30	1.18	1.21	1.22	1.23	1.23	1.24	1.24	1.24	1.25	1.26	1.26
40	1.17	1.19	1.21	1.21	1.21	1.23	1.22	1.23	1.23	1.24	1.25
50	1.15	1.18	1.19	1.19	1.21	1.21	1.22	1.22	1.22	1.24	1.24
60	1.15	1.18	1.19	1.19	1.2	1.21	1.21	1.22	1.22	1.23	1.24
70	1.15	1.17	1.17	1.19	1.2	1.2	1.21	1.22	1.21	1.23	1.24
80	1.15	1.17	1.17	1.19	1.2	1.19	1.21	1.21	1.21	1.23	1.23
90	1.14	1.17	1.17	1.18	1.19	1.19	1.21	1.21	1.21	1.23	1.23
100	1.14	1.16	1.17	1.18	1.19	1.19	1.2	1.21	1.21	1.22	1.22
200	—	1.15	1.17	1.18	1.18	1.18	1.18	1.2	1.2	1.21	1.21

signifcant level :0.05

data size \ win width	200	300	400	500	600	700	800	900	1000	2000	4000
20	1.35	1.37	1.38	1.39	1.38	1.4	1.4	1.41	1.41	1.43	1.42
30	1.32	1.35	1.36	1.36	1.37	1.38	1.37	1.39	1.38	1.4	1.39
40	1.31	1.33	1.35	1.35	1.35	1.37	1.37	1.37	1.37	1.38	1.39
50	1.28	1.31	1.32	1.33	1.34	1.34	1.35	1.36	1.36	1.37	1.38
60	1.28	1.31	1.31	1.33	1.33	1.34	1.34	1.36	1.36	1.37	1.37
70	1.29	1.31	1.31	1.32	1.33	1.33	1.34	1.36	1.35	1.37	1.37
80	1.27	1.31	1.3	1.32	1.33	1.33	1.34	1.36	1.35	1.36	1.37
90	1.27	1.31	1.3	1.32	1.32	1.33	1.34	1.36	1.34	1.36	1.36
100	1.27	1.29	1.3	1.31	1.32	1.32	1.34	1.35	1.34	1.36	1.36
200	—	1.29	1.3	1.3	1.31	1.31	1.33	1.34	1.33	1.35	1.35

signifcant level :0.01

data size \ win width	200	300	400	500	600	700	800	900	1000	2000	4000
20	1.64	1.65	1.67	1.67	1.67	1.7	1.68	1.71	1.68	1.71	1.7
30	1.6	1.63	1.65	1.64	1.62	1.66	1.65	1.65	1.65	1.68	1.7
40	1.58	1.61	1.63	1.63	1.61	1.64	1.64	1.65	1.64	1.66	1.67
50	1.57	1.6	1.6	1.6	1.61	1.61	1.62	1.63	1.63	1.66	1.66
60	1.55	1.59	1.6	1.6	1.61	1.61	1.62	1.61	1.64	1.66	1.65
70	1.55	1.58	1.6	1.59	1.6	1.61	1.61	1.6	1.64	1.64	1.64
80	1.55	1.58	1.58	1.59	1.6	1.61	1.61	1.6	1.62	1.64	1.64
90	1.54	1.57	1.58	1.58	1.59	1.6	1.61	1.6	1.63	1.64	1.64
100	1.52	1.57	1.58	1.59	1.6	1.61	1.61	1.6	1.62	1.64	1.64
200	—	1.54	1.57	1.57	1.59	1.57	1.61	1.6	1.59	1.62	1.62

References

- [1] Billingsley, P. (1968), Convergency of Probability Measures, New York : John Wiley.
- [2] Brown, R.L., Durbin, J., and Evans, J.M. (1975) "Techniques for Testing the constancy of Regression Relationship over Time", Journal of Royal Statistical Society, Series B, 37, 149-192.

- [3] Chow, G. (1960), Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, Vol. 28, No. 3, 591-605
- [4] Inclan, C. and Tiao, G. C.. (1994) "Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance", *Journal of the American Statistical Association*, September 1994. Vol. 89, no. 427, 913-923.
- [5] Krämer, Walter and Schotman, Peter (1992) "Range vs Maximum in OLS-based version of CUSUM test", *Economics letter* 40, pp 379-381
- [6] Pang, K. P. and Ting, K. M. (2003) "Improving Time series prediction by data selection" *International Conference on Computational Intelligence for Modeling, Control & Automation*, February 2003, 803-813
- [7] Pang, K. P. and Ting, K. M. (2004) "Improving the Centered CUSUMS statistic for Structural Break Detection in Time Series", *Forthcoming of 17th Australian Joint conference on Artificial Intelligence*.
- [8] Pesaran, H. and Timmermann, A. (2002) "Market timing and Return Prediction under Model instability", *Journal of Empirical Finance*, December 2002. Vol. 9, 495-510
- [9] Chu, Chia Shang J., Hornik, Kurt and Kuan, Chung Ming (1995) "MOSUM Tests for Parameter constancy", *Biometrika* 82, 3 pp. 603-617
- [10] Pesaran, H. and Timmermann, A., (2003) "How Costly is it to Ignore Breaks when Forecasting the Direction of a Time Series?," *Cambridge Working Papers in Economics 0306*, Department of Applied Economics, University of Cambridge.

Markov models for identification of significant episodes

Robert Gwadera *

Purdue University

Department of Computer Sciences

gwadera@cs.purdue.edu

Mikhail Atallah †

Purdue University

Department of Computer Sciences

mja@cs.purdue.edu

Wojciech Szpankowski ‡

Purdue University

Department of Computer Sciences

spa@cs.purdue.edu

Abstract

We propose a new method for a reliable identification of significant sequential episodes occurring *within a window of size w* in an event sequence modeled by a Markov source. As a measure of significance we use $\Omega^\exists(n, w)$, the number of windows containing the episode as a subsequence. We prove that $\Omega^\exists(n, w)$ is a sum of a φ -mixing sequence of random variables and therefore obeys the central limit theorem. This leads us to a computational formula for a threshold to identify significant episodes. The novelty of our method for Markov source stems from the fact that, instead of scoring the whole sequence using a Markov model, we compute the expected value of $\Omega^\exists(n, w)$ and its variance in order to estimate the threshold and compare it to the observed $\Omega^\exists(n, w)$. Since performance of the method critically depends on the model structure and parameters, we argue that variable-length Markov models of event streams are superior to fixed-length Markov models. We chose DNA sequences as event sources in experiments, and compared the performance of fixed-length Markov models with interpolated Markov models. This paper is an extension of our previous work in [8, 1] where we considered the problem of the reliable detection of significant episodes for memoryless sources.

Keywords: frequent episode mining, probabilistic models

1 Introduction

1.1 Episode mining

Mining episodes was introduced in [11], where the problem of finding frequent episodes in event sequences was defined. An episode was defined as a partially ordered collection of events that occur as a subsequence in a window of a given size in an event stream. The notion of an occurrence is as a subsequence rather than as a substring (that is, contiguity is not required), a requirement dictated by practical considerations because (for example) an “interesting” (e.g., suspicious) sequence of events need not be contiguous in the event stream. An arbitrary episode can be abstractly represented as a directed acyclic graph (DAG), where nodes correspond to events and directed edges define precedence among the events in the episode. Formally, such a graph defines a set of episodes whose members correspond to all distinct paths from the start vertex to end-vertices. We distinguish three types of episodes.

1. A *serial episode* is a sequence of events that occurs in the specified order. In the graph representation a serial episode corresponds to a single path from the first event of the episode to the last one.
2. A *parallel episode* is an unordered collection of events. In the graph representation a parallel episode corresponds to a single node containing all events of the episode. Formally, a parallel episode corresponds to the set of all permutations of symbols of the episode.

*The work of this author was supported by the NSF Grant CCR-0208709, and NIH grant R01 GM068959-01.

†Portions of this author's work were supported by Grants EIA-9903545, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

‡The work of this author was supported by the NSF Grant CCR-0208709, NIH grant R01 GM068959-01 and AFOSR Grant FA 8655-04-1-3074.

3. A *composite episode* corresponds to an arbitrary DAG built from an event and/or an episode by a serial and/or a parallel composition.

In the episode mining we shift the sliding window of a given size w n consecutive events in an event stream T and count the number of windows in which the episode occurred at least once as a subsequence. Note that the same episode may be present in several consecutive windows but within a particular window we count one occurrence even if there are many instances of it in that particular window. Given a window size w and an event sequence T , an episode was defined as *frequent* if its frequency, defined as the fraction of windows in which it occurred at least once, was more than a given frequency threshold τ .

In our work we are interested in episodes that are “significant” (e.g., anomalous); note that the frequency of occurrence is not enough to determine significance (e.g., an infrequent episode might have more significance than a frequent one, depending on the probabilistic characteristics of the event stream).

1.2 Previous work

In our previous papers [8, 1] we assumed that the event sequence T was generated by the Bernoulli (memoryless) source and showed how to compute the threshold τ as well as how to design the window size w such that the discovered frequent episodes are statistically significant. Observe that, for an appropriately large window size any episode will almost surely occur in every window because the probability of its existence in the window of size w $P^\exists(w)$, estimated as a fraction of windows in which the episode occurred, will be close to one. Furthermore, for an appropriately low frequency threshold any episode may be found to be frequent.

Paper [8] considered serial episodes and [1] considered sets of episodes including the special case of the parallel episode. In order to derive the threshold we analyzed $\Omega^\exists(n, w)$, the number of windows of length w , that contain an episode as a subsequence in an event sequence T after n shifts of the sliding window. Using the fact that $\Omega^\exists(n, w)$ is a sum of $w - 1$ dependent random variables we proved that appropriately normalized $\Omega^\exists(n, w)$ is normally distributed, where clearly the expected value $\mathbf{E}[\Omega^\exists(n, w)] = nP^\exists(w)$ and $P^\exists(w)$ is the probability that the episode occurs at least once in a window of length w in an event sequence T over an alphabet \mathcal{A} . We also showed that the variance $\mathbf{Var}[\Omega^\exists(n, w)] \leq cn [P^\exists(w) - (P^\exists(w))^2]$ for $c > 0$. Given the Bernoulli model of an event source, we presented the *upper threshold* for detecting significant *over-*

represented episodes $\tau_u(w) = P^\exists(w) + \frac{b\sqrt{\mathbf{Var}[\Omega^\exists(n, w)]}}{n}$ such that $P\left(\frac{\Omega^\exists(n, w)}{n} > \tau_u(w)\right) \leq \beta(b)$. That is, the

probability that the frequency of the episode $\frac{\Omega^\exists(n, w)}{n}$ is greater than the threshold $\tau_u(w)$ is smaller than $\beta(b)$, i.e., the episode is significant with probabilistic guarantee $1 - \beta(b)$. We also analogously defined the *lower threshold* $\tau_\ell(w)$ for detecting significant *under-represented* episodes such that $P\left(\frac{\Omega^\exists(n, w)}{n} < \tau_\ell(w)\right) \leq$

$\alpha(a)$. The quantity $\frac{\Omega^\exists(n, w)}{n}$ is an estimator of $P^\exists(w)$ denoted $P_e^\exists(w)$. While developing the formula for $P^\exists(w)$ we found a formula for the set of all distinct windows $\mathcal{W}^\exists(w)$ of length w containing the serial episode S of length m at least once as a subsequence. The importance of $\mathcal{W}^\exists(w)$ stems from the fact that $P^\exists(w) = \sum_{x \in \mathcal{W}^\exists(w)} P(x)$ for a Markov model of an arbitrary order including the 0-order (Bernoulli), where $P(x)$ is the probability of x as a string of symbols of length w in a given model. The advantage of the Bernoulli model versus the 1-order Markov or higher is that for the Bernoulli model $P^\exists(w)$ can be computed efficiently exploiting the structure of $\mathcal{W}^\exists(w)$ and the fact that the model requires only $|\mathcal{A}|$ probabilities of symbols of the alphabet \mathcal{A} . Using generating functions and complex asymptotics we presented an asymptotic approximation of $P^\exists(w)$, which is of the form $P^\exists(w) = 1 - \Theta(\rho^w)$ for large w and $0 < \rho < 1$. We provided fast dynamic programming algorithms for computing $P^\exists(w)$ for a serial episode and for an arbitrary set of episodes. In [8] we mined two apparently non-memoryless sources (the English alphabet and the web access data) and showed that, even for these cases, $P^\exists(w)$ closely approximated the actual $P_e^\exists(w)$, which demonstrated that the memoryless assumption did not limit the practical usefulness of the formula. In [1] we mined a large database of *Wal-Mart* transactions for sets of episodes. We also showed that the threshold mechanism indeed provides a sharp detection of significant episodes by continually injecting some episodes until they exceeded the threshold.

1.3 Present work

The present paper extends our previous work to the case of Markov sources that are more applicable and flexible than memoryless sources. The formula for the threshold for Markov models is the same as for the Bernoulli model, the difference is in using conditional probabilities to compute $P^\exists(w)$ and $\mathbf{Var}[\Omega^\exists(n, w)]$. Furthermore, we cannot use the efficient dynamic programming algorithm for computing $P^\exists(w)$ that was designed for the Bernoulli model because in Bernoulli model the probability of a symbol does not depend on its context. There

were three main new challenges that we faced and resolved in this extension. The first was theoretical: in order to use the threshold formula we had to prove that $\Omega^\exists(n, w)$ is a sum of a φ -mixing sequence of random variables meaning that the distant future is practically independent of the present and past and therefore $\Omega^\exists(n, w)$ satisfies the central limit theorem. The second challenge was algorithmic: we had to provide an algorithm for computing $P^\exists(w)$ using conditional probabilities. Finally the third challenge was to select a Markov model structure and a method of parameter estimation to ensure that the prediction of the model is accurate: we suggested variable-length Markov models and in particular, in experiments conducted on DNA sequences, we focused on the interpolated Markov model.

Given an event sequence T , we need to choose an optimal Markov model for the data and given that model we need to choose an optimal method for parameter estimation for our method for detecting significant episodes. Higher order models describe data more accurately but increase the number of excessive parameters. Using a fixed-length Markov model of order k with $|\mathcal{A}|^{k+1}$ parameters can be inefficient since for real-life data the actual memory length varies. The number of model parameters can be significantly reduced by merging equivalent states (contexts of length k) that have identical conditional probabilities. Such reduced models, first considered in [15] were termed the *variable-length Markov chains/models* or *tree models* [20, 12] since they can be conveniently represented with a tree structure. Thus, the advantage of variable-length Markov models over fixed-length models is that they efficiently capture the redundancies that are typical for real-life data. Because of that fact they are particularly well suited for our method for detecting significant episodes since we can efficiently build such models based on our empirical, expert, knowledge of the source of events. For example: sometimes we know that some symbols occur only in *n-grams* (strings of n ordered elements) and using a full fixed-length Markov model is too excessive and may drastically limit the usefulness of our method while the Bernoulli model is too inaccurate to capture the dependency of symbols in the *n-grams*.

Formally, given an alphabet $\mathcal{A} = \{a_1, a_2, \dots, a_{|\mathcal{A}|}\}$ a k -order variable-length Markov model can be represented as a *context tree* [15]. Let $\mathcal{C} = \{c_1, c_2, \dots, c_{|\mathcal{S}|}\}$ be the set of contexts in a k -order variable-length Markov model where $c_j = c_j[|c_j|] \dots, c_j[2], c_j[1]$ is the j -th context of length $1 \leq |c_j| \leq k$, written as a time-reversed string, where $c_j \in \mathcal{A}^{|c_j|}$. A context tree is a suffix tree, built from the contexts in \mathcal{C} that is called a *model*. The set of parameters of a k -order variable-length Markov model is defined as $\Theta = \{\theta_{1,1}, \theta_{1,2}, \dots, \theta_{|\mathcal{A}|,|\mathcal{C}|}\}$ where

$\theta_{i,j} = P(a_i|c_j)$ is the conditional probability of generating the symbol a_i given the context c_j subject to $\sum_{i=1}^{|\mathcal{A}|} P(a_i|c_j) = 1$. Thus, \mathcal{C} is the set of states and Θ is the set of transition probabilities in a k -order variable-length Markov chain. In [15] the *context* algorithm was presented for estimation of the minimal state space and the parameters of a variable-length Markov model. In [16] minimization of stochastic complexity of a source in a given model was suggested as a criterion for selecting an optimal (minimal) model. In [21] the *context tree weighting* algorithm was proposed for computing probability $P(T)$ of a Markov source T using an average over all possible models having orders less than a given order k . However the optimal model selection problem for the class of variable-length Markov models is still not well understood. Furthermore the parameter estimation from sparse data presents another problem.

Therefore we turn our attention to a class of variable-length Markov models called interpolated Markov model (IMM). IMM does not optimize the state space but builds a variable-length Markov model implicitly as a result of parameter estimation from sparse data. Given an alphabet of size $|\mathcal{A}|$, we could model the event stream T by a k -order fixed-length Markov model with $|\mathcal{A}|^{k+1}$ conditional probabilities to be estimated from the training data. If we are not interested in optimality of the model then a higher-order model should always do at least as well as, and frequently better than a lower-order model. In practice, when using a k -order model, if the training sequence is of length N then there are only $N - k$ strings of size $k + 1$ available to estimate $|\mathcal{A}|^{k+1}$ conditional probabilities and frequencies of some of the $|\mathcal{A}|^k$ context strings become too small or even zero. Deriving a model of too high order from such sparse data will lead to over-fitting. More formally, let $c_j = c_j[k] \dots, c_j[2], c_j[1]$ be the j -th context of length k , written as a time-reversed string, in a k -order fixed-length Markov model, where $c_i \in \mathcal{A}^k$. We estimate the conditional probabilities $P(a_i|c_j)$ using the maximum likelihood (ML) estimate given by the following formula $P(a_i|c_j) = \frac{n(c_j a_i)}{\sum_{i=1}^{|\mathcal{A}|} n(c_j a_i)}$, where $n(c_j a_i)$ is the frequency of the string $c_j a_i$ observed in the training set. Notice that even if the training set is too short to accurately estimate all probabilities, for some probabilities $P(a_i|c_j)$ the number of occurrences of the string $c_j a_i$ will be sufficient and should be accepted by the model. The problem of parameter estimation of Markov models from sparse data is known as *smoothing* and has been widely discussed in the literature on language modeling [6]. The smoothing is a technique for adjusting the maximum likelihood estimates of probabilities to produce more accurate probabilities. The

name smoothing comes from the fact that these methods tend to make the probabilities more uniform, by adjusting low probabilities upward and higher probabilities downward. Not only do smoothing methods generally prevent zero probabilities, but they also improve the accuracy of the model as the whole. Whenever a probability is estimated from a fewer counts, smoothing has the potential to significantly improve estimation.

Techniques as *back-off* [10] and *interpolation* [13] have been implemented to deal with sparse data. The back-off model backs off to lower order models depending on counts of respective contexts. The interpolated model is a Markov chain with a new structure, where a conditional probability of order k is a combination of equal and lower order probabilities weighted by interpolation parameters, giving high weight to probability estimates corresponding to high frequency contexts and lower weight to estimates corresponding to low frequency contexts. A further extension of IMM is *interpolated context model* (ICM) [7]. While in IMM we estimate the probability $P(a_i|c_j)$ of a symbol a_i based on variable length contexts immediately preceding a_i the ICM is more flexible and general by allowing to choose any contexts (not just those adjacent to a_i).

In this paper we focus on the interpolated Markov model and we compare its performance with fixed-length Markov models for detecting significant episodes. We use the notation $c_j[1 : n]$ for $n = k, k-1, \dots, 1$ to denote a suffix of length n of context c_j of length k and we omit the notation for $n = k$, i.e., we write c_j instead of $c_j[1 : k]$ in cases where k is implied. We are interested in Markov models that define conditional probabilities $P(a_i|c_j)$ as a linear combination of conditional probabilities corresponding to suffixes of c_j . The following recursion defines a value of the interpolated conditional probability in IMM:

$$P_{IMM}(a_i|c_j) = \lambda(c_j) \cdot P(a_i|c_j) + (1 - \lambda(c_j)) \cdot P_{IMM}(a_i|c_j[1 : k-1]),$$

where $0 \leq \lambda(c_j) \leq 1$ and $P(a_i|c_j)$ is the probability estimate using the maximum likelihood (ML) estimate from the training data. For contexts c_j not observed in the training data, i.e., if $n(c_j a_i) = 0$ then we set $P(a_i|c_j) = P(a_i|c_j[1 : n])$ for $n = \max_{1 \leq n \leq k} \{n | n(c_j[1 : n], a_i) > 0\}$ and this is exactly the place in the computation of parameters of an IMM where a variable-length Model is being implicitly built. The value of the parameter $\lambda(c_j)$ can be interpreted in many ways. Following [17] interpretation of the parameter depends on the following interpretations of the IMM:

- Context model interpretation: the parameters combine the predictions from contexts of varying

length. Since longer contexts support stronger predictions and shorter contexts have more accurate statistics the interpolation of the predictions of different context lengths results in more accurate prediction than from a fixed context.

- State model interpretation: the parameters are hidden transitions from a higher order Markov model to a lower Markov model where the interpolation parameters model our beliefs about how much of the past is necessary to predict a state transition in an underlying Markov source of unknown order.
- Nonuniform model interpretation: the parameters are beliefs about conditional independence with probability $(1 - \lambda(c_j))$ that the future does not depend on $c_j[k]$.

In general if the frequency of context c_j is sufficiently high, the value of $\lambda(c_j)$ is close to 1. In the opposite case $\lambda(c_j)$ is close to zero and the interpolation probability $P_{IMM}(a_i|c_j)$ gains more from $P_{IMM}(a_i|c_j[1 : k-1])$. However the problem of finding interpolation parameters is still more of an art than an exact science. In our experiments we assumed a given order of IMM and used a modification of the method based on χ^2 -test introduced in [18].

We conduct our experiments on genomic data represented as strings of nucleotide symbols over the alphabet $\mathcal{A} = \{A, C, G, T\}$. Markov models of DNA sequences have frequently been used in gene finding algorithms [18], where the interest was in finding strings of symbols instead of subsequences in the form of episodes. The novelty of our approach is that we treat the genomic sequence as a stream of symbols generated by a Markov model of an unknown order and we do not consider any biological structures as coding/non-coding regions in the DNA. Furthermore we do not score the testing sequence using a trained Markov model, as in the work on gene discovery, in order to determine whether the sequence has been generated by the model. Adapting the method of scoring the sequence for episode discovery would mean training a separate Markov model for every combination of window length and episodes type. Note that in the episode framework we consider $\Omega^\exists(n, w)$ the number of windows containing an episode as a subsequence, which is a function of a Markov chain rather than a well defined structure (coding/non-coding regions) of the sequence as in the gene discovery methods. Therefore in our method for the reliable detection of significant episodes we use a Markov model only to compute the expected value and variance of $\Omega^\exists(n, w)$ needed for the threshold computation. Because of the sequential nature of Markov sources we consider only serial episodes while using Markov models. We do not test

the threshold $\tau_u(w)$ directly in experiments by computing its value and simulating occurrences of significant episode as in [8, 1] because we already showed in [8, 1] that the accuracy of the threshold is determined by prediction accuracy of the formula for $P^\exists(w)$. Therefore we test the threshold indirectly by focusing on the predictive performance of the formula for $P^\exists(w)$ for Markov models.

Perhaps the most intriguing question is whether we can improve our detection method on DNA data in terms of accuracy by employing a Markov model rather than the Bernoulli model. As we will see in experiments the answer to this question is affirmative.

The paper is organized as follows. Section 2 presents example applications of our theory. Section 3 presents our main results containing theoretical foundation. Section 4 contains experimental results demonstrating the applicability of the derived formulas.

2 Applications of our method

There are multiple uses for the theory we developed. Given a probabilistic model of an event sequence T , example applications of our method include:

- *Designing the sliding window size:* given a priori knowledge of episodes of interest, we can select an appropriate window size w such that the discovered episodes are meaningful.
- *Validation of the sliding window size:* given a window size w and an episode (e.g., a frequent episode) discovered in the event stream T , we can validate the window size w for the discovered episode.
- *Identification of significant episodes:* given a window size w and an episode (e.g., a frequent episode) discovered in the event stream T , we can determine whether the episode is significant.
- *Episode ranking:* given a collection of episodes (e.g., all frequent episodes) discovered in the event stream T , we can rank the episodes with respect to their significance.

Given a probabilistic model of an event sequence T and an episode with observed frequency $\frac{\Omega^\exists(n,w)}{n}$, the episode can be classified using the upper threshold $\tau_u(w)$ and the lower threshold $\tau_\ell(w)$ as follows:

- *significant:*
 - if $\frac{\Omega^\exists(n,w)}{n} > \tau_u(w)$ for over-represented episodes

- if $\frac{\Omega^\exists(n,w)}{n} < \tau_\ell(w)$ for under-represented episodes

- *normal:* if $\frac{\Omega^\exists(n,w)}{n} \in [\tau_\ell(w), \tau_u(w)]$
- *meaningless:* if $\frac{\Omega^\exists(n,w)}{n} \approx 1$ and $P^\exists(w) \approx 1$ meaning that the window size w is too large.

3 Analytical results

3.1 Definition of the problem of identification of significant episodes

For clarity of the presentation we analyze only the case of a single serial episode $S = S[1]S[2] \dots S[m]$ of length m but the results can be generalized to an arbitrary set of serial episodes. Of course we do not analyze parallel episodes since they are unordered sequences.

The problem of identification of significant episodes in a Markov source T can be stated as follows.

Given:

- an alphabet $\mathcal{A} = \{a_1, a_2, \dots, a_{|\mathcal{A}|}\}$
- a k -order variable-length Markov model of the source T with parameters represented as follows:
 - $\mathcal{C} = \{c_1, c_2, \dots, c_{|\mathcal{S}|}\}$ is the set of contexts where $c_j = c_j[|c_j|] \dots c_j[2], c_j[1]$ is the j -th context of length $1 \leq |c_j| \leq k$, written as a time-reversed string and $c_j \in \mathcal{A}^{|c_j|}$.
 - $\Theta = \{\theta_{1,1}, \theta_{1,2}, \dots, \theta_{|\mathcal{A}|,|\mathcal{C}|}\}$ is the set of parameters where $\theta_{i,j} = P(a_i|c_j)$ is the conditional probability of generating the symbol a_i given the context c_j subject to $\sum_{i=1}^{|\mathcal{A}|} P(a_i|c_j) = 1$.
- $\Omega^\exists(n, w)$, the observed number of windows of length w containing at least one occurrence of a serial episode $S = S[1]S[2] \dots S[m]$ after n shifts of the window
- a level $\beta(b)$ (e.g., $\beta(b) = 10^{-5}$),

is the observed episode S significant?

In Section 3.2 we prove that $\Omega^\exists(n, w)$ is normally distributed (Theorem 1). This will allow us to compute the threshold as follows

$$(1) \quad \begin{cases} \tau_u(w) &= P^\exists(w) + \frac{b\sqrt{\text{Var}[\Omega^\exists(n,w)]}}{n} \\ \beta(b) &= \frac{1}{\sqrt{2\pi}} \int_b^\infty e^{-\frac{t^2}{2}} dt \end{cases}$$

where $\text{Var}[\Omega^\exists(n, w)] \leq [n + (2n - w)(w - 1)][P^\exists(w) - (P^\exists(w))^2]$.

Thus, if $\frac{\Omega^\exists(n,w)}{n} > \tau_u(w)$ then episode S is significant with probabilistic guarantee $1 - \beta(b)$, i.e., $P\left(\frac{\Omega^\exists(n,w)}{n} > \tau_u(w)\right) \leq \beta(b)$.

In section 3.3 we provide an algorithm for computing $P^\exists(w)$.

3.2 Central limit for $\Omega^\exists(n, w)$

In this section we show that $\Omega^\exists(n, w)$ is a sum of φ_n -mixing sequence of random variables and therefore it satisfies the central limit theorem even though independence of the random variables summing to $\Omega^\exists(n, w)$ is clearly violated.

We consider a stationary and ergodic infinite k -order Markov source T .

Definition 1 A k -order Markov source is a sequence of random variables t_1, t_2, \dots over an alphabet $\mathcal{A} = \{a_1, a_2, \dots, a_{|\mathcal{A}|}\}$ with the Markov property

$$P(t_1, \dots, t_n) = P(t_1, \dots, t_k) \cdot \prod_{i=k+1}^n P(t_i | t_{i-1}, \dots, t_{i-k})$$

where k is the minimum integer for which the Markov property holds.

A k -order fixed-length Markov source can be described by a finite state Markov chain.

Definition 2 A finite state k -order Markov chain is a sequence of random variables Q_1, Q_2, \dots , where $Q_i = (t_{i-1}, t_{i-2}, \dots, t_{i-k})$ is a symbol from a finite state alphabet \mathcal{Q} of cardinality $|\mathcal{Q}| = \mathcal{A}^k$ and there is a transition from state $Q_i = (t_{i-1}, t_{i-2}, \dots, t_{i-k})$ to state $Q_j = (t_i, t_{i-1}, \dots, t_{i-k+1})$ with transition probability $P(t_i | t_{i-1}, \dots, t_{i-k})$.

We use the Markov chain model of a k -order Markov source in the proof that $\Omega^\exists(n, w)$ is a sequence of φ -mixing sequence of random variables.

A formal definition of a φ -mixing sequence is as follows.

Definition 3 Let $\varphi_1, \varphi_2, \dots$ be a sequence of numbers such that $\varphi_n \rightarrow 0$. A stationary sequence of random variables X_1, X_2, \dots, X_n is φ -mixing if $|P(E_2 | E_1) - P(E_2)| \leq \varphi_n$ for every j and $E_1 \in (X_1, \dots, X_j)$ and every k and $E_2 \in (X_{j+n}, \dots, X_{j+n+k})$.

In this definition, E_1 is an event that depends only on X_1, \dots, X_j , and E_2 is an event that depends only on $X_{j+n}, \dots, X_{j+n+k}$. The condition requires that E_1 and E_2 are almost independent in the sense that $|P(E_2 | E_1) - P(E_2)|$ is small for large n .

Now we derive the normal limiting distribution of $\Omega^\exists(n, w)$. Observe that

$$\Omega^\exists(n, w) = \sum_{i=1}^n I_i^\exists(w)$$

where

$$I_i^\exists(w) = \begin{cases} 1 & \text{the episode } S \text{ occurs at least once as a} \\ & \text{subsequence in the window ending at} \\ & \text{position } i \text{ in } T; \\ 0 & \text{otherwise,} \end{cases}$$

where i is the relative position with respect to the first position ($i = 1$). Thus, we easily have $\mathbf{E}[I_i^\exists(w)] = P^\exists(w)$, $\mathbf{Var}[I_i^\exists(w)] = P^\exists(w) - (P^\exists(w))^2$ and $\mathbf{E}[\Omega^\exists(n, w)] = nP^\exists(w)$.

Thus, the independence of the sequence of $I_i^\exists(w)$ for $1 \leq i \leq n$ is violated twofold since:

1. observation windows overlap within $w - 1$ events meaning $|P(I_{i+k}^\exists(w) = 1 | I_i^\exists(w) = 1) - P(I_{i+k}^\exists(w) = 1)| \neq 0$ for $1 \leq k \leq w - 1$
2. the event sequence T is not memoryless meaning $|P(I_{i+k}^\exists(w) = 1 | I_i^\exists(w) = 1) - P(I_{i+k}^\exists(w) = 1)| \neq 0$ for $k > w - 1$.

For Markov sources the central limit theorem holds as long as $|P(I_{i+k}^\exists(w) = 1 | I_i^\exists(w) = 1) - P(I_{i+k}^\exists(w) = 1)| \rightarrow 0$ as $k \rightarrow \infty$, i.e., I_{i+k}^\exists and I_i^\exists are practically independent as k becomes large meaning the sequence $I_1^\exists(w), I_2^\exists(w), \dots, I_n^\exists(w)$ is φ -mixing.

Notice that according to the definition of a φ -mixing sequence the $w - 1$ -dependent sequence $I_1^\exists(w), I_2^\exists(w), \dots, I_n^\exists(w)$ is φ -mixing with $\varphi_n = 0$ for $|i - j| > w - 1$.

Given a k -order Markov event source T , $I_i^\exists(w)$ is a function of the corresponding Markov chain of order k . Therefore to prove that $I_1^\exists(w), I_2^\exists(w), \dots, I_n^\exists(w)$ is φ -mixing we use the fact that the Markov chain is φ -mixing. According to [3], let Y_1, Y_2, \dots, Y_n be a Markov chain with finite state space and positive transition probabilities p_{ij} . Let $X_i = f(Y_i)$, where f is some real function of the state space. If the initial probabilities are stationary then Y_1, Y_2, \dots, Y_n is stationary. Moreover $|p_{ij}^n - p_j| \leq \rho^n$ where $\rho < 1$. Let r be the number of states. Then according to [3] Example 27.6 X_1, X_2, \dots, X_n is φ_n -mixing with $\varphi_n = r\rho^n$.

Based on Theorem 27.5 in [3] and the fact that $I_1^\exists(w), I_2^\exists(w), \dots, I_n^\exists(w)$ is φ -mixing the central limit theorem holds for $\Omega^\exists(n, w)$ in a k -order Markov source.

Theorem 1 The random variable $\Omega^\exists(n, w)$ obeys the Central Limit Theorem in the sense that its distribution

is asymptotically normal, for $a, b = O(1)$ we have

$$\lim_{n \rightarrow \infty} P \left\{ a \leq \frac{\Omega^\exists(n, w) - \mathbf{E}[\Omega^\exists(n, w)]}{\sqrt{\mathbf{Var}[\Omega^\exists(n, w)]}} \leq b \right\} = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-\frac{t^2}{2}} dt$$

for fixed w .

Theorem 1 leads to the Formula (1) for the threshold.

3.3 Algorithm for computing $P^\exists(w)$

In this section we present an algorithm for computing $P^\exists(w)$ for Markov models.

The probability of existence of an episode can be expressed as follows

$$(2) \quad P^\exists(w) = \sum_{x \in \mathcal{W}^\exists(w)} P(x).$$

where $\mathcal{W}^\exists(w)$ is the set of all distinct windows of length w containing the episode as a subsequence. Let x_i be the i -th symbol of a window $x \in \mathcal{W}^\exists(w)$ then the probability of the window can be computed as follows

$$P(x) = P(x_0)P(x_1|x_0) \dots P(x_{k-1}|x_{k-2} \dots x_0) \cdot \prod_{i=k}^{w-1} P(x_i|x_{i-1} \dots x_{i-k})$$

where k is the order of the model. In our papers [8, 1] we gave formulas for $\mathcal{W}^\exists(w)$ for a single serial episode $S = S[1]S[2] \dots S[m]$ and for an arbitrary set of serial episodes $\mathbf{S} = \{S_1, S_2, \dots S_m\}$ respectively. In particular, $\mathcal{W}^\exists(w)$ for a single serial episode can be enumerated as follows

$$\mathcal{W}^\exists(w) = \bigcup_{\sum_{k=1}^{m+1} n_k = w-m} \overline{S[1]}^{n_1} \times S[1] \times \dots \times \overline{S[m]}^{n_m} \times S[m] \times \mathcal{A}^{n_{m+1}},$$

where \bar{a} denotes $\mathcal{A} - a$ for $a \in \mathcal{A}$. Using formula (3) directly is computationally very expensive. Furthermore computing $P(x)$ for every window x independently would be inefficient because many windows share the same prefix. Therefore we propose a computational method where we enumerate the windows according to the depth-first traversal of a trie build from the members of $\mathcal{W}^\exists(w)$ without the trailing $\mathcal{A}^{n_{m+1}}$ that contributes a factor of 1 to the computation of the probability. The idea of this method is that the probability of each distinct prefix of the set of windows $\mathcal{W}^\exists(w)$ is computed once. An example of such a trie for $S = abc$ and $w = 4$ is shown in Figure 1.

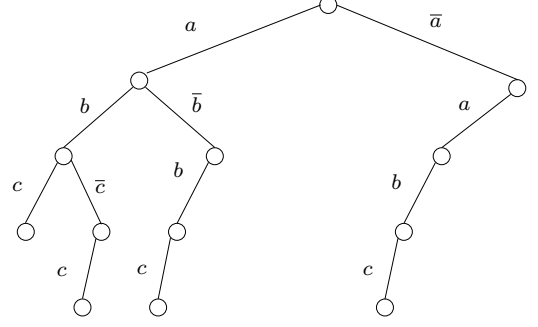


Figure 1: A trie for $\mathcal{W}^\exists(3)$ the set of windows of length $w = 4$ containing $S = abc$ as a subsequence

4 Experimental results

The ultimate measure of a statistical model is its predictive performance in the domain of interest. Therefore in experiments we compare the actual (observed) $P_e^\exists(w) = \frac{\Omega^\exists(n, w)}{n}$ value with $P^\exists(w)$ obtained using a trained model. As we stated in the introduction, we consider only a serial episode S of length m in experiments. We used an algorithm for finding occurrences of a serial episode to find $\Omega^\exists(n, w)$. To evaluate the performance of a model we used the following distance measure between two functions

$$(3) \quad d = \left[\frac{1}{r} \sum_{i=1}^r \frac{|P_e^\exists(w_i) - P^\exists(w_i)|}{P_e^\exists(w_i)} \right] 100\%$$

where $w_1 < w_2 < \dots w_r$ are the tested window sizes. We tested the prediction of Markov models on the following genomic sequences: *Haemophilus influenzae* of length 1,830,025, *Helicobacter pylori* of length 1,667,826 and *Human chromosome 22* two segments of length 234,227 and 3,661,561 respectively. We estimated the conditional probabilities using the maximum likelihood estimator for both the fixed-length and interpolated model IMM. We used *Helicobacter pylori* and the first segment of *Human chromosome 22* as training sets. For each training set we built a k -order fixed-length models and a k -order IMM for $k = 0, 1, 2, 3, 4, 5$. All our algorithms have been implemented in C++ and run under Linux operating system. The IMM algorithm is presented in Section 4.1.

4.1 Interpolated Markov model

We used a modification of the χ^2 -confidence based interpolation method introduced in the GLIMMER gene finding algorithm in [18] for computing the $\lambda(c_i)$ in the equation for the conditional probability in the

interpolated Markov model

$$P_{IMM}(a_i|c_j) = \lambda(c_j) \cdot P(a_i|c_j) + (1 - \lambda(c_j)) \cdot P_{IMM}(a_i|c_j[1:k-1]).$$

Algorithm 1: k -order IMM parameter estimation

```

input :  $n(c_j), n(c_j, a_i), N, k$ 
output:  $k$ -order  $P_{IMM}(a_i|c_j)$ 
begin
  for  $j = 1$  to  $|\mathcal{A}|^k$  do
     $th = (N - k + 1)P(c_j)$  ;
    if  $n(c_j) \geq th$  then
       $\lambda(c_j) = 1$ 
    else
       $chisquare = 0$  ;
      for  $i = 1$  to  $|\mathcal{A}|$  do
         $chisquare += \frac{(n(c_j, a_i) - n(c_j) \cdot P_{IMM}(a_i|c_j[1:k-1]))^2}{n(c_j) \cdot P_{IMM}(a_i|c_j[1:k-1])}$ ;
       $p = \text{gammp}(chisquare, \mathcal{A} - 1)$  ;
      if  $p < 0.5$  then
         $\lambda(c_j) = 0$ 
      else
         $\lambda(c_j) = \frac{p \cdot n(c_j)}{th}$ 
        for  $i = 1$  to  $|\mathcal{A}|$  do
           $P_{IMM}(a_i|c_j) = \lambda(c_j) \frac{n(c_j, a_i)}{n(c_j)} + (1 - \lambda(c_j)) \cdot P_{IMM}(a_i|c_j[1:k-1])$ 
      end
    end
  end

```

The algorithm takes as its input the following parameters: $n(c_j)$ the frequency of context c_j , $n(c_j, a_i)$ the frequency of string $c_j a_i$, k the order of the IMM and N the length of the training set. The function $\text{gammp}(chi, df)$ computes the probability that the χ^2 random variable is smaller than $chisquare$ i.e. it computes the cumulative distribution function of the χ^2 for $\mathcal{A} - 1$ degrees of freedom. The GLIMMER system used a fixed value for $th = 400$. We interpreted the threshold as the expected number of occurrences of a context c_i of length i as a string in the training set for 0-order Markov source. Thus, we set $th = \mathbf{E}[n(c_j)] = (N - k + 1)P(c_j)$, where $P(c_j)$ is the probability of the context in the 0-order Markov model. Alternatively we could use $th = (N - k + 1)P(c_j) - \sqrt{\text{Var}[n(c_j)]}$. The following sections use the IMM computed by Algorithm 1.

4.2 Fixed-length versus IMM for the same training and testing source

In this experiment we experimentally confirmed the correctness of our theoretical results including the proof of the central limit theorem, the derived formula for $P^\exists(w)$ and the algorithm for computing $P^\exists(w)$. We expected to achieve a better prediction accuracy using the 5-order (fixed-length and IMM) comparing to the 0-order. To exclude a possibility of model misbehavior (over-fitting, etc.) we used the the same sequence of *Haemophilus influenzae* as a training and testing source. We set a serial episode $S = CCGT$ and for each k -order fixed-length model for $k = 0, 1, 2, 3, 4, 5$ and 5-order IMM we computed $P^\exists(w)$ and compared to an observed $P_e^\exists(w)$ for $w = [5, 20]$ by computing the prediction error using Equation 3. The computed prediction errors are represented by a bar graph in Figure 2. Clearly the prediction error decreases monotonically starting from 1-order fixed-length model up. 5-order (fixed-length and IMM) gives the best prediction significantly outperforming the 0-order. This validates our theoretical and algorithmic results. 5-order IMM performs closely to 5-order fixed-length model since the training source was sufficiently large and the IMM did not use the lower order models.

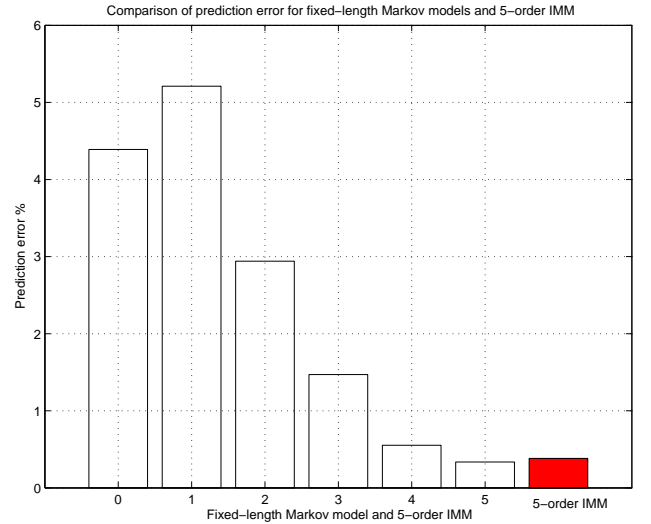


Figure 2: Prediction error d between $P^\exists(w)$ (computed) and $P_e^\exists(w)$ (observed) for a serial episode using a k -order fixed-length Markov models for $k = 0, 1, 2, 3, 4, 5$ and 5-order IMM

4.3 Fixed-length versus IMM for the same training source and a different testing source

In this experiment we compared the fixed-length 5-order with 5-order IMM. We used *Haemophilus influenzae* for computing the conditional probabilities and we tested the performance of both models on *Helicobacter pylori*. We expected the IMM to perform better than the fixed-length model because of its smoothing properties while we expected the fixed-length model to suffer from over-fitting. Also we did not expect a significant improvement in accuracy of IMM because the training set of size (1,830,025) was sufficiently large to find all context strings. We set a serial episode $S = CCGT$ and for each k -order fixed-length model for $k = 0, 1, 2, 3, 4, 5$ and 5-order IMM we compared $P^\exists(w)$ with the observed $P_e^\exists(w)$ for $w = [5, 20]$ by computing the prediction error given in Equation 3. The results, shown as a bar graph in Figure 3 confirm our expectations and the IMM performed slightly better than 5-order fixed-length model. Also 1-order fixed-length turned out to be the winner probably because there is a difference in the structure of DNA of *Helicobacter pylori* and *Haemophilus influenzae* and the 1-order captured the necessary structure without over-fitting.

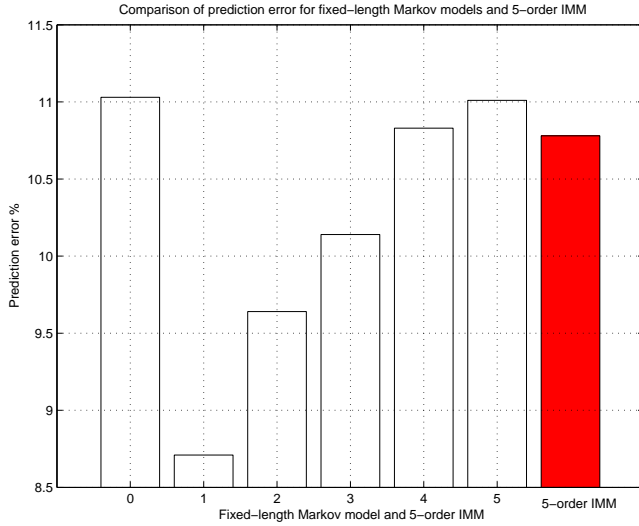


Figure 3: Prediction error d between $P^\exists(w)$ (computed) and $P_e^\exists(w)$ (observed) for a serial episode using a k -order fixed-length Markov models for $k = 0, 1, 2, 3, 4, 5$ and 5-order IMM

4.4 Fixed-length versus IMM for sparse data

In this experiment we wanted to check whether the 5-order IMM outperforms the 5-order fixed-length model when both are trained from sparse training data. To accomplish it we chose the first segment of *Human chromosome 22* of length 234,227 as a training set and we tested both models on the second segment of the same chromosome of length 3,661,561. We set a serial episode $S = CCGT$ and computed $P^\exists(w)$ for both models and compared to the observed $P_e^\exists(w)$ for $w = [5, 20]$. We plotted the results in Figure 4 to show the shape of the curves. From the figure we can see that 5-order IMM slightly better approximates the observed $P_e^\exists(w)$.

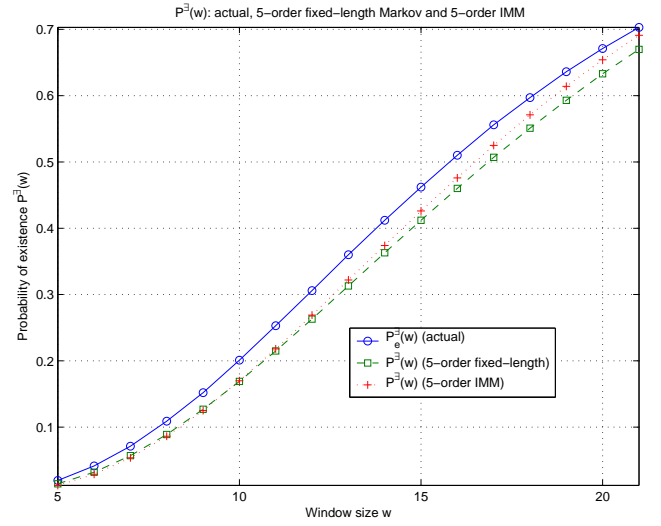


Figure 4: Observed $P_e^\exists(w)$ and computed $P^\exists(w)$ for a serial episode using 5-order fixed-length Markov model and 5-order IMM

5 Conclusions and extensions

We presented a new method for a reliable identification of significant episodes in variable-length Markov source. As a measure of significance we used $\Omega^\exists(n, w)$ the number of windows in which the episode occurred in the event stream. We proved that $\Omega^\exists(n, w)$ is a sum of so called φ -mixing random variables and obeys the central limit theorem, which leads to a computational formula for the threshold $\tau_u(w)$ for discovering significant episodes. We proposed to use variable-length Markov models with the threshold mechanism because of their flexibility for modeling a wide variety

of event sources. In particular we compared the interpolated Markov model with the fixed-length Markov model in experiments conducted on DNA sequences. We showed that the IMM slightly outperforms the fixed-length model in terms of prediction accuracy. We also showed that for DNA source the use of Markov models outperforms memoryless models in terms of accuracy in predicting occurrences of episodes even though a Markov model can be susceptible to over-fitting. The drawback of using Markov models is the high computational cost of computing the threshold. This could be overcome by using a combination of a Bernoulli model and a Markov model. In such a technique we could use a Markov model for small values of w where the accuracy of the prediction would be crucial and we could use the Bernoulli model for large w , where $P^3(w)$ for both a Markov and the Bernoulli model converge to 1.

References

- [1] M. Atallah, R. Gwadera and W. Szpankowski, Detection of significant sets of episodes in event sequences, *Fourth IEEE International Conference on Data Mining*, pages 67-74, Brighton UK.
- [2] R. Azad and M. Borodovsky, Effects of choice of DNA sequence model structure on gene identification accuracy, *Bioinformatics* 2004.
- [3] P. Billingsley (1986), *Probability and measure*, John Wiley, New York.
- [4] L. Boasson, P. Sequels, I. Guessarian, and Y. Matiyasevich (1999), Window-Accumulated Subsequence Matching Problem is Linear, *Proc. PODS*, 327-336.
- [5] S. Brin, R. Motwani, C. Silverstein, *Beyond Market Baskets: Generalizing Association Rules to Correlations*, SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona
- [6] S. Chen, J. Goodman, An Empirical Study of Smoothing Techniques for Language Modeling, Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.
- [7] A. Delcher, D. Harmon, S. Kasif, O. White, S. Salzberg, Improved microbial gene identification with GLIMMER *Nucleic Acids Research*, Vol. 27, No 23, 1999.
- [8] R. Gwadera, M. Atallah, and W. Szpankowski, Reliable detection of episodes in event sequences, In *Third IEEE International Conference on Data Mining*, pages 67-74, Melbourne Florida.
- [9] J. Han, J. Pei, Y. Yin, R. Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, *Data Mining and Knowledge Discovery*, 8, 53-87, 2004
- [10] S. Katz, Estimation of probabilities from sparse data for the language model component of a speech recognizer, *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400-401, March 1987.
- [11] H. Mannila, H. Toivonen, and A. Verkamo, Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery*, 1(3), 241-258, 1997.
- [12] A. Martin, G. Seroussi and M. Weinberger, *IEEE Transaction on Information Theory*, Vol. 50, No. 7, July 2004.
- [13] F. Jelinek, R. Mercer, Interpolated estimation of Markov source parameters from sparse data, *Proceedings of Workshop on Pattern Recognition in Practice*, pages 381-397, 1980.
- [14] M. Régnier and W. Szpankowski (1998), On pattern frequency occurrences in a Markovian sequence, *Algorithmica*, 22, 631-649.
- [15] J. Rissanen, A Universal Data Compression System, *IEEE Trans. Inform. Theory*, Vol. IT-29, No. 5, pp 656-664, 1983
- [16] J. Rissanen, Fast Universal Coding with Context Models, *IEEE Transactions on Information Theory*, Volume 45, No. 4, 1065-1071, May 1999
- [17] E. Ristad and R. Thomas, Nonuniform Markov Models, *International Conference on Acoustics, Speech, and Signal Processing*, Munich, Germany, April 20-24, 1997.
- [18] S. Salzberg, A. Delcher, S. Kasif, O. White, Microbial gene identification using interpolated Markov models, *Nucleic Acids Research*, Vol. 26, No 2, 1998.
- [19] W. Szpankowski (2001), *Average Case Analysis of Algorithms on Sequence*, John Wiley, New York.
- [20] M. Weinberger, J. Rissanen, and M. Feder, A universal finite memory source, *IEEE Trans. Inform. Theory*, vol. IT-41, pp. 643-652, May 1995. 48

- [21] F. Willems, Y. Shtarkov, and T. Tjalkens, The context-tree weighting method: Basic properties, *IEEE Trans. Inform. Theory*, vol. IT-41, pp. 653–664, May 1995 John Wiley, New York.

Efficient Mining of Maximal Sequential Patterns Using Multiple Samples *

Congnan Luo[†]

Soon M. Chung[‡]

Abstract

In this paper, we propose a new algorithm, named MSPX, which mines maximal sequential patterns by using multiple samples to effectively exclude infrequent candidates. MSPX begins with a bottom-up search. But at each pass, instead of processing all candidates, it always tries to find most of the infrequent ones effectively by counting only the potentially infrequent candidates against the whole database. After removing verified infrequent candidates, the remaining candidates are used to generate new candidates. Finally, with a top-down search, all the maximal frequent sequences can be identified efficiently. Sampling technique is used at each pass to distinguish potentially infrequent candidates. How to increase the minimum support level for the mining of samples to estimate if candidates could be infrequent is analyzed theoretically. Due to the supersequence frequency based pruning, MSPX reduces much more search space than other algorithms. Unlike the traditional single-sample methods proposed for mining frequent itemsets, MSPX uses multiple samples. Thus, it can avoid or alleviate some problems inherent in the single-sample methods. Our experiments show MSPX has very good performance and better scalability than other algorithms. Moreover, even though MSPX uses sampling, the variance of its performance is very small in multiple runs for the same task.

1 Introduction

Mining sequential patterns from large databases is an important problem in data mining. With numerous practical applications, such as consumer market-basket data analysis and web-log analysis, it has become an active research topic. Since it was introduced in [2], many algorithms have been proposed, but most of them are to discover the full set of frequent sequences.

In pure bottom-up, breadth-first search algorithms such as GSP [6] and PSP [5], only subsequence infre-

quency based pruning is used to reduce the number of candidate sequences. So, if a sequence with length l is frequent, all of its 2^l subsequences must be enumerated first. Thus, if some frequent sequences are long, the overhead of enumerating all of their subsequences is so much that mining the full set of frequent sequences is impractical. An alternative approach is mining only the maximal frequent sequences. A frequent sequence is maximal if none of its supersequences is frequent. Mining only the maximal frequent sequences is efficient because the search space can be reduced a lot by using the supersequence frequency based pruning. In interactive data mining, after mining the set of maximal frequent sequences quickly, we can selectively count the interesting patterns subsumed by this set by scanning the database just once. Moreover, managing and querying a small set of maximal patterns is easy, time-saving and space-saving.

For the association rule mining, many efficient algorithms were proposed to mine maximal frequent itemsets [4]. However, differences between the two kinds of mining make those algorithms very difficult or impossible to be applied for the maximal frequent sequence mining. For example, given a set of items, the search space for mining frequent itemsets is limited, whereas it is unlimited for sequence mining. An item can appear at most once in an itemset but it may appear multiple times in a sequence at different positions.

A critical question for the maximal frequent sequence mining is how to look ahead for longer or maximal frequent sequences at a reasonable cost. In AprioriSome and DynamicSome algorithms [2], the candidates at some passes are directly used to generate longer candidates for the next pass. Actually, it leaves the job of excluding infrequent candidates to the later passes. Since the subsequence infrequency based pruning is not performed at all, a very large number of longer infrequent sequences are generated as candidates. The cost of identifying and removing these infrequent candidates can offset the gain from the supersequence frequency based pruning. On the other hand, in GSP, all the candidates at each pass are counted. In this way, we can exclude all the infrequent candidates at each pass, and hence avoid generating many false candidates. However, we cannot benefit from the supersequence infrequency

*This research was supported in part by Ohio Board of Regents, LexisNexis, NCR, and AFRL/Wright Brothers Institute (WBI).

[†]Dept. of Computer Science and Engineering, Wright State University, Dayton, Ohio 45435, USA.

[‡]Dept. of Computer Science and Engineering, Wright State University, Dayton, Ohio 45435, USA.

based pruning. Based on these observations, a new MSPX algorithm is proposed in this paper, which emphasizes how to effectively exclude most infrequent candidates, so that the performance gain from the supersequence frequency based pruning can be maximized.

MSPX adopts the Apriori candidate generation method [1, 2, 6] and performs a bottom-up, breadth-first search first. But, instead of counting all the candidates on the whole database at each pass, it always tries to find and remove most of the infrequent candidates by counting as few candidates as possible. To achieve this, we count the candidates on a random sample drawn from the database at each pass to estimate which candidates are most potentially infrequent. Then, only the potentially infrequent candidates are verified against the whole database to remove really infrequent ones. The mining process is continued with the survived candidates to the next pass. At the end of the bottom-up phase, a superset of all frequent sequences is obtained. Starting from the border of this superset, a top-down search is performed to pick up maximal frequent sequences efficiently.

To improve the performance of MSPX, additional optimization methods are integrated: A signature technique is used to perform the subsequence infrequency based pruning when the seed set for the new candidate generation is too big to be loaded into memory. A prefix tree structure is developed to count the candidate sequences of different sizes during the database scanning, and it also facilitates the customer sequence trimming. MSPX outperforms GSP considerably, and also shows better scalability than SPAM [3] and SPADE [9]. By using multiple samples, the performance of MSPX is also much more stable than those of single-sample methods.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts of sequence mining. Section 3 reviews some related works on sequence mining. Section 4 describes the MSPX algorithm. The experimental results and performance analysis are presented in Section 5. Section 6 contains some conclusions and future work.

2 Sequence Mining

Let $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$ be a set of items. An k -itemset i is a set of k items denoted by $\{i_{m_1}, i_{m_2}, \dots, i_{m_k}\}$, where $1 \leq m_1 < m_2 < \dots < m_k \leq n$. A sequence s is an ordered list of itemsets denoted by $\langle s_1, s_2, \dots, s_k \rangle$, where each s_i , $1 \leq i \leq k$, is an itemset. A sequence $s_a = \langle a_1, a_2, \dots, a_p \rangle$ is contained in another sequence $s_b = \langle b_1, b_2, \dots, b_q \rangle$ if there exist integers $1 \leq j_1 < j_2 < \dots < j_p \leq q$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \dots, a_p \subseteq b_{j_p}$. If s_a is contained in s_b , s_a is a *subsequence* of s_b , and s_b is a *supersequence* of s_a . An item may

appear at most once in an itemset, but it may appear multiple times in different itemsets of a sequence. If there are k items in a sequence, the length of the sequence is k , and we call it a k -sequence. For example, a 3-sequence $\langle \{A\}, \{B, C\} \rangle$ is a subsequence of a 5-sequence $\langle \{C\}, \{A, D\}, \{B, C\} \rangle$. For simplicity, these two sequences can be represented as $A - BC$ and $C - AD - BC$.

Given a database \mathcal{D} of customer transactions, each transaction consists of a customer-id, transaction-time and an itemset which includes all the items purchased by the customer in that single transaction. All the transactions of a customer can be viewed as a customer sequence, where these transactions are ordered by their transaction times. We denote a customer sequence t as $\langle T_1, T_2, \dots, T_m \rangle$, which means the customer has m transactions in the database and each transaction T_i , $1 \leq i \leq m$, contains all the items purchased in that transaction. A customer supports a sequence if the sequence is contained by the customer sequence. The support for a sequence in database \mathcal{D} is defined as the fraction of total customers who support the sequence. Given a user-specified minimum support, denoted by *minsup*, a sequence is frequent if its support is greater than or equal to *minsup*. The problem of sequence mining is to find all the frequent sequences in the database with respect to a user-specified *minsup*. If a sequence is frequent and none of its supersequences is frequent, then it is a maximal frequent sequence.

Based on the above definitions, two properties are often utilized to speed up the sequence mining: 1) Any supersequence of an infrequent sequence is not frequent, so it can be pruned from the set of candidates. This is called subsequence infrequency based pruning. 2) Any subsequence of a frequent sequence is also frequent, so it can be pruned from the set of candidates. This is called supersequence frequency based pruning.

In [6], the above definition of sequence mining was generalized by incorporating time constraints, sliding time windows, and taxonomy. This generalization makes the sequence mining more complex. For example, a sequence $A - BC - D - GH$ is frequent does not necessarily mean that its subsequence $A - BC - GH$ is also frequent, because the subsequence may not satisfy the time constraints. In this research, we consider the nongeneralized sequential pattern discovery.

3 Related Work

Mining sequential patterns was introduced in [2] with AprioriAll, AprioriSome and DynamicSome algorithms. Although AprioriSome and DynamicSome try to generate and count long candidate sequences before enumerating all their subsequences, their performance is usu-

ally worse than that of AprioriAll. The reason is that too many false candidates are generated without being pruned by the subsequence infrequency based pruning. The performance gain from the supersequence frequency based pruning is not enough to offset the cost of counting so many false candidates.

GSP [6] was proposed for generalized sequence mining, and it requires multiple passes on the database. At pass k , the set of candidate k -sequences are counted on the database and frequent k -sequences are determined. Then, the candidate $(k + 1)$ -sequences are generated by joining frequent k -sequences for the next pass. This process will continue until no candidate is generated. Even though GSP is much faster than AprioriAll, it has a very high overhead of enumerating every single frequent subsequence when there are some long patterns. This is also the main weakness of other Apriori-like algorithms, such as PSP [5]. For PSP, a prefix tree was developed as the internal data structure to organize and count candidates more efficiently. The differences between the PSP's prefix tree and the one developed for our MSPX are: 1) our prefix tree is used to count candidates of different sizes, whereas PSP prefix tree is only used to count the candidates of the same size; 2) to improve the candidate counting, a bit vector is associated with our prefix tree to facilitate the customer sequence trimming; and 3) the supersequence frequency based pruning reduces the size of our prefix tree.

SPADE [9] works on the databases with a vertical id-list format, where a list of (customer-id, transaction-time) pairs are associated with each item, and the candidates are counted by intersecting the id-lists. A lattice-theoretic approach is used to decompose the search space into small pieces so that all working id-lists can be loaded into memory. SPAM [3] uses a vertical bitmap representation of the database for candidate generation and counting. A bitmap is created for each item in the database, where each bit corresponds to a transaction. If transaction j contains item i , then bit j in the bitmap for item i is set to 1; otherwise, it is set to 0. SPAM also uses a depth-first traversal of the Lexicographic sequence tree and an Apriori-based pruning of candidates.

Both SPADE and SPAM were reported more efficient than GSP. However, their performance may not be scalable in certain cases. For SPADE, if the database is in the horizontal format, where the transactions form the tuples in the database, transforming it to a vertical one requires extra disk space of roughly the same size. This may be a problem in practice if the database is large. Even if the database is in the vertical format, to efficiently count 2-sequences, SPADE proposes transforming it back to the horizontal one on the fly. This

usually requires much time and memory for very large databases and results in a performance degradation.

SPAM is claimed to be a memory-based algorithm. According to our tests, its scalability is much more sensitive to the number of items and the database size than other algorithms. The comparison between MSPX, GSP, SPADE and SPAM is presented in detail in the performance analysis section.

In [8], sampling was evaluated as an efficient way to mine an approximate set of frequent itemsets. In [7], a lowered minsup is used to mine the sample, so that the probability a frequent itemset is missed from the sample result would be small. Then, one more database scan is needed to find the misses and the overestimates.

Our proposed MSPX algorithm also uses sampling to mine maximal frequent sequences from databases. While most of other sampling-based mining algorithms use only one sample in the whole mining, MSPX uses multiple samples, one for each pass in its bottom-up phase.

4 MSPX Algorithm

We present MSPX in this section. First, we use a simple example to explain the basic idea of MSPX. Figure 1 shows how GSP mines a small database with 4 distinct items A , B , C and D . The whole mining includes 4 passes, and each pass corresponds to a level in the lattice in Figure 1. At each pass, all candidate sequences are shown and the infrequent sequences identified are in gray color. In our description, C_k denotes the set of candidate k -sequences and L_k denotes the set of frequent k -sequences.

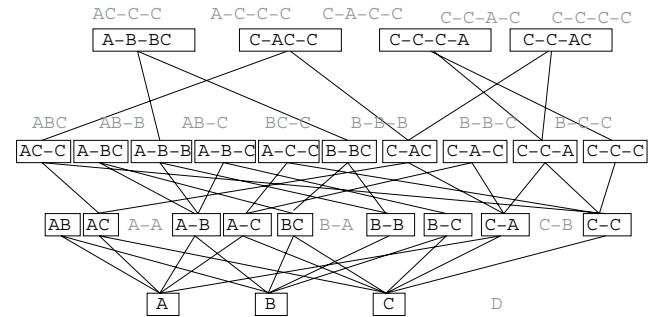


Figure 1: Candidate and Frequent Sequences in GSP

At pass 3 with C_3 , there are three possible strategies to continue the mining: I) As in GSP, we count all 17 candidate 3-sequences, and then generate C_4 from L_3 . The subsequence infrequency based pruning is fully applied because all infrequent 3-sequences are known. II) As in AprioriSome and DynamicSome, we generate C_4 directly from C_3 without counting any candidate 3-sequence. Then, the candidate 4-sequences are counted

earlier than 3-sequences. The purpose of such look-ahead is to apply the supersequence frequency based pruning. If a 4-sequence is identified as frequent, then we can avoid counting its four subsequences of length 3. For the mining task requiring many passes, we can look ahead in this way during several passes or even all passes until the longest candidates are generated. But it has been shown that this method often produces many false candidates and even causes an explosion of the candidate set size [2]. III) Suppose that, with some a priori information, we can divide C_3 into two disjoint subsets, such that the first subset contains most of the infrequent candidates in C_3 while containing as few frequent ones as possible, and the second subset contains almost only the frequent candidates in C_3 . For example, we may get the following two subsets: {ABC, AB-B, B-B-B, AC-C, AB-C, BC-C, A-B-B} and {A-BC, A-B-C, A-C-C, B-BC, B-B-C, B-C-C, C-C-C, C-AC, C-A-C, C-C-A}.

Then, we just count the candidates in the first subset on the whole database. Even though the second subset is not processed yet, we can expect that most of the infrequent candidates in C_3 should have been identified because they are already clustered into the first subset. As a next step, the candidates already identified as frequent and those not counted yet are used to generate the candidates for the next pass. That means, the candidates in the second subset, which are not counted yet, are assumed to be frequent.

While all 17 candidate 3-sequences are counted in strategy I, only 7 candidate 3-sequences are counted in strategy III. In addition, there are 12 candidate 4-sequences generated in strategy III. This number is very close to 9 of strategy I, but much smaller than 25 of strategy II. As shown in this example, strategy III provides a much more effective way to exclude infrequent candidates at pass 3 and look ahead for longer patterns. At the same time, it doesn't impose much extra overhead for excluding longer false candidates in the later passes, because the subsequence infrequency based pruning still can be applied very effectively. Thus, the explosion of the candidate set size reported for strategy II can be avoided. Based on this idea, MSPX employs strategy III at each pass, and eventually obtains the set of potential maximal frequent sequences. Then, an efficient top-down search for real maximal frequent sequences can be performed, starting with this set.

Essentially, strategies I and II can be viewed as two extreme cases of strategy III, where the counted subsets are the whole candidate set and the empty set, respectively. However, both of them ignore the possibility of each candidate to be infrequent. Actually,

strategy III shows that such information can be used to improve the effectiveness of excluding infrequent candidates. A practical question regarding strategy III is how to divide C_k into such two disjoint subsets as desired. In our research, a sampling technique is developed as a part of the proposed MSPX algorithm.

Our description of MSPX is composed of the following parts: an overview of MSPX; a new signature technique used for subsequence infrequency based pruning; the issue of dividing C_k into two disjoint subsets; the prefix tree structure and the customer sequence trimming; and finally the comparison of our sampling method with previous ones.

4.1 Overview of MSPX. MSPX includes three phases:

Initial Phase: L_1 and L_2 are determined. Candidate 3-sequences are generated from L_2 . To count candidate 2-sequences, a two-dimensional array is used. The entry at position (i, j) in the upper-triangle of the array contains the counts of three candidates $i - j$, ij and $j - i$.

Bottom-up Phase: MSPX starts the bottom-up phase from pass 3. At pass k ($k \geq 3$), a small random sample db is drawn from the database DB . We count all the candidates in C_k on db for their local supports (i.e., supports in db). Then, we choose a support level θ as the criterion to divide C_k into two subsets C_k^- and C_k^+ . All the candidates with local supports lower than θ are put into C_k^- , and others are put into C_k^+ . The notation C_k^- (C_k^+) means the candidates in C_k^- (C_k^+) are negative (positive) to be frequent. We must try to make C_k^- contain almost all the infrequent candidates while containing as few frequent ones as possible. How to achieve this will be discussed later. After determining these two subsets, we just count the candidates in C_k^- on the rest of the database DB . The really infrequent candidates are removed. A set L_k^* is constructed by including all the candidates which are already verified frequent and those in C_k^+ , which are not processed yet. Obviously, we have $L_k \subseteq L_k^*$. Then, L_k^* is used as the seed set to generate candidates for the next pass; i.e., the candidates in C_{k+1} are generated by joining the sequences in L_k^* as in GSP. At the end of this phase, we obtain a superset of all frequent sequences. The maximal sequences are extracted from this superset to construct MFS^* , the set of potential maximal frequent sequences. It is guaranteed that all maximal frequent sequences are under the border formed by MFS^* .

Top-down Phase: Starting with MFS^* , a top-down

search is performed. All the sequences in MFS^* are counted on DB . If a k -sequence ($k > 3$) is infrequent, all of its $(k - 1)$ -subsequences are considered as candidates for the next pass. For a frequent k -sequence, we stop splitting it, and put it into the set of maximal frequent sequences, MFS , if none of its supersequences is already in this set. If a subsequence of the frequent k -sequence is already in MFS , this subsequence should be removed from MFS . For a newly generated candidate $(k - 1)$ -sequence, if it has any supersequence in MFS , we remove it from further consideration. We also check if the newly generated $(k - 1)$ -sequence has any subsequence which is already identified as infrequent. If yes, this candidate must be split again. This top-down process continues until no new candidates are generated.

4.2 Candidate Generation and Pruning in the Bottom-up Phase. At pass k in the bottom-up phase, the candidates are generated in two steps:

Join Step: we generate candidate $(k + 1)$ -sequences by joining L_k^* with L_k^* . For any two k -sequences s_1 and s_2 in L_k^* , if the subsequence obtained by dropping the first item of s_1 is the same as the subsequence obtained by dropping the last item of s_2 , a new candidate is generated by extending s_1 with the last item of s_2 . The added item starts a new itemset for s_1 if it was a separate itemset in s_2 . Otherwise, it becomes a member of the last itemset in s_1 .

Prune Step: The candidate $(k + 1)$ -sequences with any subsequence of length k which is not in L_k^* are removed. If L_k^* is too large to be loaded into memory totally, we perform the partial subsequence infrequency based pruning as described below.

A weakness of GSP is the way that a large L_k is processed. When minsup is very small, L_k could be too large to be loaded into memory totally. For this case, GSP proposed to use a relational merge-join technique to generate candidates. But in this manner, subsequence infrequency based pruning cannot be applied because the whole L_k is not available in memory and retrieving the relevant portions of L_k from a disk requires too many swaps. Without subsequence infrequency based pruning, usually the performance of GSP degrades a lot. In MSPX, we adopted a new method to solve this problem. If the L_k^* at some pass requires too much memory, we assign each k -sequence in L_k^* an integer signature which is highly correlated to the content of the sequence. A simple example of generating the signature is shown below, where t is the

number of itemsets in the sequence; m_i is the number of items in the i th itemset; I_{ij} is the j th item in the i th itemset; C_i , $1 \leq i \leq t$, is the weight imposed on the i th itemset; and C_0 is the weight imposed on the total number of itemsets.

$$(C_0 * t) + \sum_{i=1}^t (C_i * m_i * \sum_{j=1}^{m_i} I_{ij})$$

All the signatures are sorted and put into an array. Compared with the case of loading the whole L_k^* into memory, the signature array requires much less space. Thus, we can load working portions of L_k^* and all the signatures into memory at the same time. When a new candidate $(k + 1)$ -sequence is generated, the signatures of its k -subsequences are computed and searched in the signature array. If any one of them is not in the array, the candidate should be removed. Since all the signatures are in memory, subsequence infrequency pruning still can be applied. It is possible that two or more k -subsequences have the same signature. However, that probability is very low. Our experiments showed that signatures are much more effective than hashing. MSPX performs much better than GSP when the seed set for the candidate generation cannot be loaded into memory totally at some passes. If the memory cannot hold all the candidates, we need to generate them in several parts. For each part, the candidates are counted on the sample. After all parts are processed, the candidates with the local support lower than θ are loaded into memory to perform the counting on the rest of the whole database.

4.3 Dividing the Candidate Set. At each pass in the bottom-up phase, we need to divide C_k into two disjoint subsets, C_k^- and C_k^+ , such that C_k^- contains most of the infrequent candidates while containing as few frequent ones as possible, and almost all the candidates in C_k^+ are frequent. To achieve this, we collect the local support of each candidate in the sample db . We set a support level θ as the criterion to estimate if a candidate could be frequent or not. If the local support of a candidate is lower than θ , it is estimated to be infrequent and put into C_k^- . Otherwise, we expect it to be frequent and put it into C_k^+ . If an infrequent candidate is misestimated as frequent, we call it an *overestimate*. On the other hand, if a frequent candidate is misestimated as infrequent, it is an *underestimate*.

Obviously, underestimates can affect only the computation overhead of the current pass. In the extreme case, even if we misestimate all frequent candidates by setting C_k^- as C_k , it just makes MSPX work like GSP at that pass. But it is hard to predict how much overestimates can affect the whole mining, because the misesti-

mated infrequent candidates in C_k^+ will be directly used to generate new candidates. A certain number of overestimates may easily increase the number of false candidates and thus make the candidate set for the next pass much bigger than the case without those overestimates. This usually results in two consequences: First, the overhead of counting all candidates on the sample in the next pass increases. Second, but more importantly, with much more false candidates, the probability of making further overestimates tends to increase. Therefore, preventing overestimates is important for MSPX because they may make the case complicated and unpredictable.

The easiest way to choose θ is setting it to the user-specified minsup. But we found out that, in practice, if there are many candidates whose supports are slightly lower than minsup, a lot of overestimates may occur. Furthermore, since the distribution characteristics of the database to be mined is usually unknown, we do not know if this will happen when the database is mined with respect to a specific minsup. Thus, it is necessary to take some measures to prevent the problem of having too many overestimates. In this research, we increase the user-specified minsup a little for θ and explore the relationship between the increment of minsup and the probability of an overestimate. We believe this theoretical analysis can provide a guideline for us in running MSPX. We must keep in mind that if we increase the user-specified minsup too much, a lot of underestimates may happen, and it contradicts our purpose of using the sampling. On the other hand, if we increase minsup too little for θ , we may have too many overestimates.

Consider an original database DB and an arbitrary sequence X . If the support of X in DB is P_X , then the probability that a customer sequence randomly selected from DB contains X is also P_X . Let's consider a random sample db with m customer sequences that are independently drawn from DB with replacement. The random variable T_X , which represents the total number of customer sequences containing X in db , has a binomial distribution of m trials with the probability of success P_X . In general, if m is greater than 30, T_X can be approximated by a normal distribution whose mean is $m * P_X$ and the standard deviation is $\sqrt{m * P_X * (1 - P_X)}$.

In MSPX, suppose that we draw a sample db with m customer sequences from DB , and then try to use the point estimator $P'_X = T_X/m$ to estimate the support of X in the population of DB . Then, P'_X is an unbiased estimator with mean $m * P_X/m = P_X$ and standard deviation $\sqrt{m * P_X * (1 - P_X)}/m = \sqrt{P_X * (1 - P_X)}/\sqrt{m}$.

If we assume the support of X in DB , P_X , is

the user-specified minsup, then P'_X , which is observed from a sample db , should be around P_X with a normal distribution as described above. If we set θ to a support level P''_X , $P''_X > P_X$, the probability that the local support of X observed in the sample is not lower than P''_X is $1 - P_Z$, where $Z = (P''_X - P_X)/\sqrt{P_X * (1 - P_X)}/\sqrt{m}$. Z is often called the z -score, and P_Z is the probability at the z -score value of Z .

Let's consider the standard deviation of P'_X , $\sqrt{P_X * (1 - P_X)}/\sqrt{m}$. The value of its part $P_X * (1 - P_X) = -(P_X - 1/2)^2 + 1/4$ is increasing in the P_X interval of $[0, 1/2]$. Since minsup is usually lower than 50%, we can assume the value of $P_X * (1 - P_X)$ is increasing in the P_X interval of $[0, minsup]$; that means, the standard deviation of P'_X is increasing in this P_X interval. If the support of another sequence Y in DB is lower than the minsup P_X (i.e., Y is actually an infrequent sequence), then both the mean and standard deviation of observed P'_Y should be smaller than those of P'_X , respectively. Therefore, compared with P'_X , the distribution curve of P'_Y is shifted left and shaper. Thus, the probability that the local support of an infrequent sequence Y observed in the sample is not lower than P''_X should be smaller than $1 - P_Z$. In other words, the probability that Y is overestimated is smaller than $1 - P_Z$.

For a specific mining job, if we specify the upper bound of the probability that an infrequent candidate is overestimated as σ (i.e., $1 - P_Z = \sigma$), then $P_Z = 1 - \sigma$. Let's denote the corresponding z -score for P_Z as $Z_{(1-\sigma)}$, which can be found from the z -score table. Then, we can compute P''_X using the formula (4.1). Actually, P''_X is the value of θ for the upper bound σ .

$$(4.1) \quad P''_X = P_X + Z_{(1-\sigma)} * \sqrt{P_X(1 - P_X)}/\sqrt{m}$$

Here, $P_X = minsup$ and $m = |db|$. For example, if we set $\sigma = 20\%$, the corresponding $Z_{(1-\sigma)}$ value is about 0.85. Our experiments show that the value of θ (i.e., P''_X) computed using the formula (4.1) often tends to be very conservative. In the above analysis, we just considered a single sample and tried to control the occurrence of overestimates at a very low level for that pass. However, it is not an easy goal to achieve unless we increase minsup very much for θ , which may cause too many underestimates.

Let's include the multiple samples into the analysis. In MSPX, if an infrequent sequence Y is overestimated at a pass, it will be used to generate new candidates. In the next pass, a new sample is drawn. If Y is not overestimated in the new sample, then all the false candidates grown from Y are not overestimated either, and hence they are removed. That means, the progressive overestimating based on Y can be avoided, and the negative effect caused by Y is limited within

these two passes. Based on this observation, we can relax our policy as follows: Overestimates are allowed at a reasonable level at the current pass, but they are strictly prevented from happening again at the next pass. With this policy, two consecutive samples are used for the analysis. Obviously, if the two samples are drawn independently, the probability that an infrequent candidate is overestimated in both samples cannot be bigger than $(1 - P_Z)^2$. Thus, with the same requirement σ on the upper bound of the probability of an overestimate, P_Z is changed from $1 - \sigma$ to $1 - \sqrt{\sigma}$. Thus, we can rewrite the formula (4.1) as

$$\theta = \text{minsup} + Z_{(1-\sqrt{\sigma})} * \sqrt{\text{minsup} * (1 - \text{minsup}) / |db|} \quad (4.2)$$

Here, $\sigma = 20\%$ means that the probability an infrequent candidate is overestimated in two consecutive samples is at most 20%. Actually, the upper bound of the probability of an overestimate in the first sample is relaxed to 45%, not the original 20%. The corresponding z -score $Z_{(1-\sqrt{\sigma})}$ is about 0.13. Our tests show that the formula (4.2) provides a tighter θ value, and MSPX works better with it in practice.

4.4 Efficient Counting of Candidates Using the Prefix Tree and the Customer Sequence Trimming. During the top-down search for maximal patterns covered by MFS^* , to reduce the number of passes, we need to count candidates of different sizes at each pass over the database. For that purpose, we developed a new prefix tree structure. Since it is much more efficient than the hash tree, we also use it to count the candidates of the same size during the bottom-up phase.

The following example shows how the prefix tree works. Suppose we have 10 candidates of length 2 or 3. The prefix tree is constructed as shown in Figure 2. Each node is associated with a pointer. If the path from the root to a node represents a candidate, the pointer points to the candidate; otherwise, it is NULL. A node may have two types of children. The “I-extension” child means the item represented by the child node is in the same itemset with the item represented by its parent node. The “S-extension” child means the item represented by the child node starts a new itemset. All the S-extension (I-extension) children of a node are linked together, and only the first child is linked to their parent node by a dashed (solid) line. For example, nodes 4 and 5 are the S-extension children of node 1, and the corresponding pathes represent the candidates $A - A$ and $A - E$, respectively. Nodes 6 and 7 are I-extension children, and their pathes represent AC and AD , respectively.

To speed up the counting, a bit vector is associated

with the prefix tree to facilitate the customer sequence trimming. In this example, we have 8 items in the database: A, B, C, D, E, F , and H . Since B, F , and G do not appear in any candidate, they should be ignored during counting. Thus, the bit vector is set as (10111001), where 1 at the i -th bit position means item i appears in the prefix tree. All the bits are initialized to 0, and the corresponding bits are set to 1 as we insert candidates into the prefix tree.

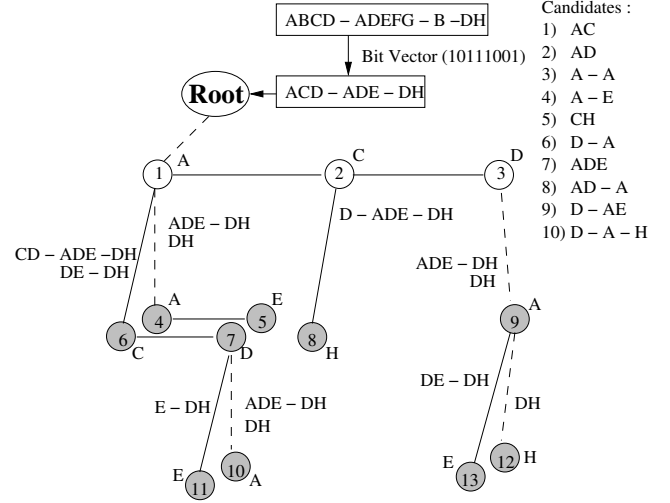


Figure 2: Prefix Tree of MSPX

Given a customer sequence $s = ABCD - ADEFG - B - DH$, we trim it to $s' = ACD - ADE - DH$ using the bit vector first. Then, a recursive method is used to count all the candidates contained in s' . At the root node, we check each item in $ACD - ADE - DH$ to see if it is in the root node's S-extension children. The first item of s' is A , and it appears as the first S-extension child of the root node. So we recursively call the count function at the root node with two sequence segments. The segment $CD - ADE - DH$ is used in the call for node 1's I-extension link, while $ADE - DH$ is for its S-extension link. Then, we can locate the second item of s' , C , at node 2. Since node 2 has no S-extension child, only one recursive call with the segment $D - ADE - DH$ is made for its I-extension link. The third item of s' , D , is the last item of the first itemset in s' . Only one call with segment $ADE - DH$ is made for node 3's S-extension link. The fourth item of s' , A , can be located at node 1 again, and we make two recursive calls. One is for the node 1's I-extension link with $DE - DH$, and the other one is for its S-extension link with DH . Then, we process the remaining items in s' , one by one, in the same way. Whenever we locate an item at some node, if the pointer associated with the node is not NULL and the count of the corresponding candidate is not

increased yet (for the current customer sequence), it should be increased.

The root node is processed differently from other nodes. At the root node, there is no constraint on which items in the customer sequence should be checked against the root's S-extension link, because the first item of a candidate can appear anywhere in the customer sequence. At other nodes, there are always some constraints. Let's see how to make recursive calls at node 1 along its I-extension link. Recall that we have made two recursive calls at the root node with segments, $CD - ADE - DH$ and $DE - DH$, for node 1's I-extension. Now we process them at node 1. Since the two segments are specified for node 1's I-extension link, we should check the items in their first itemsets, CD and DE , against node 1's I-extension link. For $CD - ADE - DH$, since C appears at node 6 which has no child, we stop there by just increasing the count of AC . Another item, D , appears at node 7. We increase the count of AD and make recursive calls for node 7's links. Since D is the last item of the first itemset in $CD - ADE - DH$, only one recursive call with the segment $ADE - DH$ is made for node 7's S-extension link. For another sequence segment $DE - DH$ at node 1, two items of the first itemset, D and E , are checked. D is located at node 7. Since the count of AD is already increased before, we should not increase it again. Two recursive calls are made at node 1 for node 7's links. One is with $E - DH$ for node 7's I-extension link and the other is with DH for the S-extension link. We can ignore E because it is not an I-extension child of node 1. This process will continue until a leaf node is reached or the sequence segment is empty.

4.5 Sampling in MSPX. The way of using the sampling in MSPX is unique compared to other previous researches [7, 8]. We use multiple samples in MSPX, instead of using a single one. However, we just collect the local supports of candidates of the same size in the sample, rather than mining the sample completely using a fixed minsup. There are some problems in the previous studies [7, 8]: 1) As only one random sample is used, if the sample does not represent the database well, it will affect the whole mining and degrade the overall performance very much. 2) It is not a surprise that the performance of the algorithms using a single sample may vary considerably from one run to another for the same mining task. 3) When mining a sample, a fixed minsup, either the user-specified minsup or a lowered one, is used. For those candidate sequences whose global supports are slightly higher or lower than the minsup, misjudgement happens frequently because their local supports in the sample often diverge from

their global supports. Then, a lot of effort is needed to identify the overestimates and the underestimates in the sample results. 4) For single-sample methods, when the minsup is very small, simply using this minsup or a lowered one to mine the sample is risky, because $minsup * |db|$ or $lowered_minsup * |db|$ used to filter the candidates during the mining of the sample could be too low. In that case, many overestimates may happen. They not only make the mining of the sample itself very difficult, but also pose a heavy overhead on verifying the sample results. We found this problem is more serious for sequence mining than for mining the frequent itemsets, because the search space of sequence mining is much bigger. Thus, the sampling method used in [7, 8] is challenged when minsup is very small. Using a large sample can relieve these problems to some extent, but it cuts the merit of sampling.

In MSPX, these problems are avoided or alleviated by using multiple samples, one for each pass in the bottom-up phase. If a sample is bad, we may need to count many underestimated candidates in C_k^- . However, it just affects the current pass. We may also overestimate many infrequent candidates and put them into C_k^+ . Then, they may generate many false candidates and affect the next pass. Fortunately, in the next pass, such false candidates will be categorized into C_k^- or C_k^+ again based on their local supports in a new sample. Thus, we still have the opportunity to stop the candidate growth caused by the overestimates made in the previous passes. Actually, the probability of choosing bad samples in a row is extremely small if the sample size is reasonable. Due to the joint contribution from multiple samples, the negative effect from one bad sample can be limited within a couple of passes, rather than the whole mining. Our experiments showed that the variance of the execution time of MSPX during 100 runs for each test is very small. The worst case of MSPX was also much better than that of single-sample methods.

In MSPX, we do not set a fixed minsup for the mining of samples. Instead, we simply collect the local supports of the candidates in the sample. Even though we use the local supports to categorize candidates into C_K^- or C_K^+ , whether to remove a candidate from the search space or not is still based on its global support after it is verified against the whole database. Thus, no frequent sequence will be missed in the bottom-up phase. This enables us to perform an efficient top-down search for all maximal patterns. Moreover, to avoid having too many overestimates, we increase the minsup a little for mining the sample. Thus, we relieve the problem that too many infrequent sequences are overestimated as frequent and placed into C_k^+ .

Initially, we concerned if MSPX would incur a lot of

overhead due to multiple samples when it is compared to the case of using a single sample. In single-sample methods, we draw one single sample but mine it in multiple passes. On the other hand, MSPX draws multiple samples, but each sample is processed in a single pass. Roughly speaking, in both methods, the sampling procedure includes three steps: 1) randomly selecting distinct customer ids for the sample, 2) loading the sample into memory, and 3) processing the sample.

The time required for step 1 is negligible. For step 2, in the traditional single-sample methods, we just need to load the sample into memory once if we have enough memory space. In MSPX, we must do it multiple times because the samples are different at different passes. However, due to the small size of the sample, the loading time is not a dominant factor. Step 3 is the dominant part in the overhead related to the sampling. Even though MSPX uses multiple samples, it does not mine each sample completely. For each sample, it just counts the candidates of the same length for that pass (i.e., candidate k -sequences for the k th pass). The single-sample methods mine the sample completely in multiple passes with respect to a minsup.

Let's simplify the situation by assuming that any sample drawn can represent the database well. In the single-sample methods, if GSP is used to mine the sample, the number of passes and the candidate set size at each pass will be similar to those of running GSP on the whole database. In the bottom-up phase of MSPX, since most infrequent candidates can be excluded at each pass as discussed before, the subsequence infrequency based pruning can be performed as effectively as in GSP. Thus, in this phase, the number of passes and the candidate set size at each pass are also similar to the case of running GSP on the whole database. Therefore, the computation costs for step 3 in both single-sample methods and our multi-sample method are actually very close to each other.

The overhead for sampling in both types of methods is actually determined by steps 2 and 3. Our tests showed that MSPX incurs a little more overhead than single-sample methods, but not much. Overall, MSPX still demonstrates its advantage in terms of the average performance, the worst-case performance and the stability in performance. Some relevant test results will be shown in the following performance analysis section.

5 Performance Analysis

To compare MSPX with other algorithms, we implemented GSP and obtained the source codes of SPAM and SPADE from their authors' web sites. In addition, we are also interested in the comparison between multi-sample method MSPX and the traditional single-

sample methods. Thus, we implemented a variant of GSP, which will be called GSP-Samp in this paper, by integrating the sampling technique in a traditional way: using GSP to mine a random sample with respect to minsup first, validating sample results to remove false patterns, then performing GSP on the database and using the longest frequent sequences found in the sample to prune candidates at each pass.

All the experiments were performed on a SuSE Linux PC with a 2.6 GHz Pentium processor and 1 Gbytes main memory. For MSPX and GSP-Samp, since sampling technique is probabilistic, we repeated each test 100 times. The average execution time of the 100 runs was reported as the performance result. The default sample size for MSPX and GSP-Samp was fixed as 10% of the test database for all experiments. For MSPX, the support level θ for the sample is computed using the formula (4.2). The upper bound of the probability that an infrequent candidate is overestimated at two consecutive passes is set as 20%, i.e. $\sigma = 0.2$. Thus, we have $Z_{(1-\sqrt{\sigma})} = Z_{0.55} = 0.13$. The databases used in our experiments are synthetically generated as in [2]. The database generation parameters are described in Table 1. For all databases, $N_S = 5000$ and $N_I = 25,000$; and the names of the databases reflect other parameter values used to generate them.

Table 1: Parameters Used in Database Generation

D	Number of customers in the database
C	Average number of transactions per customer
T	Average number of items per transaction
S	Average length of maximal potentially frequent sequences
I	Average length of maximal potentially frequent itemsets
N	Number of distinct items in the database
N_S	Number of maximal potentially frequent sequences
N_I	Number of maximal potentially frequent itemsets

5.1 Performance Comparison. We ran MSPX, GSP, SPADE and SPAM on databases with medium sizes of about 100 Mbytes. The number of items in these databases is 10,000. The test results on D400K-C10-T5-S10-I2.5-N10K database are presented in Figure 3. In our tests, SPAM could not mine these databases, and its run was terminated by the operating system. Our machine is a 32-bit system, but the user address space is limited to 2 Gbytes. In all these tests, SPAM always required more than 2 Gbytes memory, and hence caused the termination.

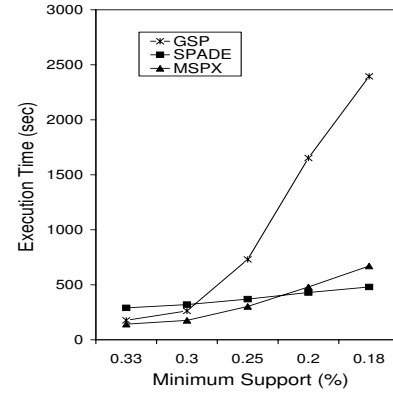
With the optimization components integrated, MSPX performs much better than GSP because it processes fewer candidates in a much more efficient way.

When the minsup is decreased, more and more candidates appear during the mining. In that case, the overhead of GSP in candidate generation, pruning, and especially counting using a huge hash tree increases drastically. For MSPX, this situation is considerably improved by using the supersequence frequency based pruning, the prefix tree structure, and the customer sequence trimming. At each pass in the bottom-up phase of MSPX, only a part of the candidates are selected to be counted on the whole database. As most infrequent sequences were identified early, the situation that too many false candidates are generated did not happen in all the tests. In the top-down phase of MSPX, the search starts with the potential maximal frequent sequences. Once a maximal frequent sequence is found, all of its subsequences are removed from the search space. Thus, the total number of candidates being counted on the whole database is much smaller than that of GSP. Figure 3(b) shows how many candidates with length greater than 2 have been counted on the whole databases in GSP and MSPX.

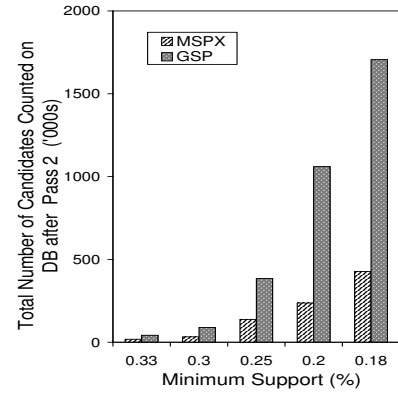
In SPADE, the subsequence infrequency based pruning is performed only partially. Compared with GSP, SPADE is expected to process more candidates. The main advantage of SPADE is the efficient counting of the candidates by intersecting the id-lists. An inefficient part of SPADE is the counting of C_2 for medium and large databases in the vertical format, which degrades the whole performance of SPADE very much. Considering both factors, we can say that if there are not enough number of candidates of length greater than 2 to be counted, SPADE cannot show its efficiency. That is why SPADE is even worse than GSP when minsup is big as shown in Figure 3(a). In these tests, MSPX performed best for large and medium minsups. Only when the minsup is very small, SPADE performed best.

5.2 Scalability Evaluation. Both SPADE and SPAM need to store a huge amount of intermediate data to save their computation cost. When the memory space requirement is over the memory size available, CPU utilization drops quickly due to the frequent swapping. Compared with them, MSPX and GSP process the customer sequences one by one, hence only a small memory space is needed to buffer the customer sequences being processed. MSPX can also handle the situation that L_k^* or C_k cannot be totally loaded into memory by using the signatures as explained in Section 4. Therefore, MSPX does not require the memory space as much as GSP, SPADE and SPAM.

Many real-life customer market-basket databases have tens of thousands of items and millions of customers, so we evaluated the scalability of the mining al-



(a) Performance



(b) Search Space (MSPX vs. GSP)

Figure 3: Tests on D400K-C10-T5-S10-I2.5-N10K

gorithms in these two aspects. First, we started with a very small database D1K-C10-T5-S10-I2.5 and changed the number of items from 500 to 10,000. The user-specified minsup was 0.5%. To run MSPX on such a small database with only 1000 customers, we selected the whole database as the sample and set θ to minsup. Since MSPX does not apply the sampling on such a small database, supersequence frequency based pruning is not performed in mining. Thus, in this case, SPADE and SPAM performed better than MSPX and GSP as long as their memory requirement is satisfied.

As the number of items is increased, SPAM shows its scalability problem. Theoretically, the memory space required to store the whole database into bitmaps in SPAM is $D * C * N / 8$ bytes. For the id-lists in SPADE, it is about $D * C * T * 4$ bytes. But we found these values are usually far less than their peak memory space requirement during the mining, because the amount of intermediate data in both algorithms is quite large.

As shown in Figure 4, even though the D1K-C10-T5-S10-I2.5-N8000 database takes only 260 Kbytes, and the theoretical memory space requirement to store the database in SPAM is about $1000 * 10 * 8000 / 8$ bytes ≈ 10 Mbytes, it could not finish the mining when the minsup was 0.5%, because it required more than 2 Gbytes of memory. Compared with SPAM, SPADE divides search space into small pieces so that only the id-lists being processed need to be loaded into memory. Another advantage of SPADE is that the id-lists become shorter and shorter with the progress in mining, whereas the length of the bitmaps does not change in SPAM. These two differences make SPADE much more space-efficient than SPAM. We also fixed the parameter N as 1000 and changed the database size from 1K to 100K customer sequences. SPAM could not mine the databases with more than 20K customers due to its memory requirement problem. Our tests showed that SPAM is very sensitive to the number of items and the number of customers, which mainly limits its applicability.

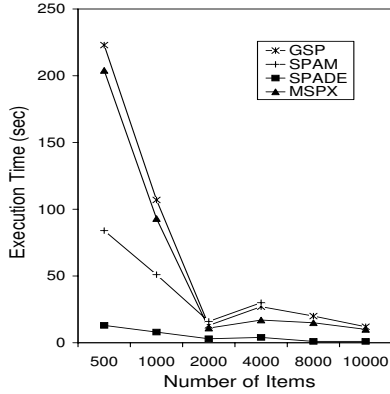


Figure 4: Scalability: Number of Items (on D1K-C10-T5-S10-I2.5, minsup=0.5%)

Second, we investigated how they perform on C10-T5-S10-I2.5-N10K when the user-specified minsup minsup is 0.18%. We fixed the number of items as 10,000 and increased the number of customers from 400,000 to 2,000,000. SPAM could not perform the mining due to its memory requirement problem. For SPADE, we partitioned the test database into multiple chunks for better performance when its size was increased. Otherwise, the counting of C_2 for a large database could be extremely time-consuming. We made each chunk contain 400,000 customers so that its size is only about 100 Mbytes, which is one tenth of our main memory size. Thus, D400K-C10-T5-S10-I2.5-N10K is processed as one chunk, D800K-C10-T5-S10-I2.5-N10K is divided into two chunks, and so on. Figure 5 shows that the scalability of MSPX and GSP are quite linear. But

SPADE cannot maintain a reasonable scalability as the database becomes larger. As the database size is increased, MSPX performs much better than the others.

When the database was relatively small with only 400,000 customers, SPADE performed best — about 20% faster than MSPX. But when the database size is increased from 1600K customers to 2000K customers, there is a sharp performance drop in SPADE, such that it is even slower than GSP. In that case, MSPX is faster than SPADE by a factor of about 8. As discussed before, counting C_2 is a performance bottleneck for SPADE, because the transformation of a large database from the vertical format to the horizontal format takes too much time. When the database is very large, the transformation also requires a large amount of memory and frequent swapping, hence the performance drops drastically. Partitioning the database can relieve this problem to some extent but does not solve it completely. Moreover, for the database with a large number of items and customers, SPADE needs more time to intersect more and longer id-lists.

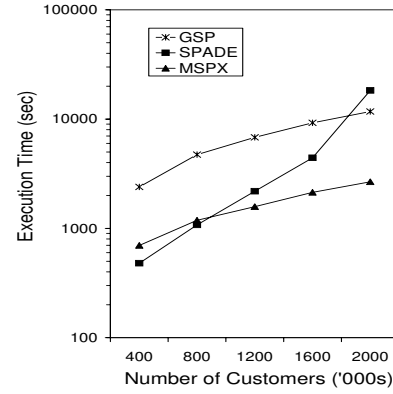


Figure 5: Scalability: Number of Customers (on C10-T5-S10-I2.5-N10K, minsup=0.18%)

Finally, we mined a large database D2000K-C10-T5-S10-I2.5-N10K, which takes about 500 Mbytes, for various minsup. This database was partitioned into 5 chunks for SPADE, and the results are shown in Figure 6. Based on our tests, we found SPADE performs best for small size databases. For medium size databases, MSPX performs better for relatively big minsup while SPADE is faster for small minsup. When the database is large, SPADE's performance drops drastically, and MSPX outperforms SPADE very much.

5.3 Multi-Sample MSPX vs. Single-Sample GSP-Samp. As far as we know, all the previous researches on the association rule mining based on the sampling used a single random sample or refined a

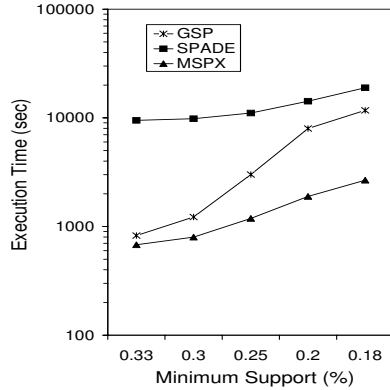


Figure 6: Performance on a Large Database D2000K-C10-T5-S10-I2.5-N10K

single big sample to a smaller one. In MSPX, multiple samples are used. We compared MSPX and GSP-Samp to see if multiple samples can avoid or alleviate the problems inherent in the single-sample methods discussed earlier. To exclude other factors affecting the performance of GSP-Samp, the signature based subsequence infrequency pruning, the prefix tree and the customer sequence trimming techniques were also used for the implementation of GSP-Samp.

Compared with GSP-Samp, MSPX has better average performance. Most importantly, the performance variance of MSPX is much smaller than that of GSP-Samp. The worst performance of MSPX also indicates that even if a few bad samples had been drawn, MSPX could successfully suppress their negative effect. Otherwise, the worst case of MSPX could have been much worse than what we observed, probably similar to the worst case of GSP-Samp. This proves that MSPX is not sensitive to a couple of bad samples because of the contribution of multiple samples.

6 Conclusions and Future Work

In this paper, we proposed an algorithm named MSPX, which mines maximal frequent sequences by effectively excluding infrequent candidates. Multiple samples are used in MSPX to avoid or alleviate some problems inherent in the algorithms using only one sample. For MSPX, we explored the relationship between the increment of the user-specified minsup for the sample and the probability of an overestimate. A theoretical guideline is given to increase the minsup for the sample in the context of multiple samples. Our extensive experiments proved that MSPX is a practical and efficient algorithm. Its excellent scalability makes it a very good candidate for mining customer market-basket databases which usually have tens of thousands of items and millions of customer sequences. More importantly, even

though MSPX is a sampling-based algorithm, the variance of its performance during multiple runs for the same mining task is usually very small. Applying the proposed idea of effectively excluding infrequent candidates and the multiple sampling technique to other sequence mining algorithms will be an interesting project.

References

- [1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. of the 20th VLDB Conf.*, 1994, pp. 487–499.
- [2] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. of Int'l Conf. on Data Engineering*, 1995, pp. 3–14.
- [3] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential Pattern Mining Using a Bitmap Representation," *Proc. of ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, 2002, pp. 429–435.
- [4] S. M. Chung and C. Luo, "Distributed Mining of Maximal Frequent Itemsets from Databases on a Cluster of Workstations," *Proc. of the 4th IEEE/ACM Int'l Symp. on Cluster Computing and the Grid – CCGrid 2004*, IEEE Computer Society Press, 2004.
- [5] F. Massegia, F. Cathala, and P. Poncelet, "The PSP Approach for Mining Sequential Patterns," *Proc. of European Symp. on Principle of Data Mining and Knowledge Discovery*, 1998, pp. 176–184.
- [6] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proc. of the 5th Int'l Conf. on Extending Database Technology*, 1996, pp. 3–17.
- [7] H. Toivonen, "Sampling Large Databases for Association Rules," *Proc. of the 22nd VLDB Conf.*, 1996, pp. 134–145.
- [8] M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara, "Evaluation of Sampling for Data Mining of Association Rules," *Proc. of the 7th Int'l Workshop on Research Issues in Data Engineering*, 1997.
- [9] M. J. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," *Machine Learning*, 42(1), 2001, pp. 31–60.

Gaussian Processes for Active Data Mining of Spatial Aggregates

Naren Ramakrishnan[†], Chris Bailey-Kellogg[#], Satish Tadepalli[†], and Varun N. Pandey[†]

[†]Department of Computer Science, Virginia Tech, Blacksburg, VA 24061

[#]Department of Computer Science, Dartmouth College, Hanover, NH 03755

Abstract

Active data mining is becoming prevalent in applications requiring focused sampling of data relevant to a high-level mining objective. It is especially pertinent in scientific and engineering applications where we seek to characterize a configuration space or design space in terms of spatial aggregates, and where data collection can become costly. Examples abound in domains such as aircraft design, wireless system simulation, fluid dynamics, and sensor networks. This paper develops an active mining mechanism, using Gaussian processes, for uncovering spatial aggregates from only a sparse set of targeted samples. Gaussian processes provide a unifying framework for building surrogate models from sparse data, reasoning about the uncertainty of estimation at unsampled points, and formulating objective criteria for closing-the-loop between data collection and data mining. Our mechanism optimizes sample selection using entropy-based functionals defined over spatial aggregates instead of the traditional approach of sampling to minimize estimated variance. We apply this mechanism on a global optimization benchmark comprising a testbank of 2D functions, as well as on data from wireless system simulations. The results reveal that the proposed sampling strategy makes more judicious use of data points by selecting locations that clarify high-level structures in data, rather than choosing points that merely improve quality of function approximation.

Keywords: spatial mining, active mining, sparse data, spatial aggregation, Gaussian processes.

1 Introduction

Many data mining applications in scientific and engineering contexts require analysis and mining of spatial datasets derived from computer simulations or field data, e.g., wireless system simulations, aircraft design configuration spaces, fluid dynamics simulations, and sensor network optimization. In contrast to traditional data mining contexts that are dominated by massive datasets, these domains are actually characterized by a *paucity* of data, owing to the cost and time involved

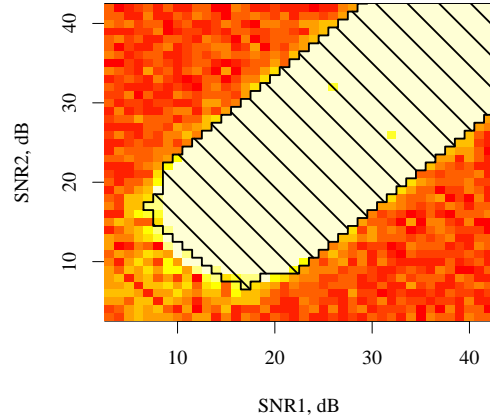


Figure 1: Mining configuration spaces from wireless system simulations. The shaded region denotes the largest portion of the configuration space where we can claim, with confidence at least 99%, that the average bit error rate (BER) is acceptable for voice-based system usage. Each ‘cell’ in the plot is the result of the spatial and temporal aggregation of hundreds of wireless system simulations, many of which take hours.

in conducting simulations or setting up experimental apparatus for data collection. Nevertheless, the computational scientist has control of *where* data can be collected; it is hence prudent in such domains to focus data collection in only those regions that are deemed important to support a high-level data mining objective.

As a concrete example, consider the characterization of WCDMA (wideband code-division multiple access) wireless system configurations for a given indoor environment. A configuration comprises many adjustable parameters, and the goal of wireless system characterization is to assess the relationship between these parameters and performance metrics such as BER (bit error rate), a measure of the number of bits transmitted in error using the system. When a wireless engineer designs a system for a given indoor environment, he or she sets an acceptable performance criterion for BER (e.g., 10^{-3} for a system designed to carry voice traffic, stricter thresholds for data traffic) and seeks a region

in the configuration space that can satisfy this criterion (see Fig. 1). To collect the data necessary for mining configuration spaces, the engineer either performs a costly Monte Carlo simulation (where a model of radio propagation in the wireless channel is embedded inside a system-wide model encapsulating wireless protocols and communications standards), or installs channel sounding equipment and system instrumentation in the environment, and actually enacts usage scenarios. In either approach, it is not feasible to first organize a voluminous body of data and subsequently perform data mining on the collected dataset. It is thus imperative that we interleave data collection and data mining and focus sampling at only those locations that maximize well-defined notions of relevance and utility. Importantly, we will not need to sample the entire configuration space, only enough so as to identify a region with acceptable confidence.

Active data selection has been investigated in a variety of contexts [4, 25]. A sampling strategy typically embodies a human assessment of where might be a good location to collect data [1, 13] or is derived from the optimization of specific design criteria [5, 17, 22]. Many of these strategies, however, are either based on utility of data for function approximation purposes [24], or are meant to be used with specific data mining algorithms and tasks (e.g., classification [10]). In this paper, we present a formal framework that casts spatial data mining as uncovering successive multi-level aggregates of data, and uses properties of higher-level structures to help close the loop between mining and data collection. This approach helps us design sampling strategies that bridge higher-level quality metrics of structures (e.g., entropy) with lower-level considerations of data samples (e.g., locations and fidelity). While we focus on spatial contexts, we point out that *spatial* can denote any dimension that affords a metric; our approach thus applies equally well to a wide range of data sets with more abstract notions of space (such as the wireless simulation example above).

Our active mining mechanism is based on the spatial aggregation language (SAL; [3]), a generic data mining framework for spatial datasets, and Gaussian processes (GPs; [27]), a powerful unifying theory for approximating and reasoning about datasets. Gaussian processes provide the ‘glue’ that enables us to perform active mining on spatial aggregates. In particular, they aid in (i) creation of *surrogate models* from data using a sparse set of samples (for cheap generation of dense approximate datasets), (ii) reasoning about the uncertainty of estimation at unsampled points, and (iii) formulation of objective criteria for active data collection.

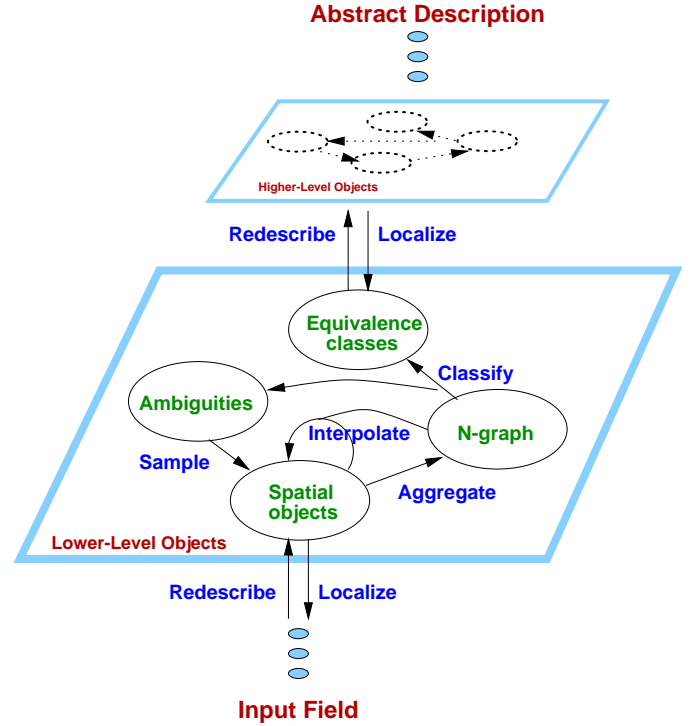


Figure 2: SAL uncovers multi-layer spatial aggregates by employing a small set of operators (a spatial mining “vocabulary”) utilizing suitable domain knowledge. A variety of spatial data mining tasks, such as vector field bundling, contour aggregation, correspondence abstraction, clustering, and uncovering regions of uniformity can be expressed as multi-level spatial aggregate computations.

1.1 Contributions: This paper builds on our prior work in [1, 23] by presenting a novel integration of Gaussian processes with SAL:

- While classical active mining work in spatial modeling focuses on quality of function approximation, the mechanism presented here focuses on clarifying high-level structures. The entropy-based sampling approach introduced in this paper is applicable to mining a broad range of spatial structures.
- Unlike traditional data mining contexts that deal with voluminous amounts of data, the mechanism is targeted at scenarios where data collection costs far outshadow data mining costs. For instance, in the wireless simulation study, each data sample requires a few hours to acquire on a cluster of workstations whereas the data mining (and sample selection optimization) algorithms as implemented here can be executed in a matter of minutes on a workstation.

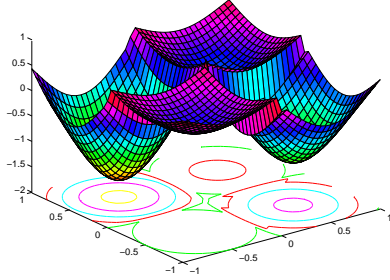


Figure 3: de Boor’s ‘pocket’ function in 2D, depicting contours around basins of local minima.

- Since Gaussian processes are re-statements of kernel-based learning methods [7] this work helps bridge the qualitative nature of SAL algorithms with rigorous quantitative methodologies necessary to evaluate and assess active mining strategies.

This work assumes a moderate background of algorithms for spatial aggregation and spatial statistical modeling. Nevertheless, Sec. 2 and Sec. 3 overview earlier work on SAL and GPs for the SIAM DM audience and instantiate such work for a motivating case study — identifying and characterizing pockets in a space (such as the wireless system design configuration space). Sec. 4 then moves to active mining and introduces two important classes of sampling strategies integrating SAL and GPs. Sec. 5 evaluates the mechanism using both synthetic and real-world datasets. Sec. 6 provides a discussion and reviews related work.

2 Spatial Aggregation Language

The Spatial Aggregation Language (SAL) [3, 28, 30] is a generic framework to study the design and implementation of spatial data mining algorithms. SAL is centered on a *field ontology*, in which the spatial data input is a field mapping from one continuum to another (e.g. 2-D temperature field: $\mathbf{R}^2 \rightarrow \mathbf{R}^1$; 3-D fluid flow field: $\mathbf{R}^3 \rightarrow \mathbf{R}^3$). SAL programs employ vision-like routines, in an *imagistic reasoning* style [29], to uncover and manipulate multi-layer geometric and topological structures in fields. Due to continuity, fields exhibit regions of uniformity, and these regions can be abstracted as higher-level structures which in turn exhibit their own continuities. Task-specific domain knowledge specifies how to uncover such uniformity, defining metrics for closeness of objects and similarity of features. For example, streamlines are connected curves of nearby points with vectors flowing in similar enough directions, while pressure cells are connected regions of similar (and extreme) enough pressure.

SAL supports structure discovery through a small set of generic operators, parameterized with domain-

specific knowledge, on uniform data types. These operators and data types mediate increasingly abstract descriptions of the input data (see Fig. 2) to form higher-level abstractions and mine patterns. The *primitives* in SAL are contiguous regions of space called *spatial objects*; the *compounds* are (possibly structured) collections of spatial objects; the *abstraction mechanisms* connect collections at one level of abstraction with single objects at a higher level. This vocabulary has proved effective for expressing the mechanisms required to uncover multi-level structures in spatial datasets in applications ranging from decentralized control design [2] and object manipulation [30] to analysis of weather data [12], diffusion-reaction morphogenesis [21], and matrix perturbation analysis [22].

The identification of structures in a field is a form of data reduction: a relatively information-rich field representation is abstracted into a more concise structural representation (e.g. pressure data points into isobar curves or pressure cells; isobar curve segments into troughs). Navigating the mapping from field to abstract description through multiple layers rather than in one giant step allows the construction of modular data mining programs with manageable pieces that can use similar processing techniques at different levels of abstraction. The multi-level mapping also allows higher-level layers to use global properties of lower-level objects as local properties of the higher-level objects. For example, the average temperature in a region is a global property when considered with respect to the temperature data points, but a local property when considered with respect to a more abstract region description. As this paper demonstrates, analysis of higher-level structures in such a hierarchy can guide interpretation of lower-level data.

Let us begin with a spatial mining task motivated by the wireless study — determining the number and locations of *pockets*, or basins of local minima, in a vector field. Fig. 3 illustrates four pockets in a field defined by Carl de Boor’s function in 2D (from [22]):

$$(2.1) \quad \alpha(\mathbf{X}) = \cos \left(\sum_{i=1}^n 2^i \left(1 + \frac{x_i}{|x_i|} \right) \right) - 2$$

$$(2.2) \quad \delta(\mathbf{X}) = \|\mathbf{X} - 0.5\mathbf{I}\|$$

$$(2.3) \quad p(\mathbf{X}) = \alpha(\mathbf{X})(1 - \delta^2(\mathbf{X})(3 - 2\delta(\mathbf{X}))) + 1$$

where \mathbf{X} is the n -dimensional point (x_1, x_2, \dots, x_n) at which the pocket function p is evaluated, \mathbf{I} is the identity n -vector, and $\|\cdot\|$ is the L_2 norm. The property of this function is that it assumes a pocket in each corner of the cube, just outside the sphere enclosed in the cube. Since the ratio of cube volume (2^n) to that of the sphere ($\pi^{n/2}/(n/2)!$) grows unboundedly, global optimization

algorithms cannot exploit any special properties and must consider every one of the 2^n corners! Hence, the de Boor function is a well-known benchmark for global optimization (esp. in high dimensions), but we focus here on a somewhat different objective of characterizing the high-level structure of the field. The algorithmic encoding of the calculus definition of local minima suggests that the four pockets in Fig. 3 can be identified via convergent flows in the gradient underlying the vector field. Let us assume we are given a dense set of samples covering the region of interest. Fig. 4 illustrates an example of key spatial aggregation operations:

- (a) Establish the input field, here by calculating the gradient field (normalized, since we’re interested only in direction in order to detect convergence).
- (b) Localize computation with a *neighborhood graph*, so that only spatially proximate objects are compared. Here, an 8-adjacency neighborhood graph is employed, which results in somewhat ‘blocky’ streamlines but fast computation.
- (c)-(f) Use a series of local computations to find *equivalence classes* of neighboring objects with similar features. Here, we systematically eliminate all neighborhood graph edges but those whose directions best match the vector direction at both endpoints. ‘Forward neighbor’ computation compares graph edge direction with the average of the vector directions, and keeps only those that are similar enough (implemented as a cosine angle similarity threshold). ‘Best forward neighbor’ at junction points then selects from among these neighbors, by a third metric combining similarity in direction with closeness in point location. Backward calculations are analogous, but deal with the predecessor along a streamline rather than the successor.
- (g) Move up a level in the spatial object hierarchy by *redescribing* equivalence classes into more abstract objects. Here, connected vectors are abstracted into curve objects, which have both a reduced representation and additional semantic properties (e.g. curvature is well-defined).
- (h) Apply the same mechanism — aggregate, classify, and redescribe — at the new level, *using the exact same operators but with different metrics*. Here, curves are grouped into coherent pockets with convergent flow. Neighborhood (not shown) is derived from neighborhood of constituent vectors, and equivalence tests direction of flow for convergence.

Notice that SAL is not a specific data mining algorithm, but rather a language to construct complex mining operations (such as in Fig. 4) from a small core set of operations. As such, the quality of results from a SAL implementation depends on suitable choices of abstraction levels and appropriate settings of any relevant parameters. For instance, in the above example, three parameters control the relationship from input field to output structures: adjacency neighborhood size (used in step (b)), angle for vector similarity (used in step (c)), and distance penalty metric (used in step (d) to combine distance with direction). For Fig. 4, we set these parameters to 1.5 (generates an 8-adjacency neighborhood), 0.75, and 0.1 respectively. This paper is not concerned with evaluating particular SAL implementations but instead focuses on using them from within an active mining context.

Localized computations are integral to SAL, and hence an effective SAL application relies on a dense set of samples covering the domain. When data is scarce, we can first build an approximation to the underlying field with the given samples, and use the approximation to generate a dense field of data (e.g., on a uniform grid). Such an approximation is called a *surrogate* model — cheap-to-compute substitutes for complex functions. One way to build surrogate models relies on Gaussian processes.

3 Gaussian Processes

The use of Gaussian processes in machine learning and data mining is a relatively new development, although their origins can be traced to spatial statistics and the practice of modeling known as kriging [14]. In contrast to global approximation techniques such as least-squares fitting, GPs are local approximation techniques, akin to nearest-neighbor procedures. In contrast to function approximation techniques that place a prior on the form of the function, GP modeling techniques place a prior on *the covariance structures* underlying the data.

The basic idea in GPs is to model a given dataset as a realization of a stochastic process. Formally, a GP is a set of random variables any finite subset of which have a (multivariate) normal distribution. For our purposes, we can think of these variables as spatially distributed (scalar) response variables t_i , one for each 2D location $\mathbf{x}_i = [x_{i1}, x_{i2}]$ where we have collected a data sample. In our vector field analysis application, t_i denotes the modeled response, i.e., the value of de Boor’s function at \mathbf{x}_i . Given a dataset $\mathcal{D} = \{\mathbf{x}_i, t_i\}, i = 1 \dots n$, and a new data point \mathbf{x}_{n+1} , a GP can be used to model the posterior $P(t_{n+1}|\mathcal{D}, \mathbf{x}_{n+1})$ (which would also be a Gaussian). This is essentially what many Bayesian modeling techniques do (e.g., least squares

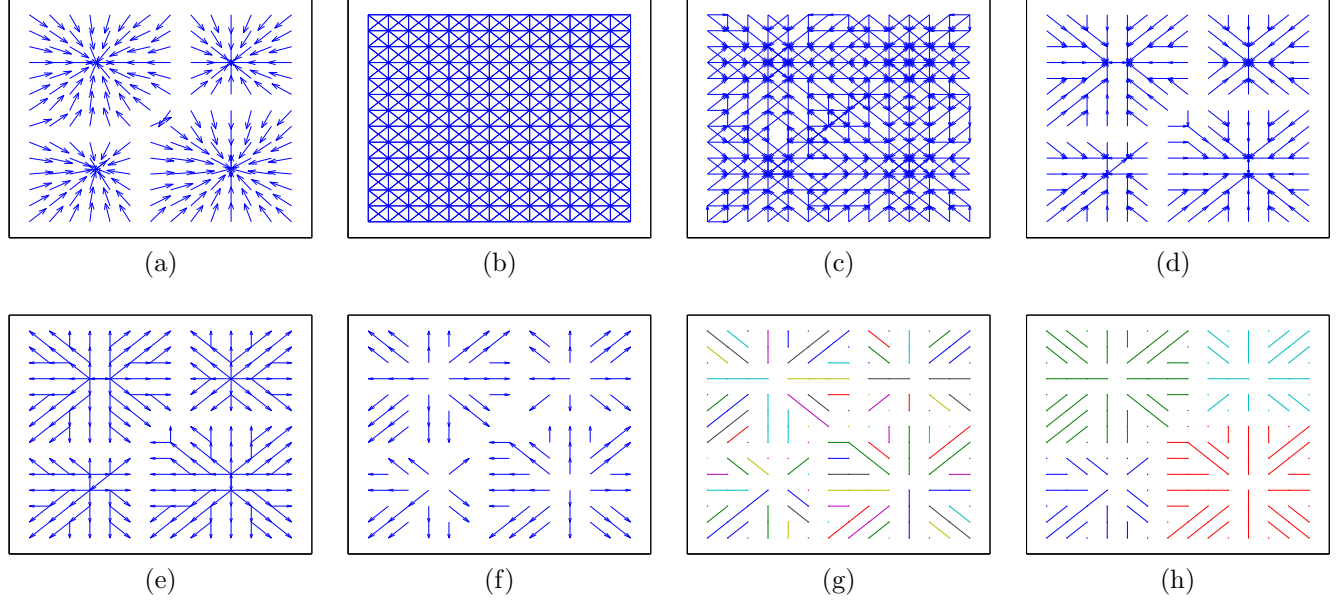


Figure 4: Example steps in SAL implementation of vector field analysis of de Boor's function. (a) Input vector field. (b) 8-adjacency neighborhood graph. (c) Forward neighbors. (d) Best forward neighbors. (e) Neighborhood graph transposed from best forward neighbors. (f) Best backward neighbors. (g) Resulting adjacencies redescribed as curves. (h) Higher-level aggregation and classification of curves whose flows converge.

approximation with normally distributed noise) but it is the specifics of how the posterior is modeled that make GPs distinct as a class of modeling techniques.

To make a prediction of t_{n+1} at a point \mathbf{x}_{n+1} , GPs place greater reliance on t_i 's from nearby points. This reliance is specified in the form of a covariance prior for the process and will be central to how we embed SAL in a broader GP framework:

$$(3.4) \text{Cov}(t_i, t_j) = \alpha \exp \left(-\frac{1}{2} \sum_{k=1}^2 a_k (x_{ik} - x_{jk})^2 \right)$$

Intuitively, this function captures the notion that response variables at nearby points must have high correlation. The reader will note that this idea of influence decaying with distance has an immediate parallel to how SAL programs localize computations. In Eq. 3.4, α is an overall scaling term whereas a_1, a_2 define the length scales for the two dimensions. However, this prior (or even its posterior) does not directly allow us to determine t_j from t_i , since the structure only captures the covariance; predictions of a response variable for new sample locations are thus conditionally dependent on the measured response variables and *their* sample locations. Hence, we must first estimate the covariance parameters (a_1, a_2 , and α) from \mathcal{D} and then use these parameters *along with* \mathcal{D} to predict t_{n+1} at \mathbf{x}_{n+1} .

3.1 Using a GP: Before covering the learning procedure for the covariance parameters (a_1, a_2 , and α), it

is helpful to develop expressions for the posterior of the response variable in terms of these parameters. Since the jpdf of the response variables $P(t_1, t_2, \dots, t_{n+1})$ is modeled Gaussian (we will assume a mean of zero), we can write:

$$P(t_1, t_2, \dots, t_{n+1} | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}, \text{Cov}_{n+1}) = \frac{1}{\lambda_1} \exp \left(-\frac{1}{2} [t_1, t_2, \dots, t_{n+1}] \text{Cov}_{n+1}^{-1} [t_1, t_2, \dots, t_{n+1}]^T \right)$$

where we ignore λ_1 as it is simply a normalizing factor. Here, Cov_{n+1} is the covariance matrix formed from the $(n+1)$ data values ($\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}$). A distribution for the unknown variable t_{n+1} can then be obtained as:

$$\begin{aligned} P(t_{n+1} | t_1, t_2, \dots, t_n, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}, \text{Cov}_{n+1}) \\ &= \frac{P(t_1, t_2, \dots, t_{n+1} | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}, \text{Cov}_{n+1})}{P(t_1, t_2, \dots, t_n | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}, \text{Cov}_{n+1})} \\ &= \frac{P(t_1, t_2, \dots, t_{n+1} | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}, \text{Cov}_{n+1})}{P(t_1, t_2, \dots, t_n | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \text{Cov}_n)} \end{aligned}$$

where the last step follows by conditional independence of $\{t_1, t_2, \dots, t_n\}$ w.r.t. \mathbf{x}_{n+1} and the part of Cov_{n+1} not contained in Cov_n . The denominator in the above expression is another Gaussian random variable, given by:

$$\begin{aligned} P(t_1, t_2, \dots, t_n | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \text{Cov}_n) = \\ \frac{1}{\lambda_2} \exp \left(-\frac{1}{2} [t_1, t_2, \dots, t_n] \text{Cov}_n^{-1} [t_1, t_2, \dots, t_n]^T \right) \end{aligned}$$

Putting it all together, we get:

$$P(t_{n+1}|t_1, t_2, \dots, t_n, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}, \text{Cov}_{n+1}) = \frac{\lambda_2}{\lambda_1} \exp \left(-\frac{1}{2} [t_1, t_2, \dots, t_{n+1}] \text{Cov}_{n+1}^{-1} [t_1, t_2, \dots, t_{n+1}]^T - \frac{1}{2} [t_1, t_2, \dots, t_n] \text{Cov}_n^{-1} [t_1, t_2, \dots, t_n]^T \right)$$

Computing the mean and variance of this Gaussian distribution, we get an estimate of t_{n+1} as:

$$(3.5) \quad \hat{t}_{n+1} = \mathbf{k}^T \text{Cov}_n^{-1} [t_1, t_2, \dots, t_n]$$

and our uncertainty in this estimate as:

$$(3.6) \quad \sigma_{\hat{t}_{n+1}}^2 = k - \mathbf{k}^T \text{Cov}_n^{-1} \mathbf{k}$$

where \mathbf{k}^T represents the n -vector of covariances with the new data point:

$$\mathbf{k}^T = [\text{Cov}(\mathbf{x}_1, \mathbf{x}_{n+1}) \text{Cov}(\mathbf{x}_2, \mathbf{x}_{n+1}) \dots \text{Cov}(\mathbf{x}_n, \mathbf{x}_{n+1})]$$

and k is the $(n+1, n+1)$ entry of Cov_{n+1} . Eqs. 3.5 and 3.6, together, give us both an approximation at any given point and an uncertainty in this approximation; they will serve as the basic building blocks for closing-the-loop between data modeling and higher level mining functionality.

The above expressions can be alternatively derived by positing a linear probabilistic model and optimizing for the MSE (mean squared error) between observed and predicted response values (e.g., see [24]). In this sense, the Gaussian process model considered here is also known as the BLUE (best linear unbiased estimator), but GPs are not restricted to linear combinations of basis functions.

To apply GP modeling to a given dataset, we must first ensure that the chosen covariance structure matches the data characteristics. We have chosen a stationary structure above under the assumption that the covariance is translation invariant. Various other functions have been studied in the literature (e.g., see [18, 19, 24]), all of which satisfy the required property of positive definiteness. The simplest covariance function yields a diagonal matrix, but this means that no data sample can have an influence on other locations, and the GP approach offers no particular advantages. In general, by placing a prior directly on the function space, GPs are appropriate for modeling ‘smooth’ functions. The terms a_1, a_2 capture how quickly the influence of a data sample decays in each direction and, thus, the length scales for smoothness.

An important point to note is that even though the GP realization is one of a random process, we can nevertheless build a GP model for deterministic functions

(like the de Boor’s function) by choosing a covariance structure that ensures the diagonal correlations to be 1 (i.e., perfect reproducibility when queried for a sample whose value is known). Also, the assumption of zero mean for the Gaussian process can be relaxed, by including a constant term (gives another parameter to be estimated) in the covariance formulation. This approach is used for our experimental studies.

3.2 Learning a GP: Learning the GP parameters $\theta = (a_1, a_2, \alpha)$ can be undertaken in the ML and MAP frameworks, or in the true Bayesian setting where we obtain a distribution over values. The log-likelihood for the parameters is given by:

$$\begin{aligned} \mathcal{L} &= \log P(t_1, t_2, \dots, t_n | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \theta) \\ &= c + \log P(\theta) - \frac{n}{2} \log(2\pi) - \frac{1}{2} \log | \text{Cov}_n | \\ &\quad - \frac{1}{2} [t_1, t_2, \dots, t_n] \text{Cov}_n^{-1} [t_1, t_2, \dots, t_n]^T \end{aligned}$$

To optimize for the parameters, we can compute partial derivatives of the log-likelihood for use with a conjugate gradient or other optimization algorithm:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial \log P(\theta)}{\partial \theta} \\ &\quad - \frac{1}{2} \text{tr} \left(\text{Cov}_n^{-1} \frac{\partial \text{Cov}_n^{-1}}{\partial \theta} \right) \\ &\quad + \frac{1}{2} [t_1, t_2, \dots, t_n] \text{Cov}_n^{-1} \frac{\partial \text{Cov}_n^{-1}}{\partial \theta} \text{Cov}_n^{-1} [t_1, t_2, \dots, t_n]^T \end{aligned}$$

where $\text{tr}(\cdot)$ denotes the trace function. In our running example, we need only estimate three parameters for θ , well within the purview of modern numerical optimization software. For larger numbers of parameters, we can resort to the use of MCMC methods [19].

4 Active Data Mining Strategies

The above section showed two important uses of GPs for spatial mining: designing a surrogate function for generating a dense field (via Eq. 3.5), and assessing uncertainties in our estimates of the function at unsampled points (using Eq. 3.6). We are now ready to formulate objective criteria for active data selection, a precursor to active mining.

4.1 Variance Reducing Designs: A simple strategy for sampling is to target locations to reduce our uncertainty in modeling, i.e., select the location that minimizes the posterior generalized variance of the function. This approach can be seen as optimizing sample

selection for the functional:

$$(4.7) \quad \Phi_V = \frac{1}{2} \log \left[\frac{\partial t}{\partial \theta} \right] \mathcal{H}^{-1} \left[\frac{\partial t}{\partial \theta} \right]^T$$

where $\left[\frac{\partial t}{\partial \theta} \right]$ is the (row) vector of sensitivities w.r.t. each GP parameter computed at a sample location, and \mathcal{H} is the Hessian (second order partial derivatives) of t , again w.r.t. the parameters. A straightforward derivation will show that optimizing Φ_V suggests a location whose ‘error bars’ σ^2 are highest.

To implement this strategy, we can adopt either a block design (optimize for K locations simultaneously), or apply it sequentially to determine one extra sampling location at a time. The former is appropriate when we can farm out function evaluations across nodes in a compute cluster, whereas the latter will track the design functional better. We adopt the sequential approach here; Fig. 5 shows this strategy for the pocket function of Fig. 3 and concomitant results from pocket mining of the surrogate model data. At each step, we determine the best sample location (from among unsampled locations on a regular grid of 21×21), build the GP model from the data collected thus far, and apply our SAL-based vector aggregation mechanism to the gradient field derived from the function values.

The initial design has one point in the center of each quadrant, and one at the center. Not surprisingly, we find a significant number (16) of basins in the gradient field. The next four points added are actually at the corners; this is because estimated variances are typically high toward the boundaries of an interpolation region. As MacKay points out [17], such a metric has a tendency to ‘repeatedly gather data at the edges of the input space.’ Continuing the sampling, we see that the 13-point design actually has the samples organized in a diagonal design (a layout that has been referred to as ‘whimsical’ [9]). The emphasis on overall quality of function approximation more than data mining is evident from the fact that it takes over 30 points before the SAL-based pocket finder can infer that there are four pockets. In further experiments not reported here, we have found that pushing the initial points outward (or inward) does not have any appreciable effect on future samplings, and the variance-based metric favors the outer envelope of the design space.

4.2 Entropy-Based Functionals: It is a classical result in experiment design (e.g., see [8]) that, for Gaussian priors, the variance-reducing design is actually equivalent to the design minimizing the (expected) posterior entropy of the distribution $t_{\overline{\mathcal{D}}|\mathcal{D}}$, where $\overline{\mathcal{D}}$ denotes the unsampled locations in \mathcal{D} . For a proof, see [16]. This criterion is also equivalent to the D-optimality de-

sign criterion in spatial statistics, under the assumption that the noise factor on all measurements is the same. MacKay generalizes this idea [17], and pre-specifies a collection of points requiring high-quality approximation; the goal then is to minimize entropy of data distribution w.r.t. these points. This strategy does not apply here since our understanding of which locations are relevant improves as active mining proceeds.

To develop a better active mining strategy, notice that our goal is the identification of regions defined by convergent flows. If we view the SAL program as an information processor that maps a data field into a class field (defined over the same underlying space), then the utility of sampling in a region is directly related to our inferential capabilities about the corresponding region in the class field. Intuitively, we should be more interested in samples that tell us something about the boundary between regions than those that capture the insides of a region, *even though the latter might have high variance in its current estimate*. Repeatedly sampling function values inside an already classified and abstracted region is not as useful as sampling to clarify an emerging boundary classification. This means that we must bridge high-level information about pockets from SAL into a preference of where to collect data.

An idea that suggests itself is to adopt variance-based design, but instead of minimizing the entropy of the data distribution, minimize the entropy of the class distribution as revealed by the SAL pocket finder. By positing a class distribution at each point, based on the class labels occupied by neighboring points, we achieve our goal of ranking locations along region boundaries higher. While this basic strategy appears reasonable, it will repeatedly gather information at the region boundaries, just as variance-based design repeatedly focuses on the edges. So a point with high entropy is a good location to sample only as long as the variance surrounding it is sufficiently high. As our confidence in the data value increases, our preference for this location should decrease even if the class entropy remains large (as it will, if it lies on a boundary). This suggests using class entropy to define a distribution $P_E(\mathbf{x})$ over points, and using that distribution to scale the variance-based design criterion:

$$(4.8) \quad \Phi_E = \frac{1}{2} \sum_{\mathbf{x}} P_E(\mathbf{x}) \log \left[\frac{\partial t}{\partial \theta} \right] \mathcal{H}^{-1} \left[\frac{\partial t}{\partial \theta} \right]^T$$

The expression inside the summation contains the same term as in Eq. 4.7 but is now evaluated across the design space and scaled by the amount of interest in location \mathbf{x} :

$$(4.9) \quad P_E(\mathbf{x}_i) \propto \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i)} P(C(\mathbf{x})) \log P(C(\mathbf{x}))$$

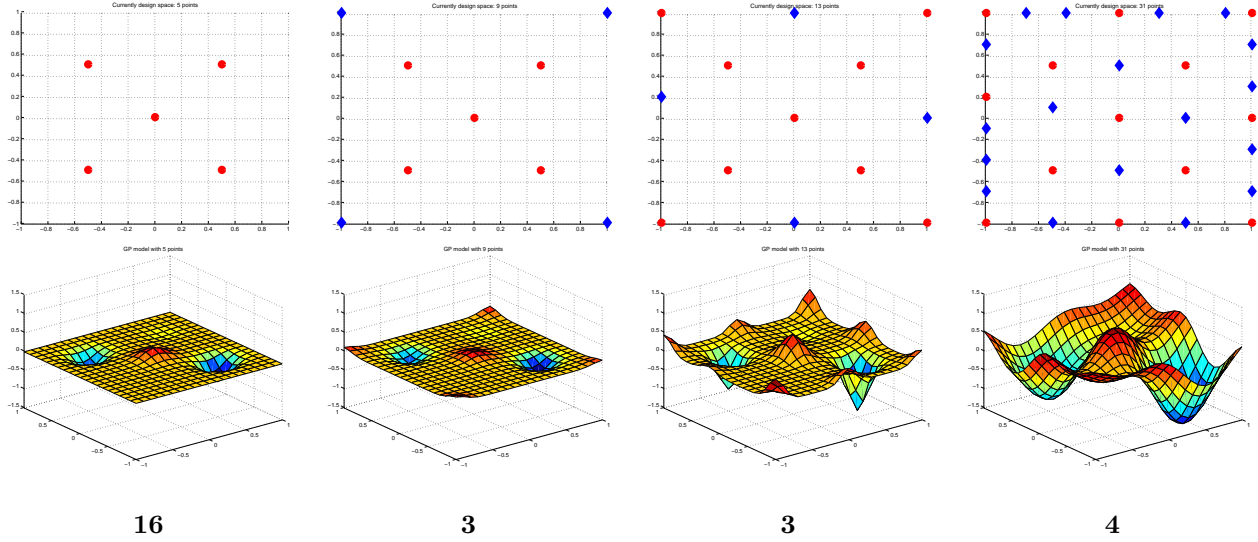


Figure 5: How variance-based sampling selects locations. (top row) initial design of 5 points, followed by snapshots taken at later stages (9, 13, and 31 points). Old sample locations are shown with red circles and new locations are shown with blue diamonds. (middle row) GP model fits to the given samples. (bottom row) Number of pockets identified by SAL pocket miner.

where $\mathcal{N}(\mathbf{x}_i)$ is a neighborhood around \mathbf{x}_i , $C(\mathbf{x})$ denotes the (flow) class of point \mathbf{x} as inferred by the SAL miner, and $P(C(\mathbf{x}))$ denotes the probability of encountering this class in the neighborhood. The proportionality constant in Eq. 4.9 must be set to ensure that $\sum P_E = 1$. Formal characterization of this criterion (i.e., convergence properties) is difficult since $P_E(\mathbf{x})$ changes during every iteration of data mining, and we do not have a model of how $P_E(\mathbf{x})$ varies across samplings. Operationally, to apply this criterion, we can identify the location that gives the highest information *gain*, given that we are intending to make a measurement at that location. Fig. 6 shows a design that optimizes Φ_E and successfully reveals all four pockets with only 11 points.

4.3 Computational Considerations: Other than any data collection costs, the primary costs to implementing the active mining mechanisms involve the nested optimizations and the necessary matrix computations. There are two optimizations per round of data collection: a multi-dimensional optimization over θ to fit the surrogate model, and a 2D optimization over \mathbf{x} to identify the next sample point. Both can be done either locally or globally, depending on our fidelity requirements and availability of resources. Here, to reduce the computational complexity in building the surrogate model, we adopt the public domain Netlab scaled conjugate gradient algorithm [18] which runs in $O(|\mathcal{D}||\theta|)$ time. While this algorithm avoids having to work with the Hessian explicitly, the active sample selection step requires the computation of the Hessian inverse, which

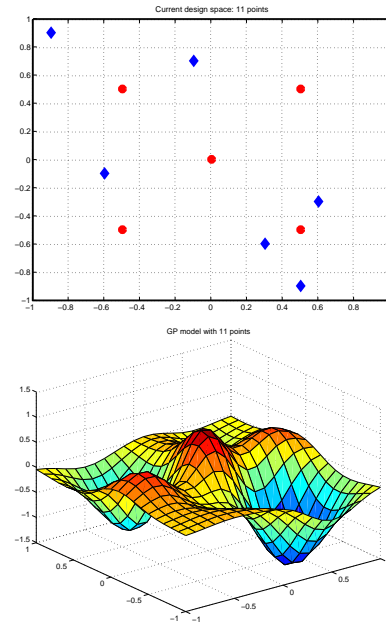


Figure 6: Active data mining with an entropy-based functional. This sampling strategy picks six additional points (top), in various quadrants; the SAL miner finds four pockets (bottom) when a GP model is constructed using the given points. Contrast this with the performance of variance-based sampling for a comparable number of samples.

takes $O(|\mathcal{D}||\theta|^2 + |\theta|^3)$ time. To reduce the cost of optimization, we use a discrete lattice search or hill climbing, restricting our attention to locations over a uniform grid. If the number of locations on the grid is $|G|$, then each round of active mining requires $O(|G||\theta|^2)$ time, plus the cost of computing the inverse Hessian. This expression applies to both variance-based mining and entropy-based mining, since the computation of $P_E(\mathbf{x})$ is just linear in $|G|$ for a fixed neighborhood calculation of entropy. Recall that this cost is typically negligible compared to the actual cost of running a simulation (to acquire a data sample).

4.4 Stopping Criteria: How do we know when to stop sampling? If a cost metric is defined over data collection, and if it can be determined that we can sample at most K points within the given resources, then we should ideally perform a K -dimensional optimization, rather than adopting a sequential sampling strategy. In the absence of such a cost-metric, a sampling strategy could terminate when the estimated dataset log likelihood is within bounds. In this paper, we primarily evaluate sampling strategies using classes of problems for which the ‘right’ answer is known, and pose questions such as: ‘starting from an initial grid, how many samples does it take to mine the right number of higher-level structures?’ The answer to this question gives us an indication of how aggressive the sampling strategy is, its stability (i.e., once mined, does it continue to mine the patterns?), and comparisons with the other strategy.

5 Experimental Results

We now present empirical results demonstrating the effectiveness of our active mining strategy on both synthetic and real datasets. We employed the Netlab suite of algorithms for GP modeling. Netlab supports a covariance formulation similar to Eq. 3.4, along with a bias term that overcomes our earlier assumption of zero mean. In addition, the model includes a noise term that can capture uncertainties in individual measurements; while this is not required for the deterministic functions considered here, it ensures that the numerical computation doesn’t become unstable. All GP parameters are given a relatively broad Gaussian prior. A surrogate model was fit on a regularly spaced grid (more below), with a limit of 100 iterations for conjugate gradient search. The SAL parameters were set to $(1.5, 0.75, 0.1)$, as before. The standard variance-based sampling has no adjustable parameters; a fixed 8-adjacency neighborhood was utilized for defining $P(\mathbf{x})$ in entropy-based sampling. Optimization for Φ_V and Φ_E was conducted over the same grid as the domain of the surrogate function.

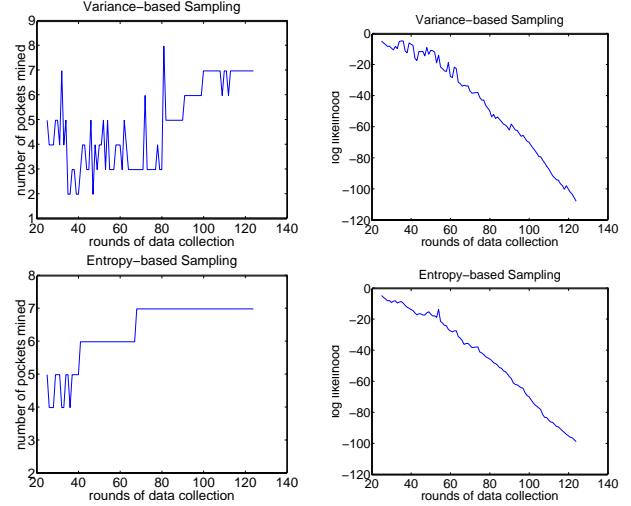


Figure 8: Pocket mining performance on the 7-pocket function from Fig. 7. (top) variance-based and (bottom) entropy-based sampling. (left) number of pockets found and (right) negative log-likelihood.

5.1 Synthetic Datasets: For the synthetic benchmark, we adopted the suite of test functions from [11], an ACM TOMS algorithm to readily generate classes of functions with known local and global minima. The algorithm systematically distorts a convex quadratic function with cubic or quintic polynomials to yield continuously differentiable (D-type) and twice continuously differentiable (D2-type) functions over the closed interval $[-1, 1]^2$. Since our active mining proceeds by discrete search over a pre-defined grid, we evaluated the generated functions over a regular 21×21 grid in $[-1, 1]^2$ ($|G| = 441$) and used these function values as the ‘oracle’ that is queried by the active mining mechanism. We verified whether in each instance, the SAL miner is able to resolve all pockets when given a complete 21×21 dataset. This is necessary because the radii of the basins of attraction interact with the spacing of the sampling grid, and hence influence the number of samples available for aggregation by the SAL miner. We found that the pocket miner is able to resolve only those generated functions that have up to 7 local minima; functions with more (e.g., 8–12) local minima use only a handful of points (typically 3–9) to represent some of their pockets, too few to be aggregated into a flow class under the SAL miner’s parameter settings. Hence, we pruned the automatically generated functions by requiring that each local minima have at least 12 samples per pocket, when sampled over the 21×21 grid. This yields a collection of 43 functions (21 D-type and 22 D2-type), with numbers of pockets ranging from 4 to 7. Fig. 7 depicts some of these functions.

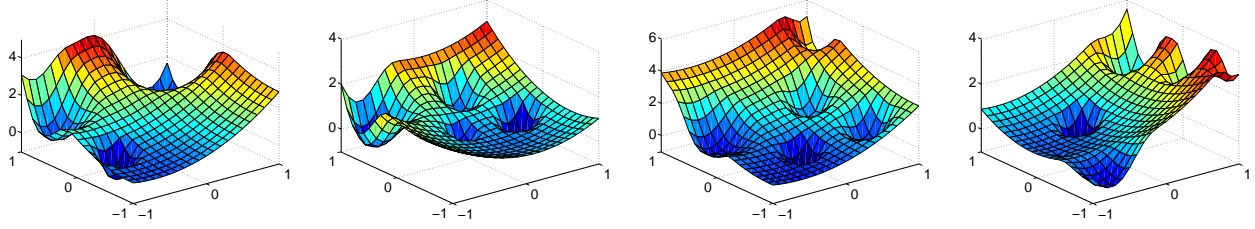


Figure 7: Example test functions with 4, 5, 6, and 7 pockets (respectively). Note that the viewpoint chosen makes visible only some of the pockets in these functions.

Both algorithms were initially seeded with a 5^2 design, comprising 25 points (about 5% of the design space of 441 points). Sampling was conducted for an additional 100 sample values (a total of 125 points, or about 25% of the design space). We reasoned that this is a good interval over which to monitor the performance of the sampling strategies, as even a regularly spaced grid covering 25% of the design space would mine the pockets correctly! Fig. 8 reveals the results for the 7-pocket function of Fig. 7. Both sampling strategies systematically reduce the (negative) log likelihood (as estimated from the GP model parameters) but variance-based sampling shows more oscillatory behavior w.r.t. the number of pockets mined. On close inspection, we found that this strategy goes through stages where adjacent pockets are periodically re-grouped around sample values (which are mostly at the boundaries), causing rapid fluctuations in the SAL miner’s output. We say that this strategy is more prone to ‘being surprised.’ The number of pockets stabilizes around 7 only toward the end of the data collection interval. In contrast, the entropy-based sampling first mines the seven pockets with 68 points, and proceeds to stabilize beyond this point. Similar results have been observed with other test functions.

Next, we analyzed the performance of both algorithms across all 43 test functions. We tested for what fraction of the datasets the mining was correct by, and stayed correct following, a given number of rounds of sampling. Our hypothesis was that the D2-type functions, being smoother, are more easily modeled using GPs and should lend themselves to more aggressive sampling strategies. In addition, the entropy-based sampling strategy should be more effective w.r.t. number of rounds than the variance-based sampling. Fig. 9 shows that this is indeed the case.

5.2 Mining Wireless System Configuration Spaces: Our second application involves characterization of configuration spaces of wireless system designs (see again Fig. 1). The goal is to understand the joint

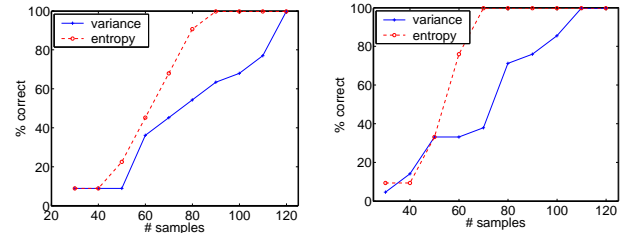


Figure 9: Overall pocket mining performance (fraction of cases correctly identified) with increasing number of samples, for (left) D-type and (right) D2-type functions.

influence of selected configuration parameters on system performance. This can be achieved by identifying spatial aggregates in the configuration space, aggregating low level simulation data (typically multiple samples per configuration point) into regions of constrained shape. In particular, the setup in Fig. 1 is from a study designed to evaluate the performance of STTD (space-time transmit diversity) wireless systems, where the base station uses two transmitter antennas separated by a small distance, in an attempt to improve received signal strength. In this application, the aim is to assess how the power imbalance between the two branches impacts the performance (measured by bit error rate, BER) of the simulated system, across a range of signal-to-noise ratios (SNRs). When the signal components are significant compared to the noise components, and when the SNR ratios of the two branches are comparable, then it is well known that the system would yield high quality of BER performance. What is not so clear is how the performance will degrade as the SNRs move apart. Posed in the spatial aggregation framework, this objective translates into identifying and characterizing (in terms of width, or power imbalance) the pocket in the central portion of the configuration space. Identifying and characterizing other pockets is not as important, since some of them will actually contain suboptimal configurations.

We adopt an experimental methodology similar to that in the previous case studies, and created an ‘oracle’ from the simulation data described in [26].

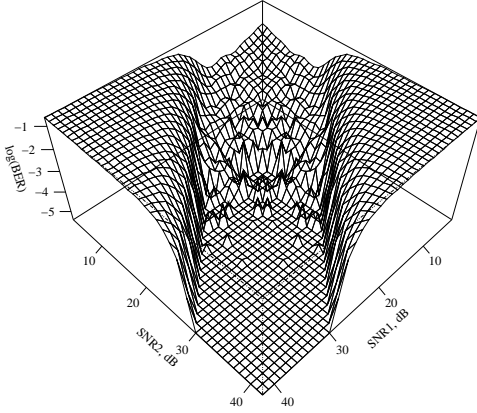


Figure 10: Estimates of BER performance in a space of wireless system configurations.

Fig. 10 demonstrates that the dataset is quite noisy, especially when the SNR values are low. The design of the oracle, surrogate model building, and sample selection all employ a 55×55 grid over the configuration space (SNR levels ranging from 3dB to 58dB for each antenna). Both variance-based sampling and entropy-based sampling were initialized using a 11^2 design (about 4% of the configuration space). Sampling was conducted for an additional 650 points, yielding a total of 771 points (25% of the design space, as with the earlier studies). For each round of active mining, we determined the majority class occupied by points having equal SNR and determined the maximum width of this class. This measure was periodically tracked across the rounds of data collection. Fig. 11 shows how the sampling strategies fare compared to the correct estimate of 12dB, as reported in [26] by applying a spatial data mining algorithm over the entire dataset. Entropy-based sampling once again selects data that systematically clarify the nature of the pockets, and cause a progressive widening of the trough in the middle. However, it doesn't mine the ideal width of 12dB (within the given samples). We reason that this is because the GP model has difficulty approximating the steep edge of the basin. Variance-based sampling fares worse and demonstrates a slower growth of width across samples. This application highlights the utility of our framework for mining both qualitative and quantitative properties of spatial aggregates.

6 Discussion

This paper has presented a novel integration of approaches from three areas, namely spatial structure discovery, probabilistic modeling using GPs, and active data mining. The spatial aggregation language provides a methodology for identifying multi-level structures in field data, Gaussian processes provide a prob-

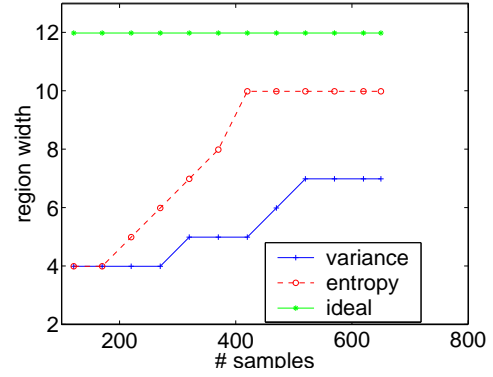


Figure 11: Performance of active mining strategies on wireless simulation data, characterizing width of the main pocket in Fig. 10 with increasing numbers of samples.

abilistic basis for reasoning about uncertainty in field data, and active data mining closes the loop to optimize new samples for uncertainty in field data as well as information content relevant to high-level structures. Entropy-based sampling is suitable whenever we can define information-theoretic functionals over spatial aggregates. In this paper, we have primarily focused on characterizing region boundaries, but this same strategy is applicable to any case where we expect locations with spatial proximity but informational impurity, e.g., identifying breaks and fissures in volumetric data, picking outliers from geographical maps, and detecting violations of coherence in spatio-temporal datasets.

There are several extensions to the work presented here. First, our assumption of sampling over a defined grid can be relaxed and the scope of active mining can be expanded to include subsampling. Second, the modeling of vector fields using GPs warrants further investigation, in particular to address the issue of how to model data fields given only (or also) derivative information or when the underlying function is not smooth or differentiable. Other investigators have done related work in this area [6]. Third, we assume here that the model (of flow classes) posited by SAL is correct, and use this information to drive the sampling. To overcome this assumption, we must create a probabilistic model of SAL's computations (including uncertainty and non-determinism in aggregation procedures) and integrate this model with the GP model for the data fields. Instantiating SAL to popular spatial mining algorithms investigated in the data mining community (e.g. [15, 20]) and applying them in an active mining context is a final direction we are pursuing. These and similar ideas will help establish the many ways in which mathematical models of data approximation can be integrated with data mining algorithms.

Acknowledgements

This work is supported in part by US NSF grants IIS-0237654, EIA-9984317, and IBN-0219332. The wireless simulation dataset is courtesy of Alex Verstak.

References

- [1] C. Bailey-Kellogg and N. Ramakrishnan. Ambiguity-Directed Sampling for Qualitative Analysis of Sparse Data from Spatially Distributed Physical Systems. In *Proc. IJCAI*, pages 43–50, 2001.
- [2] C. Bailey-Kellogg and F. Zhao. Influence-Based Model Decomposition for Reasoning about Spatially Distributed Physical Systems. *Artificial Intelligence*, Vol. 130(2):pages 125–166, 2001.
- [3] C. Bailey-Kellogg, F. Zhao, and K. Yip. Spatial Aggregation: Language and Applications. In *Proc. AAAI*, pages 517–522, 1996.
- [4] K. Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, pages 59–66, 2003.
- [5] D.A. Cohn, Z. Ghahramani, and M.I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, Vol. 4:pages 129–145, 1996.
- [6] D. Cornford, I.T. Nabney, and C.K.I. Williams. Adding Constrained Discontinuities to Gaussian Process Models of Wind Fields. In *Proceedings of NIPS*, pages 861–867, 1998.
- [7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
- [8] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian Prediction of Deterministic Functions, with Applications to the Design and Analysis of Computer Experiments. *J. Amer. Stat. Assoc.*, Vol. 86:pages 953–963, 1991.
- [9] R.G. Easterling. Comment on ‘Design and Analysis of Computer Experiments’. *Statistical Science*, 4(4):425–427, 1989.
- [10] J. Garcke and M. Griebel. Data Mining with Sparse Grids using Simplicial Basis Functions. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 87–96, 2001.
- [11] M. Gaviano, D.E. Kvasov, D. Lera, and Y.D. Sergeyev. Algorithm 829: Software for Generation of Classes of Test Functions with Known Local and Global Minima for Global Optimization. *ACM Transactions on Mathematical Software*, Vol. 29(4):pages 469–480, Dec 2003.
- [12] X. Huang and F. Zhao. Relation-Based Aggregation: Finding Objects in Large Spatial Datasets. In *Proceedings of the 3rd International Symposium on Intelligent Data Analysis*, 1999.
- [13] J.-N. Hwang, J.J. Choi, S. Oh, and R.J. Marks II. Query-based Learning Applied to Partially Trained Multilayer Perceptrons. *IEEE Transactions on Neural Networks*, Vol. 2(1):pages 131–136, 1991.
- [14] A.G. Journel and C.J. Huijbregts. *Mining Geostatistics*. Academic Press, New York, 1992.
- [15] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical Clustering using Dynamic Modeling. *IEEE Computer*, Vol. 32(8):pages 68–75, 1999.
- [16] J. Koehler and A. Owen. Computer Experiments. In S. Ghosh and C. Rao, editors, *Handbook of Statistics: Design and Analysis of Experiments*, pages 261–308. North Holland, 1996.
- [17] D.J. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, Vol. 4(4):pages 590–604, 1992.
- [18] I.T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, 2002.
- [19] R.M. Neal. Monte Carlo Implementations of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of Statistics, University of Toronto, Jan 1997.
- [20] R.T. Ng and J. Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14(5):pages 1003–1016, 2002.
- [21] I. Ordoñez and F. Zhao. STA: Spatio-Temporal Aggregation with Applications to Analysis of Diffusion-Reaction Phenomena. In *Proc. AAAI*, pages 517–523, 2000.
- [22] N. Ramakrishnan and C. Bailey-Kellogg. Sampling Strategies for Mining in Data-Scarce Domains. *IEEE/AIP CISE*, Vol. 4(4):pages 31–43, 2002.
- [23] N. Ramakrishnan and C. Bailey-Kellogg. Gaussian Process Models of Spatial Aggregation Algorithms. In *Proc. IJCAI*, pages 1045–1051, 2003.
- [24] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, Vol. 4(4):pages 409–435, 1989.
- [25] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, Vol. 2:pages 45–66, 2001.
- [26] A. Verstak et al. Using Hierarchical Data Mining to Characterize Performance of Wireless System Configurations. Technical Report cs.CE/0208040, CoRR, Aug 2002.
- [27] C.K.I. Williams. Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 599–621. MIT Press, Cambridge, MA, 1998.
- [28] K.M. Yip and F. Zhao. Spatial Aggregation: Theory and Applications. *JAIR*, Vol. 5:pages 1–26, 1996.
- [29] K.M. Yip, F. Zhao, and E. Sacks. Imagistic Reasoning. *ACM Computing Surveys*, Vol. 27(3):pages 363–365, 1995.
- [30] F. Zhao, C. Bailey-Kellogg, and M.P.J. Fromherz. Physics-Based Encapsulation in Embedded Software for Distributed Sensing and Control Applications. *Proceedings of the IEEE*, 91:40–63, 2003.

Correlation Clustering for Learning Mixtures of Canonical Correlation Models

Xiaoli Z. Fern*

Carla E. Brodley†

Mark A. Friedl‡

Abstract

This paper addresses the task of analyzing the correlation between two related domains X and Y . Our research is motivated by an Earth Science task that studies the relationship between vegetation and precipitation. A standard statistical technique for such problems is Canonical Correlation Analysis (CCA). A critical limitation of CCA is that it can only detect linear correlation between the two domains that is globally valid throughout both data sets. Our approach addresses this limitation by constructing a mixture of local linear CCA models through a process we name *correlation clustering*. In correlation clustering, both data sets are clustered simultaneously according to the data's correlation structure such that, within a cluster, domain X and domain Y are linearly correlated in the same way. Each cluster is then analyzed using the traditional CCA to construct local linear correlation models. We present results on both artificial data sets and Earth Science data sets to demonstrate that the proposed approach can detect useful correlation patterns, which traditional CCA fails to discover.

1 Introduction

In Earth science applications, researchers are often interested in studying the correlation structure between two domains in order to understand the nature of the relationship between them. The inputs to our correlation analysis task can be considered as two data sets X and Y whose instances are described by feature vectors \vec{x} and \vec{y} respectively. The dimension of \vec{x} and that of \vec{y} do not need to be the same, although there must be a one-to-one mapping between instances of X and instances of Y . Thus, it is often more convenient to consider these two data sets as one compound data set whose instances are described by two feature vectors \vec{x} and \vec{y} . Indeed, throughout the remainder of this paper, we will refer to the input of our task as one data set, and the goal is to study how the two sets of features are correlated to each other.

Canonical Correlation Analysis (CCA) [4, 6] is a

multivariate statistical technique commonly used to identify and quantify the correlation between two sets of random variables. Given a compound data set described by feature vectors \vec{x} and \vec{y} , CCA seeks to find a linear transformation of \vec{x} and a linear transformation of \vec{y} such that the resulting two new variables are maximally correlated.

In Earth science research, CCA has been often applied to examine whether there is a cause-and-effect relationship between two domains or to predict the behavior of one domain based on another. For example, in [13] CCA was used to analyze the relationship between the monthly mean sea-level pressure (SLP) and sea-surface temperature (SST) over the North Atlantic in the months of December, January and February. This analysis confirmed the hypothesis that atmospheric SLP anomalies cause SST anomalies.

Because CCA is based on *linear* transformations, the scope of its applications is necessarily limited. One way to tackle this limitation is to use nonlinear canonical correlation analysis (NLCCA) [5, 8]. NLCCA applies nonlinear functions to the original variables in order to extract correlated components from the two sets of variables. Although promising results have been achieved by NLCCA in some Earth science applications, it tends to be difficult to apply such techniques because of the complexity of the model and the lack of robustness due to overfitting [5].

In this paper we propose to use a mixture of local linear correlation models to capture the correlation structure between two sets of random variables (features). Mixtures of local linear models not only provide an alternative solution to capturing nonlinear correlations, but also have the potential to detect correlation patterns that are significant only in a part (a local region) of the data. The philosophy of using multiple local linear models to model global nonlinearity has been successfully applied to other statistical approaches with similar linearity limitations such as principal component analysis [12] and linear regression [7]. Our approach uses a two-step procedure. Given a compound data set, we propose to first solve a clustering problem that partitions the data set into clusters such that each cluster contains instances whose \vec{x} features and \vec{y} features are linearly correlated. We then independently apply CCA

*School of Elect. and Comp. Eng., Purdue University, West Lafayette, IN 47907, USA

†Dept. of Comp. Sci., Tufts University, Medford, MA 02155, USA

‡Dept. of Geography, Boston University, Boston, MA, USA

to each cluster to form a mixture of correlation models that are locally linear.

In designing this two-step process, we need address the following two critical questions.

1. Assume we are informed *a priori* that we can model the correlation structure using k local linear CCA models. *How should we cluster the data in the context of correlation analysis?*
2. In real-world applications, we are rarely equipped with knowledge of k . *How can we decide how many clusters there are in the data or whether a global linear structure will suffice?*

Note that the goal of clustering in the context of correlation analysis is different from traditional clustering. In traditional clustering, the goal is to group instances that are similar (as measured by certain distance or similarity metric) together. In contrast, here we need to group instances based on how their \vec{x} features and \vec{y} features correlate to each other, i.e., instances that share similar correlation structure between the two sets of features should be clustered together. To differentiate this clustering task from traditional clustering, we name it *correlation clustering*¹ and, in Section 3 we propose an iterative greedy k -means style algorithm for this task.

To address the second question, we apply the technique of cluster ensembles [2] to our correlation clustering algorithm, which provides a user with a visualization of the results that can be used to determine the proper number of clusters in the data. Note that our correlation clustering algorithm is a k -means style algorithm and as such may have many locally optimal solutions—different initializations may lead to significantly different clustering results. By using cluster ensembles, we can also address the local optima problem of our clustering algorithm and find a stable clustering solution.

To demonstrate the efficacy of our approach, we apply it to both artificial data sets and real world Earth science data sets. Our results on the artificial data sets show that (1) the proposed correlation clustering algorithm is capable of finding a good partition of the data when the correct k is used and (2) cluster ensembles provide an effective tool for finding k . When applied to the Earth science data sets, our technique detected significantly different correlation patterns in comparison to what was found via traditional CCA. These results led our domain expert to highly interesting hypotheses that merit further investigation.

The remainder of the paper is arranged as follows. In Section 2, we review the basics of CCA. Section 3 introduces the intuitions behind our correlation clustering algorithm and formally describes the algorithm, which is then applied to artificially constructed data sets to demonstrate its efficacy in finding correlation clusters from the data. Section 4 demonstrates how cluster ensemble techniques can be used to determine the number of clusters in the data and address the local optima problem of the k -means style correlation clustering algorithm. Section 5 explains our motivating application, presents results, and describes how our domain expert interprets the results. Finally, in Section 6 we conclude the paper and discuss future directions.

2 Basics of CCA

Given a data set whose instances are described by two feature vectors \vec{x} and \vec{y} , the goal of CCA is to find linear transformations of \vec{x} and linear transformations of \vec{y} such that the resulting new variables are maximally correlated.

In particular, CCA constructs a sequence of pairs of strongly correlated variables $(u_1, v_1), (u_2, v_2), \dots, (u_d, v_d)$ through linear transformations, where d is the minimum dimension of \vec{x} and \vec{y} . These new variables u_i 's and v_i 's, named *canonical variates* (sometimes referred to as canonical factors). They are similar to principal components in the sense that principal components are linear combinations of the original variables that capture the most variance in the data and in contrast canonical variates are linear combinations of the original variables that capture the most correlation between two sets of variables.

To construct these canonical covariates, CCA first seeks to transform \vec{x} and \vec{y} into a pair of new variables u_1 and v_1 by the linear transformations:

$$u_1 = (\vec{a}_1)^T \vec{x}, \text{ and } v_1 = (\vec{b}_1)^T \vec{y}$$

where the transformation vectors \vec{a}_1 and \vec{b}_1 are defined such that $\text{corr}(u_1, v_1)$ is maximized subject to the constraint that both u_1 and v_1 have unit variance.² Once $\vec{a}_1, \vec{b}_1; \dots; \vec{a}_i, \vec{b}_i$ are determined, we then find the next pair of transformations \vec{a}_{i+1} and \vec{b}_{i+1} such that the correlation between $(\vec{a}_{i+1})^T \vec{x}$ and $(\vec{b}_{i+1})^T \vec{y}$ is maximized with the constraint that the resulting u_{i+1} and v_{i+1} are uncorrelated with all previous canonical variates.³ Note that the correlation between u_i and v_i becomes weaker as i increases. Let r_i represent the correlation between the i th pair of canonical variates, we have $r_i \geq r_{i+1}$.

¹Note that the term *correlation clustering* has also been used by [1] as the name of a technique for traditional clustering.

²This constraint ensures unique solutions.

³This constraint ensures that the extracted canonical variates contain no redundant information.

It can be shown that to find the projection vectors for canonical variates, we only need to find the eigenvectors of the following matrices:

$$M_x = (\Sigma_{xx})^{-1} \Sigma_{xy} (\Sigma_{yy})^{-1} \Sigma_{yx}$$

and

$$M_y = (\Sigma_{yy})^{-1} \Sigma_{yx} (\Sigma_{xx})^{-1} \Sigma_{xy}$$

The eigenvectors of M_x , ordered according to decreasing eigenvalues, are the transformation vectors $\vec{a}_1, \vec{a}_2, \dots, \vec{a}_d$ and the eigenvectors of M_y are $\vec{b}_1, \vec{b}_2, \dots, \vec{b}_d$. In addition, the eigenvalues of these two matrices are identical and the square-root of the i -th eigenvalue $\sqrt{\lambda_i} = r_i$, i.e., the correlation between the i -th pair of canonical variates u_i and v_i . Note that in most applications, only the first few most significant pairs of canonical variates are of real interest. Assume that we are interested in the first d pairs of variates, we can represent all the useful information of the linear correlation structure as a model M , defined as

$$M = \{(u_j, v_j), r_j, (\vec{a}_j, \vec{b}_j) : j = 1 \dots d\}$$

where (u_j, v_j) represent the j th pair of canonical variates, r_j is the correlation between them and (\vec{a}_j, \vec{b}_j) represent the projection vectors for generating them. We refer to M as a CCA model.

Once a CCA model is constructed, the next step is for the domain experts to examine the variates as well as the transformation vectors in order to understand the relationship between the two domains. This can be done in different ways depending on the application. In our motivating Earth science task, the results of CCA can be visualized as colored maps and interpreted by Earth scientist. We explain this process in Section 5.

3 Correlation Clustering

In this section, we first explain the basic intuitions that led to our algorithm and formally present our k -means style correlation clustering algorithm. We then apply the proposed algorithm to artificially constructed data sets and analyze the results.

3.1 Algorithm Description Given a data set described by two sets of features \vec{x} and \vec{y} , and the prior knowledge that the correlation structure of the data can be modeled by k local linear models, the goal of correlation clustering is to partition the data into k clusters such that *for instances in the same cluster the features of \vec{x} and \vec{y} are linearly correlated in the same way*. The critical question is how should we cluster the data to reach this goal. Our answer is based on the following important intuitions.

Table 1: A correlation clustering algorithm

Input:	a data set of n instances, each described by two random vectors \vec{x} and \vec{y} k , the desired number of clusters
Output:	k clusters and k linear CCA models, one for each cluster
Algorithm:	<ol style="list-style-type: none"> 1. Randomly assign instances to the k clusters. 2. For $i = 1 \dots k$, apply CCA to cluster i to build $M^i = \{(u_j, v_j), r_j, (a_j, b_j) : j = 1 \dots d\}$, i.e., the top d pairs of canonical variates, the correlation r between each pair, and the corresponding d pairs of projection vectors. 3. Reassign each instance to a cluster based on its \vec{x} and \vec{y} features and the k CCA models. 4. If no assignment has changed from previous iteration, return the current clusters and CCA models. Otherwise, go to step 2.

Intuition 1: If a given set of instances contains multiple correlation structures, applying CCA to this instance set will not detect a strong linear correlation.

This is because when we put instances that have different correlation structure together, the original correlation patterns will be weakened because they are now only valid in part of the data. Conversely, if CCA detects strong correlation in a cluster, it is likely that the instances in the cluster share the same correlation structure. This suggests that we can use the strength of the correlation between the canonical variates extracted by CCA to measure the quality of a cluster. Note that it is computationally intractable to evaluate all possible clustering solutions in order to select the optimal one. This motivates us to examine a k -means style algorithm. Starting from a random clustering solution, in each iteration, we build a CCA model for each cluster and then reassign each instance to its most appropriate cluster according to its \vec{x} and \vec{y} features and the CCA models. In Table 1, we describe the basic steps of such a generic correlation clustering procedure.

The remaining question is how to assign instances to their clusters. Note that in traditional k -means clustering, each iteration reassigns instances to clusters according to the distance between instances and cluster centers. For correlation clustering, minimizing the

Table 2: Procedure of assigning instances to clusters

-
-
1. For each cluster i and its CCA model M^i , described as $\{(u_j^i, v_j^i), r_j^i, (\vec{a}_j^i, \vec{b}_j^i) : j = 1 \cdots d\}$, construct d linear regression models $v_j^i = \beta_j^i * u_j^i + \alpha_j^i, j = 1 \cdots d$, one for each pair of canonical variates.
 2. Given an instance (\vec{x}, \vec{y}) , for each cluster i , compute the instance's canonical variates under M^i as $u_j = (\vec{a}_j^i)^T \vec{x}$ and $v_j = (\vec{b}_j^i)^T \vec{y}, j = 1 \cdots d$, and calculate \hat{v}_j as $\hat{v}_j = \beta_j^i * u_j + \alpha_j^i, j = 1 \cdots d$, and the weighted err^i
$$err^i = \sum_{j=1}^d \frac{r_j^i}{r_1^i} * (v_j - \hat{v}_j)^2,$$
 where $\frac{r_j^i}{r_1^i}$ is the weight for the j th prediction error.
 3. Assign instance (\vec{x}, \vec{y}) to the cluster minimizing err^i .
-
-

distance between instances and their cluster centers is no longer our goal. Instead, our instance reassignment is performed based on the intuition described below.

Intuition 2: If CCA detects strong a correlation pattern in a cluster, i.e., the canonical variates u and v are highly correlated, we expect to be able to predict the value of v from u (or vice versa) using a linear regression model.

This is demonstrated in Figure 1, where we plot a pair of canonical variates with correlation 0.9. Shown as a solid line is the linear regression model constructed to predict one variate from the other. Intuition 2 suggests that, for each cluster, we can compute its most significant pair of canonical variates (u_1, v_1) and construct a linear regression model to predict v_1 from u_1 . To assign an instance to its proper cluster, we can simply select the cluster whose regression model best predicts the instance's variate v_1 from it's variate u_1 . In some cases, we are interested in the first few pairs of canonical variates rather than only the first pair. It is thus intuitive to construct one linear regression model for each pair, and assign instances to clusters based on the combined prediction error. Note that because the correlation r_i between variate v_i, u_i decreases as i increase, we set the weight for the i^{th} error to be $\frac{r_i}{r_1}$. In this manner, the weight for the prediction error between u_1 and v_1 is always one, whereas the weights for the ensuing ones will be smaller depending on the strength

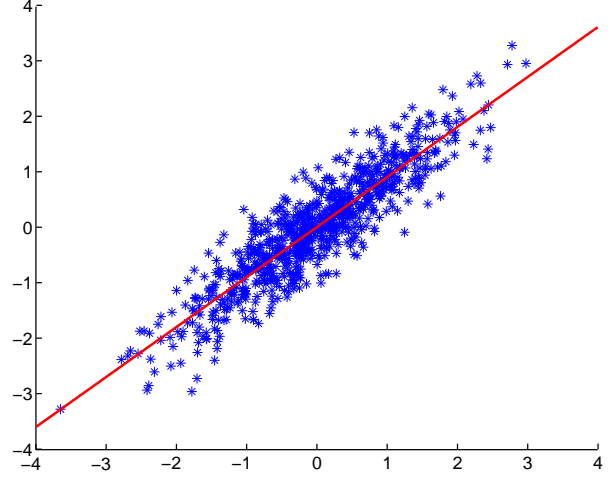


Figure 1: Scatter plot of a pair of canonical variates ($r = 0.9$) and the linear regression model constructed to predict one variate from another.

of the correlations. This ensures that more focus is put on the canonical variates that are more strongly correlated. In Table 2, we describe the exact procedure for reassigning instances to clusters.

Tables 1 and 2 complete the description of our correlation clustering algorithm. To apply this algorithm, the user needs to specify d , the number of pairs of canonical variates that are used in computing the prediction errors and reassigning the instances. Based on our empirical observations with both artificial and real-world datasets, we recommend that d be set to be the same as or slightly larger than the total number of variates that bear interest in the application. In our application, our domain expert is interested in only the top two or three pairs of canonical variates, consequently we used $d = 4$ as the default choice for our experiments.

The proposed correlation clustering algorithm is a greedy iterative algorithm. We want to point out that it is not guaranteed to converge. Specifically, after re-assigning instances to clusters at each iteration, there is no guarantee that the resulting new clusters will have more strongly correlated variates. In our experiments, we did observe fluctuations in the objective function, i.e., the weighted prediction error. But, fluctuations typically occur only after an initial period in which the error computed by the objective function quickly decreases. Moreover, after this rapid initial convergence, the ensuing fluctuations are relatively small. Thus we recommend that one specify a maximum number of iterations, and in our experiments we set this to be 200 iterations.

Table 3: An artificial data set and results

	Data Sets		Global CCA	Mixture of CCA	
	D_1	D_2		clust. 1	clust. 2
r_1	0.85	0.9	0.521	0.856(.001)	0.904(.001)
r_2	0.6	0.7	0.462	0.619(.001)	0.685(.004)
r_3	0.3	0.4	0.302	0.346(.003)	0.436(.003)

3.2 Experiments on Artificial Data Sets To examine the efficacy of the proposed correlation clustering algorithm, we apply it to artificially generated data sets that have pre-specified nonlinear correlation structures. We generate such data by first separately generating multiple component data sets, each with a different linear correlation structure, and then mixing these component data sets together to form a composite data set. Obviously the resulting data set’s correlation structure is no longer globally linear. However, a properly constructed mixture of local linear models should be able to separate the data set into the original component data sets and recover the correlation patterns in each part. Therefore, we are interested in (1) testing whether our correlation clustering algorithm can find the correct partition of the data, and (2) testing whether it can recover the original correlation patterns represented as the canonical variates, and (3) comparing its results to the results of global CCA on the composite data set.

In Table 3, we present the results of our correlation clustering algorithm and traditional CCA on a composite data set formed by two component data sets, each of which contains 1000 instances. We generate each component data set as follows.⁴ Given the desired correlation values r_1 , r_2 , and r_3 , we first create a multivariate Gaussian distribution with six random variables $u_1, u_2, u_3, v_1, v_2, v_3$, where u_i and v_i are intended to be the i th pair of canonical variates. We set the covariance matrix to be:

$$\begin{pmatrix} 1 & 0 & 0 & r_1 & 0 & 0 \\ 0 & 1 & 0 & 0 & r_2 & 0 \\ 0 & 0 & 1 & 0 & 0 & r_3 \\ r_1 & 0 & 0 & 1 & 0 & 0 \\ 0 & r_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & r_3 & 0 & 0 & 1 \end{pmatrix}$$

This ensures that $\text{corr}(u_j, v_j) = r_j$, for $j = 1, 2, 3$ and $\text{corr}(u_i, u_j) = \text{corr}(v_i, v_j) = \text{corr}(u_i, v_j) = 0$ for $i \neq j$. We then randomly sample 1000 points from this joint Gaussian distribution and form the final vector of \vec{x} using linear combinations of u_j ’s and the vector of \vec{y} using linear combinations of v_j ’s.

⁴The matlab code for generating a component data set is available at <http://www.ecn.purdue.edu/~xz>

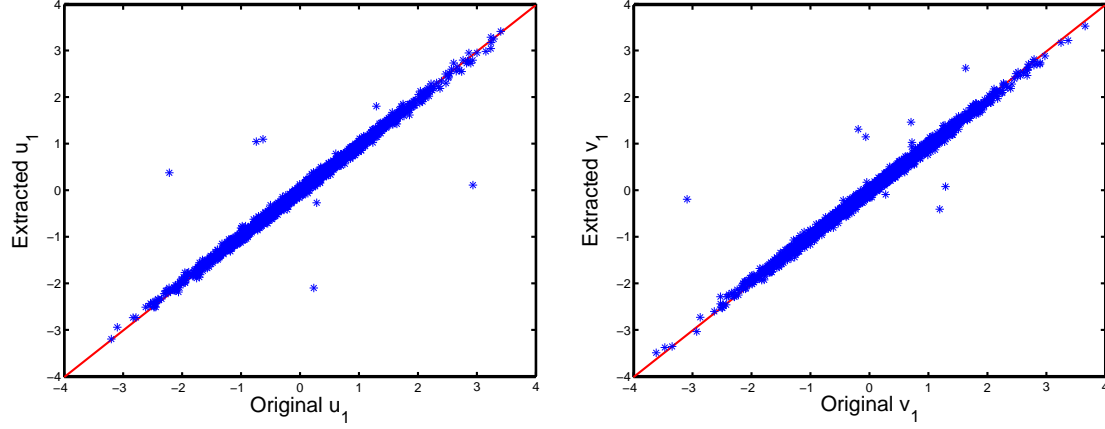
Columns 2 and 3 of Table 3 specify the correlation between the first three pairs of canonical variates of each of the constructed datasets, D_1 and D_2 . These are the values that were used to generate the data. We applied the traditional CCA to the composite data set (D_1 and D_2 combined together) and we report the top three detected canonical correlations in Column 4. We see from the results that, as expected, global CCA is unable to extract the true correlation structure from the data.

The last two columns of Table 3 show the results of applying the proposed correlation clustering algorithm to the composite data set with $k = 2$ and $d = 4$. The results, shown in Columns 5 and 6 are the average over ten runs with different random initializations (the standard deviations are shown in parentheses). We observe that the detected canonical correlations are similar to the true values. In Figure 2, We plot the canonical variates extracted by our algorithm (y axis) versus the true canonical variates (x axis) and the plots of the first two pairs of variates are shown. We observe that the first pair of variates extracted by our algorithm are very similar to the original variates. This can be seen by noticing that for both u_1 and v_1 most points lie on or are close to the line of unit slope (shown as a red line). For the second pair, we see more deviation from the red line. This is possibly because our algorithm put less focus on the second pair of variates during clustering. Finally, we observe that the clusters formed by our algorithm correspond nicely to the original component data sets. On average, only 2.5% of the 2000 instances were assigned to the wrong cluster.

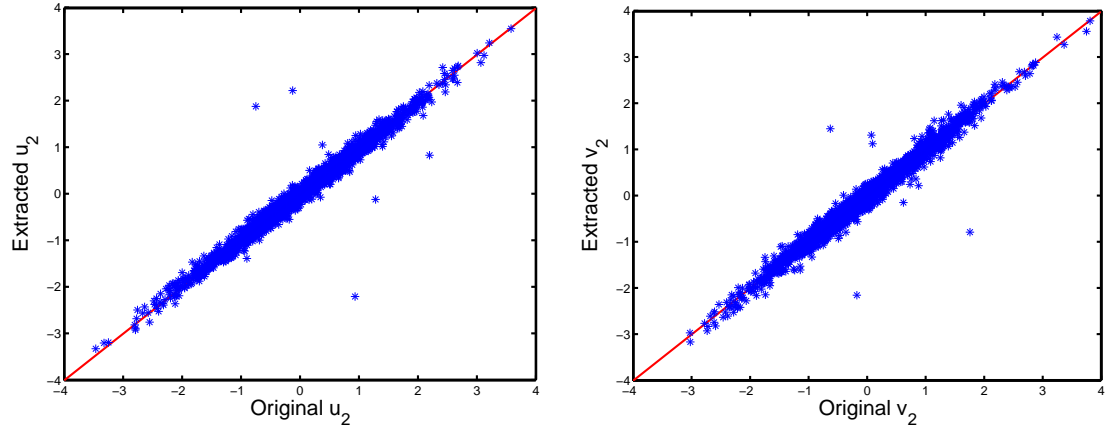
These results show that our correlation clustering algorithm can discover local linear correlation patterns given prior knowledge of k , the true number of clusters in the data. Our algorithm performs consistently well on artificially constructed data sets. This is in part due to the fact that these data sets are highly simplified examples of nonlinearly correlated data. In real applications, the nonlinear correlation structure is often more complex. Indeed, when applied to our Earth science data sets, we observe greater instability of our algorithm—different initializations lead to different clustering solutions. We conjecture that this is because our clustering algorithm is a k -means style greedy algorithm and has large number of locally optimal solutions.

4 Cluster Ensembles for Correlation Clustering

In this section we address a problem in the practical application of the proposed correlation clustering algorithm—identification of the number of clusters in the data. A complicating factor is that because we are



(a). The first pair of canonical variates



(b). The second pair of canonical variates

Figure 2: Comparing the first two pairs of canonical variates extracted by our mixture of CCA algorithm and the original canonical variates.

dealing with a k -means style greedy algorithm there may be many locally optimal solutions. In particular, different initializations may lead to different clusters. In this section we show how to apply cluster ensemble techniques to address these issues.

The concept of cluster ensembles has recently seen increasing popularity in the clustering community [11, 2, 10, 3], in part because it can be applied to any type of clustering as a generic tool for boosting clustering performance. The basic idea is to generate an ensemble of different clustering solutions, each capturing some structure of the data. The anticipated result is that by combining the ensemble of clustering solutions, a better final clustering solutions can be obtained. Cluster ensembles have been successfully applied to determine the number of clusters [10] and to improve clustering performance for traditional clustering tasks [11, 2, 3].

Although our clustering tasks are significantly different from traditional clustering in terms of the goal, we believe similar benefits can be achieved by using cluster ensembles.

To generate a cluster ensemble, we run our correlation clustering algorithm on a given data set with $k=2$ for r times, each run starting from a different initial assignment, where r is the size of the ensemble. We then combine these different clustering solutions into a $n \times n$ matrix S , which describes for each pair of instances the frequency with which they are clustered together (n is the total number of instances in the data set.) As defined, each element of S is a number between 0 and 1. We refer to it as a similarity matrix because $S(i, j)$ can be considered as the similarity (correlation similarity instead of the conventional similarity) between instances i and j .

After the similarity matrix is constructed, we can then visualize the matrix using a technique introduced by [10] to help determine how many clusters there are in the data. This visualization technique has two steps. First, it orders the instances such that instances that are similar to each other are arranged to be next to each other. It then maps the 0-1 range of the similarity values to a gray-scale such that 0 corresponds to white and 1 corresponds to black. The similarity matrix is then displayed as an image, in which darker areas indicate strong similarity and lighter areas indicate little to no similarity. For example, if all clustering solutions in the ensemble agree with one another perfectly, the similarity matrix S will have similarity value 1 for these pairs of instances that are from the same cluster and similarity value 0 for those from different clusters. Because the instances are ordered such that similar instances are arranged next to each other, the visualization will produce black squares along the diagonal of the image. For a detailed description of the visualization technique, please refer to [10].

To demonstrate the effect of cluster ensembles on our correlation clustering, we generate three artificial data sets using the same procedure as described in Section 3.2. These three data sets contain one, two, and three correlation clusters respectively. We apply our correlation clustering algorithm 20 times with different initializations and construct a similarity matrix for each data set. In Figure 3 we show the images of the resulting similarity matrices for these three data sets and make following observations.

- For the one-cluster data set, shown in Figure 3 (a), the produced similarity matrix does not show any clear clustering pattern. This is because our correlation clustering algorithm splits the data randomly in each run—by combining the random runs through the similarity matrix, we can easily reach the conclusion that the given data set contains only one correlation cluster.
- For the two-cluster data set, shown in Figure 3 (b), First, we see two dark squares along the diagonal, indicating there are two correlation clusters in the data. This shows that, as we expect, the similarity matrix constructed via cluster ensembles reveal information about the true number of clusters in the data.

In addition to the two dark diagonal squares, we also see small gray areas in the image, indicating that some of the clustering solutions in the ensemble disagree with each other on some instances. This is because different initializations sometimes lead to different local optimal solutions. Further,

we argue that these different solutions sometimes make different mistakes—combining them can potentially correct some of the mistakes and produce a better solution.⁵ Indeed, our experiments show that, for this particular two-cluster data set, applying the average-link agglomerative clustering to the resulting similarity matrix reduces the clustering error rate from 2.0% (the average error rate of the 20 clustering runs) to 1.1%. In this case, cluster ensembles corrected for the local optima problem of our correlation clustering algorithm. Cluster ensembles have been shown to boost the clustering performance for traditional clustering tasks, here we confirm that correlation clustering can also benefit from cluster ensembles.

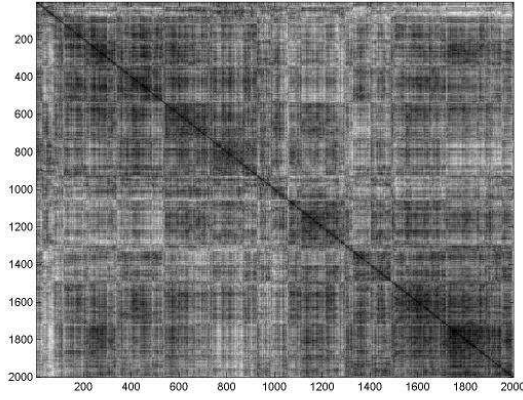
- For the last data set, shown in Figure 3 (c), we see three dark squares along the diagonal, indicating that there are three correlation clusters in the data.

Comparing to the two-cluster case, we see significantly larger areas of gray. In this case, our correlation clustering algorithm was asked to partition the data into two parts although the data actually contains three clusters. Therefore, it is not surprising that many of the clustering solutions don't agree with each other because they may split or merge clusters in many different ways when different initializations are used, resulting in much larger chance for disagreement. However, this does not stop us from finding the correct number of clusters from the similarity matrix. Indeed, by combining multiple solutions, these random splits and merges tend to cancel out each other and the true structure of the data emerges.

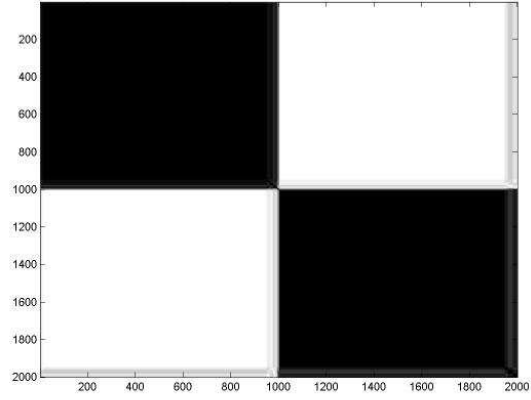
With the help of the similarity matrix, now we know there are three clusters in the last data set. We then constructed another cluster ensemble for this data set, but this time we set $k=3$ for each clustering run. The resulting similarity matrix S' is shown in Figure 3 (d). In this case, the average error rate achieved by the individual clustering solutions in the ensemble is 7.5% and the average-link agglomerative clustering algorithm applied on S' reduces the error rate to 6.8%.

To conclude, cluster ensembles help to achieve two goals. First, they provide information about the true structure of the data. Second, they help improve clustering performance of our correlation clustering algorithm.

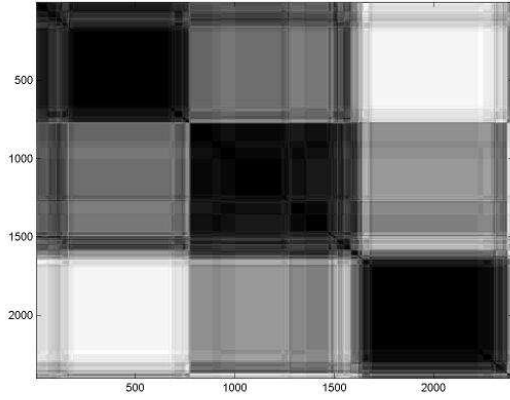
⁵It should be noted that if the different solutions make the same mistakes, these mistakes will not be corrected by using cluster ensembles.



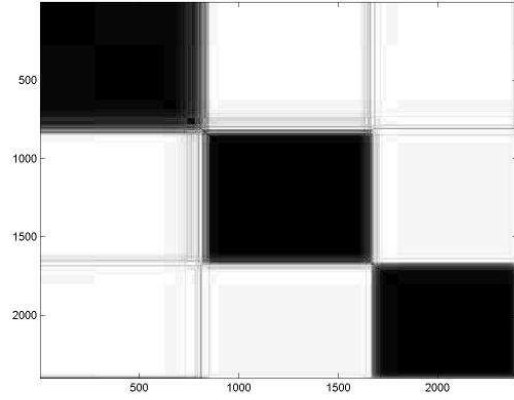
(a)



(b)



(c)



(d)

Figure 3: Visualization of similarity matrices: (a). S for the one-cluster data set; (b). S for the two-cluster data set ; (c). S for the three-cluster data set, and (d). S' for the three-cluster data set

5 Experiments on Earth Science Data Sets

We have demonstrated on artificial data sets that our correlation algorithm is capable of finding locally linear correlation patterns in the data. In this section, we apply our techniques to Earth science data sets. The task is to investigate the relationship between the variability in precipitation and the dynamics of vegetation. Below, we briefly introduce the data sets and then compare our technique to traditional CCA.

In this study, the standardized precipitation index (SPI) is used to describe the precipitation domain and the normalized difference vegetation index (NDVI) is used to describe the vegetation domain [9]. The data for both domains are collected and aligned at monthly time intervals from July 1981 to October 2000 (232 months). Our analysis is performed at continental level for the

continents of North America, South America, Australia and Africa. For each of these continents, we form a data set whose instances correspond to time points. For a particular continent, the feature vector \vec{x} records the SPI value at each grid location of that continent, thus the dimension of \vec{x} equals the number of grid locations of that continent. Similarly, \vec{y} records the NDVI values. Note that the dimensions of \vec{x} and \vec{y} are not equal because different grid resolutions are used to collect the data. The effect of applying our technique to the data is to cluster the data points in time. This is motivated by the hypothesis that during different time periods the relationship between vegetation and precipitation may vary.

For our application, a standard way to visualize CCA results is to use colored map. In particular, to

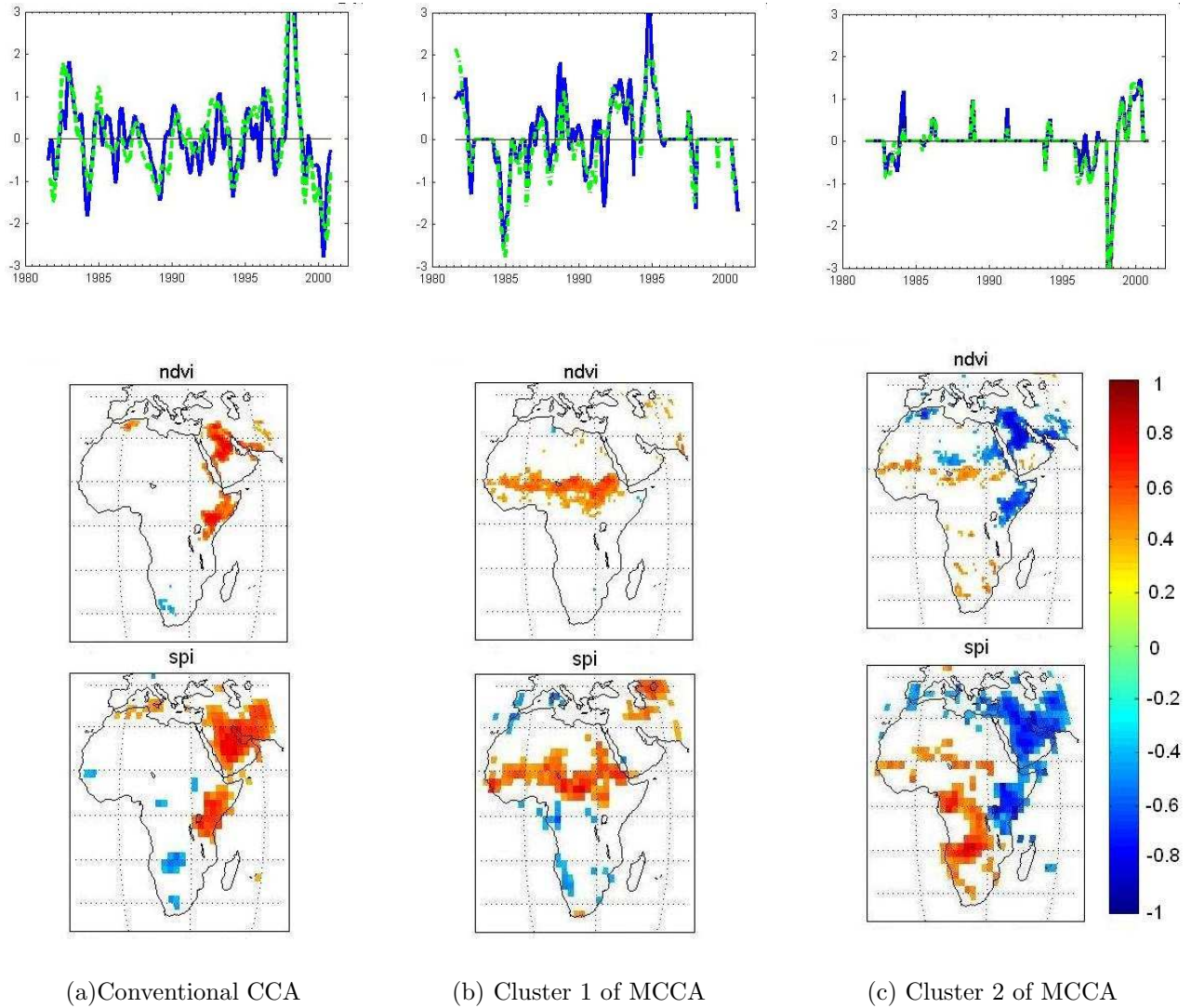


Figure 4: The results of conventional CCA and Mixture of CCA (MCCA) for Africa. Top panel shows the NDVI and SPI canonical variates (time series). Middle and bottom panel show the NDVI and SPI maps.

analyze a pair of canonical variates, which are in this case a pair of correlated time series, one for SPI and one for NDVI. We produce one map for SPI and one map for NDVI. For example, to produce a map for SPI, we take the correlation between the time series of the SPI canonical variate and the SPI time series of each grid point, generating a value between -1 (negative correlation) and 1 (positive correlation) for each grid point. We then display these values on the map via color coding. Areas of red (blue) color are positively (negatively) correlated with the SPI canonical variate. Considered together, the NDVI map and SPI map identify regions where SPI correlates with NDVI. Since our technique produces local CCA models, we can visualize each cluster using the same technique.

Note that an exact geophysical interpretation of the produced maps is beyond the scope of this paper. To do so, familiarity with the geoscience terminologies and concepts is required from our audience. Instead, we will present the maps produced by traditional CCA and the maps produced by our technique, as well as plots of the time series of the SPI and NDVI canonical variates. Finally, a high level interpretation of the results is provided by our domain expert. For brevity, the rest of our discussion will focus on the continent of Africa, which is a representative example where our method finds patterns of interest that were not discovered by traditional CCA.

We apply our technique to the data set of Africa by setting $k=2$ and constructing a cluster ensemble of

size 200.⁶ The final two clusters were obtained using the average-link agglomerative algorithm applied to the similarity matrix.

Figure 4 (a) shows the maps and the NDVI and SPI time series generated by traditional CCA. Figures 4 (b) and (c) show the maps and the time series for each of the two clusters. Note that each of the maps is associated with the first pair of canonical variates for that dataset/cluster. Inspection of the time series and the spatial patterns that are associated with the canonical variates for each cluster demonstrates that the mixture of CCA approach provides information that is clearly different from results produced by conventional CCA. For Africa, the interannual dynamics in precipitation are strongly influenced by a complex set of dynamics that depend on El-Nino and La Nina, and on the resulting sea surface temperature regimes in Indian Ocean and southern Atlantic ocean off the coast of west Africa. Although exact interpretation of these results requires more study, the maps of Figures 4 (b) and (c) show that the proposed approach was able to isolate important quasi-independent modes of precipitation-vegetation co-variability that linear methods are unable to identify. As shown in [9], conventional CCA is effective in isolating precipitation and vegetation anomalies in eastern Africa associated with El-Nino, but less successful in isolating similar patterns in the Sahelian region of western Africa. In contrast, Figures 4 (b) and (c) show that the mixture of CCA technique isolates the pattern in eastern Africa, and additionally identifies a mode of co-variability in the Sahel that is probably related to ocean-atmosphere dynamics in the southern Atlantic ocean.

6 Conclusions and Future Work

This paper presented a method for constructing mixtures of local CCA models in attempt to address the limitations of the conventional CCA approach. We developed a correlation clustering algorithm, which partitions a given data set according to the correlation between two sets of features. We further demonstrated that cluster ensembles can be used to identify the number of clusters in the data and ameliorate the local optima problem of the proposed clustering algorithm. We applied our technique to Earth science data sets. In comparison to traditional CCA, our technique led to interesting and encouraging new discoveries in the data.

As an ongoing effort, we will closely work with our domain expert to verify our findings in the data from

a geoscience viewpoint. For future work, we would also like to apply our technique to more artificial and real-world data sets that have complex nonlinear correlation structure. Finally, we are developing a probabilistic approach to learning mixture of CCA models.

References

- [1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.
- [2] X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
- [3] X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the Twenty First International Conference on Machine Learning*, pages 281–288, 2004.
- [4] H. Hotelling. Relations between two sets of variants. *Biometrika*, 28:321–377, 1936.
- [5] W. Hsieh. Nonlinear canonical correlation analysis by neural networks. *Neural Networks*, 13:1095–1105, 2000.
- [6] R.A. Johnson and D. W. Wichern. *Applied multivariate statistical analysis*. Prentice Hall, 1992.
- [7] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.
- [8] P. L. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(5):365–377, 2000.
- [9] A. Lotsch and M. Friedl. Coupled vegetation-precipitation variability observed from satellite and climate record. *Geophysical research letter*, In submission.
- [10] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52:91–118, 2003.
- [11] A. Strehl and J. Ghosh. Cluster ensembles - A knowledge reuse framework for combining multiple partitions. *Machine Learning Research*, 3:583–417, 2002.
- [12] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11, 1999.
- [13] E. Zorita, V. Kharin, and H. von Storch. The atmospheric circulation and sea surface temperature in the north atlantic area in winter: Their interaction and relevance for iberian precipitation. *Journal of Climate*, 5:1097–1108, 1992.

⁶We use large ensemble sizes for the Earth science data sets because they contain a small number of instances, making it computationally feasible and also larger ensemble sizes ensure that the clusters we found in the data are not obtained by chance.

On Periodicity Detection and Structural Periodic Similarity

Michail Vlachos

Philip Yu

Vittorio Castelli

IBM. T.J. Watson Research Center
19 Skyline Dr, Hawthorne, NY

Abstract

This work motivates the need for more flexible structural similarity measures between time-series sequences, which are based on the extraction of important periodic features. Specifically, we present non-parametric methods for accurate periodicity detection and we introduce new periodic distance measures for time-series sequences. The goal of these tools and techniques are to assist in detecting, monitoring and visualizing structural periodic changes. It is our belief that these methods can be directly applicable in the manufacturing industry for preventive maintenance and in the medical sciences for accurate classification and anomaly detection.

1 Introduction

In spite of the fact that in the past decade we have experienced a profusion of time-series distance measures and representations [9], the majority of them attempt to characterize the similarity between sequences based solely on shape. However, it is becoming increasingly apparent that *structural* similarities can provide more intuitive sequence characterizations that adhere more tightly to human perception of similarity.

While shape-based similarity methods seek to identify homomorphic sequences using the original raw data, structure-based methodologies are designed to find latent similarities, possibly by transforming the sequences into a new domain, where the resemblance can be more apparent. For example, in [6] the authors use change-point-detection signatures for identifying sequences that exhibit similar structural changes. In [7] Kalpakis, et al., use the *cepstrum* for clustering sequences that share a similar underlying ARIMA generative process. Keogh, et al. [10], employ a compression-based dissimilarity measure that is effectively used for clustering and anomaly detection. Finally, Vlachos, et al. [15] consider structural similarities that are based on burst features of time-series sequences.

In this work we consider methods for efficiently capturing and characterizing the periodicity and periodic similarity of time-series. Such techniques can be applicable in a variety of disciplines, such as manufacturing, natural sciences and medicine, which acquire and

record large amounts of periodic data. For the analysis of such data, first there is a need for accurate periodicity estimation, which can be utilized either for anomaly detection or for prediction purposes. Then, a structural distance measure should be deployed that can effectively incorporate the periodicity for quantifying the degree of similarity between sequences. A periodic measure can allow for more meaningful and accurate clustering and classification, and can also be used for interactive exploration (and visualization) of massive periodic datasets. Let us consider areas where periodic measures can be applicable:

■ In *natural sciences*, many processes manifest strong or weak periodic behavior, such as tidal patterns (oceanography), sunspots (astronomy), temperature changes (meteorology), etc. Periodic analysis and periodicity estimation is an important aspect in these disciplines, because they can suggest potential anomalies or help understand the causal relationship between different processes. For example, it is well established that solar variability greatly affects the climate change. In fact the solar cycle (sunspot numbers) presents striking resemblance to the northern hemisphere land temperatures [4].

■ In *medicine*, where many biometric measures (e.g., heartbeats) exhibit strong periodicities, there is a great interest in detecting periodic anomalies. Disturbances of similar periodic patterns can be noted in many degenerative diseases; for example, it has been noted that Tourette's syndrome patients exhibit elevated eyeblink rate [14], while people affected by Parkinson's disease show symptoms of gait disturbances [1]. The tools that we provide here, can significantly enhance the early detection of such changes.

■ Finally, periodic analysis is an indispensable tool in *automotive, aviation and manufacturing* industries for machine monitoring and diagnostics [12]. Predictive maintenance can be possible by examination of the vibration spectrum caused by its rotating parts. Therefore, a change in the periodic structure of machine vibrations can be a good indicator of machine wear and/or of an incipient failure.

This work targets similar applications and provides tools that can significantly ease the “mining” of useful information. Specifically, this paper makes the following contributions:

1. We present a novel automatic method for accurate periodicity detection in time-series data. Our algorithm is the first one that exploits the information in both periodogram and autocorrelation to provide accurate periodic estimates without upsampling.
2. We introduce new periodic distance measures that exploit the power of the dominant periods, as provided by the Fourier Transform. By ignoring the phase information we can provide more compact representations, that also capture similarities under time-shift transformations.
3. Finally, we present comprehensive experiments demonstrating the applicability and efficiency of the proposed methods, on a variety of real world datasets (online query logs, manufacturing diagnostics, medical data, etc.).

2 Background

We provide a brief introduction to harmonic analysis using the discrete Fourier Transform, because we will use these tools as the building blocks of our algorithms.

2.1 Discrete Fourier Transform. The normalized Discrete Fourier Transform of a sequence $x(n)$, $n = 0, 1 \dots N - 1$ is a sequence of complex numbers $X(f)$:

$$X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi k n}{N}}, \quad k = 0, 1 \dots N - 1$$

where the subscript k/N denotes the frequency that each coefficient captures. Throughout the text we will also utilize the notation $\mathcal{F}(x)$ to describe the Fourier Transform. Since we are dealing with real signals, the Fourier coefficients are symmetric around the middle one (or to be more exact, they will be the complex conjugate of their symmetric). The Fourier transform represents the original signal as a linear combination of the complex sinusoids $s_f(n) = \frac{e^{j2\pi f n/N}}{\sqrt{N}}$. Therefore, the Fourier coefficients record the amplitude and phase of these sinusoids, after signal x is projected on them.

We can return from the frequency domain back to the time domain, using the inverse Fourier transform $\mathcal{F}^{-1}(x) \equiv x(n)$:

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(f_{k/N}) e^{j \frac{2\pi k n}{N}}, \quad k = 0, 1 \dots N - 1$$

Note that if during this reverse transformation we discard some of the coefficients (e.g., the last k), then the outcome will be an approximation of the original sequence (Figure 1). By carefully selecting which

coefficients to record, we can perform a variety of tasks such as compression, denoising, etc.

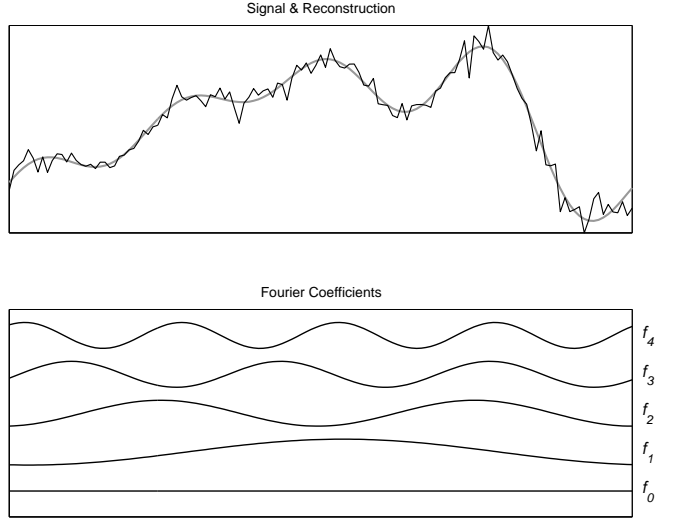


Figure 1: Reconstruction of a signal from its first 5 Fourier coefficients

2.2 Power Spectral Density Estimation. In order to discover potential periodicities of a time-series, one needs to examine its *power spectral density* (PSD or power spectrum). The PSD essentially tells us how much is the expected signal power at each frequency of the signal. Since period is the inverse of frequency, by identifying the frequencies that carry most of the energy, we can also discover the most dominant periods. There are two well known estimators of the PSD; the periodogram and the circular autocorrelation. Both of these methods can be computed using the DFT of a sequence (and can therefore exploit the Fast Fourier Transform for execution in $O(N \log N)$ time).

2.2.1 Periodogram Suppose that X is the DFT of a sequence x . The *periodogram* \mathcal{P} is provided by the squared length of each Fourier coefficient:

$$\mathcal{P}(f_{k/N}) = \|X(f_{k/N})\|^2 \quad k = 0, 1 \dots \lceil \frac{N-1}{2} \rceil$$

Notice that we can only detect frequencies that are at most half of the maximum signal frequency, due to the Nyquist fundamental theorem. In order to find the k dominant periods, we need to pick the k largest values of the periodogram.¹

¹Due to the assumption of the Fourier Transform that the data is periodic, proper windowing of the data might be necessary for achieving a more accurate harmonic analysis. In this work we will sidestep this issue, since it goes beyond the scope of this paper. However, the interested reader is directed to [5] for an excellent review of data windowing techniques.

Each element of the periodogram provides the power at frequency k/N or, equivalently, at period N/k . Being more precise, each DFT ‘bin’ corresponds to a *range* of periods (or frequencies). That is, coefficient $X(f_{k/N})$ corresponds to periods $[\frac{N}{k} \dots \frac{N}{k-1}]$. It is easy to see that the resolution of the periodogram becomes very coarse for longer periods. For example, for a sequence of length $N = 256$, the DFT bin margins will be $N/1, N/2, N/3, \dots = 256, 128, 64$ etc.

Essentially, the accuracy of the discovered periods, deteriorates for large periods, due to the increasing width of the DFT bins (N/k). Another related issue is *spectral leakage*, which causes frequencies that are not integer multiples of the DFT bin width, to disperse over the entire spectrum. This can lead to ‘false alarms’ in the periodogram. However, the periodogram can still provide an accurate indicator of important short (to medium) length periods. Additionally, through the periodogram it is easy to automate the extraction of important periods (peaks) by examining the statistical properties of the Fourier coefficients (such as in [15]).

2.2.2 Circular Autocorrelation. The second way to estimate the dominant periods of a time-series x , is to calculate the circular AutoCorrelation Function (or ACF), which examines how similar a sequence is to its previous values for different τ lags:

$$ACF(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} x(\tau) \cdot x(n + \tau)$$

Therefore, the autocorrelation is formally a convolution, and we can avoid the quadratic calculation in the time domain by computing it efficiently as a dot product in the frequency domain using the normalized Fourier transform:

$$ACF = \mathcal{F}^{-1} \langle X, X^* \rangle$$

The star (*) symbol denotes complex conjugation.

The ACF provides a more fine-grained periodicity detector than the periodogram, hence it can pinpoint with greater accuracy even larger periods. However, it is not sufficient by itself for automatic periodicity discovery for the following reasons:

1. Automated discovery of important peaks is more difficult than in the periodogram. Approaches that utilize forms of autocorrelation require the user to manually set the significance threshold (such as in [2, 3]).
2. Even if the user picks the level of significance, multiples of the same basic period also appear as peaks. Therefore, the method introduces many false alarms that need to be eliminated in a post-processing phase.
3. Low amplitude events of high frequency may appear less important (i.e., have lower peaks) than high

amplitude patterns, which nonetheless appear more scarcely (see example in fig. 2).

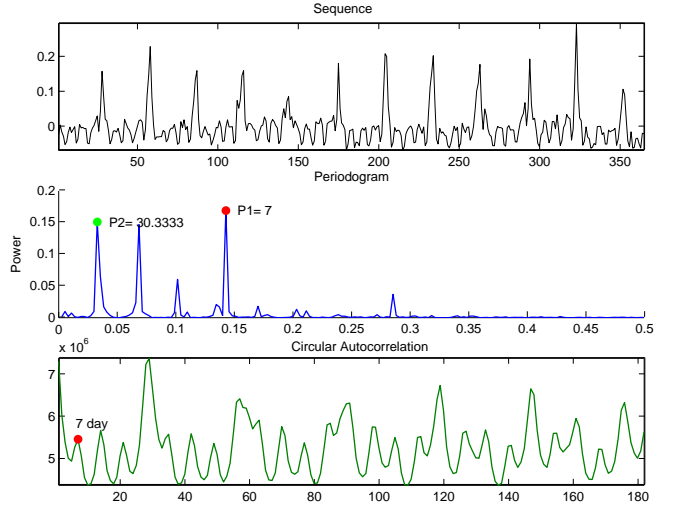


Figure 2: The 7 day period is latent in the autocorrelation graph, because it has lower amplitude (even though it happens with higher frequency). However, the 7 day peak is very obvious in the Periodogram.

The advantages and shortcomings of the periodogram and the ACF are summarized in Table 1.

From the above discussion one can realize that although the periodogram and the autocorrelation cannot provide sufficient spectral information separately, there is a lot of potential when both methods are combined. We delineate our approach in the following section.

3 Our Approach

We utilize a two-tier approach, by considering the information in both the autocorrelation and the periodogram. We call this method AUTOPERIOD. Since the discovery of important periods is more difficult on the autocorrelation, we can use the periodogram for extracting period candidates. Let’s call the period candidates ‘hints’. These ‘hints’ may be false (due to spectral leakage), or provide a coarse estimate of the period (remember that DFT bins increase gradually in size); therefore a verification phase using the autocorrelation is required, since it provides a more fine-grained estimation of potential periodicities. The intuition is that if the candidate period from the periodogram lies on a hill of the ACF then we can consider it as a valid period, otherwise we discard it as false alarm. For the periods that reside on a hill, further refinement may be required if the periodicity hint refers to a large period.

Figure 3 summarizes our methodology and Figure 4 depicts the visual intuition behind our approach with a working example. The sequence is obtained from the

Method	Easy to threshold	Accurate short periods	Accurate large periods	Complexity
Periodogram	yes	yes	no	$O(N \log N)$
Autocorrelation	no	yes	yes	$O(N \log N)$
Combination	yes	yes	yes	$O(N \log N)$

Table 1: Concise comparison of approaches for periodicity detection.

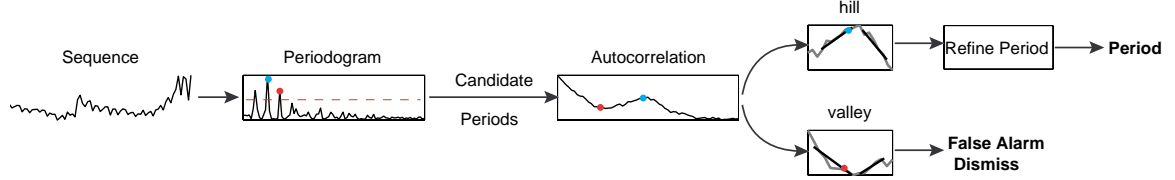


Figure 3: Diagram of our methodology (AUTOPERIOD method)

MSN query request logs and represents the aggregate demand for the query ‘Easter’ for 1000 days after the beginning of 2002. The demand for the specific query peaks during Easter time and we can observe one yearly peak. Our intuition is that periodicity should be approximately 365 (although not exactly, since Easter is not celebrated at the same date every year). Indeed the most dominant periodogram estimate is $333.33 = (1000/3)$, which is located on a hill of the ACF, with a peak at 357 (the correct periodicity -at least for this 3 year span). The remaining periodic hints can be discarded upon verification with the autocorrelation.

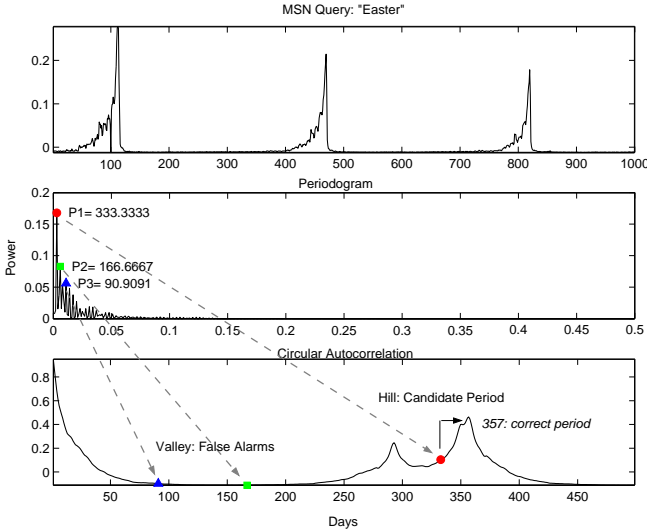


Figure 4: Visual demonstration of our method. Candidate periods from the periodogram are verified against the autocorrelation. Valid periods are further refined utilizing the autocorrelation information.

Essentially, we have leveraged the information of both metrics for providing an accurate periodicity detector. In addition, our method is computationally effi-

cient, because both the periodogram and the ACF can be directly computed through the Fast Fourier Transform of the examined sequence in $O(N \log N)$ time.

3.1 Discussion. First, we need to clarify succinctly that the use of the combined periodogram and autocorrelation does not carry additional information than each metric separately. This perhaps surprising statement can be verified by noting that:

$$\langle X, X^* \rangle = \|X\|^2$$

Therefore, the autocorrelation is the inverse Fourier transform of the periodogram, which means that the ACF can be considered as the dual of the periodogram, from the time into the frequency domain. In essence, our intention is to solve each problem in its proper domain; (i) the period significance in the frequency domain, and (ii) the identification of the exact period in the time domain.

Another issue that we would like to clarify is the reason that we are not considering a (seemingly) simpler approach for accurate periodicity estimation.

Looking at the problem from a signal processing perspective, one could argue that the inability to discover the correct period is due to the ‘coarse’ sampling of the series. If we would like to increase the resolution of the DFT, we could ‘sample’ our dataset at a finer resolution (upsampling). Higher sampling rate essentially translates into padding the time-series with zeros, and calculating the DFT of the longer time-series. Indeed, if we increase the size of the example sequence from 1000 to 16000, we will be able to discover the correct periodicity which is 357 (instead of the incorrect 333, given in the original estimate).

However, upsampling also imposes a significant performance overhead. If we are interested in obtaining online periodicity estimates from a data stream, this alternative method may result in a serious system

bottleneck. We can see this analytically; the time required to compute the FFT of a sequence with length 2^x is in the order of $2^x \log 2^x = x2^x$. Now let's assume that we pad the sequence with zeros increasing its length 16 times (just like in our working example). The FFT now requires time in the order of $(x+4)2^{x+4}$, which after algebraic calculations translates into 2 orders of magnitude additional time.

Using our methodology, we do not require higher sampling rates for the FFT calculation, hence keeping a low computational profile.

3.2 Discovery of Candidate Periods. For extracting a set of candidate periodicities from the periodogram, one needs to determine an appropriate power threshold that should distinguish only the dominant frequencies (or inversely the dominant periods). If none of the sequence frequencies exceeds the specific threshold (i.e., the set of periodicity 'hints' is empty), then we can regard the sequence as non-periodic.

In order to specify which periods are important, we first need to identify how much of the signal energy is attributed to random mechanisms, that is, everything that could not have been attributed to a random process should be of interest.

Let us assume that we examine a sequence x . The outcome of a permutation on the elements of x is a sequence \tilde{x} . The new sequence will retain the first order statistics of the original sequence, but will not exhibit any pattern or periodicities, because of the 'scrambling' process (even though such characteristics may have existed in sequence x). Anything that has the structure of \tilde{x} is not interesting and should be discarded, therefore at this step we can record the maximum power (p_{max}) that \tilde{x} exhibits, at any frequency f .

$$p_{max} = \arg \max_f \|\tilde{X}(f)\|^2$$

Only if a frequency of x has more power than p_{max} can be considered interesting. If we would like to provide a 99% confidence interval on what frequencies are important, we should repeat the above experiment 100 times and record for each one the maximum power of the permuted sequence \tilde{x} . The 99th largest value of these 100 experiments, will provide a sufficient estimator of the power threshold p_T that we are seeking. Periods (in the original sequence periodogram) whose power is more than the derived threshold will be considered:

$$p_{hint} = \{N/k : \mathcal{P}(f_{k/N}) > p_T\}$$

Finally, an additional period 'trimming' should be performed for discarding periods that are either too large or too small and therefore cannot be considered reli-

able. In this phase any periodic hint greater than $N/2$ or smaller than 2 is removed.

Figure 5 captures a pseudo-code of the algorithm for identifying periodic hints.

```

1 periods = getPeriodHints(Q)
2 {
3   k = 100; // number of permutations
4   maxPower = {}; // empty set
5   periods = {};
6
7   for i = 1 to k
8   {
9     Qp = permute(Q);
10    P = getPeriodogram(Qp);
11
12    power = max(P.power);
13    maxPower.add(power);
14  }
15
16  percentile = 99;
17  maxPower.sort; // ascending
18  P_threshold = maxPower(maxPower.length*(percentile/100));
19
20  P = getPeriodogram(Qp);
21
22  for i = 1 to P.length
23  {
24    if (P[i].power > P_threshold)
25      periods.add(P); // new candidate period
26  }
27
28  // period trimming
29  N = Q.length;
30  for i = 1 to periods.length
31  {
32    if (periods[i].hint >= N/2 || periods[i].hint <= 2)
33      periods[i].erase();
34  }
35
36  return periods;
37 }

```

Figure 5: Algorithm `getPeriodHints`

In [15] another algorithm for detection of important periods was proposed, which follows a different concept for estimating the periodogram threshold. The assumption there was that the periodogram of non-periodic time-series will follow an exponential distribution, which returned very intuitive period estimates for real world datasets. In our experiments, we have found the two algorithms to return very comparable threshold values. However, because the new method does not make any assumptions about the underlying distribution, we expect it to be applicable for a wider variety of time-series processes.

Examples: We use sequences from the MSN query logs (yearly span) to demonstrate the usefulness of the discovered periodic hints. In Figure 6(a) we present the demand of the query 'stock market', where we can distinguish a strong weekly component in the periodogram. Figure 6(b) depicts the query 'weekend' which does not contain any obvious periodicities. Our method can set the threshold high enough, therefore avoiding false

alarms.

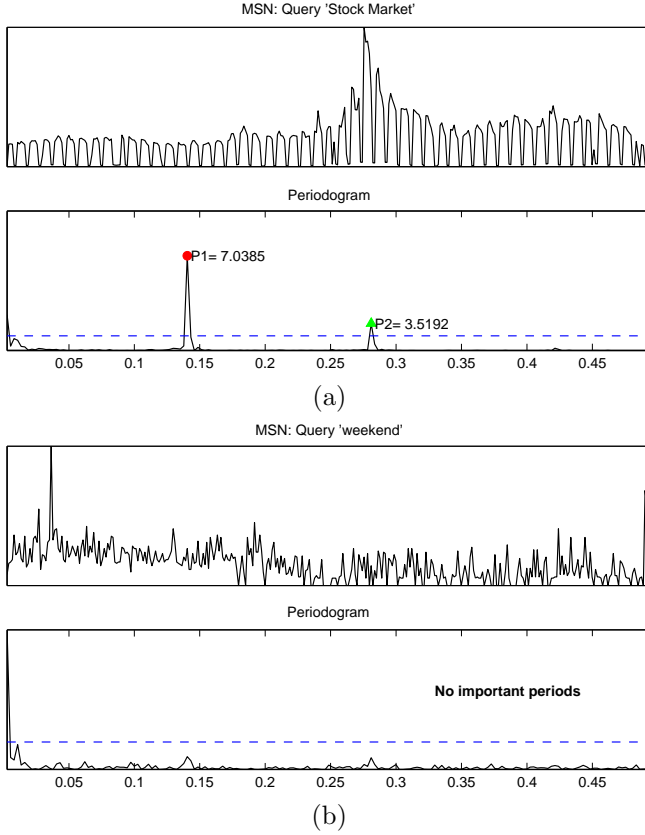


Figure 6: (a) Query 'stock market' (2002): Weekly periodic hint is identified. (b) Query 'weekend' (2002): No significant periodicities are spotted.

3.3 Verification of Candidate Periods. After the periodogram peaks have been identified, we have obtained a candidate set of periodicities for the examined sequence. The validity of these periods will be verified against the autocorrelation. An indication that a period is important, can be the fact that the corresponding period lies on a *hill* of the autocorrelation. If the period resides on a *valley* then it can be considered spurious and therefore safely discarded.

After we discover that a periodicity 'hint' resides on a hill of the autocorrelation, we can refine it even further by identifying the closest peak (i.e., local maximum). This is a necessary step, because the correct periodicity (i.e., peak of the hill) might not have been discovered by the periodogram, if it was derived from a 'wide' DFT bin. This is generally true for larger periods, where the resolution of the DFT bins drops significantly. We will explicate further, how to address the above issues:

3.3.1 Validity of Periodicity Hint. The significance of a candidate period ideally can be determined by examining the curvature of the ACF around the candidate period p . The autocorrelation is concave downward, if the second derivative is negative in an open interval $(a \dots b)$:

$$\frac{\partial^2 ACF(x)}{\partial x^2} < 0, \text{ for all } x \in (a \dots b), a < p < b$$

Nevertheless, small perturbations of the ACF due to the existence of noise, may invalidate the above requirement. Will will seek a more robust estimator of the curvature by approximating the ACF in the proximity of the candidate period with two linear segments. Then it is sufficient to examine if the approximating segments exhibit an upward-downward trend, for identifying a concave downward pattern (i.e., a hill).

The segmentation of a sequence of length N into k linear segments can be computed optimally using a dynamic programming algorithm in $O(N^2k)$ time, while a greedy merge algorithm achieves results very close to optimal in $O(N \log N)$ time [8]. For this problem instance, however, one can employ a simpler algorithm, because we require only a two segment approximation for a specific portion of the ACF.

Let \hat{S}_a^b be the linear regression of a sequence x between the positions $[a \dots b]$ and $\epsilon(\hat{S}_a^b)$ be the error introduced by the approximating segment. The best split position t_{split} is derived from the configuration that minimizes the total approximation error:

$$t_{split} = \arg \min_t \epsilon(\hat{S}_1^t) + \epsilon(\hat{S}_{t+1}^n)$$

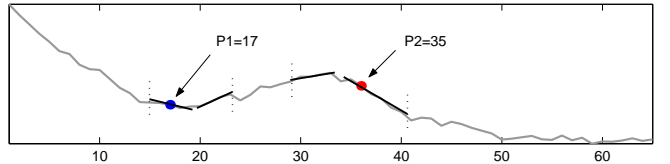


Figure 7: Segmentation of two autocorrelation intervals into two linear segments. The left region indicates a concave upward trend ('valley') while the right part consists of a concave downward trend ('hill'). Only the candidate period 35 can be considered valid, since it is located on a hill.

There is still the issue of the width of the search interval on the ACF, that is how much should we extend our search for a hill around the candidate period. Since the periodicity hint might have leaked from adjacent DFT bins (if it was located near the margin of the bin) we also examine half of the adjacent bins as well. Therefore, for a hint at period N/k , we examine the range $R_{N/k}$ of the ACF for the existence of a hill:

$$R_{N/k} = [\frac{1}{2}(\frac{N}{k+1} + \frac{N}{k}) - 1, \dots, \frac{1}{2}(\frac{N}{k} + \frac{N}{k-1}) + 1]$$

3.3.2 Identification of closest Peak. After we have ascertained that a candidate period belongs on a hill and not on a valley of the ACF, we need to discover the closest peak which will return a more accurate estimate of the periodicity hint (particularly for larger periods). We can proceed in two ways; the first one would be to perform any hill-climbing technique, such as gradient ascent, for discovering the local maximum. In this manner the local search will be directed toward the positive direction of the first derivative. Alternatively, we could derive the peak position directly from the linear segmentation of the ACF, which is already computed in the hill detection phase. The peak should be located either at the end of the first segment or at the beginning of the second segment.

We have implemented both methods for the purposes of our experiments and we found both of them to report accurate results.

4 Extension for Streaming Data.

Even though we have presented the AUTOPERIOD algorithm for static time-series, it can be easily extended for a streaming scenario, by adapting an incremental calculation of the Fourier Transform. Incremental Fourier computation has been a topic of interest since the late 70s and it was introduced by Papoulis [13] under the term ‘Momentary Fourier Transform’ (MFT). MFT covered the aggregate (or growing) window case, however recent implementations also deal with the sliding window case, such as in [16, 11]. Incremental AUTOPERIOD requires only constant update time per DFT coefficient, and linear space for recording the window data.

5 Accuracy of Results

We use several sequences from the MSN query logs to perform convincing experiments regarding the accuracy of our 2-tier methodology. The specific dataset is ideal for our purposes because we can detect a number of different periodicities according to the demand pattern of each query.

The examples in Figure 8 demonstrate a variety of situations that might occur when using both the periodogram and autocorrelation.

■ *Query ‘Easter’(MSN)*: Examining the demand for a period of 1000 days, we can discover several periodic hints above the power threshold in the periodogram. In this example, the autocorrelation information refines the original periodogram hint (from 333 \rightarrow 357). Additional hints are rejected because they reside on ACF valleys (in the figure only the top 3 candidate periods are displayed for reasons of clarity).

■ *Query ‘Harry Potter’(MSN)*: For the specific query although there are no observed periodicities (duration 365 days), the periodogram returns 3 periodic hints, which are mostly attributed to the burst pattern during November when the movie was released. The hints are classified as spurious upon verification with ACF.

■ *Query ‘Fourier’(MSN)*: This is an example where the periodogram threshold effectively does not return candidate periods. Notice that if we had utilized only the autocorrelation information, it would have been more troublesome to discover which (if any) periods were important. This represents another validation that our choice to perform the period thresholding in the frequency space was correct.

■ *Economic Index (Stock Market)*: Finally, this last sequence from a stock market index illustrates a case where both the periodogram and autocorrelation information concur on the single (albeit weak) periodicity.

Through this experimental testbed we have demonstrated that AUTOPERIOD can provide very accurate periodicity estimates without upsampling the original sequence. In the sections that follow, we will show how it can be used in conjunction with periodic similarity measures, for interactive exploration of sequence databases.

6 Structure-Based Similarity and Periodic Measures

We introduce structural measures that are based on *periodic* features extracted from sequences. Periodic distance measures can be used for providing more meaningful structural clustering and visualization of sequences (whether they are periodic or not). After sequences are grouped in ‘periodic’ clusters, using a ‘drill-down’ process the user can selectively apply the AUTOPERIOD method for periodicity estimation on the sequences or clusters of interest. In the experimental section we provide examples of this methodology using hierarchical clustering trees.

Let us consider first the utility of periodic distance measures with an example. Suppose that one is examining the similarity between the two time-series of Figure 9. When sequence *A* exhibits an upward trend, sequence *B* displays a downward drift. Obviously, the Euclidean distance (or inner product) between sequences *A* and *B*, will characterize them as very different. However, if we exploit the frequency content of the sequences and evaluate their periodogram, we will discover that it is almost identical. In this new space, the Euclidean distance can easily identify the sequence similarities. Even though this specific example could have been addressed in the original space using the Dynamic Time Warping

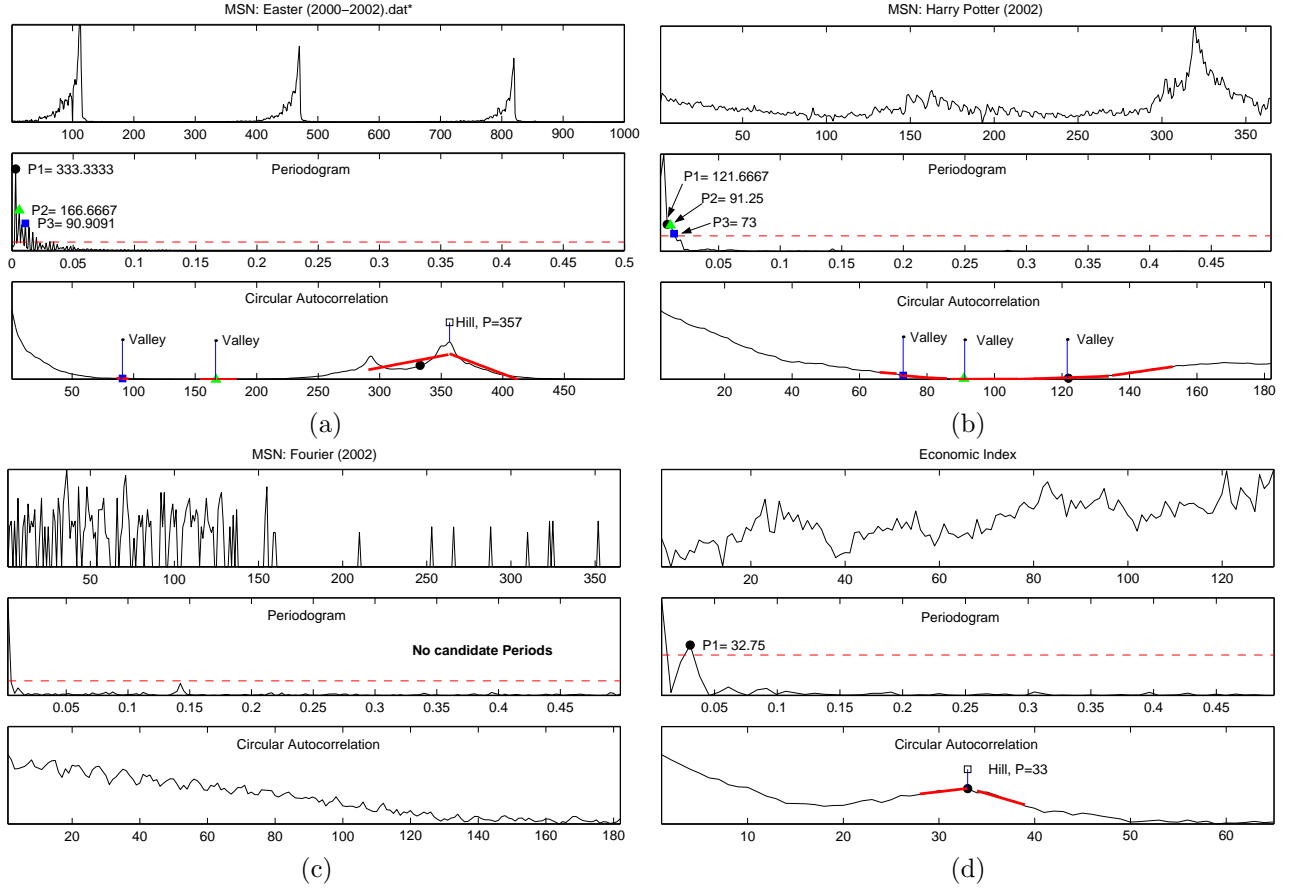


Figure 8: Periodicity detection results of the AUTOPERIOD method.

(DTW) distance, we have to note that our method is significantly more efficient (in terms of both time and space) than DTW. Additionally, periodic measures can address more subtle similarities that DTW cannot capture, such as *different* patterns/shapes occurring at periodic (possibly non-aligned) intervals. We will examine cases where the DTW fails in the sections that follow.

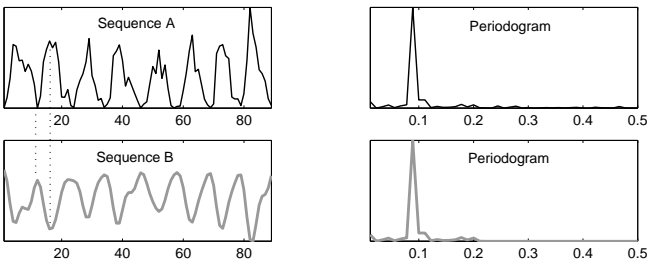


Figure 9: Sequences *A* and *B* are very distant in a Euclidean sense in the time domain. A transformation in the frequency domain (using a view of the periodogram) reveals the structural similarities.

The new measure of structural similarity that we

present, exploits the *power* content of only the most dominant periods/frequencies. By considering the most powerful frequencies, our method concentrates on the most important structural characteristics, effectively filtering out the negative influence of noise, and eventually allowing for expedited distance computation. Additionally, the omission of the phase information renders the new similarity measure shift invariant in the time domain. We can therefore discover time-series with similar patterns, which may occur at different chronological instants.

6.1 Power Distance (pDist). For comparing the periodic structure of two sequences, we need to examine how different is their harmonic content. We achieve this by utilizing the periodogram and specifically the frequencies with the highest energy.

Suppose that X is the Fourier transform of a sequence x with length n . We can discover the k largest coefficients of X by computing its periodogram $\mathcal{P}(X)$ and recording the position of the k frequencies with the highest power content (parameter k depends

on the desired compression factor). Let us denote the vector holding the positions of the coefficients with the largest power p^+ (so $p^+ \subset [1 \dots n]$). To compare x with any other sequence q , one needs to examine how similar energies they carry in the dominant periods of x . Therefore, we evaluate $\mathcal{P}(Q(p^+))$, that describes a sequence holding the equivalent coefficients as the vector $\mathcal{P}(X(p^+))$. The distance **pDist** between these two vectors captures the periodic similarity between sequences x and q :

$$pDist = \|\mathcal{P}(Q(p^+)) - \mathcal{P}(X(p^+))\|$$

Example: Let x and q be two sequences and let their respective Fourier Transforms be $X = \{(1+2i), (2+2i), (1+i), (5+i)\}$ and $Q = \{(2+2i), (1+i), (3+i), (1+2i)\}$. The periodogram vector of X is: $\mathcal{P}(X) = \|X\|^2 = (5, 8, 2, 26)$. The vector holding the positions of X with highest energy is $p^+ = (2, 4)$ and therefore $\mathcal{P}(X(p^+)) = (0, 8, 0, 26)$. Finally, since $\mathcal{P}(Q) = (8, 2, 10, 5)$ we have that: $\mathcal{P}(Q(p^+)) = (0, 2, 0, 5)^2$.

In order to meaningfully compare the power content of two sequences we need to normalize them, so that they contain the same amount of total energy. We can assign to any sequence $x(n)$ unit power, by performing the following normalization:

$$\hat{x}(n) = \frac{x(n) - \frac{1}{N} \sum_{i=1}^N x(i)}{\sqrt{\sum_{i=1}^N (x(n) - \frac{1}{N} \sum_{i=1}^N x(i))^2}}, \quad n = 1, \dots, N$$

The above transformation will lead to zero mean value and sum of squared values equal to 1. Parseval's theorem dictates that the energy in the time domain equals the energy in the frequency domain, therefore the total energy in the frequency domain should also be unit:

$$\|\hat{x}\|^2 = \|\mathcal{F}(\hat{x})\|^2 = 1$$

After this normalization, we can more meaningfully compare the periodogram energies.

Indexability: Although in this work we are not going to discuss now to index the **pDist**, we would like to note that this is possible. The representation that we are proposing, utilizes a different set of coefficients for every sequence. While indexing might appear problematic using space partitioning indices such as R-trees (because they operate on a fixed set of dimensions/coefficients), such representations can be easily indexed using metric tree structures, such as VP-Tree or M-Tree (more details can be found in [15]).

7 Periodic Measure Results

We present extensive experiments that show the usefulness of the new periodic measures and we compare them with widely used shape based measures or newly introduced structural distance measures.

7.1 MSN query logs. Using 16 sequences which record the yearly demand of several keywords at the MSN search engine, we perform the hierarchical clustering which is shown in Figure 10. In the dendrogram derived using the **pDist** as the distance function, we can notice a distinct separation of the sequences/keywords into 3 classes. The first class contains no clear periodicities (no specific pattern in the demand of the query), while the second one exhibits only bursty seasonal trends (e.g., during Christmas). The final category of queries are requested with high frequency (weekly period) and here we can find keywords such as 'cinema', 'bank', 'Bush' etc.

We utilize an extended portion of the same dataset for exploring the visualization power of periodic distance measures. Using the pairwise distance matrix between a set of MSN keyword demand sequences (365 values, year 2002), we evaluate a 2D mapping of the keywords using Multidimensional Scaling (Figure 11). The derived mapping shows the high discriminatory efficacy of the **pDist** measure; seasonal trends (low frequencies) are disjoint from periodic patterns (high frequencies), allowing for a more structural sequence exploration. Keywords like 'fall', 'Christmas', 'lord of the rings', 'Elvis', etc, manifest mainly seasonal bursts, which need not be aligned in the time axis. On the contrary, queries like 'dry cleaners' or 'Friday' indicate a natural weekly repeated demand. Finally, some queries do not exhibit any obvious periodicities within a year's time (e.g., 'icdm', 'kdd', etc).

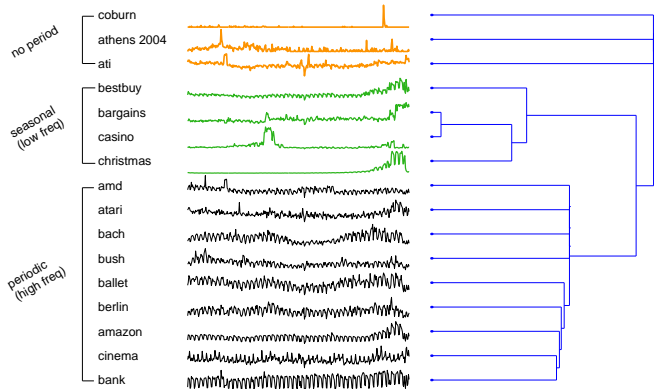


Figure 10: Dendrogram based on periodic features

²The zeros are placed in the vectors for clarity reasons. In the actual calculations they can be omitted.

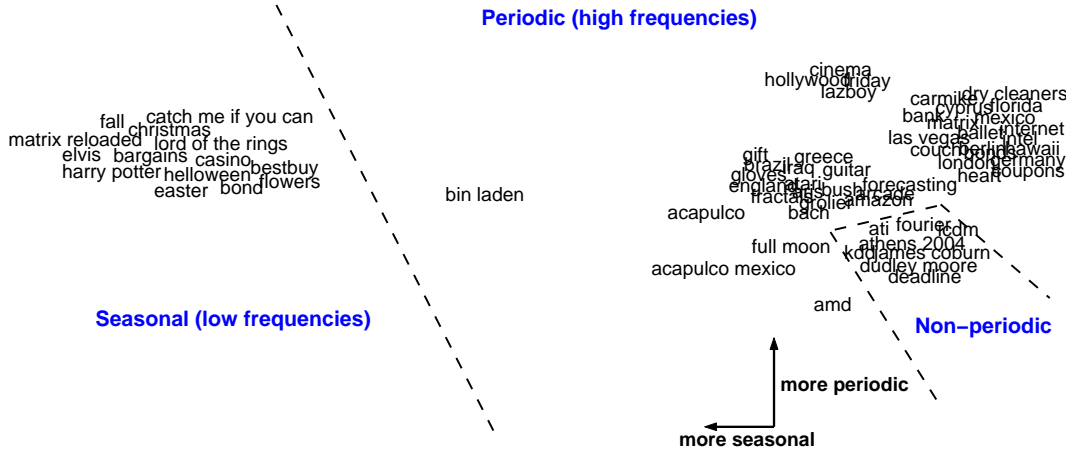


Figure 11: Mapping on 2D of pairwise distances between several sequences. The similarity measure utilized was the power based similarity. We can clearly distinguish a separation between periodic and seasonal trends.

7.2 Structured + Random Mixture. For our second experiment we use a combination of periodic time-series that are collected from natural sciences, medicine and manufacturing, augmented by pairs of random noise and random walk data.

All datasets come in pairs, hence, when performing a hierarchical clustering algorithm on this dataset, we expect to find a direct linkage of each sequence pair at the lower level of the dendrogram. If this happens we consider the clustering correct. The dataset consists of 12 pairs, therefore a measure of the clustering accuracy can be the number of correct pair linkages, over twelve, the number of total pairs.

Figure 12 displays the resulting dendrogram for the `pDist` measure, which achieves a perfect clustering. We can also observe that pairs derived from the same source/process are clustered together as well, in the higher dendrogram level (Power Demand, ECG, MotorCurrent etc). After the clustering, we can execute the `AUTOPERIOD` method and annotate the dendrogram with the important periods of every sequence. Some sequences, like the random walk or the random data, do not contain any periodicities, which we indicate with an empty set $\{\}$. When both sequences at the lower level display the same periodicity, a single set is displayed on the bifurcation for clarity.

For many datasets that came into 2 pairs (power demand, video surveillance, motor current), all 4 instances instances demonstrated the same basic period (as suggested by the `AUTOPERIOD`). However, the periodic measure can effectively separate them into two pairs, because the power content of the respective frequencies was different.

For example, in the video surveillance dataset, both actors display a periodic movement every 30 units

(drawing a gun from a holster). However, because the male person performs the movement with wider ‘arches’ (because of different body structure), the periodic measure can distinguish his movement, due to the higher energy content. The above example indicates that analogous periodic measures could be effectively used for biometric characterization, since every individual tends to have a distinct intrinsic rhythm (e.g., when typing on the keyboard, performing repetitive moves, speaking, etc).

On the sunspot sequence set the `AUTOPERIOD` estimates of 89 and 84 units may appear erroneous at first glance, because of our knowledge that the solar cycles range from 10 to 12 years. However, this is not the case because the 1000 sequence points record sunspot measurements of approximately 120 years. After the proper rescaling the estimates of 89 and 84 yield periodicities close to 11 and 10 years respectively.

Euclidean	DTW	Cepstrum	CDM	pDist
0.16	0.66	0.75	1	1

Table 2: Clustering accuracy for the dataset of fig. 12

On the same dataset the accuracy results for Euclidean, DTW, Cepstrum and CDM compression based measure [10] are given in table 2. CDM is the only one that also achieves perfect clustering. However, it should be noted that while all other methods operate on the original dimensional space (using 1000 points), `pDist` works on a very lower dimensional space, using only 50 numbers to describe each sequence, after a 20x compression of the data.

7.3 ECG datasets. Our last experiment is performed on the MIT-BIH Arrhythmia dataset. We use

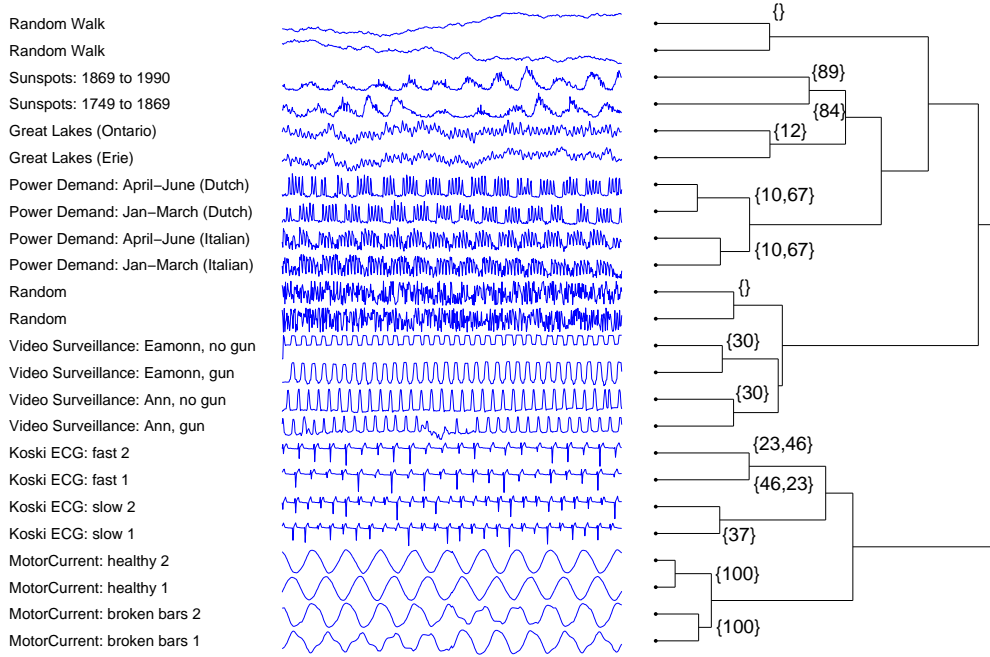


Figure 12: The `pDist` measure produces an accurate dendrogram based on the periodic structural characteristics of a dataset. The lower dendrogram levels are also annotated by the periods discovered as important, by a subsequent run of the `AUTOPERIOD` method.

two sets of sequences; one with 2 classes of heartbeats and another one with three (figures 13, 14). We present the dendrogram of the `pDist` measure and the DTW, which represents possibly one of the best shape based distance measures. To tune the single parameter of the DTW (corresponding to the maximum warping length) we probed several values and here we report the one that returned the best clustering.

For both dataset instances, `pDist` again returns an accurate clustering, while DTW seems to perform badly on the high level dendrogram aggregations, hence not leading to perfect class separation. The Euclidean distance reported worse results. The CDM measure is accurate on the 2 class separation test but does not provide a perfect separation for the 3 class problem (see the original paper [10] for respective results).

7.4 Distance Measures Overview. The experiments have testified to the utility of periodic measures for exploration of sequence databases. The only real contender to the `pDist` measure is the compression-based CDM measure. However, compared to CDM our approach presents some favorable advantages: (i) it does not require any discretization phase (we operate on the original data), (ii) it is meaningful for both long and short sequences (CDM performs better on longer sequences) (iii) it can be easily extended for streaming

sequences, using incremental Fourier Transform computation (iv) it provides additional sequence information in the form of periodic estimates.

8 Conclusion

We have presented methods for accurate periodicity estimation and for characterization of structural periodic similarity between sequences. It is our belief that these methods will find many applications for interactive exploration of time-series databases and for classification or anomaly detection of periodic sequences (e.g., in auto manufacturing, biometrics and medical diagnosis).

Acknowledgements: We are thankful to MSN and Microsoft for letting us use a portion of the MSN query logs. We also wish to thank Eamonn Keogh for numerous suggestions and for kindly donating his dendrogram code.

References

- [1] G. Ebersbach, M. Sojer, F. Valdeoriola, J. Wissel, J. Müller, E. Tolosa, and W. Poewe. Comparative analysis of gait in parkinson’s disease, cerebellar ataxia and subcortical arteriosclerotic encephalopathy. In *Brain*, Vol. 122, No. 7, 1349-1355, July 1999.
- [2] M. G. Elfeke, W. G. Aref, and A. K. Elmagarmid.

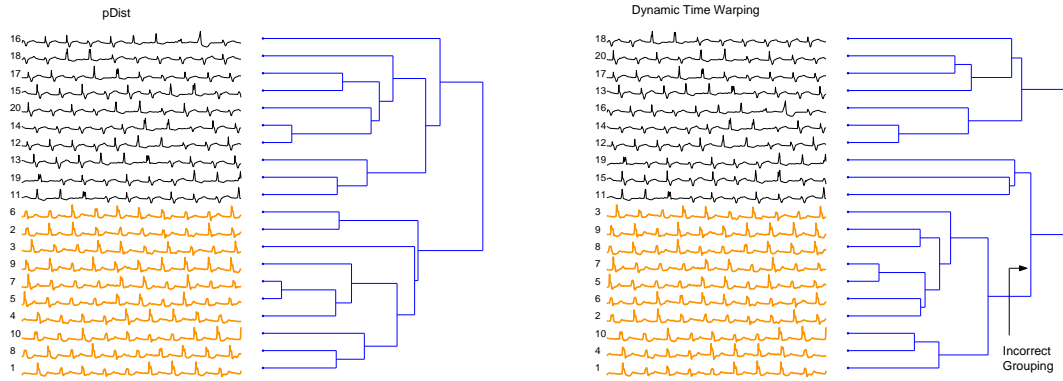


Figure 13: 2 class ECG problem: DTW provides incorrect grouping

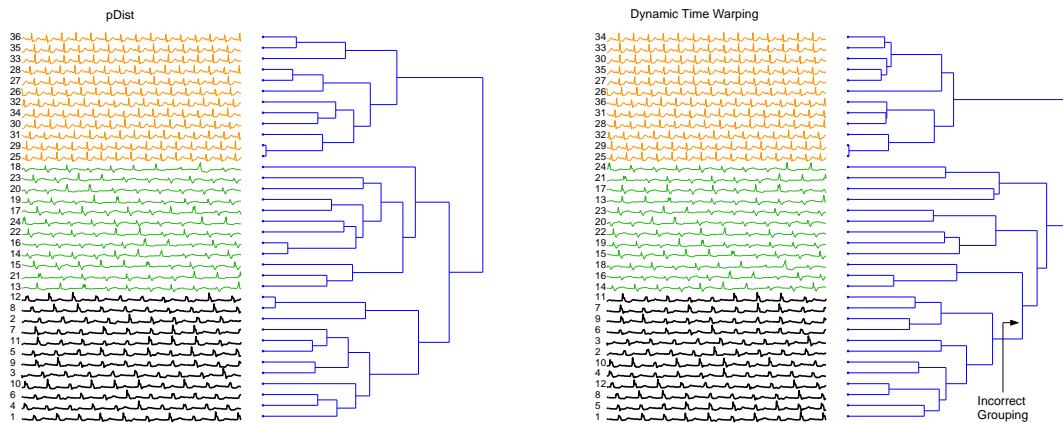


Figure 14: 3 class ECG problem: only pDist provides correct clustering into 3 groups

- Using Convolution to Mine Obscure Periodic Patterns in One Pass. In *Proc. of EDBT*, 2004.
- [3] F. Ergün, S. Muthukrishnan, and S. C. Sahinalp. Sublinear methods for detecting periodic trends in data streams. In *LATIN*, 2004.
 - [4] E. Friss-Cristensen and K. Lassen. Length of solar cycle - An Indicator of solar-activity closely related with climate. In *Science*, 254, pages 698–700, 1991.
 - [5] F. J. Harris. On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. In *Proc. of the IEEE*, Vol. 66, No 1, 1978.
 - [6] T. Idé and K. Inoue. Knowledge Discovery from Time-Series Data using Nonlinear Transformations. In *Proc. of the 4th Data Mining Workshop (Japan Soc. for Software Science and Technology*, 2004.
 - [7] K. Kalpakis, D. Gada, and V. Puttagunta. Distance Measures for Effective Clustering of ARIMA Time-Series. In *Proc. of ICDM*, 2001.
 - [8] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Proc. of ICDM*, 2001.
 - [9] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *Proc. of SIGKDD*, 2002.
 - [10] E. Keogh, S. Lonardi, and A. Ratanamahatana. Towards parameter-free data mining. In *Proc. of SIGKDD*, 2004.
 - [11] M. Kontaki and A. Papadopoulos. Efficient similarity search in streaming time sequences. In *SSDBM*, 2004.
 - [12] J. S. Mitchell. *An introduction to machinery analysis and monitoring*. PennWell Publ. Co., 1993.
 - [13] A. Papoulis. *Signal Analysis*. McGraw-Hill, 1977.
 - [14] J. Tulena, M. Azzolini, J. de Vriesa, W. H. Groeneveld, J. Passchier, and B. van de Wetering. Quantitative study of spontaneous eye blinks and eye tics in Gilles de la Tourette's syndrome. In *Journal of Neurol. Neurosurg. Psychiatry* 1999,67:800-802.
 - [15] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identification of Similarities, Periodicities & Bursts for Online Search Queries. In *Proc. of SIGMOD*, 2004.
 - [16] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, 2002.

Cross Table Cubing: Mining Iceberg Cubes from Data Warehouses*

Moonjung Cho

State University of New York at Buffalo, U.S.A.
mcho@cse.buffalo.edu

Jian Pei

Simon Fraser University, Canada
jpei@cs.sfu.ca

David W. Cheung

The University of Hong Kong, China
dcheung@csis.hku.hk

Abstract

All of the existing (iceberg) cube computation algorithms assume that the data is stored in a single base table, however, in practice, a data warehouse is often organized in a schema of multiple tables, such as star schema and snowflake schema. In terms of both computation time and space, materializing a universal base table by joining multiple tables is often very expensive or even unaffordable in real data warehouses. In this paper, we investigate the problem of computing iceberg cubes from data warehouses. Surprisingly, our study shows that computing iceberg cube from multiple tables directly can be even more efficient in both space and runtime than computing from a materialized universal base table. We develop an efficient algorithm, CTC (for Cross Table Cubing) to tackle the problem. An extensive performance study on synthetic data sets demonstrates that our new approach is efficient and scalable for large data warehouses.

1 Introduction

Given a base table $B(D_1, \dots, D_n, M)$ and an aggregate function, where D_1, \dots, D_n are n dimensions and M is a measure, a data cube consists of the complete set of group-bys on any subsets of dimensions and their aggregates using the aggregate function. A data cube in practice is often huge due to the very large number of possible dimension value combinations. Since many detailed aggregate cells whose aggregate values are too small may be trivial in data analysis, instead of computing a complete cube, an iceberg cube can be computed, which consists of only the set of group-bys whose aggregates are no less than a user-specified aggregate threshold.

In the previous studies, several efficient algorithms have been proposed to compute (iceberg) cubes efficiently from a *single* base table, with simple or complex measures, such as BUC [1] and H-Cubing [4]. All of them assume that the data is stored in a single base table. However, a data warehouse in practice is often organized in a schema of multiple tables, such as star schema or snowflake schema. Although mining iceberg cube from single table becomes more and more efficient, such algorithms cannot be applied directly to real data warehouses in many applications.

K	A_1	\dots	A_n	M
k	$a_{1,1}$	\dots	$a_{1,n}$	m_1
\dots	\dots	\dots	\dots	\dots
k	$a_{l,1}$	\dots	$a_{l,n}$	m_l

Table F

K	B_1	\dots	B_m
k	b_1	\dots	b_m

Table D

A_1	\dots	A_n	K	B_1	\dots	B_m	M
$a_{1,1}$	\dots	$a_{1,n}$	k	b_1	\dots	b_m	m_1
\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots
$a_{l,1}$	\dots	$a_{l,n}$	k	b_1	\dots	b_m	m_l

Universal base table $B = F \bowtie D$

Figure 1: A simple case of computing iceberg cube from two tables.

EXAMPLE 1. (INTUITION) Consider computing the iceberg cube from tables F and D in Figure 1. Suppose attribute M is the measure. A rudimentary method may first compute a universal base table $B = F \bowtie D$, as also shown in the figure, and then compute the iceberg cube from B . However, such a rudimentary method may suffer from two non-trivial costs.

Space cost. As shown in the figure, the tuple in table D is replicated l times in the universal base table B , where l is the number of tuples in the fact table. Moreover, every attribute in the tables appears in the universal base table. Thus, the universal base table is wider than any table in the original database. In real applications, there can be a large number of tuples in the fact table, and hundreds of attributes in the database. Then, the dimension information may be replicated many times, and the universal base table may be very wide – containing hundreds of attributes.

Time cost. The large base table may have to be scanned many times and many combinations of attributes may have to be checked. As the universal base table can be much wider and larger than the original tables, the computation time can be dramatic.

Can we compute iceberg cubes directly from F and D without materializing the universal base table B ? The following two observations help.

First, for any combination of attributes in table D , the aggregate value is $m = \text{aggr}(\{m_1, \dots, m_l\})$. Therefore, if m satisfies the iceberg condition, then every combination of attributes in D is an iceberg cell. Here, we compute these iceberg cells using table D only, which contains only 1 tuple. In the rudimentary method, we have to use many tuples in table B to compute these iceberg cells.

Second, for any iceberg cell involving attributes in table F , the aggregate value can be computed from table F only. In other words, if we find an iceberg cell in F , we can

*This research is supported in part by NSF Grant IIS-0308001, a President's Research Grant, an Endowed Research Fellowship Award and a startup grant in Simon Fraser University. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

enumerate a whole bunch of iceberg cells by inserting more attributes in D and the aggregate value retains. Please note that we only use F , which has only $(n + 1)$ attributes, to compute these iceberg cells. In the rudimentary method, we have to compute these iceberg cells using a much wider universal base table B .

Although the observations here are based on an oversimplified case, as shown in the rest of the paper, the observations can be generalized. ■

In this paper, we make the following contributions. First, we address the problem of mining iceberg cubes from data warehouses of multiple tables. We use star schema as an example. Our approach can be easily extended to handle other schemas in data warehouses, such as snowflake schema.

Second, we develop an efficient algorithm, CTC (for Cross Table Cubing), to compute iceberg cubes. Our method does not need to materialize the universal base table. Instead, CTC works in three steps. First, CTC propagates the information of keys and measure to each dimension table. Second, the local iceberg cube in each table is computed. Last, the global iceberg cube is derived from the local ones. We show that CTC is more efficient in both space and runtime than computing iceberg cube from a materialized universal base table.

Last, we conduct an extensive performance study on synthetic data sets to examine the efficiency and the scalability of our approach. The experimental results show that CTC is efficient and scalable for large data warehouses.

The rest of the paper is organized as follows. In Section 2, we formulate the problem and review related work. Algorithm CTC is developed in Section 3 by examples. A performance study is briefly reported in Section 4.

2 Problem Definition and Related Work

Without loss of generality, we assume that the domains of the attributes in the tables are exclusive, except for the foreign keys K_i in the fact table F referencing to the primary keys K_i in dimension table D_i .

EXAMPLE 2. (STAR SCHEMA) Consider the data warehouse DW in Figure 2. We will use this data warehouse as the running example in the rest of the paper.

The star schema is shown in Figure 2(a). In data warehouse DW , the fact table $Fact$ has 3 dimensions, namely A , B and E . The measure is M . Dimensions B and E reference to dimension tables D_1 and D_2 , respectively. In data warehouse DW , the universal base table $T_{base} = Fact \bowtie D_1 \bowtie D_2$ is shown in Figure 3. ■

Let $B = (A_1, \dots, A_m, M)$ be a universal base table, where A_1, \dots, A_m are either dimensions or attributes in dimension tables. A cell $c = (a_1, \dots, a_m)$ is called an *aggregate cell*, where $a_i \in A_i$ or $a_i = *$ ($1 \leq i \leq m$). The *cover* of c is the set of tuples in B that match all non- $*$ a_i 's, i.e., $cov(c) = \{t \in B \mid \forall a_i \neq *, t.A_i = a_i\}$.

For an aggregate function $aggr()$ on the domain of M , $aggr(c) = aggr(cov(c))$.

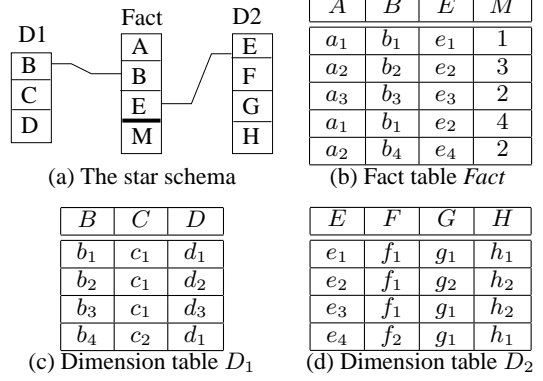


Figure 2: Data warehouse DW as the running example.

A	B	C	D	E	F	G	H	M
a_1	b_1	c_1	d_1	e_1	f_1	g_1	h_1	1
a_2	b_2	c_1	d_2	e_2	f_1	g_2	h_2	3
a_3	b_3	c_1	d_3	e_3	f_1	g_1	h_2	2
a_1	b_1	c_1	d_1	e_2	f_1	g_2	h_2	4
a_2	b_4	c_2	d_1	e_4	f_2	g_1	h_1	2

Figure 3: The universal base table T_{base} .

For an iceberg condition C , where C is defined using some aggregate functions, a cell c is called an *iceberg cell* if c satisfies C . An *iceberg cube* is the complete set of iceberg cells.

EXAMPLE 3. (ICEBERG CUBE) In base table T_{base} (Figure 3), for aggregate cell $c = (*, b_1, *, d_1, *, f_1, *, *)$, $cov(c)$ contains 2 tuples, the first and the fourth tuples in T_{base} , since they match c in dimensions B , D and F . We have $COUNT(cov(c)) = 2$.

Consider iceberg condition $C \equiv (COUNT(c) \geq 2)$. Aggregate cell c satisfies the condition and thus is in the iceberg cube. ■

Problem definition. The problem of computing iceberg cube from data warehouse is that, given a data warehouse and an iceberg condition, compute the iceberg cube. Limited by space, we only discuss data warehouses in star schema in this paper. ■

For aggregate cells $c = (a_1, \dots, a_m)$ and $c' = (a'_1, \dots, a'_m)$, c is called an *ancestor* of c' and c' a *descendant* of c if for any $a_i \neq *, a'_i = a_i$ ($1 \leq i \leq m$), denoted by $c' \sqsubseteq c$. An iceberg condition C is called *monotonic* if for any aggregate cell c , if C holds for c , then C also holds for every ancestor of c . In this paper, we only consider monotonic iceberg conditions, such as $COUNT(c) \geq v$, $MAX(c) \geq v$, $MIN(c) \leq v$, $SUM(c) \geq v$ (if the domain of the measure consists of only non-negative numbers).

Many approaches have been proposed to compute data cubes efficiently from scratch (e.g., [6, 1]). In general, they speed up the cube computation by sharing partitions, sorts, or partial sorts for group-bys with common dimensions.

Fang et al. [2] proposed the concept of iceberg queries and developed some sampling algorithms to answer such

queries. Beyer and Ramakrishnan [1] introduced the problem of iceberg cube computation in the spirit of [2] and developed algorithm BUC. BUC conducts bottom-up computation and can use the monotonic iceberg conditions to prune.

H-cubing [4] uses a hyper-tree data structure called H-tree to compress the base table. The H-tree can be traversed bottom-up to compute iceberg cubes. It also can prune unpromising branches of search using monotonic iceberg conditions. Moreover, a strategy was developed in [4] to use weakened but monotonic conditions to approximate non-monotonic conditions to compute iceberg cubes. The strategies of pushing non-monotonic conditions into bottom-up iceberg cube computation were further improved by Wang et al. [5]. A new strategy, divide-and-approximate, was developed.

All of the previous studies on computing (iceberg) cubes assume that *a universal base table is materialized*. However, many real data warehouses are stored in tens or hundreds of tables. *It is often unaffordable to compute and materialize a universal base table for iceberg cube computation*. This observation motivates the study in this paper.

3 CTC: A Cross Table Cubing Algorithm

For an iceberg cell c with respect to a monotonic iceberg condition, its projections on the fact table and the dimension tables must also be local iceberg cells. Instead of directly computing the iceberg cube from a universal base table, we can compute local iceberg cubes from the fact table and the dimension tables, respectively. Then, we can try to derive the global iceberg cube from the local ones.

Based on the above observation, algorithm CTC works in three steps. First, the aggregate information is propagated from the fact table to each dimension tables. Then, the iceberg cubes in the propagated dimension tables as well as in the fact table (i.e., the *local iceberg cubes*) are mined independently using the same iceberg cube condition. Last, the iceberg cells involving attributes in multiple dimension tables are derived from the local iceberg cubes.

3.1 Propagation Across Tables

EXAMPLE 4. (PROPAGATING AGGREGATE INFORMATION) Consider our running example data warehouse (Figure 2) again. To propagate the aggregate information from the fact table *Fact* to the dimension tables D_1 and D_2 , we create a new attribute *Count* in every dimension table. By scanning the fact table once, the number of occurrences of each foreign key value in the fact table can be counted. Such information is registered in the column of *Count* in the dimension tables, as shown in Figure 4. Hereafter, the propagated dimension tables are denoted as PD_1 and PD_2 , respectively, to distinguish from the original dimension tables.

In the rest of the computation, we only use the fact table and the propagated dimension tables PD_1 and PD_2 . We will show that the iceberg cube computed from these three tables is the same as the one computed from the universal base table. ■

B	C	D	$Count$
b_1	c_1	d_1	2
b_2	c_1	d_2	1
b_3	c_1	d_3	1
b_4	c_2	d_1	1

(a) Propagated PD_1

E	F	G	H	$Count$
e_1	f_1	g_1	h_1	1
e_2	f_1	g_2	h_2	2
e_3	f_1	g_1	h_2	1
e_4	f_2	g_1	h_1	1

(b) Propagated PD_2

Figure 4: The propagated dimension tables.

This computation of the aggregates on the keys is implemented as group-by aggregate queries on the key attributes in the fact table. Only the fact table is needed to conduct such queries. The aggregate information is appended to the records in the dimension tables after the aggregates are computed. In general, we extend every dimension table to include a measure column.

CTC never really joins multiple tables. Instead, it only conducts group-by queries on each key attribute and propagates the aggregates to the corresponding dimension table. When there are multiple dimension tables, propagating the aggregates is much more cheaper than joining multiple tables and materializing a universal base table. We notice that there are efficient indexing techniques to join tables in star schema fast. Most of those techniques can also be used to propagate the aggregates to dimension tables efficiently.

3.2 Computation of Local Iceberg Cubes Local iceberg cubes on propagated dimension tables can be computed using an adaption of any algorithms for iceberg cube computation. For each iceberg cell c , we maintain the histogram of primary key values that the tuples in $cov(c)$ carry.

EXAMPLE 5. (COMPUTING LOCAL ICEBERG CUBE)

We can compute the iceberg cube on propagated dimension table PD_2 (Figure 4(b)) with respect to condition $C \equiv COUNT(c) \geq 2$ using algorithm BUC [1]. One advantage of computing iceberg cubes on propagated dimension tables is that *one tuple in the propagated dimension table may summarize multiple tuples in the corresponding projection of the universal base table*. Thus, we reduce the number of tuples in the computation.

For each iceberg cell, we record the histogram of primary key values that the tuples in the cell carry. For example, for iceberg cell $(*, f_1, *, *)$ with count 4, we record the set of primary key values $\{e_1, e_2, e_3\}$ that the tuples having f_1 carry. This is called the *signature* of the iceberg cell. It will be used in the future to derive global iceberg cells. To maintain the signature, we can use a vector of m bits for every iceberg cell, where m is the number of distinct values appearing in attribute E (the primary key attribute) in table PD_2 . ■

Let D be a dimension table and K be a primary key attribute such that K is used in the fact table as the foreign key referencing to D . For an iceberg cell c in D , the *signature* of c , denoted as $c.sig$, is the set of primary key values (i.e., values in K) that appear in the tuples in $cov(c)$ in D . Clearly, to maintain the signatures in D , we only need m bits, where m is the number of distinct values in K that

appear in the fact table. m is at most the number of tuples in D , and no more than the cardinality of K .

3.3 Computation of Global Iceberg Cubes The set of global iceberg cells can be divided into two exclusive subsets: the ones having some non- $*$ values on the dimension attributes in the fact table, and the ones whose projections on the fact table are $(*, \dots, *)$. We handle them separately.

EXAMPLE 6. (ICEBERG CELL INVOLVING FACT TABLE)

Now, we consider the iceberg cells that contain some non- $*$ values in the dimension attributes in fact table *Fact*. To find such iceberg cells, we first apply an iceberg cube computing algorithm, such as BUC [1], to the fact table.

For example, we find $(a_1, *, *) : 2$ is an iceberg cell in the fact table. In the cover of $(a_1, *, *)$ (i.e., the first and the fourth tuples in Figure 2(b)), b_1 appears in attribute B , which references to dimension table D_1 . Thus, for any local iceberg cell c in PD_1 whose signature contains b_1 , such as $(b_1, *, *)$, $(*, c_1, *)$, and $(*, c_1, d_1)$, the join¹ of $(a_1, *, *)$ and c , such as $(a_1, b_1, *, *, *, *, *)$, $(a_1, *, c_1, *, *, *, *, *)$ and $(a_1, *, c_1, d_1, *, *, *, *)$, must be a global iceberg cell of count 2 (yielding to the measure of the iceberg cell in the fact table). As another example, iceberg cell $(a_1, *, c_1, *, *, *, *, *)$, e_1 and e_2 appear in attribute E , which reference to dimension table D_2 . Thus, for any local iceberg cell c in PD_2 whose signature contains e_1 or e_2 , such as $(*, f_1, *, *)$, can be a global iceberg cell, if the overlap of the signatures can lead to an aggregate value satisfying the iceberg condition. Then, we can further join them to get iceberg cell $(a_1, *, c_1, *, *, f_1, *, *)$.

In such a recursive way, we can find all the global iceberg cells that contain some values in the attributes in fact table *Fact*. Limited by space, we omit the details here. ■

We never need to join the fact table with any dimension tables to generate a global iceberg cell. Instead, we join the local iceberg cells based on the signatures. Recall that we maintain the signatures using bitmap vectors, the matching of signatures is efficient. To facilitate matching, we also index the iceberg cells in the dimension tables by their signatures. Another advantage of the algorithm is that, a local iceberg cell is found only once but is used many times to join with other local iceberg cells to form global ones. If we compute the global iceberg cells from the universal base table, we may have to search the same portion of the universal base table for the (local) iceberg cell many times for different global iceberg cells. The cross table algorithm eliminates the redundancy in the computation.

EXAMPLE 7. (JOINING LOCAL ICEBERG CELLS) We consider how to compute the global iceberg cells in data warehouse *DW* (Figure 2) that do not contain any non- $*$ value in

¹Let c_1 and c_2 be aggregate cells on tables T_1 and T_2 , respectively, such that if T_1 and T_2 have any common attribute then c_1 and c_2 have the same value in every such common attribute. The join of c_1 and c_2 , denoted as $c_1 \bowtie c_2$, is the tuple c such that (1) for any attribute A that c_1 has a non- $*$ value, c has the same value as c_1 on A ; (2) for any attribute B that c_2 has a non- $*$ value, c has the same value as c_2 on B ; (3) c has value $*$ in all other attributes.

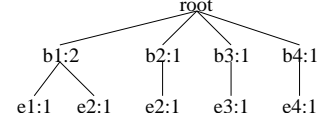


Figure 5: The H-tree for foreign key attribute values.

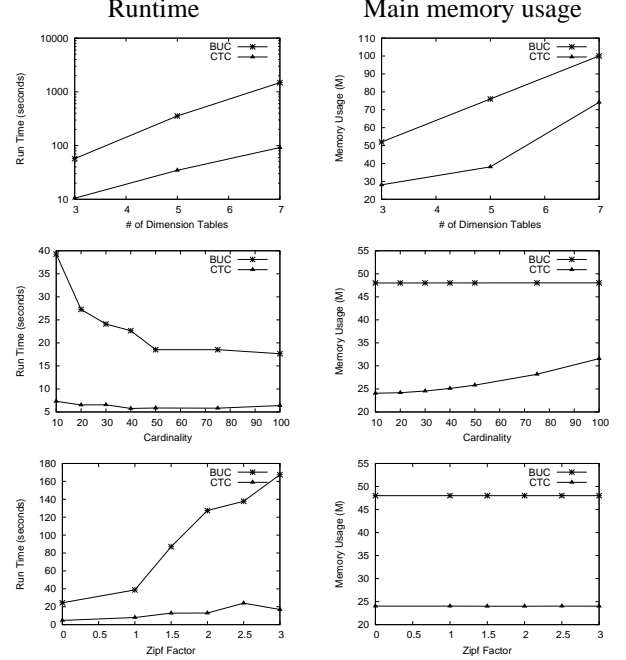


Figure 6: Experimental results – Part 1.

attributes in the fact table. Those global iceberg cells can be divided into two subsets: (1) the descendants of some local iceberg cells in PD_1 , and (2) the descendants of some local iceberg cells in PD_2 but not descendant of any local iceberg cells in PD_1 . In both cases, we only consider the cells that do not contain any non- $*$ value in the key attributes.

To find the first subset, we consider the local iceberg cells in PD_1 one by one. For example, $(*, c_1, *)$ is a local iceberg cell in PD_1 with signature $\{b_1, b_2, b_3\}$. To find the local iceberg cells in PD_2 that can be joined with $(*, c_1, *)$ to form a global iceberg cell, we should collect all the tuples in the fact table that contain either b_1 , b_2 or b_3 , and find their signature on attribute E .

Clearly, to derive the signature on attribute E for a local iceberg cell in table PD_1 by collecting the tuples in the fact table is inefficient, since we have to scan the fact table once for each local iceberg cell. To tackle the problem, we build an H-tree [4] using only the foreign key attributes in the fact table, as shown in Figure 5.

With the H-tree, for a given signature on attribute B , it is efficient to retrieve the corresponding signature on attribute E . For example, for $(*, c_1, *)$, its signature (on B) is $\{b_1, b_2, b_3\}$. From the H-tree, we can retrieve its signature on E is $\{e_1, e_2, e_3\}$, i.e., the union of the nodes at level E that are descendants of b_1 , b_2 or b_3 .

Then, we can search the iceberg cells in dimension table PD_2 . For example, iceberg cell $(*, *, g_1*)$ in dimension

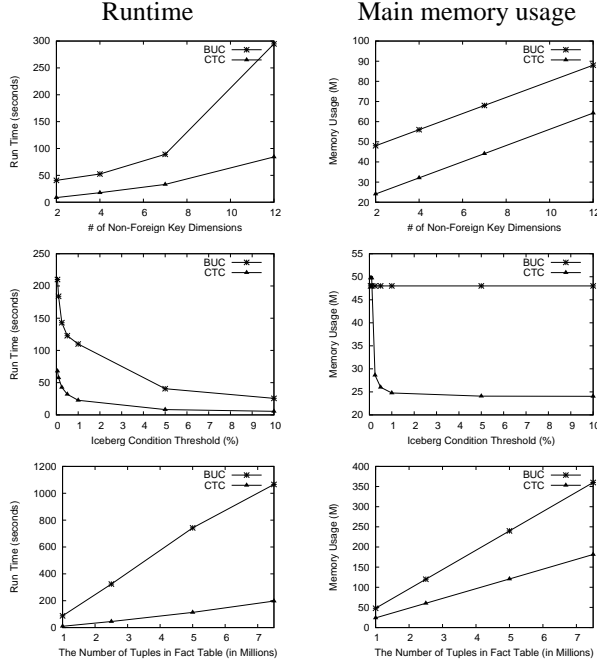


Figure 7: Experimental results – Part 2.

table PD_2 has signature $\{e_1, e_3, e_4\}$. The intersection of the two signatures is $\{e_1, e_3\}$. From the H-tree, we know that the total aggregate of tuples having e_1 or e_3 and b_1, b_2 or b_3 is 2 (the sum of the first and the fourth leaf nodes in the H-tree). Thus, the two iceberg cells can be joined and $(*, *, c_1, *, *, *, g_1, *)$ is a global iceberg cell.

Moreover, if we have more than 2 foreign key attributes, once all the global iceberg cells that are descendants of local iceberg cells in dimension table PD_1 are computed, the level of attribute B in the H-tree can be removed and the remaining sub-trees can be collapsed according to the next attribute, E . That will further reduce the tree size and search cost.

The second subset of global iceberg cells, i.e., the ones that are descendants of some local iceberg cells in PD_2 , but not of PD_1 , are exactly $(*, *, *, *) \bowtie c$, where c is a local iceberg cell in PD_2 . ■

The space complexity of the H-tree in CTC is $O(kn)$, where k is the number of dimension tables and n is the number of tuples in the fact table. In many cases, the H-tree is smaller than the fact table and much smaller than the universal base table. The signatures of local iceberg cells can be stored on disk and do not have to be maintained in main memory.

4 Experimental Results

In this section, we briefly report an extensive performance study on computing iceberg cubes from data warehouses in star schema, using synthetic data sets. All the experiments are conducted on a Dell Latitude C640 laptop computer with a 2.0 GHz Pentium 4 processor, 20 G hard drive, and 512 MB main memory, running Microsoft Windows XP operating

system. We compare two algorithms: BUC [1] and CTC. Both algorithms are implemented in C++.

We generate synthetic data sets following the Zipf distribution. By default, the fact table has 5 dimensions, 1 million tuples and the cardinality of each dimension is set to 10; we set 3 dimension tables, and each dimension table has 3 attributes; the Zipf factor is set to 1.0.

In a data warehouse generated by the above data generator, if there are n dimensions in the fact table and k dimension tables ($n \geq k$), and there are l attributes in each dimension table, then the universal base table has $(l \cdot k + (n - k))$ dimensions. Thus, by default, a data warehouse has 11 dimensions. In all our experiments, we use aggregate function `count()`. Therefore, the domain, cardinality and distribution on the measure attribute have no effect on the experimental results. By default, we set the iceberg condition to “`COUNT(*) ≥ number of tuples in fact table × 5%`”.

In all our experiments, the runtime of CTC is the elapsing time that CTC computes iceberg cube from multiple tables, including the CPU time and I/O time. However, the runtime of BUC is only the time that *BUC computes iceberg cube from the universal base table*, including the CPU time and I/O time. That is, *the time of deriving the universal table is not counted in the BUC runtime*. We believe that such a setting does not bias towards CTC.

To simplify the comparison, we assume that the universal base table can be held into main memory in our experiments. When the universal base table cannot be held into main memory, the performance of BUC will be degraded substantially. CTC does not need to store all the tables in main memory. Instead, it loads tables one by one. The local iceberg cells can be indexed and stored on disk. One major consumption of main memory in CTC is to store the H-tree for the fact table. As shown before, the H-tree is often smaller than the fact table and much smaller than the universal base table. When the H-tree is too large to fit into main memory, the disk management techniques as discussed in [4] and also the techniques for disk-based BUC can be applied.

The experimental results are shown in Figures 6 and 7, while the curves are self-explained. By the extensive performance study using synthetic data sets, we show that CTC is consistently more efficient and more scalable than BUC. The performance of BUC in our experiments is consistent in trend with the results reported in [1].

References

- [1] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *SIGMOD'99*.
- [2] M. Fang et al. Computing iceberg queries efficiently. In *VLDB'98*.
- [3] J. Gray et al. Data cube: A relational operator generalizing group-by, cross-tab and sub-totals. In *ICDE'96*.
- [4] J. Han et al. Efficient computation of iceberg cubes with complex measures. In *SIGMOD'01*.
- [5] K. Wang et al. Pushing aggregate constraints by divide-and-approximate. In *ICDE'03*.
- [6] Y. Zhao et al. An array-based algorithm for simultaneous multidimensional aggregates. In *SIGMOD'97*.

Decision Tree Induction in High Dimensional, Hierarchically Distributed Databases

Amir Bar-Or, Assaf Schuster, Ran Wolff
Faculty of Computer Science
Technion, Israel
{abaror, assaf, ranw}@cs.technion.ac.il

Daniel Keren
Department of Computer Science
Haifa University, Israel
dkeren@cs.haifa.ac.il

Abstract

Classification based on decision trees is one of the important problems in data mining and has applications in many fields. In recent years, database systems have become highly distributed, and distributed system paradigms such as federated and peer-to-peer databases are being adopted. In this paper, we consider the problem of inducing decision trees in a large distributed network of high dimensional databases. Our work is motivated by the existence of distributed databases in healthcare and in bioinformatics, and by the vision that these database are soon to contain large amounts of genomic data, characterized by its high dimensionality. Current decision tree algorithms would require high communication bandwidth when executed on such data, which is not likely to exist in large-scale distributed systems. We present an algorithm that sharply reduces the communication overhead by sending just a fraction of the statistical data. A fraction which is nevertheless sufficient to derive the exact same decision tree learned by a sequential learner on all the data in the network. Extensive experiments using standard synthetic SNP data show that the algorithm utilizes the high dependency among attributes, typical to genomic data, to reduce communication overhead by up to 99%. Scalability tests show that the algorithm scales well with both the size of the dataset, the dimensionality of the data, and the size of the distributed system.

Keywords: data mining, distributed algorithms, decision trees, classification, high dimension data.

1 Introduction

The analysis of large databases requires automation. Data mining tools have been shown to be useful for this task, in a variety of domains and architectures. It has recently been shown that data mining tools are extremely useful for the analysis of genomic data as well [12]. Since the number of genomic databases and the amount of data in them increases rapidly, there is a dire need for data mining tools designed specifically to

target genomic data specifically.

Classification, the separation of data records into distinct classes, is apparently the most common data mining task, and decision tree classifiers are perhaps the most popular classification technique. Some recent works have shown that classification can be used to analyze the effect of genomic, clinical, environmental, and demographic factors on diseases, response to treatment, and the risk of side effects [9]. Providing efficient decision tree induction algorithms suitable for genomic data is therefore an important goal.

One interesting aspect of genomic databases is that they are often distributed over many locations. The main reason for this is that they are produced by a variety of independent institutions. While these institutions often allow a second party to browse their databases, they will rarely allow this party to *copy* them. There could be a number of reasons for this: the need to retain the privacy of personal data recorded in the database, through questions regarding its ownership, or even because the sheer size of the data makes copying non-permissively costly in CPU, disk I/O or network bandwidth.

Our lead example in this paper is the task of mining genomically enriched electronic medical records (EMRs). Within a few years it is expected that each patient's medical record will contain a genomic fingerprint. This fingerprint will be used mainly to optimize treatment and predict side effects. Existing genomic fingerprinting techniques, such as single nucleotide polymorphisms (SNPs) and Gene Expression Microarrays, yield records with tens of thousands of entries that are usually interpreted as binary (normal/abnormal allele or active/inactive gen, respectively). It is common perception that an illness or treatment side effect can many times relate to just single SNPs or to the expression of few genes.

Data mining of genomically enriched EMRs would be needed for the identification of unknown correlations

and for the development of new drugs. It would best be performed on a national scale, using EMRs gathered by many different health maintenance organizations (HMOs). This would naturally extend the functionality of systems such as RODS and NRDM [13] which already collect and analyze health data at a regional (RODS) and national (NRDM) scale. RODS, for example, accesses the database of tens of hospitals using the HL7 protocol to retrieve statistical information and detect disease outbreaks. Nevertheless, it is unlikely that an HMO would allow systems such as RODS to download its entire database. Hence, the need for distributed algorithms.

A distributed decision tree induction algorithm is one that executes on several computers, each with its own database partition. The outcome of the distributed algorithm is a decision tree which is the same as, or at least comparable with, a tree that would be induced were the different partitions collected to a central place and processed using a sequential decision tree induction algorithm. Since decision tree induction poses modest CPU requirements, the performance of the algorithm would usually be dictated by its communication requirements.

Previous work on distributed decision tree induction usually focused on tight clusters of computers, or even on shared memory machines [4–6, 10, 11]. When a wide area distributed scenario was considered, all these algorithms become impractical because they use too much communication and synchronization. A kind of decision tree induction algorithm which is more efficient in a wide area system employs meta-learning. However, these produce a heuristic approximation rather than the optimal result produced by the former algorithms and, thus, are not considered in this paper. Because genomic databases contain many (thousands) attributes for each data instance and can be expected to be distributed over many distant locations, current distributed decision tree induction algorithms are ill-fit for them.

In this paper we describe a new distributed decision tree algorithm, Distributed Hierarchical Decision Tree (DHDT). DHDT is executed by a collection of agents which correlate with the natural hierarchy of a national virtual organization. For instance, the leaf level agents may correspond to different HMOs (or clinics within an HMO) while upper levels correspond to regional, state and national levels of the organization. DHDT focuses on reducing the volume of data sent from each level to the next while preserving perfect accuracy (i.e., the resulting decision tree is not an approximation). When tested on genomic SNP data with one thousand SNPs in each data record, DHDT usually collects data about only about a dozen of the SNPs – a 99% decrease in

bandwidth requirements. The algorithm is suitable for any high dimension data, provided that the correlations in it are sparse as they are in genomic data. Both the hierarchical organization and the communication efficiency of DHDT give it excellent scalability at no decrease in accuracy.

2 Sequential Decision Tree Induction

The decision tree model was first introduced by Hunt et al. [3], and the first sequential algorithm was presented by Quinlan [7]. This basic algorithm used by most of the existing decision tree algorithms is given here.

Given a training set of examples, each tagged with a class label, the goal of an induction algorithm is to build a decision tree model that can predict with high accuracy the class label of future unlabeled examples. A decision tree is composed of nodes, where each node contains a test on an attribute, each branch from a node corresponds to a possible outcome of the test, and each leaf contains a class prediction. Attributes can be either *numerical* or *categorical*. In this paper, we deal only with categorical attributes. Numerical attributes can be discretized and treated as categorical attributes; however, the discretization process is outside the scope of this paper.

A decision tree is usually built in two phases: A growth phase and a pruning phase. The tree is grown by recursively replacing the leaves by test nodes, starting at the root. The attribute to be tested at a node is chosen by comparing all the available attributes and greedily selecting the attribute that maximizes some heuristic measure, denoted as the *gain function*. The minimal and sufficient information for computing most of the gain functions is usually contained in a two-dimensional matrix called the *crosstable* of attribute i . The $[v, c]$ entry of the crosstable contains the number of examples for which the value of the attribute is v and the value of the class attribute is c .

The decision tree built in the growth phase can "overfit" the learning data. As the goal of classification is to accurately predict new cases, the pruning phase generalizes the tree by removing sub-trees corresponding to statistical noise or variation that may be particular only to the training data. This phase requires much less statistical information than the growth phase; thus it is by far less expensive. Our algorithm integrates a tree generalization technique suggested in PUBLIC [8], which combines the growing and pruning stages while providing the same accuracy as the post-pruning phase. In this paper, we focus on the costly growth phase.

2.1 Gain Functions The most popular gain functions are information gain [7], which is used by Quin-

lan's ID3 algorithm, and the Gini Index, which is used by Brieman's Cart algorithm, among others.

Consider a set of examples S that is partitioned into M disjoint subsets (classes) C_1, C_2, \dots, C_M such that $S = \bigcup_{i=1}^M C_i$ and $C_i \cap C_j = \emptyset$ for every $i \neq j$. The estimated probability that a randomly chosen instance $s \in S$ belongs to class C_j is $p_j = \frac{|C_j|}{|S|}$, where $|X|$ denotes the cardinality of the set X . With this estimated probability, two measures of impurity are defined: $entropy(S) = -\sum_j p_j \log p_j$, and $Gini(S) = \sum_j p_j^2$.

Given one of the impurity measures defined above, the gain function measures the reduction in the impurity of the set S when it is partitioned by an attribute \mathbf{A} as follows: $Gain_{\mathbf{A}}(S) = \sum_{v \in Values(\mathbf{A})} \frac{|S_v|}{|S|} Imp(S_v)$, where $Values(\mathbf{A})$ is the set of all possible values for attribute \mathbf{A} , S_v is the subset of S for which attribute \mathbf{A} has the value v , and $Imp(S)$ can be $entropy(S)$ or $Gini(S)$.

3 Bounds on the Gain Functions

The bounds given in this section bound the gain function of a population that is the union of several disjoint subpopulations on which only partial information is available. By using them we can avoid collecting the crosstables of many of the attributes whose gain, as indicated by the bounds, cannot be large enough to change the result.

3.1 Notations The bounds given below are defined for a single attribute of a single decision tree leaf node. Therefore, we simplify the notations by removing references to the attribute and the decision tree node. Let P be a population of size n and let $\{P_1, P_2\}$ be a partition of P into two subpopulations of sizes n_1, n_2 respectively. Let the crosstables of populations P_1, P_2, P be defined as:

$$\begin{aligned} \vec{P}_1(value, class) &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \\ \vec{P}_2(value, class) &= \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}, \\ \vec{P}(value, class) &= \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix} \text{ respectively.} \end{aligned}$$

Here, $a_{i,j}$ and $b_{i,j}$ denote the number of learning examples with value i and class j in \vec{P}_1 and \vec{P}_2 , respectively.

In the algorithm described here we rely on the following two bounds, the proof of which is omitted due to space considerations:

THEOREM 3.1. *Let P be a population of size n , and $\{P_1, P_2, \dots, P_k\}$ a partition of P into k subpopulations*

of sizes n_1, n_2, \dots, n_k respectively. Let $G()$ denote the gain function (information gain or Gini index). Then an upper bound on $G(P)$ is given by:

$$G(P) \leq \frac{\sum_{i=1}^k n_i G(P_i)}{\sum_{i=1}^k n_i}.$$

THEOREM 3.2. *Let P be a population of size n , and $\{P_1, P_2\}$ a partition of P into two subpopulations of sizes n_1, n_2 respectively. Assume that the candidate split divides P_1 into two subsets, P_1^{left} and P_1^{right} , with sizes n_1^{left} and n_1^{right} respectively. Let $G()$ denote the gain function (information gain or Gini index). Then, lower bounds on $G(P)$ is given by:*

$$G(P) \geq \frac{G(P_1)}{\left[1 + \frac{n_2}{n_1}\right] \left[1 + \frac{n_2}{\min\{n_1^{left}, n_1^{right}\}}\right]}$$

4 Distributed Hierarchical Decision Tree

The distributed hierarchical decision tree (DHDT) algorithm runs on a group of computers, connected through a wide-area network such as the Internet. Each computer has its own local database. The goal of DHDT is to derive exactly the same decision tree learned by a sequential decision tree learner on the collection of all data in the network. We assume a homogeneous database schema for all databases, which can be provided transparently, if required, by ordinary federated system services. The algorithm relies on a (possibly overlay) communication tree that spans all computers in the group. The communication tree can be maintained by a spanning tree algorithm or can utilize the natural hierarchy of the network. For reasons of locality, communication between nodes in the lower levels of the spanning tree is often cheaper than communication between nodes in the upper levels. Thus, a "good" algorithm will use more communication at the bottom than at the top of the tree. We further assume that during the growth phase of the decision tree, the databases and the communication tree remain static.

Every computer in the group employs an entity called *Agent* that is in charge of computing the required statistics from the local database and participates in the distributed algorithm. Agents collect statistical data from their children agents and from the local database and send it to their parent agent at its request.

The root agent is responsible for developing the decision tree and making the split decisions for the new decision tree leaves. First, the root agent decides whether a decision tree leaf has to be split according to one or more stopping conditions (e.g., if the dominance of the majority class has already reached a certain threshold) or according to the PUBLIC method [8], which avoids splitting a leaf once it knows it may be pruned eventually. The class distribution vector,

which holds the number of examples that belong to each distinct class in the population, is sufficient for computing these functions, and thus it is aggregated by the agents over the communication tree to the root agent.

Definitions

- D1. *border* = maximal lower bound of all attributes which were not sent to the parent
- D2. *borderAttribute* = the attribute whose lower bound defines the border
- D3. If agent is root then
- D4. ExtraCondition = There is only a single attribute A_i where $UpperBound(A_i) \geq border$ or $max_i(UpperBound(A_i)) = border$
- D5. Else
- D6. ExtraCondition = $G_u^i < border$ for all children

Algorithm

Phase 1: Starts when a new leaf is born

- 01. Receive information from all children
- 02. While (not (*border* defines a clear separation and ExtraCondition)) do
- 03. If ($G_u^i > border$) then
- 04. request $child_i$ to lower its border and send new information
- 05. Else if (*border* does not define a clear separation and crosstable of *borderAttribute* has only partial information)
- 06. request information for *borderAttribute* from children who did not send complete information
- 07. Else
- 08. request information for all attributes that cross the *border*
- 09. End if
- 10. Receive information from all children
- 11. End while
- 12. Return attributes A_i where $LowerBound(A_i) \geq border$

Phase 2: Starts when an agent receives a request for more information from its parent

- 01. If (parent requires more information for attribute $attr_i$) then
- 02. If (crosstable of $attr_i$ was not sent to parent) then
- 03. Send parent the crosstable of $attr_i$
- 04. Else
- 05. request information for $attr_i$ from children who sent partial information regarding $attr_i$
- 06. Else (the case where parent requests that the border be lowered)
- 07. Update *border* and *borderAttribute* and start phase 1.
- 08. Endif

Algorithm 1: DESAR Algorithm

Recall that if a decision tree leaf has to be split, the split must be done by the attribute with the highest gain in the combined database of the entire network. All that is required to decide on the splitting attribute is thus an

agreement as to which attribute has the maximal gain; the actual gain of each attribute does not need to be computed. To reach agreement, the agents participate in a distributed algorithm called DESAR (Distributed Efficient Splitting Attribute Resolver). For each new leaf that has to be developed, DHDt starts a new instance of DESAR to find the best splitting attribute. We proceed to describe the DESAR algorithm.

4.1 Distributed Efficient Splitting Attribute Resolver

To find the best splitting attribute while minimizing communication complexity, DESAR aggregates only a subset of the attribute crosstables over the communication tree to the root agent. The algorithm starts when the agents receive a message that is broadcast down the communication tree (initiated by the root and transmitted by each agent to all its children), asking for the development of a new leaf in the decision tree. Then, each agent waits for messages from its children. When messages are received from all children, the agent combines the received crosstables with its own local crosstables, picks the most *promising* attributes on the basis of its aggregated data, and sends the corresponding crosstables to its parent agent.

Algorithm 1 describes DESAR pseudocode, uniformly executed by all agents. For space considerations, we only provide pseudo-code here.

5 Experimental Evaluation

The DHDt algorithm is designed to run on datasets with a large number of attributes, such as the genomically enriched EMR. However, such data is not yet available for large-scale data mining. Therefore, we adopted an approach common in bioinformatics studies on the association of phenotype with SNP data. In this approach, synthetic SNP data is generated by a theoretical model, and then one SNP serves as the phenotype we wish to classify. Since some diseases are correlated strongly with a single SNP variation, learning a model which predicts an SNP's allele is equivalent to learning a model which predicts one of these diseases. We synthesized the SNP data using two data generators ([1,2]) with typical parameters to generate two datasets, where each of the generators uses a different theoretical model.

Each dataset contains 250,000 examples describing a single population. A single SNP is described by a binary attribute where '0' denotes the most common allele. An example is composed of 1000 SNPs. An arbitrary SNP is designated the class attribute. The experiments were performed on a simulation of a communication tree that spans all agents in the system. At the beginning of each experiment, each agent builds its local database by sampling a small fraction of the simu-

lated dataset, thus emulating a specific subpopulation.

Unless otherwise stated, experiments have been run assuming a spanning communication tree of degree three and height six, totaling in 1093 Agents. Each agent has a database (population) of 5000 samples and 1000 attributes (alleles) per sample. The average resulting decision tree had 25 nodes and a misclassification rate of 3%.

5.1 Experiments Our first experiment measures the average communication overhead of a single split decision (i.e., a single run of the DESAR algorithm) in terms of the number of messages and the number of sent crosstables. These results are compared with previous distributed decision tree algorithms which collect and aggregate the crosstables of all attributes.

Our algorithm demonstrates an average reduction of more than 99% in the number of transmitted bytes, with only a small increase in the average number of sent messages (1.2 per Agent per decision tree node). These results are summarized in Fig. 1.

Additional experiments have proved the algorithm is scalable with respect to the size of the network, the number of total attributes, and the size of the local databases. For space considerations, we only present results for network size scalability (Fig. 2).

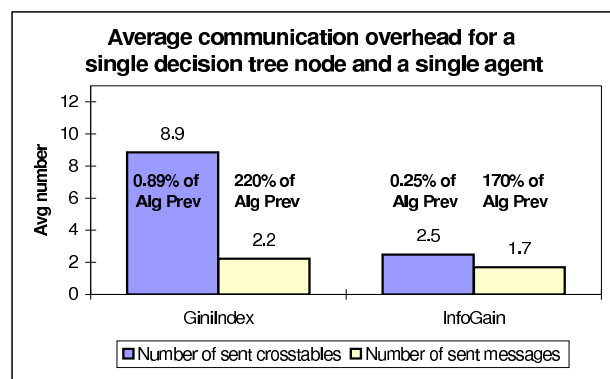


Figure 1: Average communication overhead, comparing to [11] which sends 1000 attributes in a single message

References

- [1] G. Greenspan and D. Geiger. Model-based inference of haplotype block variation. *RECOMB*, pages 131–137, 2003.
- [2] R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18:337–338, 2002.
- [3] E. B. Hunt, J. Marin, and P. T. Stone. *Experiments in Induction*. Academic Press, 1966.

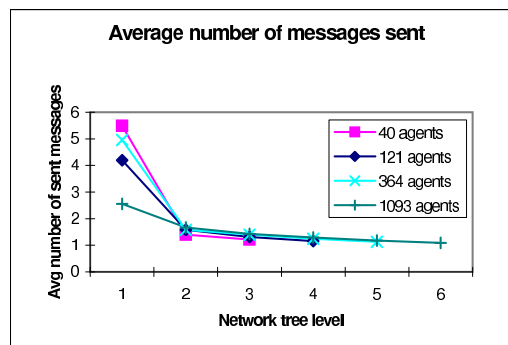


Figure 2: Scalability in network size. The above figure show the distribution of the average communication overhead over the network tree levels for different network sizes (Gini index).

- [4] R. Jin and G. Agrawal. Communication and memory efficient parallel decision tree construction. In *Proc. of the 3rd (SDM)*, 2003.
- [5] M. V. Joshi, G. Karypis, and V. Kumar. A new scalable and efficient parallel classification algorithm for mining large datasets. In *Proc. of International Parallel Processing Symposium*, 1998.
- [6] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proc. of the Fifth Int'l Conference on Extending Database Technology, Avignon, France*, 1996.
- [7] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [8] Rajeev Rastogi and Kyuseok Shim. PUBLIC: A decision tree classifier that integrates building and pruning. *Data Mining and Knowledge Discovery*, 4(4):315–344, 2000.
- [9] N. J. Risch. Searching for genetic determinants in the new millennium. In *Nature* 405, pages 847–856, 2000.
- [10] J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proc. of the 22nd VLDB Conf.*, 1996.
- [11] A. Srivastava, E.-H. (S.) Han, V. Kumar, and V. Singh. Parallel formulations of decision-tree classification algorithms. *Data Mining and Knowledge Discovery: An International Journal*, 3:237–261, 1999.
- [12] W. Stihlinger, O. Hogl, H. Stoyan, and M. Muller. Intelligent data mining for medical quality management. In *the Fifth Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2000), Workshop Notes of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, pp. 55–67, 2000.
- [13] M. M. Wagner, J. M. Robinson, F.-C. Tsui, J. U. Espino, and W. R. Hogan. Design of a national retail data monitor for public health surveillance. *Journal of the American Medical Informatics Association JAMIA*, 10(5):409–418, Sep/Oct 2003.

Slope One Predictors for Online Rating-Based Collaborative Filtering

Daniel Lemire*

Anna Maclachlan†

January 9, 2005

Abstract

Rating-based collaborative filtering is the process of predicting how a user would rate a given item from other user ratings. We propose three related slope one schemes with predictors of the form $f(x) = x + b$, which precompute the average difference between the ratings of one item and another for users who rated both. Slope one algorithms are easy to implement, efficient to query, reasonably accurate, and they support both online queries and dynamic updates, which makes them good candidates for real-world systems. The basic SLOPE ONE scheme is suggested as a new reference scheme for collaborative filtering. By factoring in items that a user liked separately from items that a user disliked, we achieve results competitive with slower memory-based schemes over the standard benchmark EachMovie and Movielens data sets while better fulfilling the desiderata of CF applications.

Keywords: Collaborative Filtering, Recommender, e-Commerce, Data Mining, Knowledge Discovery

1 Introduction

An online rating-based Collaborative Filtering CF query consists of an array of (item, rating) pairs from a single user. The response to that query is an array of predicted (item, rating) pairs for those items the user has not yet rated. We aim to provide robust CF schemes that are:

1. easy to implement and maintain: all aggregated data should be easily interpreted by the average engineer and algorithms should be easy to implement and test;
2. updateable on the fly: the addition of a new rating should change all predictions instantaneously;
3. efficient at query time: queries should be fast, possibly at the expense of storage;
4. expect little from first visitors: a user with few ratings should receive valid recommendations;

5. accurate within reason: the schemes should be competitive with the most accurate schemes, but a minor gain in accuracy is not always worth a major sacrifice in simplicity or scalability.

Our goal in this paper is not to compare the accuracy of a wide range of CF algorithms but rather to demonstrate that the Slope One schemes simultaneously fulfill all five goals. In spite of the fact that our schemes are simple, updateable, computationally efficient, and scalable, they are comparable in accuracy to schemes that forego some of the other advantages.

Our Slope One algorithms work on the intuitive principle of a “popularity differential” between items for users. In a pairwise fashion, we determine how much better one item is liked than another. One way to measure this differential is simply to subtract the average rating of the two items. In turn, this difference can be used to predict another user’s rating of one of those items, given their rating of the other. Consider two users A and B , two items I and J and Fig. 1. User A gave item I a rating of 1, whereas user B gave it a rating of 2, while user A gave item J a rating of 1.5. We observe that item J is rated more than item I by $1.5 - 1 = 0.5$ points, thus we could predict that user B will give item J a rating of $2 + 0.5 = 2.5$. We call user B the predictee user and item J the predictee item. Many such differentials exist in a training set for each unknown rating and we take an average of these differentials. The family of slope one schemes presented here arise from the three ways we select the relevant differentials to arrive at a single prediction.

The main contribution of this paper is to present slope one CF predictors and demonstrate that they are competitive with memory-based schemes having almost identical accuracy, while being more amenable to the CF task.

2 Related Work

2.1 Memory-Based and Model-Based Schemes

Memory-based collaborative filtering uses a similarity measure between pairs of users to build a prediction, typically through a weighted average [2, 12, 13, 18]. The chosen similarity measure determines the accuracy of the prediction and numerous alternatives have been studied [8]. Some potential drawbacks of memory-based CF include

*Université du Québec

†Idilia Inc.

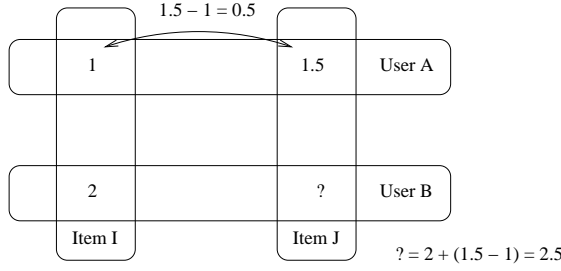


Figure 1: Basis of SLOPE ONE schemes: User A's ratings of two items and User B's rating of a common item is used to predict User B's unknown rating.

scalability and sensitivity to data sparseness. In general, schemes that rely on similarities across users cannot be precomputed for fast online queries. Another critical issue is that memory-based schemes must compute a similarity measure between users and often this requires that some minimum number of users (say, at least 100 users) have entered some minimum number of ratings (say, at least 20 ratings) including the current user. We will contrast our scheme with a well-known memory-based scheme, the Pearson scheme.

There are many model-based approaches to CF. Some are based on linear algebra (SVD, PCA, or Eigenvectors) [3, 6, 7, 10, 15, 16]; or on techniques borrowed more directly from Artificial Intelligence such as Bayes methods, Latent Classes, and Neural Networks [1, 2, 9]; or on clustering [4, 5]. In comparison to memory-based schemes, model-based CF algorithms are typically faster at query time though they might have expensive learning or updating phases. Model-based schemes can be preferable to memory-based schemes when query speed is crucial.

We can compare our predictors with certain types of predictors described in the literature in the following algebraic terms. Our predictors are of the form $f(x) = x + b$, hence the name "slope one", where b is a constant and x is a variable representing rating values. For any pair of items, we attempt to find the best function f that predicts one item's ratings from the other item's ratings. This function could be different for each pair of items. A CF scheme will weight the many predictions generated by the predictors. In [14], the authors considered the correlation across pairs of items and then derived weighted averages of the user's ratings as predictors. In the simple version of their algorithm, their predictors were of the form $f(x) = x$. In the regression-based version of their algorithm, their predictors were of the form $f(x) = ax + b$. In [17], the authors also employ predictors of the form $f(x) = ax + b$. A natural extension of the work in these two papers would be to consider predictors of the form $f(x) = ax^2 + bx + c$. Instead, in this paper, we use naïve pre-

dictors of the form $f(x) = x + b$. We also use naïve weighting. It was observed in [14] that even their regression-based $f(x) = ax + b$ algorithm didn't lead to large improvements over memory-based algorithms. It is therefore a significant result to demonstrate that a predictor of the form $f(x) = x + b$ can be competitive with memory-based schemes.

3 CF Algorithms

We propose three new CF schemes, and contrast our proposed schemes with four reference schemes: PER USER AVERAGE, BIAS FROM MEAN, ADJUSTED COSINE ITEM-BASED, which is a model-based scheme, and the PEARSON scheme, which is representative of memory-based schemes.

3.1 Notation We use the following notation in describing schemes. The ratings from a given user, called an *evaluation*, is represented as an incomplete array u , where u_i is the rating of this user gives to item i . The subset of the set of items consisting of all those items which are rated in u is $S(u)$. The set of all evaluations in the training set is χ . The number of elements in a set S is $card(S)$. The average of ratings in an evaluation u is denoted \bar{u} . The set $S_i(\chi)$ is the set of all evaluations $u \in \chi$ such that they contain item i ($i \in S(u)$). Given two evaluations u, v , we define the scalar product $\langle u, v \rangle$ as $\sum_{i \in S(u) \cap S(v)} u_i v_i$. Predictions, which we write $P(u)$, represent a vector where each component is the prediction corresponding to one item: predictions depend implicitly on the training set χ .

3.2 Baseline Schemes One of the most basic prediction algorithms is the PER USER AVERAGE scheme given by the equation $P(u) = \bar{u}$. That is, we predict that a user will rate everything according to that user's average rating. Another simple scheme is known as BIAS FROM MEAN (or sometimes NON PERSONALIZED [8]). It is given by

$$P(u)_i = \bar{u} + \frac{1}{card(S_i(\chi))} \sum_{v \in S_i(\chi)} v_i - \bar{v}.$$

That is, the prediction is based on the user's average plus the average deviation from the user mean for the item in question across all users in the training set. We also compare to the item-based approach that is reported to work best [14], which uses the following adjusted cosine similarity measure, given two items i and j :

$$sim_{i,j} = \frac{\sum_{u \in S_{i,j}(\chi)} (u_i - \bar{u})(u_j - \bar{u})}{\sum_{u \in S_{i,j}(\chi)} (u_i - \bar{u})^2 \sum_{u \in S_{i,j}(\chi)} (u_j - \bar{u})^2}$$

The prediction is obtained as a weighted sum of these measures thus:

$$P(u)_i = \frac{\sum_{j \in S(u)} |sim_{i,j}| (\alpha_{i,j} u_j + \beta_{i,j})}{\sum_{j \in S(u)} |sim_{i,j}|}$$

where the regression coefficients $\alpha_{i,j}, \beta_{i,j}$ are chosen so as to minimize $\sum_{u \in S_{i,j}(u)} (\alpha_{i,j} u_j \beta_{i,j} - u_i)^2$ with i and j fixed.

3.3 The PEARSON Reference Scheme Since we wish to demonstrate that our schemes are comparable in predictive power to memory-based schemes, we choose to implement one such scheme as representative of the class, acknowledging that there are many documented schemes of this type. Among the most popular and accurate memory-based schemes is the PEARSON scheme [13]. It takes the form of a weighted sum over all users in χ

$$P(u)_i = \bar{u} + \frac{\sum_{v \in S_i(\chi)} \gamma(u, v) (v_i - \bar{v})}{\sum_{v \in S_i(\chi)} |\gamma(u, v)|}$$

where γ is a similarity measure computed from Pearson's correlation:

$$\text{Corr}(u, w) = \frac{\langle u - \bar{u}, w - \bar{w} \rangle}{\sqrt{\sum_{i \in S(u) \cap S(w)} (u_i - \bar{u})^2 \sum_{i \in S(u) \cap S(w)} (w_i - \bar{w})^2}}.$$

Following [2, 8], we set

$$\gamma(u, w) = \text{Corr}(u, w) |\text{Corr}(u, w)|^{p-1}$$

with $p = 2.5$, where p is the Case Amplification power. Case Amplification reduces noise in the data: if the correlation is high, say $\text{Corr} = 0.9$, then it remains high ($0.9^{2.5} \cong 0.8$) after Case Amplification whereas if it is low, say $\text{Corr} = 0.1$, then it becomes negligible ($0.1^{2.5} \cong 0.003$). Pearson's correlation together with Case Amplification is shown to be a reasonably accurate memory-based scheme for CF in [2] though more accurate schemes exist.

3.4 The SLOPE ONE Scheme The slope one schemes take into account both information from other users who rated the same item (like the ADJUSTED COSINE ITEM-BASED) and from the other items rated by the same user (like the PER USER AVERAGE). However, the schemes also rely on data points that fall neither in the user array nor in the item array (e.g. user A 's rating of item I in Fig. 1), but are nevertheless important information for rating prediction. Much of the strength of the approach comes from data that is *not* factored in. Specifically, only those ratings by users who have rated some common item with the predictee user and only those ratings of items that the predictee user has also rated enter into the prediction of ratings under slope one schemes.

Formally, given two evaluation arrays v_i and w_i with $i = 1, \dots, n$, we search for the best predictor of the form $f(x) = x + b$ to predict w from v by minimizing $\sum_i (v_i + b - w_i)^2$. Deriving with respect to b and setting the derivative to zero, we get $b = \frac{\sum_i w_i - v_i}{n}$. In other words, the constant b must be chosen to be the average difference between the two arrays. This result motivates the following scheme.

Given a training set χ , and any two items j and i with ratings u_j and u_i respectively in some user evaluation u (annotated as $u \in S_{j,i}(\chi)$), we consider the average deviation of item i with respect to item j as:

$$\text{dev}_{j,i} = \sum_{u \in S_{j,i}(\chi)} \frac{u_j - u_i}{\text{card}(S_{j,i}(\chi))}.$$

Note that any user evaluation u not containing both u_j and u_i is not included in the summation. The symmetric matrix defined by $\text{dev}_{j,i}$ can be computed once and updated quickly when new data is entered.

Given that $\text{dev}_{j,i} + u_i$ is a prediction for u_j given u_i , a reasonable predictor might be the average of all such predictions

$$P(u)_j = \frac{1}{\text{card}(R_j)} \sum_{i \in R_j} (\text{dev}_{j,i} + u_i)$$

where $R_j = \{i | i \in S(u), i \neq j, \text{card}(S_{j,i}(\chi)) > 0\}$ is the set of all relevant items. There is an approximation that can simplify the calculation of this prediction. For a dense enough data set where almost all pairs of items have ratings, that is, where $\text{card}(S_{j,i}(\chi)) > 0$ for almost all i, j , most of the time $R_j = S(u)$ for $j \notin S(u)$ and $R_j = S(u) - \{j\}$ when $j \in S(u)$. Since $\bar{u} = \sum_{i \in S(u)} \frac{u_i}{\text{card}(S(u))} \simeq \sum_{i \in R_j} \frac{u_i}{\text{card}(R_j)}$ for most j , we can simplify the prediction formula for the SLOPE ONE scheme to

$$P^{S1}(u)_j = \bar{u} + \frac{1}{\text{card}(R_j)} \sum_{i \in R_j} \text{dev}_{j,i}.$$

It is interesting to note that our implementation of SLOPE ONE doesn't depend on how the user rated individual items, but only on the user's average rating and crucially on which items the user has rated.

3.5 The WEIGHTED SLOPE ONE Scheme One of the drawbacks of SLOPE ONE is that the number of ratings observed is not taken into consideration. Intuitively, to predict user A 's rating of item L given user A 's rating of items J and K , if 2000 users rated the pair of items J and L whereas only 20 users rated the pair of items K and L , then user A 's rating of item J is likely to be a far better predictor for item L than user A 's rating of item K is. Thus, we define the WEIGHTED SLOPE ONE prediction as the following weighted average

$$P^{wS1}(u)_j = \frac{\sum_{i \in S(u) - \{j\}} (\text{dev}_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u) - \{j\}} c_{j,i}}$$

where $c_{j,i} = \text{card}(S_{j,i}(\chi))$.

3.6 The BI-POLAR SLOPE ONE Scheme While weighting served to favor frequently occurring rating patterns over infrequent rating patterns, we will now consider

favoring another kind of especially relevant rating pattern. We accomplish this by splitting the prediction into two parts. Using the WEIGHTED SLOPE ONE algorithm, we derive one prediction from items users liked and another prediction using items that users disliked.

Given a rating scale, say from 0 to 10, it might seem reasonable to use the middle of the scale, 5, as the threshold and to say that items rated above 5 are liked and those rated below 5 are not. This would work well if a user's ratings are distributed evenly. However, more than 70% of all ratings in the EachMovie data are above the middle of the scale. Because we want to support all types of users including balanced, optimistic, pessimistic, and bimodal users, we apply the user's average as a threshold between the users liked and disliked items. For example, optimistic users, who like every item they rate, are assumed to dislike the items rated below their average rating. This threshold ensures that our algorithm has a reasonable number of liked and disliked items for each user.

Referring again to Fig. 1, as usual we base our prediction for item J by user B on deviation from item I of users (like user A) who rated both items I and J . The BI-POLAR SLOPE ONE scheme restricts further than this the set of ratings that are predictive. First in terms of items, only deviations between two liked items or deviations between two disliked items are taken into account. Second in terms of users, only deviations from pairs of users who rated both item I and J and who share a like or dislike of item I are used to predict ratings for item J .

The splitting of each user into user likes and user dislikes effectively doubles the number of users. Observe, however, that the bi-polar restrictions just outlined necessarily reduce the overall number of ratings in the calculation of the predictions. Although any improvement in accuracy in light of such reduction may seem counter-intuitive where data sparseness is a problem, failing to filter out ratings that are irrelevant may prove even more problematic. Crucially, the BI-POLAR SLOPE ONE scheme predicts nothing from the fact that user A likes item K and user B dislikes this same item K .

Formally, we split each evaluation in u into two sets of rated items: $S^{like}(u) = \{i \in S(u) | u_i > \bar{u}\}$ and $S^{dislike}(u) = \{i \in S(u) | u_i < \bar{u}\}$. And for each pair of items i, j , split the set of all evaluations χ into $S_{i,j}^{like} = \{u \in \chi | i, j \in S^{like}(u)\}$ and $S_{i,j}^{dislike} = \{u \in \chi | i, j \in S^{dislike}(u)\}$. Using these two sets, we compute the following deviation matrix for liked items as well as the derivation matrix $dev_{j,i}^{dislike}$.

$$dev_{j,i}^{like} = \sum_{u \in S_{j,i}^{like}(\chi)} \frac{u_j - u_i}{card(S_{j,i}^{like}(\chi))},$$

The prediction for rating of item j based on rating of item i is either $p_{j,i}^{like} = dev_{j,i}^{like} + u_i$ or $p_{j,i}^{dislike} = dev_{j,i}^{dislike} + u_i$ depending

on whether i belongs to $S^{like}(u)$ or $S^{dislike}(u)$ respectively. The BI-POLAR SLOPE ONE scheme is given by

$$p^{bpS1}(u)_j = \frac{\sum_{i \in S^{like}(u) - \{j\}} p_{j,i}^{like} c_{j,i}^{like} + \sum_{i \in S^{dislike}(u) - \{j\}} p_{j,i}^{dislike} c_{j,i}^{dislike}}{\sum_{i \in S^{like}(u) - \{j\}} c_{j,i}^{like} + \sum_{i \in S^{dislike}(u) - \{j\}} c_{j,i}^{dislike}}$$

where the weights $c_{j,i}^{like} = card(S_{j,i}^{like})$ and $c_{j,i}^{dislike} = card(S_{j,i}^{dislike})$ are similar to the ones in the WEIGHTED SLOPE ONE scheme.

4 Experimental Results

The effectiveness of a given CF algorithm can be measured precisely. To do so, we have used the All But One Mean Average Error (MAE) [2]. In computing MAE, we successively hide ratings one at a time from all evaluations in the test set while predicting the hidden rating, computing the average error we make in the prediction. Given a predictor P and an evaluation u from a user, the error rate of P over a set of evaluations χ' , is given by

$$MAE = \frac{1}{card(\chi')} \sum_{u \in \chi'} \frac{1}{card(S(u))} \sum_{i \in S(u)} |P(u^{(i)}) - u_i|$$

where $u^{(i)}$ is user evaluation u with that user's rating of the i th item, u_i , hidden.

We test our schemes over the EachMovie data set made available by Compaq Research and over the Movielens data set from the Grouplens Research Group at the University of Minnesota. The data is collected from movie rating web sites where ratings range from 0.0 to 1.0 in increments of 0.2 for EachMovie and from 1 to 5 in increments of 1 for Movielens. Following [8, 11], we used enough evaluations to have a total of 50,000 ratings as a training set (χ) and an additional set of evaluations with a total of at least 100,000 ratings as the test set (χ'). When predictions fall outside the range of allowed ratings for the given data set, they are corrected accordingly: a prediction of 1.2 on a scale from 0 to 1 for EachMovie is interpreted as a prediction of 1. Since Movielens has a rating scale 4 times larger than EachMovie, MAEs from Movielens were divided by 4 to make the results directly comparable.

The results for the various schemes using the same error measure and over the same data set are summarized in Table 1. Various subresults are highlighted in the Figures that follow.

Consider the results of testing various baseline schemes. As expected, we found that BIAS FROM MEAN performed the best of the three reference baseline schemes described in section 3.2. Interestingly, however, the basic SLOPE ONE scheme described in section 3.4 had a higher accuracy than BIAS FROM MEAN.

The augmentations to the basic SLOPE ONE described in sections 3.5 and 3.6 do improve accuracy over EachMovie. There is a small difference between SLOPE ONE and

Scheme	EachMovie	Movielens
BI-POLAR SLOPE ONE	0.194	0.188
WEIGHTED SLOPE ONE	0.198	0.188
SLOPE ONE	0.200	0.188
BIAS FROM MEAN	0.203	0.191
ADJUSTED COSINE ITEM-BASED	0.209	0.198
PER USER AVERAGE	0.231	0.208
PEARSON	0.194	0.190

Table 1: All Schemes Compared: All But One Mean Average Error Rates for the EachMovie and Movielens data sets, lower is better.

WEIGHTED SLOPE ONE (about 1%). Splitting dislike and like ratings improves the results 1.5–2%.

Finally, compare the memory-based PEARSON scheme on the one hand and the three slope one schemes on the other. The slope one schemes achieve an accuracy comparable to that of the PEARSON scheme. This result is sufficient to support our claim that slope one schemes are reasonably accurate despite their simplicity and their other desirable characteristics.

5 Conclusion

This paper shows that an easy to implement CF model based on average rating differential can compete against more expensive memory-based schemes. In contrast to currently used schemes, we are able to meet 5 adversarial goals with our approach. Slope One schemes are easy to implement, dynamically updateable, efficient at query time, and expect little from first visitors while having a comparable accuracy (e.g. 1.90 vs. 1.88 MAE for MovieLens) to other commonly reported schemes. This is remarkable given the relative complexity of the memory-based scheme under comparison. A further innovation of our approach are that splitting ratings into dislike and like subsets can be an effective technique for improving accuracy. It is hoped that the generic slope one predictors presented here will prove useful to the CF community as a reference scheme.

Note that as of November 2004, the WEIGHTED SLOPE ONE is the collaborative filtering algorithm used by the Bell/MSN Web site inDiscover.net.

References

- [1] D. Billsus and M. Pazzani. Learning collaborative information filterings. In *AAAI Workshop on Recommender Systems*, 1998.
- [2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Fourteenth Conference on Uncertainty in AI*. Morgan Kaufmann, July 1998.
- [3] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR 2002*, 2002.
- [4] S. H. S. Chee. Rectree: A linear collaborative filtering algorithm. Master’s thesis, Simon Fraser University, November 2000.
- [5] S. H. S.g Chee, J. H., and K. Wang. Rectree: An efficient collaborative filtering method. *Lecture Notes in Computer Science*, 2114, 2001.
- [6] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *Proc. of the thirty-fourth annual ACM symposium on Theory of computing*, pages 82–90. ACM Press, 2002.
- [7] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
- [8] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of Research and Development in Information Retrieval*, 1999.
- [9] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *International Joint Conference in Artificial Intelligence*, 1999.
- [10] K. Honda, N. Sugiura, H. Ichihashi, and S. Araki. Collaborative filtering using principal component analysis and fuzzy clustering. In *Web Intelligence*, number 2198 in *Lecture Notes in Artificial Intelligence*, pages 394–402. Springer, 2001.
- [11] Daniel Lemire. Scale and translation invariant collaborative filtering systems. *Information Retrieval*, 8(1):129–150, January 2005.
- [12] D. M. Pennock and E. Horvitz. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *IJCAI-99*, 1999.
- [13] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. ACM Computer Supported Cooperative Work*, pages 175–186, 1994.
- [14] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommender algorithms. In *WWW10*, 2001.
- [15] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *WEBKDD ’00*, pages 82–90, 2000.
- [16] B.M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental svd-based algorithms for highly scaleable recommender systems. In *ICCIT’02*, 2002.
- [17] S. Vucetic and Z. Obradovic. A regression-based approach for scaling-up personalized recommender systems in e-commerce. In *WEBKDD ’00*, 2000.
- [18] S.M. Weiss and N. Indurkha. Lightweight collaborative filtering method for binary encoded data. In *PKDD ’01*, 2001.

Sparse Fisher Discriminant Analysis for Computer Aided Detection

M. Murat Dundar*
Glenn Fung*
Jinbo Bi*
Sandilya Sathyakama*
Bharat Rao*

Abstract

We describe a method for sparse feature selection for a class of problems motivated by our work in Computer-Aided Detection (CAD) systems for identifying structures of interest in medical images. We propose a sparse formulation for Fisher Linear Discriminant (FLD) that scales well to large datasets; our method inherits all the desirable properties of FLD, while improving on handling large numbers of irrelevant and redundant features. We demonstrate that our sparse FLD formulation outperforms conventional FLD and two other methods for feature selection from the literature on both an artificial dataset and a real-world Colon CAD dataset.

Keywords: fisher linear discriminant, sparse formulation, feature selection

1 Problem Specification.

Over the last decade, Computer-Aided Detection (CAD) systems have moved from the sole realm of academic publications, to robust commercial systems that are used by physicians in their clinical practice to help detect early cancer from medical images. The growth has been fueled by the Food and Drug Administration's (FDA) decision to grant approval in 1998 for a CAD system that detected breast cancer lesions from mammograms (scanned x-ray images) [1]. Since then a number of CAD systems have received FDA approval. Virtually all these commercial CAD systems, focus on detection (or more recently diagnosis [2]) of breast cancer lesions for mammography.

Typically, CAD systems are used as "second readers" – the physician views the image to identify potential cancers (the "first read"), and then reviews the CAD marks to determine if any additional cancers can be found. In order to receive clinical acceptance and to actually be used in the daily practice of a physician,

it is immediately obvious that CAD systems must be efficient (for instance, completing the detections in the minutes taken by the physician during the "first read") and have very high sensitivity (the whole point of CAD is to boost the physician's sensitivity, which is already fairly high – 80%-90% for colon cancer – to the high 90's).

Physicians detect cancers by visually extracting shape and texture based features, that are often qualitative rather than quantitative from the images (hereafter, "image" and "volume" are used interchangeably in this document). However, there are usually no definitive image-processing algorithms that exactly correspond to the precise, often subtle, features used intuitively by physicians. To achieve high sensitivity and specificity, CAD researchers must necessarily consider a very large number of experimental image processing features. Therefore, a typical training dataset for a CAD classifier is extremely unbalanced (significantly less than 1% of the candidates are positive), contains a very large number of candidates (several thousand), each described by many features (100+), most of which are redundant and irrelevant.

A large number of features provides more control over the discriminant function. However, even with our "large" training sample, the high-dimensional feature space is mostly empty [3]. This allows us to find many classifiers that perform well on the training data, but it is well-known that few of these will generalize well. This is particularly true of nonlinear classifiers that represent more complex discriminant functions. Furthermore, many computationally expensive nonlinear classification algorithms (e.g. nonlinear SVM, neural networks, kernel-based algorithms) do not scale well to large datasets. When the potential pitfalls of designing a classifier and the characteristics of the data are considered, it appears safer to train a CAD system with a linear classifier. This is empirically demonstrated in our previous study [4] where we compare the generalization capability of some linear and nonlinear classification al-

*Computer Aided Diagnosis and Therapy, Siemens Medical Solutions Inc, USA

gorithms on a CAD dataset.

Fisher Linear Discriminant (FLD) [5] is a well-known classification method that projects high-dimensional data onto a line and performs classification in this one dimensional space. This projection is obtained by maximizing the ratio of between and within class scatter matrices – the so called *Rayleigh quotient*. As a linear classifier it is rather robust against feature redundancy and noise and has an order of complexity $O(ld^2)$ (l is the number of training samples in the dataset and d is the number of features in the feature set).

In this study we propose a sparse formulation of FLD where we seek to eliminate the irrelevant and redundant features from the original dataset within a *wrapper* framework [6]. To achieve sparseness, earlier studies focused on direct optimization of an objective function consisting of two terms: the goodness of fit and the regularization term. In order to avoid overfitting by excessively maximizing the goodness of fit, a regularization term commonly expressed as ℓ_0 – *norm* [7], [8] or ℓ_1 – *norm* [9], [10] of the discriminant vector is added to the objective function. Optimization of this objective function generates sparse solutions, i.e. a solution that depends only on a subset of the features.

Our approach achieve sparseness by introducing regularity constraints into the problem of finding FLD. Since we maintain the original formulation of FLD as we introduce the regularization constraints, the proposed technique can scale to very large datasets (on the order of hundred thousand samples). Casting this problem as a biconvex programming problem provides us a more direct way of controlling the size of the feature subset selected. This problem is iteratively solved and once the algorithm stops the nonzero elements of the solution indicates features that are relevant to classification task at hand, and their value quantifies the degree of this relevancy. The proposed algorithm inherits all desirable characteristics of FLD while improving on handling large number of redundant and irrelevant features. This makes the algorithm numerically more stable and improve its prediction performance.

The rest of this paper is organized as follows. In the next section, we discuss the need for a linear classifier and briefly review the Fisher Linear Discriminant (FLD). We also introduce our notion of sparse FLD, where we seek to eliminate the redundant and irrelevant features from the original training set using a wrapper approach. In Section 3 we review the concept and formulation of FLD. In Section 4 we modify the conventional FLD problem so as to achieve sparseness and propose an iterative feature selection algorithm based on our the sparse formulation. Finally we present experi-

mental results on an artificial dataset and a ColonCAD dataset, and compare our approach with conventional FLD and also with two well-known methods from the literature for feature selection.

2 Fisher's Linear Discriminant

Let $X_i \in R^{d \times l}$ be a matrix containing the l training data points on d -dimensional space and l_i the number of labeled samples for class w_i , $i \in \{\pm\}$. FLD is the projection α , which maximizes,

$$(2.1) \quad J(\alpha) = \frac{\alpha^T S_B \alpha}{\alpha^T S_W \alpha}$$

where

$$S_B = (m_+ - m_-)(m_+ - m_-)^T$$

$$S_W = \sum_{i \in \{\pm\}} \frac{1}{l_i} (X_i - m_i e_{l_i}^T) (X_i - m_i e_{l_i}^T)^T$$

are the between and within class scatter matrices respectively and

$$m_i = \frac{1}{l_i} X_i e_{l_i}$$

is the mean of class w_i and e_{l_i} is an l_i dimensional vector of ones.

Transforming the above problem into a convex quadratic programming problem provides us some algorithmic advantages. First notice that if α is a solution to (2.1), then so is any scalar multiple of it. Therefore to avoid multiplicity of solutions, we impose the constraint $\alpha^T S_B \alpha = b^2$, which is equivalent to $\alpha^T (m_+ - m_-) = b$ where b is some arbitrary positive scalar. Then the optimization problem of (2.1) becomes,

$$\begin{aligned} \text{Problem 1 : } \quad & \min_{\alpha \in R^d} \quad \alpha^T S_W \alpha \\ & \text{s.t.} \quad \alpha^T (m_+ - m_-) = b \end{aligned}$$

For binary classification problems the solution of this problem is $\alpha^* = \frac{b S_W^{-1} (m_+ - m_-)}{(m_+ - m_-)^T S_W^{-1} (m_+ - m_-)}$. In what follows we propose a sparse formulation of FLD. The proposed approach incorporates a regularization constraint on the conventional algorithm and seeks to eliminate those features with limited impact on the objective function.

3 Sparse Fisher Discriminant Analysis

If we require α to be nonnegative, the 1-norm of α can be calculated as $\alpha^T e_l$. With the new constraints Problem 1 can be updated as follows,

$$\begin{aligned} \text{Problem 2: } \min_{\alpha \in R^d} \quad & \alpha^T S_W \alpha \\ \text{s.t.} \quad & \alpha^T (m_+ - m_-) = b \\ & \alpha^T e_l \leq \gamma, \alpha \geq 0 \end{aligned}$$

We denote the feasible set associated with Problem 1 by $\Omega_1 = \{\alpha \in R_d, \alpha^T (m_+ - m_-) = b\}$ and that associated with Problem 2 by $\Omega_2 = \{\alpha \in R_d, \alpha^T (m_+ - m_-) = b, \alpha^T e_l \leq \gamma, \alpha \geq 0\}$ and observe that $\Omega_2 \subset \Omega_1$. Then we define $\delta_{max} = \max_i \frac{b}{(m_+ - m_-)_i}$ and $\delta_{min} = \min_i \frac{b}{(m_+ - m_-)_i}$ where $i = \{1, \dots, d\}$. The set Ω_2 is empty whenever $\delta_{max} < 0$ or $\delta_{min} > \gamma$. In addition to the feasibility constraints $\gamma < \delta_{max}$ should hold in order to achieve a sparse solution. In what follows we introduce a linear transformation which will ensure $\delta_{max} > 0$ and standardize the sparsity constraint.

We define a linear transformation such that $x \mapsto Dx$. With this transformation Problem 2 takes the following form,

$$\begin{aligned} \text{Problem 3: } \min_{\alpha \in R^d} \quad & \alpha^T D S_W D \alpha \\ \text{s.t.} \quad & \alpha^T D (m_+ - m_-) = b \\ & \alpha^T e_l \leq \gamma, \alpha \geq 0 \end{aligned}$$

Note that both $\bar{\delta}_{min}$ and $\bar{\delta}_{max}$ are nonnegative and hence both feasibility constraints are satisfied when $\gamma > \bar{\delta}_{min}$. For $\gamma > d$ the globally optimum solution α^* to Problem 3 is $\alpha^* = [1, \dots, 1]^T$, i.e. nonsparse solution. For $\gamma < d$ sparse solutions can be obtained. Unlike Problem 2 where the upper bound on γ depends on mean vectors here the upper bound is d , i.e. the number of features.

The above sparse formulation is indeed a biconvex programming problem.

$$\begin{aligned} \text{Problem 4: } \min_{\alpha, a \in R^d} \quad & \alpha^T (S_W * (aa^T)) \alpha \\ \text{s.t.} \quad & \alpha^T ((m_+ - m_-) * a^T) = b \\ & \alpha^T e_l \leq \gamma, \alpha \geq 0 \end{aligned}$$

where $*$ is an element-wise product. We first initialize $\alpha = [1, \dots, 1]^T$ and solve for a^* , i.e. the solution to Problem 1, then we fix a^* and solve for α^* , i.e. the solution to Problem 3.

4 The Iterative Feature Selection Algorithm

Successive feature elimination can be obtained by iteratively solving the above biconvex programming problem.

- (0) Set $\alpha^0 = e_n, d^0 = d, \gamma \ll d$

For each iteration i do the following:

- (i) Select the d^i features with α_j^i values greater than $\epsilon, d^i \leq d^{i-1}$.
- (ii) Calculate the class scatter matrices and means in the $d^i - \text{dimensional}$ feature space.
- (iii) Solve Problem 4 to obtain a^i .
- (iv) Fix a to a^i and update the class scatter matrices and means.
- (v) Solve Problem 4 to obtain α^i .

Stop when all α_j^i , for $j = 1, 2, \dots, d^i$ are greater than $\epsilon = 1e - 16$.

Since at each iteration we truncate α the above algorithm is not guaranteed to converge. However at any iteration i when $d^i \leq \gamma$ no sparseness would be achieved and hence all α_j^i would be equal to one. Therefore the algorithm is guaranteed to stop at the latest when $d^i \leq \gamma$.

5 Experimental Results and Discussion

5.1 A Toy Example This experiment is adapted from [11]. The probability of $y = 1$ or $y = -1$ is equal. The first three features x_1, x_2, x_3 are drawn as $x_i = yN(i, 5)$. Note that only one of these features is relevant for discriminating one class from the other, the other two are redundant. The rest of the features are drawn as $x_i = N(0, 20)$. Note that these features are noise. The noise features are added to the feature set one by one allowing us to observe the gradual change in the prediction capability of both approaches.

We initialize $d = 3$, i.e. start with the first three features and proceed as follows. We generate 200 samples for training and 1000 samples for testing. Then we train and test both approaches and record the corresponding prediction errors. Next we increase d by one and repeat the above procedure until we reach $d = 20$. For the proposed approach we select the best two features. The error bars in Figure 1 are obtained by repeating the above process 100 times for each d each time using a different training and testing set.

Looking at the results, as d gets larger and noise features are added to the feature set the performance of the conventional FLD deteriorates significantly whereas the average prediction error for the proposed formulation remains around its initial level with some increase in the standard deviation. Also 90% of the time the proposed formulation selects feature two and three together. These are the two most powerful features in the set.

5.2 Example 2: Colon Cancer

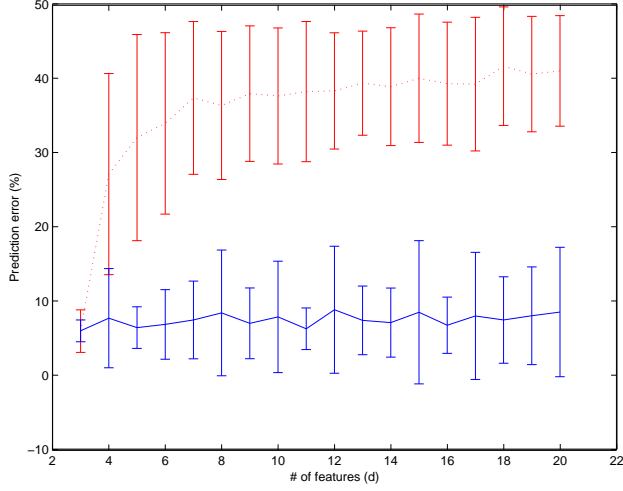


Figure 1: Testing Error vs l for the Artificial Data. Full dimensionality and two-dimensional feature subset compared. The dotted curve corresponds to Conventional FLD, the solid curve corresponds to proposed sparse approach

5.2.1 Data Sources and Domain Description

The database of high-resolution CT images used in this study were obtained from NYU Medical Center, Cleveland Clinic Foundation, and two EU sites in Vienna and Belgium. The 163 patients were randomly partitioned into two groups: training ($n=96$) and test ($n=67$). The test group was sequestered and only used to evaluate the performance of the final system.

Training Data Patient and Polyp Info: There were 96 patients with 187 volumes. A total of 76 polyps were identified in this set with a total number of 9830 candidates.

Testing Data Patient and Polyp Info: There were 67 patients with 133 volumes. A total of 53 polyps were identified in this set with a total number of 6616 candidates. A combined total of 207 features are extracted for each candidate by three imaging scientists.

5.2.2 Feature Selection and Classification:

In this experiment we consider three feature selection algorithms in a wrapper framework and compare their prediction performance on the Colon Dataset. These techniques are namely, the sparse formulation proposed in this study (SFLD), the sparse formulation introduced in [9] for Kernel Fisher Discriminant with linear loss and linear regularizer (SKFD) and a greedy sequential forward-backward feature selection algorithm [12] implemented with FLD (GFLD).

5.3 Results and Discussion: Even though we choose the computationally least expensive model for SKFD this approach failed to run with the original training set. Thus we were forced to run SKFD on a smaller subset of the training dataset where we included all the positive candidates and a random subset of size 1000 of the negative candidates. The 5 algorithms we ran were

1. SFLD on the original training set.
2. GFLD on the original training set.
3. Conventional on the original training set.
4. SKFD on the subset training set.
5. SFLK on the subset training set (denoted as SFLD-sub).

The ROC curves in Figure 2 demonstrates the LOPO performance of the each algorithm and those in Figure 3 show the performance on the test data set. Table 1 shows the number of features selected (d), the area of the ROC curve scaled by 100 (Area) and the sensitivity corresponding to 90% specificity (Sens) for all algorithms considered in this study.

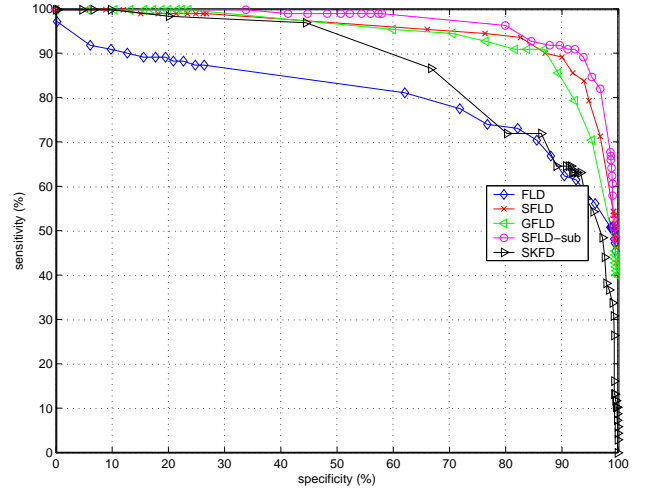


Figure 2: ROC curves for Training Results (LOPO results)

These results show that Sparse (SFLD) and SFLD-sub clearly outperform the greedy and conventional FLD and SKFD both on the training and testing datasets. Although SFLD-sub performs better than SFLD on the training data, SFLD generalizes slightly better on the testing data. This is not surprising because SFLD-sub uses a subset of the original training

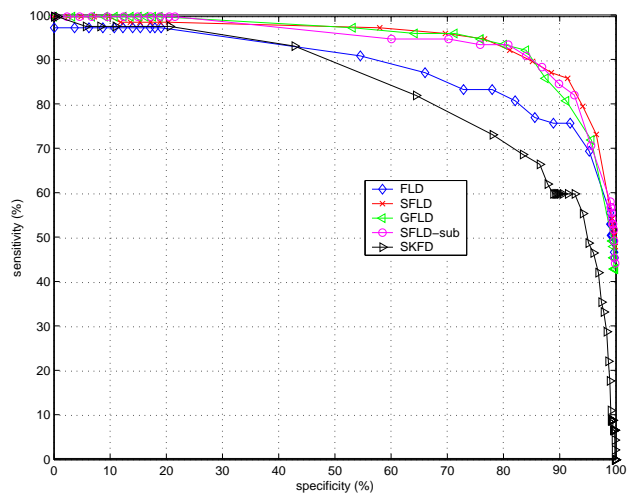


Figure 3: ROC curves for Testing Results

Table 1: The number of features selected (d), the area of the ROC curve scaled by 100 (Area) and the sensitivity corresponding to 90% specificity (Sens) is shown for all algorithms considered in this study. The values in parenthesis show the corresponding values for the testing results.

Algorithm	d	Area	Sens (%)
SFLD	25	94.8 (94.9)	89 (87)
SFLD-sub	17	94.7 (94.1)	92 (85)
GFLD	17	94.3 (94.7)	85 (83)
SKFD	18	88.0 (82.0)	65 (60)
FLD	207	80.3 (89.1)	63 (77)

data. GFLD performs almost equally well with SFLD-sub and SFLD algorithms but the difference is hidden in the computational cost required to select the features in GFLD. The computational cost of GFLD is proportional to d^3 whereas that of SFLD is proportional to d^2 .

6 Conclusions

In this study we proposed a sparse formulation of famous Fisher Linear Discriminant and applied this technique to a Colon dataset. Experimental results favor the proposed algorithm over two other feature selection/regularization techniques implemented in the FLD framework both in terms of prediction accuracy and the computational cost for large data sets. Future study will focus on obtaining sparse solutions in an iterative scheme without truncating the discriminant vector which will in turn guarantee convergence.

References

- [1] J. Roehrig, *The Promise of CAD in Digital Mammography*, European Journal of Radiology, 31 (1999), pp. 35-39.
- [2] S. Buchbinder, I. Leichter, R. Lederman, B. Novak, P. Bamberger, M. Sklair-Levy, G. Yarmish, and S. Fields *Computer-aided Classification of BI-RADS Category 3 Breast Lesions*, Radiology, 230 (2004), pp. 820-823.
- [3] C. Lee and D. Landgrebe *Analyzing High Dimensional Multispectral Data*, IEEE Transactions on Geoscience and Remote Sensing, 31 (1993), pp. 792-800.
- [4] M. Dundar, G. Fung, L. Bogoni, M. Macari, A. Megibow, B. Rao *A Methodology for Training and Validating a CAD System and Potential Pitfalls*, In Proc. CARS, (2004), pp. 1010-1014.
- [5] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Progress, San Diego, CA, 1990.
- [6] G. John, R. Kohavi, K. Pfleger, *Irrelevant Features and the Subset Selection Problem*, In Proc. of ICML, (1994).
- [7] J. Weston, A. Elisseeff, B. Scholkopf, M. Tipping *Use of the Zero-Norm with Linear Models and Kernel Methods*, Journal of Machine Learning Research, 3 (2003), pp. 1439-1461.
- [8] P. Bradley and O. Mangasarian *Feature Selection via Concave Minimization and Support Vector Machines*, Proc. of 15th International Conference on Machine Learning, (1998), pp. 82-90.
- [9] S. Mika, G. Ratsch, K. Muller *A Mathematical Programming Approach to the Kernel Fisher Algorithm*, Proc. NIPS 13, (2001), pp. 591-597.
- [10] J. Bi, K. Bennett, M. Embrechts, C. Breneman, M. Song *Dimensionality Reduction via Sparse Support Vector Machines*, Journal of Machine Learning Research, 3 (2003), pp. 1229-1243.
- [11] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio and V. Vapnik, *Feature Selection for SVMs*, Advances in Neural Information Processing Systems., 13, pp. 668-674.
- [12] J. Kittler, *Feature Set Search Algorithms*, Pattern Recognition and Signal Processing, Sijhoff and Noordhoff, the Netherlands, 1978.

Expanding the Training Data Space Using Emerging Patterns and Genetic Methods

Hamad Alhammady and Kotagiri Ramamohanarao

Department of Computer Science and Software Engineering
The University of Melbourne
Parkville, Victoria 3010, Australia
Email: {hhammady, rao}@cs.mu.oz.au

Abstract. Classification is a major problem in machine learning. Many classifiers have been developed recently. However, the performance of these classifiers is proportional to the knowledge obtained from the training data. As a result, traditional classifiers can not perform very well when the training data space is very limited. In this paper, we propose a new approach to expand the training data space (ETDS) using emerging patterns (EPs) [4] and genetic methods (GMs) [7]. EPs are those itemsets whose supports in one class are significantly higher than their supports in the other classes. GMs are evolutionary methods that incorporate computational techniques inspired by biology [8]. We combine the power of EPs and GMs to expand the training data space before applying standard classifiers. The expansion process is performed by generating more training instances using four techniques. An extensive experimental evaluation carried out on a number of datasets shows that our approach has a great impact on the performance of many traditional classifiers.

Keywords: emerging patterns, genetic methods

1 Introduction

The problem of classification is one of the most important research topics in KDD (Knowledge Discovery in Databases). Different classifiers have been proposed recently to solve this problem. These classifiers are based on one general idea; they obtain knowledge from the training dataset, and then they use this knowledge in a certain way to classify a test instance. Generally, traditional classifiers differ on the way they use the knowledge obtained from the training dataset. They have no control over the amount of knowledge which has a great impact on the classification performance, and which depends on the number of available instances in the training dataset.

In this paper, we employ emerging patterns (EPs) [4] and genetic methods (GMs) [7] to expand the data space of the training sets. EPs are a new kind of patterns introduced recently [4]. EPs can be used in many applications such as rare-class classification [1] [2].

They are defined as itemsets whose supports increase significantly from one class to another. The discriminating power of EPs can be measured by their growth rates. The growth rate of an EP is the ratio of its support in a certain class over that in another class. Usually the discriminating power of an EP is proportional to its growth rate.

2 Emerging Patterns and Classification

Let $obj = \{a_1, a_2, a_3, \dots, a_n\}$ is a data object following the schema $\{A_1, A_2, A_3, \dots, A_n\}$. $A_1, A_2, A_3, \dots, A_n$ are called *attributes*, and $a_1, a_2, a_3, \dots, a_n$ are *values* related to these attributes. We call each pair (attribute, value) an *item*.

Let I denote the set of all items in an encoding dataset D . *Itemsets* are subsets of I . We say an instance Y contains an itemset X , if $X \subseteq Y$.

Definition 2.1. Given a dataset D , and an itemset X , the support of X in D , $s_D(X)$, is defined as

$$s_D(X) = \frac{\text{count}_D(X)}{|D|} \quad (1)$$

where $\text{count}_D(X)$ is the number of instances in D containing X .

Definition 2.2. Given two different classes of datasets D_1 and D_2 . Let $s_i(X)$ denotes the support of the itemset X in the dataset D_i . The growth rate of an itemset X from D_1 to D_2 , $GR_{D_1 \rightarrow D_2}(X)$, is defined as

$$GR_{D_1 \rightarrow D_2}(X) = \begin{cases} 0, & \text{if } s_1(X)=0, s_2(X)=0 \\ \infty, & \text{if } s_1(X)=0, s_2(X) \neq 0 \\ \frac{s_2(X)}{s_1(X)}, & \text{otherwise} \end{cases} \quad (2)$$

Definition 2.3. Given a growth rate threshold $p > 1$, an itemset X is said to be a *p-emerging pattern* (*p-EP* or simply EP) from D_1 to D_2 if $GR_{D_1 \rightarrow D_2}(X) \geq p$.

Let $C = \{c_1, \dots, c_m\}$ is a set of *class labels*. A *training dataset* is a set of data objects such that, for each object *obj*, there exists a class label $c_{obj} \in C$ associated with it. A *classifier* is a function from attributes $\{A_1, A_2, A_3, \dots, A_n\}$ to class labels $\{c_1, \dots, c_m\}$, that assigns class labels to unseen examples.

3 Expanding the Training Data Space

3.1 Overview

We propose a new approach to improve the classification performance. The key idea is to expand the training data space by generating more training instances. We present four methods to generate additional training instances. These methods involve using emerging patterns (EPs) and genetic methods (GMs). We use an existing algorithm [3] to mine EPs. The generation methods are presented in subsections 3.2, 3.3, 3.4, and 3.5.

3.2 Method 1: Generation by Superimposing EPs

This method was first introduced in our earlier work [2]. The main difference is that our work in [2] is dedicated to deal with rare-class datasets. Hence the generation is performed for the rare class only. In this paper we deal with balanced datasets, and the generation is performed for all classes. Given a training dataset and a set of mined EPs, the following steps are repeated for every class in the dataset:

- The attribute values (considered as itemsets consisting of one attribute) that have the highest growth rates in the current class are found.
- The set of EPs related to current class is divided into a number of groups such that EPs in each group have attribute values for most of the elements in the attribute set.
- The new instances are generated by combining the EPs in each group. If a value for an attribute is missing from a group, it is substituted by the value that has the highest growth rate for the same attribute (found in step 1).

Figure 1 shows an example of this process. Suppose we have a dataset consisting of 7 attributes $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. The values that have the highest growth rate for these attributes in the current class are $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. Table a in figure 1 represents a group of three EPs. These EPs are $e_1\{(A_1=1), (A_2=x_1)\}$, $e_2\{(A_5=y_1), (A_6=2), (A_7=3)\}$, and $e_3\{(A_2=x_2), (A_3=4), (A_5=y_2)\}$. Notice that none of these EPs has a value for attribute A_4 . As a result, the value

that has the highest growth rate for attribute A_4 , v_4 , will be used as described earlier. The three EPs and the value chosen for attribute A_4 are combined to generate four new instances for the current class. The intersected values (x_1 and x_2 for attribute A_2 , and y_1 and y_2 for attribute A_5) are used one after the other to generate the new instances. The four generated instances are shown in table b in figure 1.

	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇
e ₁	1	x ₁					
e ₂					y ₁	2	3
e ₃		x ₂	4		y ₂		

(a)

A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇
1	x ₁	4	v ₄	y ₁	2	3
1	x ₁	4	v ₄	y ₂	2	3
1	x ₂	4	v ₄	y ₁	2	3
1	x ₂	4	v ₄	y ₂	2	3

(b)

Figure 1: Example of generation by superimposing EPs

3.3 Method 2: Generation by Crossover

Consider two instances and a randomly chosen breaking point. The values of these instances are switched before the breaking point. Figure 2 shows an example of this process. Suppose that we have two instances $I_1 \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$ and $I_2 \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, and the breaking point is randomly chosen to be between the third and fourth attributes. Then the two generated instances are $I_3 \{b_1, b_2, b_3, a_4, a_5, a_6, a_7\}$ and $I_4 \{a_1, a_2, a_3, b_4, b_5, b_6, b_7\}$.

Random breaking point ↓

I ₁	a ₁	a ₂	a ₃	a ₄	a ₅	a ₆	a ₇
I ₂	b ₁	b ₂	b ₃	b ₄	b ₅	b ₆	b ₇

I ₃	b ₁	b ₂	b ₃	a ₄	a ₅	a ₆	a ₇
I ₄	a ₁	a ₂	a ₃	b ₄	b ₅	b ₆	b ₇

Figure 2: Example of generation by crossover

3.4 Method 3: Generation by Mutation

In this method, we mutate an instance with the highest-growth-rate values. The first step is to find the attribute values that have the highest growth rates in the current

class. A random binary number is chosen and overlapped over the instance. All attribute values in the instance that match 1's in the random binary number are replaced by the values that have the highest growth rates. Figure 3 shows an example of this method. Suppose that we have an instance $I_1 \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$. The highest-growth-rate values are $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. The random binary number is (1001011). The values in I_1 that match 1's in the random binary number (namely, the first, fourth, sixth, and seventh attributes) are replaced with the highest-growth-rate values that they match.

I_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Random binary number	1	0	0	1	0	1	1
Highest-growth-rate values	v_1	v_2	v_3	v_4	v_5	v_6	v_7
I_2	v_1	a_2	a_3	v_4	a_5	v_6	v_7

Figure 3: Example of generation by mutation

3.5 Method 4: Generation by Mutation and EPs

In this method, we mutate an instance with an EP. All values in the instance are replaced with their matched values in the EP. Figure 4 shows an example of this method. Suppose that we have an instance $I_1 \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$. We want to mutate this instance with an EP $e \{e_2, e_3, e_6\}$. We replace the values in I_1 (namely, the second, third, and sixth values) with their matched values in e .

I_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7
e		e_2	e_3			e_6	
I_2	a_1	e_2	e_3	a_4	a_5	e_6	a_7

Figure 4: Example of generation by mutation and EPs

3.6 Generation Constraints and Fitness Function

Using the generation methods explained in subsections 3.2, 3.3, 3.4, and 3.5, a large number of training instances can be generated. Our aim is to generate a reasonable number of high-quality training instances. We propose four constraints to achieve this aim. These constraints are:

1. Instead of using the whole range of EPs, we use a certain percentage (α) of the best-quality EPs in method 1 (generation by superimposing EPs) and method 4 (generation using mutation and EPs).

2. Instead of using the whole range of the training data, we use a certain percentage (β) of the best-quality instances in method 2 (generation using crossover), method 3 (generation using mutation), and method 4 (generation using mutation and EPs).
3. Adding the best-quality generated instances to the training dataset, and discarding the bad-quality generated instances.
4. Stopping the generating process when the training dataset is increased by a certain percentage (χ).

The quality of EPs and instances (constraints 1, 2, and 3) can be measured by their fitness functions. A fitness function is defined as a measure to evaluate the elements (EPs or data). For the EPs (constraint 1), the fitness function can be measured by the growth rate. The growth rate of an EP is proportional to its discriminating power. Hence, it indicates how good this EP is. The fitness function of an EP (growth rate) is defined by equation 2.

The fitness function of a data instance can be measured by the average support of the attribute values in this instance. Suppose that we have an instance $I \{a_1, a_2, a_3, \dots, a_n\}$. We first find the supports of all the attribute values (from a_1 to a_n). Then we average these supports to obtain a measure that tells how good the instance is. This fitness function is defined by equation 3.

$$Fitness(I) = \frac{\sum_{i=1}^n s(a_i)}{n} \quad (3)$$

We use this function to choose the best-quality instances to be used in the generating process (constraint 2). Moreover, we use it to evaluate (keep or discard) the instances generated by our proposed methods (constraint 3).

4 Experimental Evaluation

In order to investigate the performance of our proposed system, we carry out a number of experiments. We use 30 datasets from UCI repository of machine learning databases [5]. We also use C4.5, boosting (C4.5), and SVM [6] as base learners. The testing method is stratified 10-cross-validation. Each experiment is carried out as follows:

- Run a base learner $\{M\}$.
- Apply method 1 (generation by superimposing EPs) and run the base learner $\{M1\}$.

- Apply method 2 (generation by crossover) and run the base learner {M2}.
- Apply method 3 (generation by mutation) and run the base learner {M3}.
- Apply method 4 (generation by mutation and EPs) and run the base learner {M4}.
- Apply the union of the four methods and run the base learner {M*}.

The last method (M*) is explained as follows. We use our four methods (presented in section 3) to generate more training instances. We use the fitness function (presented in subsection 3.6) to evaluate the generated instances and choose the best instances resulted from the four methods.

The results of our experiments are shown in tables 2, 3, and 4. Taking into consideration that no classification method can outperform all others in all datasets, and that different classifiers deal differently with a dataset, we can draw some interesting points from results as follows:

- Using C4.5, M3 wins on 8 datasets, M* wins on 7 datasets, M1 wins on 6 datasets, M4 wins on 4 datasets, M2 wins on 3 datasets, and M wins on 2 datasets.
- Using boosting (C4.5), M1 wins on 8 datasets, M* wins on 6 datasets. M3 wins on 5 datasets, both M2 and M4 win on 4 datasets, and M wins on 3 datasets.
- Using SVM, M* wins on 8 datasets, Both M1 and M3 win on 6 datasets, M2 wins on 5 datasets, M4 wins on 3 datasets, and M wins on 2 datasets.
- Considering the three classifiers, M*, M1, and M3 are the most powerful methods as they win on 21, 20, and 19 cases, respectively. M2 wins on 12 cases, M4 wins on 11 cases, and M wins on 7 cases.
- Our approach as a whole (considering the winning method for each dataset) outperforms the base learner in 28 datasets out of the 30 datasets used in our experiments. This significant result is achieved in the three sets of experiments (C4.5 experiments, boosting experiments, and SVM experiments).
- In terms of the average accuracy, M* outperforms all other methods using the three classifiers.
- Our proposed methods increase the accuracy significantly in some cases. For example, the increase in C4.5 experiments is up to 5.6% (Hayes-roth dataset). In boosting experiments, the increase is up to 5% (Pima dataset). In SVM experiments, the increase is up to 4.1% (Iono dataset).

5 Conclusions

In this paper, we propose a new approach to expand the training data space. Our approach, called ETDS, introduces the idea of generating new training instances using emerging patterns (EPs) and genetic methods (GMs). It uses the advantage of EPs, which have a strong discriminating power, and GMs, which have a great reproduction ability, to expand the training data space by generating new instances for the training set. We introduce four methods to achieve this aim. We experimentally demonstrate that our approach has a great impact on improving the results of other classification methods such as C4.5, boosting, and SVM. Furthermore, our approach can be thought of as an avenue to extend the applications of EPs and GMs to cover a wide range of problems in data mining.

Table 2: Results of C4.5 experiments

Dataset	C4.5					
	M	M1	M2	M3	M4	M*
Adult	86.1	88.3	87.2	87.7	86.4	87.8
Australian	84.3	84.9	85.7	84.4	83.9	86.9
Breast	94.6	96.3	94.6	95.1	95.9	95.9
Cleve	73.8	72.6	73.2	74.9	74.6	74.8
Credit card	85.3	86.4	85.2	86.7	87.2	86.9
Crx	85.3	85.8	85.1	83.7	84.6	85.5
Diabetes	73.4	71.6	71.8	72.8	72.2	74.7
Flags	57.5	59.3	59.7	57.9	57.1	59.3
German	69.6	69.9	70.2	71.3	70.6	72.6
Glass	64.7	66.2	64.9	64.1	65.3	66.1
Hayes-roth	70.2	70.7	72.5	75.3	75.8	75.5
Heart	80.6	81.1	80.4	81.3	80.9	81.1
Hepatitis	81.8	80.9	80.2	81.7	82.6	82.4
Horse	85.2	85.4	85.4	85.9	85.7	85.6
Hypo	99.3	97.2	98.1	96.8	98.4	99.4
Iono	89.4	88.6	89.5	89.1	89.8	91.1
Labor	76.9	76.2	77.3	77.8	76.9	77.5
Liver	58.1	59.3	59.8	58.5	61.3	60.6
Machine	87	87.5	89.5	89.2	88.4	89.2
Pima	74	76.2	74	73.4	74.7	76.1
Segment	93.5	93.6	92.7	94.8	94.1	94.4
Sick	98.7	97.5	97.7	98.9	98.1	98.6
Sonar	75.3	76.8	76.3	77.6	75.4	77.4
Staimage	85.2	84.7	85.5	86.1	84.9	85.8
Tic-tac-toe	84.2	86.5	88.1	84.6	87.1	87.8
Vehicle	71.2	72.8	69.5	70.2	70.7	72.9
Votes	77.8	78.9	77.4	78.5	78.1	78.8
Wine	84.2	85.6	83.1	84.8	85.2	85.9
Yeast	49.9	48.1	48.5	49.4	48.6	49.7
Zoo	93	91	89	91	90	92
Average	79.67	79.99	79.73	80.11	80.15	82.50

Table 3: Results of boosting experiments

Dataset	Boosting(C4.5)					
	M	M1	M2	M3	M4	M*
Adult	86.6	88.7	86.9	86.1	87.3	88.6
Australian	82.3	83.4	81.9	84.1	83.1	83.8
Breast	95.9	97.1	96.3	95.7	96.3	96.9
Cleve	80.8	79.9	81.2	79.2	83.1	82.7
Credit card	84.9	86.4	84.4	85.6	85.1	85.7
Crx	82.8	83.2	83.7	82.8	82.4	83.5
Diabetes	72	72	72.8	72.3	72.2	72.4
Flags	54.9	57.8	55.1	55.3	57.2	57.7
German	71.8	73.1	71.2	73.3	72.4	72.8
Glass	74.1	74.9	76.5	74.3	75.2	76.4
Hayes-roth	78.3	79.5	78.5	78.5	78.1	79.7
Heart	76.2	77.4	76.5	79.2	77.2	78.9
Hepatitis	79.2	79.7	80.5	81.3	80.1	82.4
Horse	82.2	82	81.9	82.2	84.3	84.1
Hypo	98.9	98.1	98.5	98.1	97.3	98.8
Iono	92	89.8	91.5	87.7	89.2	92.7
Labor	82	82.6	84.1	84.8	82	84.6
Liver	61	61.5	60.7	60.4	61.3	61.4
Machine	90.2	92.6	90.9	90	90.9	92.4
Pima	71.5	75.2	73.3	75	73.7	76.5
Segment	94.8	94.2	95.2	94.9	96.3	97.7
Sick	98.7	98.6	98.1	98.9	98.4	98.7
Sonar	78.2	78.7	79.2	78.1	79.5	79.4
Staimage	85.9	86.2	86.5	85.4	86.7	86.5
Tic-tac-toe	95.4	97.8	94.9	94.4	95.3	97.5
Vehicle	73	73.5	74.2	73.5	73.7	75.3
Votes	77.9	79.1	76.8	78.6	78.2	78.8
Wine	91.5	90.1	92.7	91.8	91.8	92.6
Yeast	50.2	49.3	48.6	48.3	48.2	48.9
Zoo	98	95	95	91	93	97
Average	81.37	81.91	81.58	81.36	81.65	82.81

Table 4: Results of SVM experiments

Dataset	SVM					
	M	M1	M2	M3	M4	M*
Adult	87.1	86.7	85.3	86.2	85.3	87.5
Australian	85.1	86.8	84.6	85.9	86.2	86.6
Breast	95.9	94.6	95.5	95.1	94.3	95.7
Cleve	84.4	86.3	85.1	85.8	84.2	86.7
Credit card	84.6	85.1	84.3	85.9	85.1	85.8
Crx	84.7	85.2	84.2	84.9	84.9	84.9
Diabetes	72.8	73.6	72.8	72.9	73.2	73.3
Flags	60.6	59.2	60.1	59.2	57.7	59.9
German	74.6	74.7	74.1	75.9	75.1	75.7
Glass	75.7	76.1	75.1	75.8	76.9	77.8
Hayes-roth	84.7	84.1	85.6	86.1	85.2	86.9
Heart	85.1	86.7	84.3	84.6	84.3	86.5
Hepatitis	83.3	82.9	85.2	82.9	83.7	84.8
Horse	86.9	86.4	86.3	87.7	87.5	87.5
Hypo	98.9	97.7	98.9	98.9	99.2	98.9
Iono	88.5	89.3	91.1	89.5	88.7	92.6
Labor	97.4	97.6	97.1	97.9	96.4	97.6
Liver	66.8	67.1	66.5	66.5	65.8	66.8
Machine	93.7	93.5	93.1	92.8	92.4	93.8
Pima	73.6	74.2	75.8	74.5	73.9	75.7
Segment	95.5	95.9	97.1	95.9	95.6	96.9
Sick	93.9	95.4	94.3	94.3	93.2	96.6
Sonar	77.7	78.3	78.2	78.9	77.3	78.6
Staimage	86.7	87.3	88.8	87.7	86.7	88.2
Tic-tac-toe	98.3	98.8	98.3	98.1	97.4	98.7
Vehicle	68.5	68.8	69.9	68.8	66.1	69.8
Votes	64.7	64.1	64.7	65.6	65.4	65.9
Wine	95.4	94.2	95.1	97.1	95.9	96.9
Yeast	52.7	51.5	52.1	52.7	52.9	52.7
Zoo	97	93	92	92	98	97
Average	83.16	83.17	83.18	83.33	82.95	84.21

References

- [1] H. Alhammady, and K. Ramamohanarao. The Application of Emerging Patterns for Improving the Quality of Rare-class Classification. In Proceedings of 2004 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '04), Sydney, Australia.
- [2] H. Alhammady, and K. Ramamohanarao. Using Emerging Patterns and Decision Trees in Rare-class Classification. In Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04), Brighton, UK.
- [3] H. Fan, and K. Ramamohanarao. An Efficient Single-Scan Algorithm For Mining Essential Jumping Emerging Patterns for Classification. In Proceedings of 2002 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '02), Taipei, Taiwan.
- [4] G. Dong, and J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In Proceedings of 1999 International Conference on Knowledge Discovery and Data Mining (KDD '99), San Diego, CA, USA.
- [5] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases. Department of Information and Computer Science, University of California at Irvine, CA, 1999. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [6] I. H. Witten, E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo, CA., 1999.
- [7] Z. Michalewicz. Genetic Algorithms + Data Structure = Evolution Programs. Springer-Verlag, Berlin, 1996.
- [8] M. Obitko, and P. Slavik. Visualization of Genetic Algorithms in a Learning Environment. In Proceedings of 1999 Spring Conference on Computer Graphics, (SCCG '99). Bratislava, Slovakia.

Making Data Mining Models Useful to Model Non-Paying Customers of Exchange Carriers

Wei Fan¹

Janak Mathuria²

Chang-tien Lu²

¹ IBM T.J.Watson Research Center, Hawthorne, NY 10532

weifan@us.ibm.com

² Virgia Tech Northen Virginia Center, Computer Science, Church Falls, VA 20043

{janakm, ctlu}@vt.edu

Abstract

Due to both limitations of technologies and the nature of the problems, data mining may not be able to solve a problem completely in a way as one wishes. When this happens, we need to first understand the actual need of business, characteristic of available partial solution, and then make compromises between the technology solution and business needs. A majority of the papers published in data mining conferences and journals seem to concentrate only on the success side of the story. In this paper, we discuss our experiences and the complete process from near failure to success when applying inductive learning techniques to predict non-paying customers of competitive local exchange carriers (CLEC's), currently at 20%. Experiments with a number of state-of-the-art methods and algorithms found that most customers were labeled as paying on time. Cost-sensitive learning is not possible since the target company cannot define a cost-model. Finally, after discussing with the billing department, a compromised but still useful solution is to predict the probability that someone will default. The billing team can use the predicted score to prioritize collection efforts as well as to predict cash flow. We have found that two randomized decision tree ensemble methods (Fan's random decision tree and a probabilistic extension of Breiman's random forest) are consistently more accurate in posterior probability estimation than single decision tree based probability calibration methods. The software, both Fan's RDT and probabilistic extension of random forest, as well as a longer version of this paper will be made available by the contacting author.

1 Introduction

The enactment of the US Telecommunication Act of 1996 has separated infrastructure provider and service provider in order to break up monopoly and create more competitive market for consumers. With this act,

a service provider to consumer is not necessarily the owner of the physical infrastructure, and an infrastructure provider can lease their lines to multiple service providers. A consumer can choose among many competing service providers without knowing about the underlying infrastructure. After this act was implemented, two kinds of telecommunication companies were created: ILEC and CLEC. ILEC is short for Incumbent Local Exchange Carrier. An ILEC is a telephone company that owns the infrastructure (phone lines, switches, etc) and may also provide local service to end users. An example of ILEC is Verizon. CLEC is short for Competitive Local Exchange Carrier, a telephone company that does not own any infrastructure but rents the infrastructure from an incumbent local exchange carrier (ILEC) and provides services. An incomplete list of CLEC's include Paetec, CBeyond, Alligance, X0, Level 3, and Cypress Communications. There has been an explosion in the number of CLEC's since 1996. As compared to Incumbent Local Exchange Carriers (ILEC's), CLEC's have always been more susceptible to the risk of a very high level of Days Sales Outstanding (DSO) and customer non-payment (at approximately 20%).

We were approached by one CLEC company to build a model to predict which customer will default. It is important to distinguish between two important concepts: "late in payment" and "default". Assume that the closing date of a CLEC company is always on the 1st of the month and the due date is always on the 21st of the same month, and someone does not have any previous balance. The customer receives his current month's statement with the closing date of March 1st. The current amount due is the total charge incurred during the previous billing period from Feb 2nd to March 1st. If he pays this due amount by the 21st, the account is considered "current". However, if the due amount is not paid in full by March 21st but before the next due date of April 21st, the account is marked

as late in payment. If by the next due date of April 21st, the due amount by March 21st is still not paid in full, the account will be labelled as “default”. Besides possibly reporting to a credit agency, there is almost nothing a CLEC company can do to a late customer (but not default) since the customer is granted a grace period of 30 days before being labelled as defaulted. Even a customer is defaulted, CLEC companies are required under law to send letters out and wait for the customers to remain defaulted for 3 full months before the company are legally allowed to cut off service and submit the case to a third party collection agency.

2 Problem Description and Modeling

The back office database of the target company collects call detail history as well as billing and payment history of each customer. The task is simply stated as to “predict if a customer will default in 60 days on the current month’s due solely based on the available billing and calling history”. Ideally, all available history for a particular customer could be used to train the predictive model. However, this may be not be necessary and practical for the billing department. When a customer does not pay for the due amount for three consecutive months, by law, the company can cut off the service and refer the case to a collection agency. Different customers have been with the company for different amount of time, hence there is different amount of information to mine for each customer. In the same time, the target company only has limited database capability and cannot afford large number of sophisticated queries. In the end, the billing department agrees that four months of data is feasible for them and useful under “legal terms”. Assuming that it is currently in March, we use the history data from Dec to March to predict if someone will default in May (or in 60 days from March). We define the following 16 features for each of the four months queried from the database. It is very important to point out that one of these features (the amount outstanding in the >60 days aging bucket) is to label which customer defaults and cannot be used for training.

- Billing Balance Related Features

1. The amount outstanding in the past 0-30 days.
2. The amount outstanding in the 31-60 days.
3. The amount outstanding > 60 days.

- Call Profile Related Features

4. The non-usage revenue for this month.
5. The total number of local calls.
6. The total number of long distance (LD) calls. Long distance calls includes regional toll, inter-state calls and international calls. Long distance calls are generally more expensive than local calls.

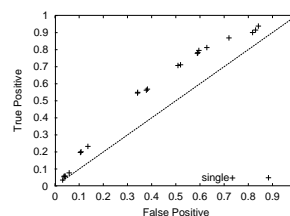
7. The revenue from local calls.
8. The revenue from LD calls.
9. The number of local calls to the 5 most called local numbers
10. The number of LD calls to the 5 most called LD numbers
11. The total amount of local revenue from the calls to the top 5 local numbers.
12. The total amount of LD revenue from the calls to the top 5 LD numbers.
13. The number of after hours local calls. After hour calls are defined as calls originating on weekends or between 6 PM to 8 AM on weekdays.
14. The number of after hours LD calls.
15. The revenue from after hours local calls.
16. The revenue from after hours LD calls.

3 Initial Trial, Error, and Failure Experiences

At the start of the project, the billing department aims to catch as many default customers as possible without incurring a large number of false positives. Since the dataset is not skewed at all (approximately 20% non-paying customers), we experimented with decision trees (both commercial C5.0 and free C4.5), naive Bayes, Ripper, Bagging on C4.5, numerical version of AdaBoost on C4.5, and Random Forest (with subset ratio of 0.5). We used the first half of the data (from early months) as training and the remaining half of the data (belonging to later months) for testing. Unfortunately, all these methods nearly predict every customer as “paying on time”. Detailed results can be found in the long version of this paper available upon request from the contacting author. As a twist to the traditional solutions that minimize traditional 0-1 loss, a different approach is to use “probability output” instead of class label output. Probability is a rather continuous output and we may be able to choose a decision threshold with reasonable true and false positive rates. For decision trees, assuming that n_c is the number of examples with class c in a node, and n is the total number of examples in the same node, then the probability that \mathbf{x} is a member of class c is simply

$$(3.1) \quad P(c|\mathbf{x}) = \frac{n_c}{n}$$

Figure 1: Complete ROC of Single Decision Tree that outputs probabilities



Assuming that t is a decision threshold to predict \mathbf{x} as class c , i.e., $P(c|\mathbf{x}) \geq t$. Since there are limited number of nodes in a decision tree and each node can output just one probability value for any examples classified by this node, the probability output by decision trees is not completely continuous, and the resultant ROC plot is not continuous either. We draw a complete ROC plot to study if the use of probability will improve the performance. To draw a complete ROC plot, we track all unique probability output of a model and use these values as the decision threshold t . However, as shown in Figure 1, the single tree's performance is only slightly better than that of random guessing.

Re-evaluate the Empirical Need We experimented with feature selection and a few other variations of feature vector, none of them seemed to help. We realized that these results may just be the fact of this particular problem. Some problems are simply stochastic, or the true model will produce different labels for the same data item at different times. When this happens, we will never be able to have 100% accurate model. The best we can do is to predict the label that happens the most often under traditional 0-1 loss, or the label that minimizes a given loss function under cost-sensitive loss. In order to make the model useful, we spoke to the billing department and were told that the cost model would be very difficult to develop and will probably be over simplified. However, a reliable score that predicts the true probability that someone will default in two months will help both the billing department and the Chief Financial Officer. For the billing department, if the probability output of a model is a reliable estimate of the true probability, it will help the collection team to better prioritize on which customers to expend their efforts when the predicted default customers become late in payment. As mentioned above, CLECs only have limited staff and resources for their collection efforts. Often, when presented with a large list of customers who are late in making their payments, the collection team has very little way of knowing which customers to contact first and very little hope of reaching each and every customer. Focusing on the customers most likely to fall substantially behind on paying their dues would help in preventing outright default in many cases or in cases when this is inevitable, would at least help the company minimizing losses by discontinuing services at the earliest. For the CFO, the estimated probability can be used to compute expected cash flows in two months. If an account has an outstanding balance of \$100 and a 30% chance to default, the expected payment from this account in two months will be $\$100 \times 0.7 = \70 . We sum up over all the outstanding balances and will have a good estimate of actually payment in two

months. We experimented with three methods to estimate the posterior probability, including Fan's random decision tree [Fan et al., 2003], a variation of Breiman's random forest [Breiman, 2001], the set of calibrated single decision tree probability methods by Zadrozny and Elkan [Zadrozny and Elkan, 2001].

4 Estimating Posterior Probabilities

We have considered the following methods to estimate posterior probability.

Fan's Random Decision Tree Random Decision Trees or RDT was originally proposed by Fan in [Fan et al., 2003]. The idea of RDT exhibits significant difference from the way conventional decision trees are constructed. RDT constructs multiple decision trees *randomly*. When constructing one particular tree, contrary to the use of purity check functions by traditional algorithms, e.g., information gain, gini index and others, RDT chooses a remaining feature randomly. A discrete feature can be chosen only once in a decision path starting from the root of the tree to the current node. A continuous feature can be chosen multiple times in the same decision path, but each time, a different decision threshold is chosen. The tree stops growing if either the current node becomes empty or the depth of the tree exceeds some predefined limit. Since both feature and decision threshold for continuous features are chosen randomly, random decision trees constructed at different times are very likely to be different. If either every features is categorical or continuous features are discretized, the number of different random trees are bounded. When there are continuous features and random decision threshold is picked, the number of different random trees is potentially unlimited. The depth of the tree is limited to be up to the number of features in order to give each feature equal opportunity to be chosen in any decision path.

During classification, each random tree computes a posterior probability from the leaf node as shown in Eq 3.1. The posterior probability outputs from multiple decision trees are averaged as the final probability estimation. In order to make a decision, a well-defined loss function is needed. For example, if the loss function is 0-1 loss or traditional accuracy, the class label with the highest posterior probability will be predicted. As stated in [Fan et al., 2003], typically 30 random trees should give satisfactory results and more trees may not be necessary. However, experimental studies have shown 10 random trees produce results that sufficiently close to that of 30 random trees. Although quite different from well-accepted methods that employ purity check functions to construct decision trees, random trees have been shown to have accuracy comparable

to or higher than bagging and random forest but at a fraction of the training cost, and has been independently implemented and confirmed separately by Ian Davidson and his student, Tony Fei Liu and Kai Ming Ting, Ed Greengrass, Xinwei Li and Aijun An.

Probabilistic Extension of Breiman’s Random Forest Random forest [Breiman, 2001] introduces randomness into decision tree by i) training multiple trees from bootstraps and ii) randomly sampling a subset of remaining features (those not chosen yet by a decision path) and then choosing the best splitting criteria from this feature subset. The chosen size of the subset is provided by the user of random forests. Random forests performance simple voting on the final prediction the same way as bagged decision trees. In this paper, we propose a probabilistic variation of random forest. Instead of training from bootstraps and predicting class label as in Breiman’s random forest, each tree in our version is trained from the original dataset, and outputs posterior probability. Similar to random decision trees, the probabilities from all trees in the forest are averaged as the final probability output. Our variation is called Random Forest+.

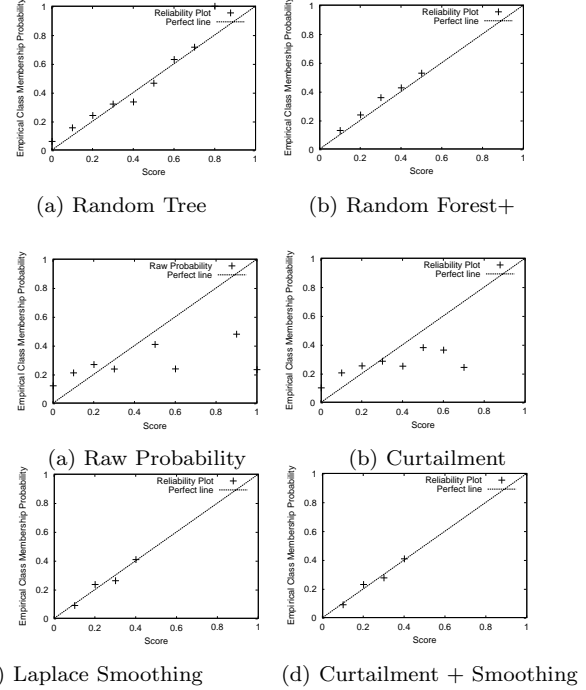
Zadrozny and Elkan’s Calibrated Probabilities for Decision Trees Raw probability or original probability of decision trees is defined in Eq 3.1. Smoothed probability considers the base probability in the data and is defined as: $P(c|\mathbf{x}) = \frac{n_c + m \cdot b}{n + m}$, where b is the base rate of positive examples in the training set (i.e., 20% for our data) and m is chosen to be 100 (suggested by Zadrozny and Elkan [Zadrozny and Elkan, 2001]). Curtailment stops searching down the tree if the current node has less than v examples and uses the current node to compute probabilities. As discussed and suggested in [Zadrozny and Elkan, 2001], the exact value v is not critical if v is chosen small enough. We have used $v = 100$ in our experiments. Curtailment plus smoothing is a combination of curtailment and smoothing.

5 Experiments

Since the problem is time-sensitive in nature, the training set ought to be taken from earlier months while the testing set ought to be taken from remaining later months. Traditional CV is not entirely applicable for this situation. Instead, we use different amount of training and testing data: 50% training- 50% testing, 15% training - 85% testing, as well as 85% training - 15% testing. This will not only give us an idea of the performance on different datasets as well as different amount of data.

Figures 2 to 4 show the “reliability plot” of ensemble-based methods and calibrated decision tree

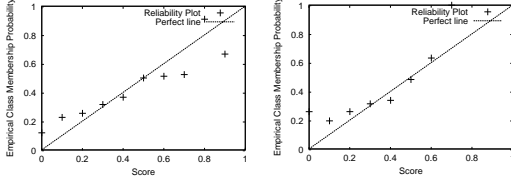
Figure 2: 50% Training and 50% Testing



methods under three combinations of training and testing data sets. “Reliability plot” shows how reliable the “score” of a model is in estimating the empirical probability of an example \mathbf{x} to be a member of a class y . To draw a reliability plot, for each unique score value predicted by the model, we count the number of examples (N) in the data having this same score, and the number (n) among them having class label y . Then the empirical class membership probability is simply $\frac{n}{N}$. Most practical datasets are limited in size; some scores may just cover a few number of examples and the empirical class membership probability can be extremely over or under estimated. To avoid this problem, we normally divide the range of the score into continuous bins and compute the empirical probability for examples falling into each bin. In Figures 2 to 4, the percentage of features sampled by Random Forest+ is 50%. In the extended version of this paper, we have results to sample different percentage of features and 50% appears to be the most accurate.

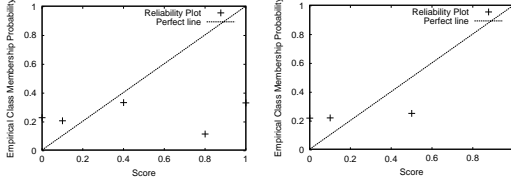
Comparing different methods for different combinations of training and testing data size, random decision tree and random forest+ are the most reliable and stable methods, as shown in the top two plots of Figures 2 to 4. Their probability estimation is insensitive to the amount of training data, i.e., 15%, 50% or 85%. The four calibrated decision tree methods (the bottom four

Figure 3: 15% Training and 85% Testing



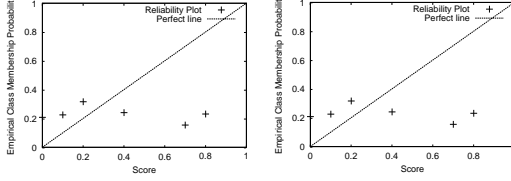
(a) Random Tree

(b) Random Forest+



(a) Raw Probability

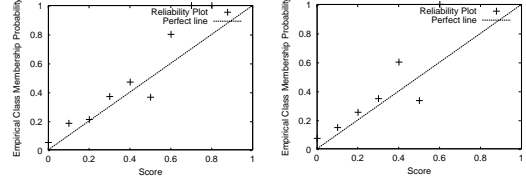
(b) Curtailment



(c) Laplace Smoothing

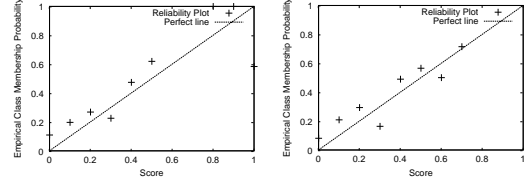
(d) Curtailment + Smoothing

Figure 4: 85% Training and 15% Testing



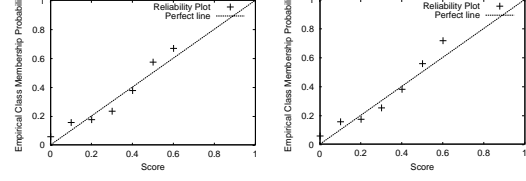
(a) Random Tree

(b) Random Forest+



(a) Raw Probability

(b) Curtailment



(c) Laplace Smoothing

(d) Curtailment + Smoothing

plots in Figures 2 to 4) appear to be very sensitive to the amount of training data. When the training data size is small, 15% in our experiment or approximately 670 data items, all other method except for random decision tree and random forest appears random and do not a clear pattern patterns. However, when the training data increases to 85%, every tested method appear to have a pattern that is well correlated with the perfect line.

6 Conclusions

In this paper, we formulated the problem to mine defaulted customers for competitive local exchange carriers or CLEC. We discussed the empirical importance, i.e., managing collection efforts as well as projecting cash flow, of this effort for the survival of these companies. We detailed the complete feature construction process to model the calling, billing and payment history from back office raw data. We also addressed the important problem of how to make a data mining model useful for a business based on the nature of the problem and the actual performance of a mined model. We evaluated many methods and found that the most successful method is to use random decision and random forest plus to estimate the true probability that a customer will default. The probability estimation by both methods have been found to be reliable under two different

customer groups and very different amount of training data. This solution helps the CLEC in two important ways. The predicted score prioritizes the collection effort by the billing company. The product of the due amount by the probability of default gives a good estimate of the cash flow in the future. In the algorithm part of this paper, we find that both random decision tree and random forest plus are reliable and stable in estimating probabilities even when the amount of data is extremely small. However, single decision trees are extremely sensitive to the amount of training data. When the data is small, their probability output are close to random, which prohibits the application of Zadrozny and Elkan's calibration methods.

References

- [Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.
- [Fan et al., 2003] Fan, W., Wang, H., Yu, P. S., and Ma, S. (Nov 2003). Is random model better? on its accuracy and efficiency. In *Proceedings of Third IEEE International Conference on Data Mining (ICDM-2003)*, Melbourne, FL.
- [Zadrozny and Elkan, 2001] Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of Eighteenth International Conference on Machine Learning (ICML'2001)*.

Matrix Condition Number Prediction with SVM Regression and Feature Selection

Shuting Xu, * Jun Zhang †

Department of Computer Science, University of Kentucky,
Lexington, KY 40506-0046, USA

Abstract

Condition number of a matrix is an important measure in numerical analysis and linear algebra. The general approach to obtaining it is through direct computation or estimation. The time and memory cost of such approaches are very high, especially for large size matrices. We propose a totally different approach to estimating the condition number of a sparse matrix. That is, after computing the features of a matrix, we use support vector regression (SVR) to predict its condition number. We also use feature selection strategies to further reduce the response time and improve accuracy. We design a feature selection criterion which combines the weights from SVR with the weights from comparison of matrices with their preconditioned counterparts. Our preliminary experiments show that the results are encouraging.

Key words: condition number, support vector machine, feature selection, preconditioning

1 Introduction

The condition number $k(A)$ of a nonsingular matrix A with respect to a matrix norm is formally defined as $\|A\| \cdot \|A^{-1}\|$. In this paper, we only stress on the condition number corresponding to 1-norm $k_1(A)$. If $k(A)$ is relatively small, then the matrix A is called a *well-conditioned matrix*, but if $k(A)$ is large, then A is an *ill-conditioned matrix*. Condition number is a widely used matrix feature in many areas, such as in numerical analysis and linear algebra. There are several ways to obtain the condition number of a matrix. The direct method is to compute A^{-1} first and then multiply its norm with the norm of A . However, computing $k(A)$

for even the simplest matrix norms is three times as expensive as solving $Ax = b$ in the first place. Another method is using a less expensive algorithm to estimate the condition number. There are some estimation algorithms in literature [2]. For example, LAPACK uses subroutine SGECON to compute the condition number. Its time cost is $O(n^2)$ extra beyond the $O(n^3)$ cost of solving $Ax = b$, where n is the dimension of A . If the size of the matrix is relatively large (e.g., $n > 20000$), the memory will be depleted before the computation is completed on our SunBlade 150 workstations. In most cases, the estimated condition number is within a factor of 10 of the true condition number, but there exist some counter-examples with large estimation errors.

We propose a new approach to estimating the condition number of a *sparse* matrix - predicting condition number from matrix features using data mining techniques. The predictor used is SVM regression (SVR) [9, 10]. We also apply some feature selection methods [1, 7] to further reduce the time cost and improve precision. The method proposed is especially suitable for building an online condition number query system. The training of SVR can be done in background, thus the response time just includes the time to compute matrix features and predict the result through trained model, which is much faster than using the traditional methods mentioned above.

2 SVM Regression

SVM regression is an approach to predicting real-valued outputs. In ε -SV regression [10], the goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible [9]. The problem can be transformed to the following convex optimization problem:

$$\text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*),$$

*The research work of S. Xu was supported by NSF under grant ACR-0234270. E-mail: sxu2@uky.edu, URL: <http://www.csr.uky.edu/~sxu2>.

†Correspondent. The research work of J. Zhang was supported in part by NSF under grants CCR-0092532 and ACR-0202934, by DOE under grant DE-FG02-02ER45961, and by the University of Kentucky Faculty Research Support Program. E-mail: jzhang@cs.uky.edu, URL: <http://www.cs.uky.edu/~jzhang>.

$$\text{subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i, \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0, \\ C > 0, \end{cases}$$

where C determines the trade-off between the flatness of $f(x)$ and the amount up to which deviation larger than ε is tolerated. ξ_i and ξ_i^* are slack variables accounting for errors. Kernel functions are applied to map input space into some feature space. The commonly used kernel functions are: Polynomial kernel: $K(x, x_i) = (\langle x, x_i \rangle + c)^d$, RBF kernel: $K(x, x_i) = e^{-\frac{\|x - x_i\|^2}{2\sigma^2}}$, and Neural Network kernel: $K(x, x_i) = \tanh(\eta \langle x, x_i \rangle + \vartheta)$.

3 Matrix Feature Extraction

The features of a matrix used are directly related to the precision of the prediction system. We will compute the features of a matrix first and then use such information to predict its condition number. We have extracted about 60 features such as structure, value, bandwidth and diagonal related statistics. Yet there may be more useful features that we can extract in the future and add them into the feature space. For more detailed description on the matrix features, please see [11].

4 Feature Selection

Our experiments show that the accuracy of the condition number predicted based on all the features seems to be good. But such features may contain some redundant information. We apply three feature selection methods to remove such redundancy. Feature selection may also bring other benefits [1, 3, 7]: reduce the computation time, save memory space, remove noise and possibly optimize the prediction accuracy. For an online condition number prediction system, it is crucial to lower the response time and improve precision.

4.1 Correlation Correlation is one of the simplest feature selection methods. It computes the correlation of the input vector x_i and the target vector y as follows:

$$Cor_i = \frac{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)^2 \sum_{k=1}^m (y_k - \bar{y})^2}},$$

where the bar stands for an average over the index k . Correlation criteria can only detect linear dependencies between variables and target [1].

4.2 Weights from SVR There have been some feature selection methods based on the weights from the SVM classification model [3, 6]. Using the weights from SV-regression works in the same way. Like in neural

networks, the output prediction is of the form:

$$\text{predict}(x) = G\left(\sum_j w_j x_j + b\right),$$

where $G(x)$ is an activation function. A feature j with a larger weight w_j has more effect on the prediction than the feature with a smaller weight [6]. Shih *et al.* justified in [8] that the features with higher weights are more influential in determining the width of the margin. Thus $\|w\|^2$ is a suitable criterion for feature selection.

4.3 Combinational method It is known that a good preconditioner can improve the condition number of a matrix. We compare the condition number as well as the matrix features of the original matrix and the preconditioned matrix to find out which features contribute more to the improvement of the condition number. Such features have larger influence on the condition number and should be kept in feature selection. Assume we have l matrix examples, and k^A represents the condition number vector for all the original matrices, while k^M represents the condition number vector for all the preconditioned matrices. v_i^A is the vector of features for the i th original matrix, likewise, v_i^M is the vector of features for the i th preconditioned matrix. Then we can obtain the weight vector w_{cmp} to rank the features:

$$(4.1) \quad w_{cmp} = \sum_{i=1}^l (|k_i^A - k_i^M| * |v_i^A - v_i^M|).$$

w_{cmp} seems to be a reasonable criterion for feature selection, however, as we cannot successfully construct a useful preconditioner for all the general matrices, w_{cmp} is biased towards the features of the matrices that can be preconditioned. To remedy this problem, we propose to use the weight w_{comb} , which combines w_{cmp} and the weight from SVM regression w_{SVM} , as

$$(4.2) \quad w_{comb} = \text{nml}(w_{cmp}) + \text{nml}(w_{SVM}),$$

where the function $\text{nml}(x)$ normalizes vector x . Thus w_{comb} is the sum of the normalized w_{cmp} and w_{SVM} .

5 Experiments and Results

In this section, we report our experiments on the accuracy and response time of the condition number prediction methods. We use *SVM^{Light}* [4] for SVM regression. There are 277 matrices from Matrix Market [5] tested in the experiments. We use altogether 60 matrix features. The experiments are carried out on a SunBlade 150 workstation.

5.1 Accuracy First, we test how accurate the predicted condition numbers are, compared with the directly computed condition numbers. The accuracy is

obtained using 5-fold cross validation. The feature selection criteria used are correlation, w_{SVM} , w_{cmp} , and w_{comb} . We compare them together with the SVR without feature selection on three kernels: linear kernel, polynomial kernel and RBF kernel. Here all the feature selection methods choose 50% of the features.

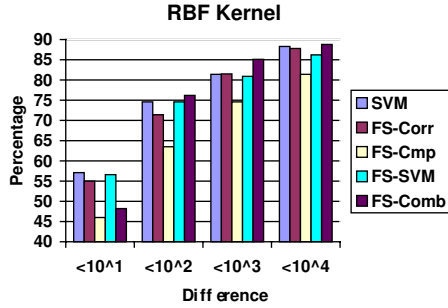


Figure 1: Comparison of accuracy with a RBF kernel.

Figure 1 shows the accuracy comparison using a RBF kernel ($\gamma = 0.1$). The figure illustrates the percentage of all matrices for which the relative differences between the computed values and the predicted values of the condition number are within 10^1 , 10^2 , 10^3 , 10^4 , respectively. For the RBF kernel, w_{cmp} does not work well. Its accuracy is the lowest. For all the other methods, we can safely say that more than 70% of the matrices have relative differences smaller than 10^2 . Among them, feature selection with w_{comb} works best for all the difference scales except the first one. Using feature selection with w_{comb} , 76.2% of the matrices have relative differences smaller than 10^2 . It has better accuracy than using SVR alone, although it only uses half of the features. Other feature selection methods can also obtain similar accuracy to that obtained from SVR without feature selection.

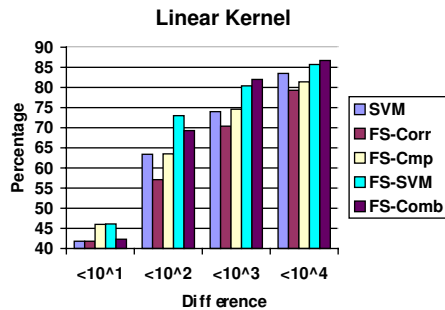


Figure 2: Comparison of accuracy with a linear kernel.

Figure 2 displays the accuracy comparison using a linear kernel. Here both feature selection criteria w_{comb} and w_{SVM} work well. Their accuracy are higher than

using SVR alone for all the difference scales. For the first two difference scales, w_{SVM} is better than w_{comb} , while for the last two, w_{comb} exceeds w_{SVM} . Correlation criterion performs worst for the linear kernel.

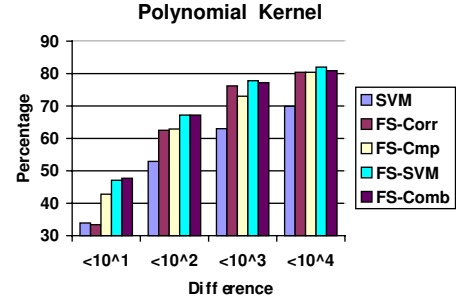


Figure 3: Comparison of accuracy with a polynomial kernel.

The accuracy obtained using a polynomial kernel ($d = 2$) is depicted in Figure 3. In this figure, all the feature selection methods obtain much better accuracy than without feature selection. Here w_{SVM} and w_{comb} perform similarly, both are better than the other methods.

Put these 3 figures together, we can see that the best accuracy is obtained using the RBF kernel, then the linear kernel, and the polynomial kernel does not seem to fit for this job. For the RBF kernel SVR without feature selection works rather well, thus the advantage of the feature selection methods over it is moot. However, for the polynomial kernel, when SVR without feature selection performs poorly, using feature selection methods can remarkably improve accuracy. Among the feature selection methods, the performance of the criteria using w_{comb} or w_{SVM} are consistently good.

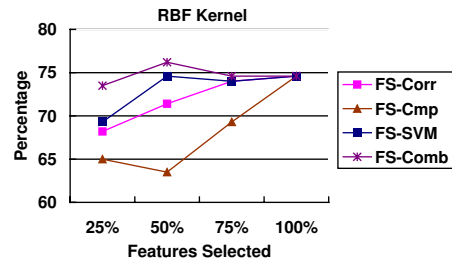


Figure 4: Comparison of accuracy with a RBF kernel with different percentage of features selected.

Next, we compare the performance of the feature selection methods with different amount of features selected. We test the accuracy using 25%, 50%, 75% of the features respectively, and make comparison with us-

ing 100% of the features, that is, running SVM without feature selection. Here we choose the percentage of matrices with relative condition number differences smaller than 10^2 as accuracy. Figure 4 illustrates the results obtained with a RBF kernel ($\gamma = 0.1$). Only feature selection with correlation has the property that with more features used the system becomes more accurate. For feature selection using w_{comb} and w_{SVM} , choosing 50% of the features seems to be optimal, with which they gain the highest accuracy. For feature selection using w_{cmp} it is an opposite story, choosing 50% of the features yields the worst results.

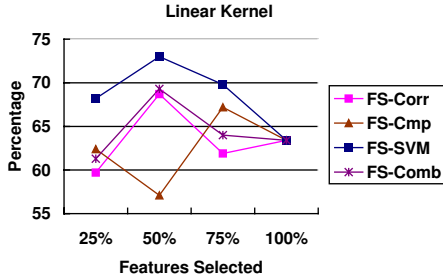


Figure 5: Comparison of accuracy with a linear kernel with different percentage of features selected.

For linear kernel, all feature selection criteria except w_{cmp} seem to find their optimized feature sets with 50% of the features. They get the best accuracy with 50% of the features. w_{cmp} , like using RBF kernel, performs the worst with 50% of the features (see Figure 5).

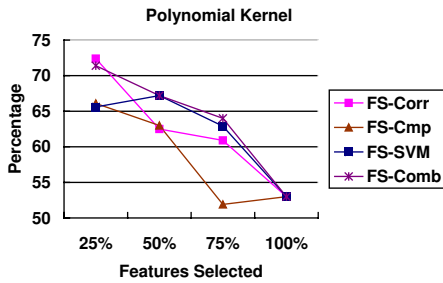


Figure 6: Comparison of accuracy with a polynomial kernel with different percentage of features selected.

In Figure 6, nearly all the feature selection criteria get their best accuracy with 25% of the features. Even with the only exception w_{SVM} , the accuracy obtained with 25% of the features is very close to its best accuracy obtained with 50% of the features. Figure 6 explains why polynomial kernel ($d = 2$) does not work as well as RBF kernel and linear kernel in Figures 1 - 3. In these 3 figures, 50% of the features are used. Thus almost all the feature selection methods find their optimized

feature sets for RBF kernel and linear kernel, but not for polynomial kernel. The feature selection methods work well with 25% of the features for polynomial kernel. Figure 6 also suggests that polynomial kernel is worthwhile to try, as the fewer features used, the less the response time.

5.2 Response Time Given a matrix, the time used to obtain the condition number is referred to as response time. The response time for the LAPACK method is the time to compute the condition number using LAPACK routines. The response time for the prediction method includes the time to compute matrix features and the time for prediction. Here we also compare the response time for prediction using the whole matrix features and using half of the features selected based on w_{SVM} .

Table 1: Average response time (in seconds).

T_{LAPACK}	$T_{predict-all}$	$T_{predict-FS}$
99.23	6.56	6.32

Table 1 shows the average response time for the 277 matrices used in our tests. T_{LAPACK} and $T_{predict}$ denote the CPU time (in seconds) used by LAPACK and the prediction method respectively. The prediction methods are 15 times faster than using LAPACK on average. 6 seconds is also an acceptable time for an online query system. Prediction with feature selection is only slightly faster than without feature selection. Using half of the features does not mean reducing the time cost in half. In our system, we usually compute a group of features in a function. Thus the time cost for calculating one feature using the function is the same as calculating all the features provided by the function. We need to do code optimization to make feature selection more beneficial in response time.

The prediction method is especially advantageous in response time for large size matrices. For example, in Table 2 the average response time for the 78 matrices with size larger than 2000 is around 6 minutes, while using the prediction methods, the response time is only about 20 seconds. $NumMat$ denotes the number of matrices. A matrix with size greater than 1000 is large in our experiments though this may not be true in reality. As LAPACK will run out of memory on our computers with matrices of size larger than 20000, we can only test matrices under this size for comparison.

Table 3 gives some examples of how much the prediction methods may exceed the LAPACK method in computation time. $LgCN_{LAPACK}$ represents the 10-based logarithm of condition number computed by LAPACK while $LgCN_{predict}$ is the predicted 10-based

Table 2: Average response time for large size matrices (in seconds).

Size	NumMat	T_{LAPACK}	$T_{predict-all}$	$T_{predict-FS}$
≥ 1000	119	227.22	15.17	14.62
≥ 2000	78	340.74	22.81	21.99

Table 3: Performance comparison for some large size matrices.

Name	n	Non-zeros	T_{LAPACK}	$T_{predict-all}$	$LgCN_{LAPACK}$	$LgCN_{predict-all}$
GEMAT11	4929	33108	236.7	2.9	8.057436	8.1577207
LNS_3937	3937	25407	2977.0	1.4	14.417698	14.517881
PSMIGR_1	3140	543160	2129.8	15.8	9.858676	9.9595671

logarithm of condition number. For instance, LAPACK uses 2977 seconds to compute the condition number of the matrix LNS_3937, the prediction method only needs 1.4 seconds. The relative difference of condition numbers obtained by these two methods is only about $10^{0.1}$. Although this difference may not be expected for all matrices, it is exactly our motivation for using prediction to obtain the condition numbers for general sparse matrices.

6 Concluding Remarks

In this paper we proposed a new approach to estimating the condition number of a matrix - predicting them from the matrix features. We use SVM regression with feature selection. The experiments show that around 75% of the matrices can be predicted with a relative difference from the computed condition number within 10^2 . The accuracy is low compared with direct computation or estimation, but it may be sufficient for those people who just want to know whether the matrix is *well-conditioned* or *ill-conditioned*. The advantage of the prediction method is that the response time is very low, especially for large size matrices. Thus it is desirable for an online condition number query. It is also fitted for the intelligent preconditioner recommendation system (IPRS) system [11, 12]. In IPRS, the condition number is used as one of the matrix features to predict the solvability of a matrix, thus it is crucial to obtain it with a low time cost. We also tried several feature selection methods. We designed a combinational feature selection criterion which uses both the weights from SVR and from comparison of a matrix and its preconditioned counterpart. The experimental results show that using feature selection can reduce the time cost and improve or maintain the accuracy. The combinational feature selection criterion is one of the best methods tested.

References

- [1] I. Guyon, A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(2003), 1157-1182.
- [2] N. J. Higham. Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. *ACM Trans. Math. Soft.*, 14, 1988, pp. 381-396.
- [3] R. Jin, H. Liu. Robust feature induction for support vector machines. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.
- [4] T. Joachims. Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges and A. Smola (ed.), MIT-Press, 1999.
- [5] <http://math.nist.gov/MatrixMarket/>
- [6] D. Mladenović, J. Brank, M. Grobelnik, N. Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of SIGIR'04*, Sheffield, UK, July, 2004.
- [7] L. C. Molina, L. Belanche, A. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan, 2002.
- [8] L. Shih, Y. Chang, J. Rennie, et al. Not too hot, not too cold: The Bundled-SVM is just right! In *Workshop on Text Learning (TextML-2002)*, Sydney, Australia, 2002.
- [9] A. J. Smola, B. Schölkopf. A tutorial on support vector regression. *NeuroCOLT Technical Report Series*, NC2-TR-1998-030, 1998.
- [10] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.
- [11] S. Xu, E. Lee, J. Zhang. An interim analysis report on preconditioners and matrices. Technical Report No. 388-03, Department of Computer Science, University of Kentucky, Lexington, KY, 2003.
- [12] S. Xu, E. Lee, J. Zhang. Designing and building an intelligent preconditioner recommendation system (a progress report). In *Abstracts of the 2003 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, Napa, CA, 2003.

Cluster Validity Analysis of Alternative Results from Multi-Objective Optimization

Yimin Liu Tansel Özyer Reda Alhajj Ken Barker

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
{liuyi, ozyer, alhajj, barker}@cpsc.ucalgary.ca

Abstract

This paper investigates validity analysis of alternative clustering results obtained using the algorithm named Multi-objective K-Means Genetic Algorithm (MOKGA). The reported results are promising. MOKGA gives the optimal number of clusters as a solution set. The achieved clustering results are then analyzed and validated under several cluster validity techniques proposed in the literature. The optimal clusters are ranked for each validity index. The approach is tested by conducting experiments using three well-known data sets. The obtained results for each dataset are compared with those reported in the literature to demonstrate the applicability and effectiveness of the proposed approach.

Keywords: clustering, gene expression data, genetic algorithms, multi-objective optimization, validity analysis.

1. Introduction

Traditional clustering algorithms, in general, do not produce alternative solutions, and most of them do not lead to the optimal number of clusters in the dataset that they work on. For example, hierarchical clustering method can get the heuristic overview of a whole dataset, but it cannot relocate objects that may have been 'incorrectly' grouped at an early stage. It cannot tell the optimal number of clusters nor give the non-dominated set. *K*-means needs the number of clusters as a predefined parameter, and it may give local optimal solutions because it is a local search from a random initial partitioning. SOM has the same disadvantage in that it requires the number of clusters be given *a priori*. Clearly, a clustering algorithm is needed to get the global pareto optimal solution set required to give users the best overview of the whole dataset according to the number of clusters and their quality. Further, it is required to get clustering results with the optimal number of clusters.

The main contribution of this paper is a new clustering approach that considers multiple objectives in the process and its application for clustering microarray and other datasets; we have tested our approach on three data sets. The proposed approach has two components. 1) Multi-objective *K*-means Genetic Algorithm (MOKGA) based clustering approach, which delivers a pareto optimal clustering solution set without taking weight values into account. Otherwise, users need to consider several trials weighting with different values until a satisfactory result is obtained. 2) Cluster validity analysis employed to evaluate the obtained

candidate optimal number of clusters, by applying some of the well-known cluster validity techniques, namely Silhouette, *C* index, Dunn's index, DB index, SD index and S-Dbw index, to the clustering results obtained from MOKGA. It provides one or more options for the optimal number of clusters.

The applicability and effectiveness of the proposed clustering approach and clustering validity analysis process are demonstrated by conducting experiments using three datasets: namely Fig2data, cancer (NCI60), and Leukaemia data sets available at Genomics Department of Stanford University, UCI machine learning repository.

The balance of the paper is organized as follows. Section 2 is devoted to the development of the new clustering system MOKGA. Section 3 reports experimental results to illustrate the applicability, performance and effectiveness of the system. Section 4 discusses advantages of the proposed approach in comparison with other existing methods. Section 5 is summary and conclusions.

2. The Proposed Clustering Approach

The proposed clustering approach named Multi-Objective Genetic *K*-means algorithm (MOKGA) has been developed on the basis of the Fast Genetic *K*-means Algorithm (FGKA) [8] and the Niche Pareto Genetic Algorithm [5].

After running the multi-objective *K*-means genetic algorithm, the Pareto-optimal front giving the optimal number of clusters as a solution set can be obtained. The system then analyzes the clustering results found with respect to six cluster validity techniques proposed and well documented in the literature, namely Silhouette, *C* index, Dunn's index, SD index, DB index, and S-Dbw index.

MOKGA uses a list of parameters to drive the evaluation procedure as in other genetic types of algorithms: including population size (the number of chromosomes), t_{dom} (the number of comparison set) representing the assumed non-dominated set, crossover, mutation probability, and the number of iterations for the execution of the algorithm to obtain the result. Subgoals can be defined as fitness functions, and instead of scalarizing them to find the goal as the overall fitness function with the user defined weight values, it is expected that the system can find the set of best solutions, i.e., the Pareto-optimal front. By using the specified formulas, at each generation, each chromosome in the population is evaluated and assigned a value for each fitness function.

Initially, the *current generation* is assigned to zero. Each chromosome takes the *number of clusters* parameter within the range 1 to the maximum number of clusters given by the user. A population with the specified number of chromosomes is created randomly by using the method described by Rousseeuw [11], where data points are randomly assigned to each cluster at the beginning and the rest of the points are randomly assigned to clusters. By using this method, we can avoid generating illegal strings, which means some clusters do not have any pattern in the string.

Using the current population, the next population is generated and the generation number is incremented by 1. During the next generation, the current population performs the Pareto domination tournament to get rid of the worst solutions from the population. Crossover, mutation, and the k-means operator [8] are then performed to reorganize each object's assigned cluster number. Finally, we will have twice the number of individuals after the Pareto domination tournament. The ranking mechanism used by Zitzler in [2] is applied to satisfy the elitism and diversity preservation. This halves the number of individuals.

The first step in the construction of the next generation is the selection using Pareto domination tournaments. In this step, two candidate items picked among (*population size* - t_{dom}) individuals participate in the Pareto domination tournament against the t_{dom} individuals for the survival of each chromosome in the population. In the selection part, t_{dom} individuals are randomly picked from the population. Two chromosome candidates are randomly selected from the current population except those in the comparison set (*population size* - t_{dom}), and each of the candidates is compared against each individual in the comparison set t_{dom} . If one candidate has larger total within-cluster variation fitness and larger number of cluster values than all the chromosomes in the comparison set, then it is dominated by the comparison set and will be deleted from the population permanently. Otherwise, it resides in the population.

After the Pareto domination tournament, the dominated chromosome is deleted from the population. The next step is crossover: one point crossover is used in the employed multi-objective genetic clustering approach. An index into the chromosome is selected and all data beyond that point in the chromosome are swapped between the two parent chromosomes. The resulting chromosomes are the children.

Mutation is applied to the population in the next step by randomly changing the values in the chromosome according to probability distribution.

The K-means operator is applied last to reanalyze each chromosome gene's assigned cluster value. It calculates the cluster centre for each cluster and re-assigns each gene to the closest cluster to each instance in the gene. Hence, K-means operator is used to speed up the convergence process by replacing a_n by a_n' , for $n=1$ to N simultaneously, where a_n' is the closest to object X_n in Euclidean distance.

After all operators have been applied, twice the number of individuals remains. After having the Pareto dominated tournament, we cannot give an exact number equal to the initial population size because at each generation randomly picked candidates are selected for the survival test leading to

the deletion of one or both, in case dominated. To half the number of individuals, the ranking mechanism proposed by Zitzler [2] is employed: individuals obtained after crossover, mutation, and K-means operator are ranked; and the best individuals are picked for population of the next generation.

The approach picks the first l individuals by considering the elitism and diversity among $2l$ individuals. Pareto fronts are ranked. Basically, we find the Pareto-optimal front and remove individuals of the Pareto-optimal front from the $2l$ set and place them in the population to run in the next generation. In the remaining sets, we get the first Pareto-optimal front and put it in the population and so on. Since we try to get the first l individuals, the last Pareto-optimal front may have more individuals required to complete the number of individuals to l . We handle the diversity automatically. We rank them and reduce the objective dimension into one. We then sum the normalized value of the objective functions for each individual. These are sorted in increasing order and each individual's total difference from its individual pairs is calculated. The individuals are placed in population based on decreasing differences, and then we keep placing from the top as many individuals as we need to complete the number of individuals in the population to l . The reason for doing this is to take the crowding factor into account automatically so that individuals occurring closer to others are unlikely to be picked.

This method was also suggested as a solution for the elitism and diversity for improvement in NSGA-II. For example, in order to get 20 chromosomes from the population, we select 10 chromosomes from the Pareto front, delete them from the current population, then get 8 chromosomes from the Pareto front in the current population, delete them from the population. Suppose that we have 6 in the current population, we take 2 chromosomes that have the largest distance to their neighbours using the ranking method mentioned above. Finally, if the maximum number of generations is reached, or the Pareto front remains stable for 50 generations, then the process is terminated; otherwise the next generation is performed.

3. Experimental Results

To evaluate the performance and efficiency of the proposed system consisting of the MOKGA clustering approach and conduct cluster validity analysis on the obtained alternative results, experiments were conducted on a computer with the following features: Pentium @4 with 2.00 GHz CPU, 512 MB RAM and running Windows XP. The system was implemented using MS Visual C++. The running platform is Microsoft Visual Studio.NET 2003.

Three widely gene expression datasets, namely Fig2data, cancer (NCI60), and Leukaemia have been used to test the performance and accuracy of the system. Fig2data data is used for clustering genes, while cancer (NCI60) and Leukaemia data sets are used for group cell samples.

3.1 Fig2data Dataset

Fig2data dataset is the time course of serum stimulation of primary human fibroblasts. It contains the expression data

for 517 genes of which expression changed substantially in response to serum. Each gene has 19 expressions ranging from 15 minutes to 24 hours [1, 6].

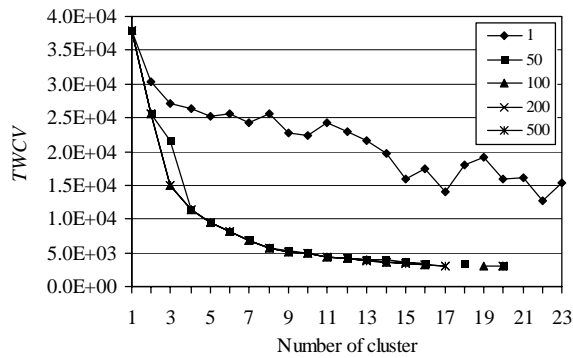


Figure 3.1 Pareto-fronts for Fig2data dataset

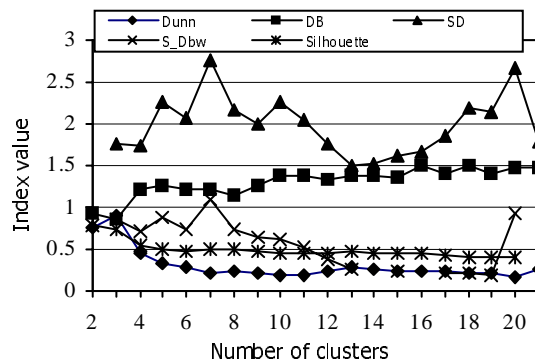


Figure 3.2 Fig2data dataset cluster validity results using Dunn, DB, SD, S_Dbw and Silhouette indexes

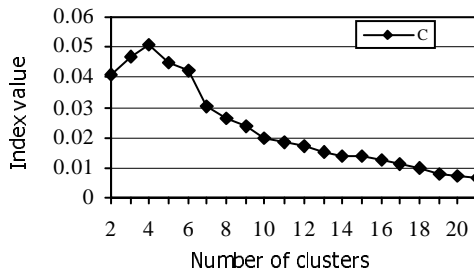


Figure 3.3 Fig2data dataset cluster validity results using C index

Lu *et al* [8] applied the Fast Genetic K-means Algorithm to Fig2data. They selected as their parameter setting: mutation probability = 0.01, population size = 50, and generation = 100. As a result, they obtained fast clustering process.

In our tests, MOKGA has been applied to Fig2data dataset. Experiments were conducted with the following parameters: population size = 150, t_{dom} (number of comparison set = 10) and crossover = 0.8, mutation = 0.005, gene mutation rate = 0.005, and threshold = 0.0001, which is applied to check if the population stops evolution after 50 generations and if the process needs to be stopped. The range of [1, 25] was picked to find the optimal number of clusters. The corresponding experimental results are reported

in Figure 3.1. They also show how the system converges to a Pareto optimal front.

Figure 3.2 and Figure 3.3 report validity results and reflect comparisons with the studies described elsewhere [6, 8]. The study by Iyer *et al* [6] show that the optimal number of clusters for Fig2data is 10. Consistently, results in this paper indicate that it ranks among the best ones for C index, and the number of 10 clusters is among the best for other indices as well. According to Maria *et al* [4], SD, S_Dbw, DB, Silhouette, and Dunn indices cannot handle properly arbitrarily shaped clusters, so they do not always give satisfactory results.

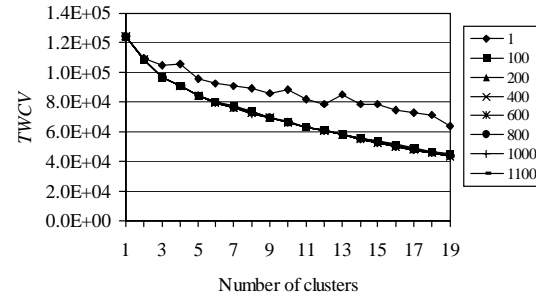


Figure 3.4 Pareto-fronts for Cancer dataset

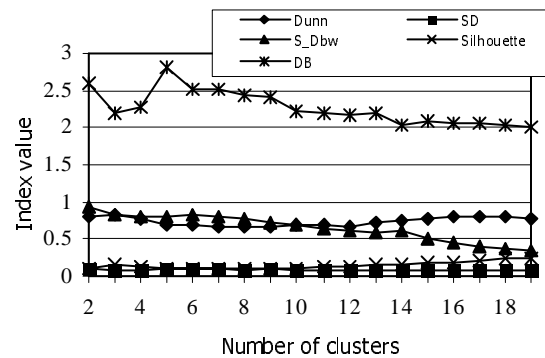


Figure 3.5 Cancer dataset cluster validity results using Dunn, DB, SD, S_Dbw and Silhouette indexes

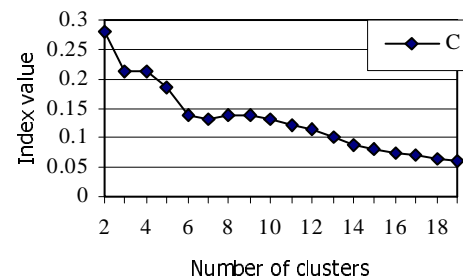


Figure 3.6 Cancer dataset cluster validity results using C index

3.2 Cancer (NCI60) dataset

NCI60 is a gene expression database for the molecular pharmacology of cancer. It contains 728 genes and 60 cell lines derived from cancers of colorectal, renal, ovarian, breast, prostate, lung, and central nervous system origin, leukaemias and melanomas. Growth inhibition is assessed

from changes in total cellular protein after 48 hours of drug treatment using a sulphorhodamine B assay. The patterns of drug activity across the cell lines provide information on mechanisms of drug action, resistance, and modulation [13].

The study by Scherf [13] uses an average-linkage algorithm and a metric based on the growth inhibitory activities of the 1,400 compounds for the cancer dataset. The authors observed 15 distinct branches at an average inter-cluster correlation coefficient of at least 0.3. In this method, the correlation parameter was used to control the clustering results. It might be hard to decide if it is an unsupervised clustering task.

In our tests, MOKGA has been run for the Cancer dataset with the following parameters: population size = 100, t_{dom} (number of comparison set = 10) and crossover = 0.8, mutation = 0.005, gene mutation rate = 0.005, and threshold = 0.0001, which is used to check if the population stops evolution for 50 generations and if the process needs to be stopped. The range of [1, 20] was picked to find the optimal number of clusters.

Changes in the Pareto-optimal front after running the algorithm are displayed in Figure 3.4. It demonstrates convergence to an optimal Pareto-optimal front.

Figures 3.5 and Figure 3.6 show the average results obtained. For the cancer (NCI60) dataset, we have 15 in the Pareto optimal front; this value also ranks the sixth for DB index, fifth for SD index and the fifth for C index. These are consistent with the results reported in [13]. Having some indexes values not good demonstrates the fact that index values are highly dependent on the shape of the clusters.

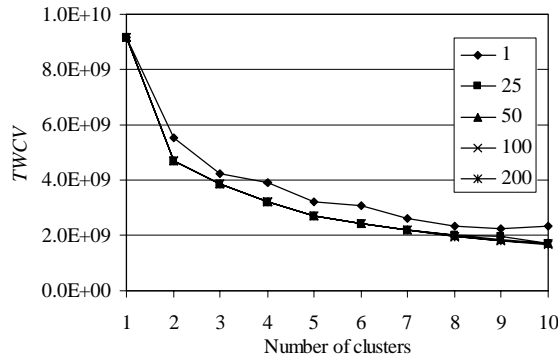


Figure 3.7. Pareto-fronts for Leukaemia dataset

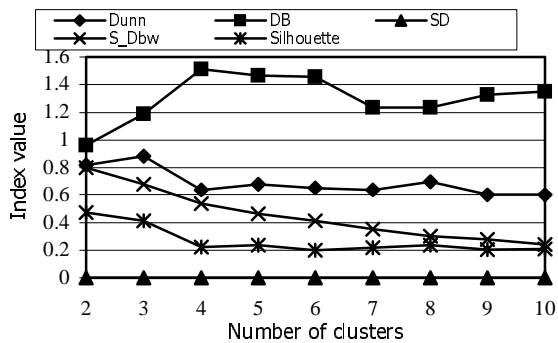


Figure 3.8 Leukemia dataset cluster validity results using Dunn, DB, SD, S_Dbw and Silhouette indexes

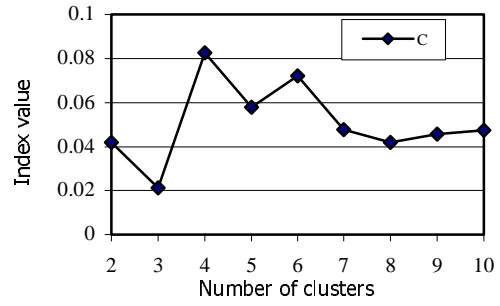


Figure 3.9 Leukemia dataset cluster validity results using C index

3.3 Leukaemia dataset

The third microarray dataset used in this paper is the Leukemia dataset, which has 38 acute leukemia samples and 50 genes. The purposes of the testing include clustering cell samples into groups and finding subclasses in the dataset.

The study by Golub *et al* [3] uses Self-Organizing Maps (SOMs) to group the Leukemia dataset. In this approach, the user specifies the number of clusters to be identified. SOM finds an optimal set of centroids around which the data points appear to aggregate. It then partitions the data set with each centroid defining a cluster consisting of the data points nearest to it. Golub [3] got two clusters acute myeloid leukemia (AML) and acute lymphoblastic leukaemia (ALL), as well as the distinction between B-cell and T-cell ALL, which means that the optimal number of clusters is 2 or 3 (with subclasses).

The proposed genetic algorithm-based approach has been run for the Leukemia dataset with the following parameters: population size = 100, t_{dom} (number of comparison set = 10) and crossover = 0.8, mutation = 0.005, gene mutation rate = 0.005, and threshold = 0.01, which is used to check if the population stops evolution for 50 generations and if the process needs to be stopped. The range of [1, 10] was picked for finding the optimal number of clusters. Changes in the Pareto-optimal front are displayed in Figure 3.7. It demonstrates how the system converges to an optimal Pareto-optimal front.

The Leukaemia dataset clustering results shown in Figure 3.8 and Figure 3.9 indicate the same conclusions reported in [3] by Golub *et al*. They also indicate that 2 (AML and ALL) is the best number of clusters after the validity analysis with Dunn index, DB index, SD index, and Silhouette and 3 (AML, B-cell ALL and T-cell ALL) is the second best. C index shows that 2 is the best cluster number and 3 is the second. It can be seen from Figure 3.8 that S_Dbw is an exception. SD index gives good values, but S_Dbw does not. This indicates that the inter-cluster density for number of clusters taken 2 and 3 is not high for the 38 samples. Experimental results in this paper also indicate that S_Dbw index is not suitable to test small datasets with fewer than 40 instances.

4. General Evaluation and Comparison

In this section, the MOKGA system is compared with other methods on basis of the results obtained for the same datasets. For instance, according to [6], Fig2data has 10

clusters. The proposed approach gave the same result using C index clustering validity method. Cancer data has 15 clusters according to the result in [13]. MOKGA produces the same result using the DB index. The optimal number of clusters of Leukemia dataset is 2 or 3. MOKGA reported the same results using Dunn, DB, SD, and Silhouette indexes.

Since MOKGA has been developed on the basis of Fast Genetic K-mean Algorithm (FGKA) [8] and Niche Pareto Genetic Algorithm (NPGA), MOKGA and FGKA share many features: both are evolutionary algorithms; they have the same mutation and K-mean operators; and they both use the Total Within-Cluster Variation (TWCV) for the fitness value evaluation.

According to the results, MOKGA and FGKA got similar TWCV values, MOKGA obviously need more generations to get the stable state, this might be because MOKGA is optimizing chromosomes with different number of clusters altogether.

MOKGA has some advantages over FGKA and GKA: it can find the Pareto optimal front, which allows us to get an overview of the entire clustering possibilities and to get the optimal clustering results in one run; it does not need the number of clusters as a parameter, which is very important because clustering is an unsupervised task, and we usually do not have any idea about the number of clusters before the clustering of gene expression data. These two issues are real concerns for FGKA, GKA and most of the other clustering algorithms.

Both MOKGA and K-means Algorithm minimize the overall within-cluster dispersion by iterative reallocation of cluster members. MOKGA has some advantages over K-means algorithm: it can find the Pareto optimal front; it does not need the number of clusters as a parameter; MOKGA can find global optimal solutions using mutation and crossover operators. MOKGA combines both the advantages of genetic algorithm and advantages of the K-means algorithm: by using GA operators it can get global optimal solutions, and by using K-means operators MOKGA can get solutions much faster.

5. Summary and Conclusions

The MOKGA approach proposed in this paper has been developed on the basis of the Niche Pareto optimal and fast K-means genetic algorithm. By using MOKGA, it is aimed at finding the Pareto-optimal front sought to help the user to obtain several alternative solutions at once. Then, cluster validity index values are evaluated for each Pareto-optimal front value, which is considered the optimal *number of clusters* value. MOKGA overcomes the difficulty of determining the weight of each objective function taking part in the fitness when dealing with this multiple objectives problem. Otherwise, the user would have been expected to do many trials with different weighting of objectives as in traditional genetic algorithms. This method also gives the users an overview of different numbers of clusters, which may help them in finding subclasses and optimal number of clusters in a single run, whereas traditional methods like SOM, K-means, Hierarchical clustering algorithms and GCA can not find optimal number of clusters, or need it as a

prespecified parameter. MOKGA is less susceptible to the shape or continuity of the Pareto front. It can easily deal with discontinuous or concave Pareto fronts. These two issues are real concerns for mathematical programming techniques, like model-based approaches such as Bayesian method and mixed model-based clustering algorithms.

References

- [1] K. Chen, L. Liu, "Validating and Refining Clusters via Visual Rendering Gene Expression Data of the Genomic Resources," *Proc. of IEEE International Conference on Data Mining*, pp.501-504, 2003.
- [2] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," *Doctoral Thesis* ETH NO. 13398, Zurich: Swiss Federal Institute of Technology, 1999.
- [3] T. R. Golub, et al, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, 286, pp.531-537, 1999.
- [4] M. Halkidi, Y. Batistakis and M. Vazirgiannis, "Clustering Validity Checking Methods: Part II," *SIGMOD Record*, Vol.31, No.3, pp.19-27, 2002.
- [5] J. Horn, N. Nafpliotis and D. E. Goldberg, "A Niche Pareto Genetic Algorithm for Multiobjective Optimization," *Proc. of IEEE CEC*, Vol.1, pp.82-87, Piscataway, NJ, 1994.
- [6] V.R. Iyer, et al, "The transcriptional program in the response of human fibroblasts to serum," *Science*, 283(5398), pp.83-7, 1999.
- [7] Y. Liu, T. Özger, R. Alhajj and K. Barker, "Multi-objective Genetic Algorithm based Clustering Approach and Its Application to Gene Expression Data," *Proc. of ADVIS*, Springer-Verlag, Oct. 2004.
- [8] Y. Lu, et al, "FGKA: A Fast Genetic K-means Clustering Algorithm," *Proc. of ACM Symposium on Applied Computing*, Cyprus, pp.162-163, 2004.
- [9] U. Möller, D. Radke, F. Thies, Testing the significance of clusters found in gene expression data. *Proc. of European Conference on Computational Biology*, Paris, pp.26-30, 2003.
- [10] T. Özger, Y. Liu, R. Alhajj and K. Barker, "Validity Analysis of Clustering Obtained Using Multi-Objective Genetic Algorithm," *Proc. of ISDA*, Springer-Verlag, Hungary, Aug. 2004.
- [11] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *Journal of Comp App. Math*, Vol.20, pp.53-65, 1987.
- [12] E. H. Ruspini, "Numerical methods for fuzzy clustering," *Inform. Science*, vol.2, pp.319-350. 1970.
- [13] U. Scherf, et al, "A Gene Expression Database for the Molecular Pharmacology of Cancer," *Nat Genet*, Vol.24, pp.236-44, 2000.
- [14] B. Stein, S. Meyer and F. Wissbrock, "On Cluster Validity and the Information Need of Users," *Proc. of the International Conference on Artificial Intelligence and Applications*, Spain, Sep. 2003.
- [15] W. Shannon, R. Culverhouse J. Duncan, "Analyzing microarray data using cluster analysis," *Pharmacogenomics*, Vol.4, No.1, pp.41-52, 2003.

ClosedPROWL: Efficient Mining of Closed Frequent Continuities by Projected Window List Technology

Kuo-Yu Huang, Chia-Hui Chang and Kuo-Zui Lin*

Abstract

Mining frequent patterns in databases is a fundamental and essential problem in data mining research. A continuity is a kind of causal relationship which describes a definite temporal factor with exact position between the records. Since continuities break the boundaries of records, the number of potential patterns will increase drastically. An alternative approach is to mine closed frequent continuities. Mining closed frequent patterns has the same power as mining the complete set of frequent patterns, while substantially reducing redundant rules to be generated and increasing the effectiveness of mining. In this paper, we propose a method called projected window list technology for the mining of frequent continuities. We present a closed frequent continuity mining algorithm, ClosedPROWL. Experimental result shows that our algorithm is more efficient than previously proposed algorithms.

Temporal databases, association rules, Mining methods and algorithms

1 Introduction

Mining frequent patterns in databases is a fundamental and essential problem in data mining. Over the past few years, a considerable number of studies have been made in frequent pattern mining. There are various directions in pattern mining, such as frequent itemsets, sequential patterns, frequent episodes [4], periodic patterns [3], frequent continuities [2, 5], etc. The fundamental paradigm of association rule mining (e.g. frequent itemsets) identifies correlations between objects in transaction databases (market baskets) without taking any ordering of the objects into account. Such rules can be useful for decisions concerning product pricing, promotions, store layout and many others.

In addition to the mining tasks on transaction databases, there are also works on temporal association mining, which concerns the occurrences of events along time, e.g. frequent episodes, periodic patterns, frequent

continuities, etc. To distinguish these two kinds of mining tasks, prior researches [5] use the term intra-transaction associations for the former mining tasks and inter-transaction association for the latter ones. As suggested in [5], inter-transaction associations are better for trend prediction than intra-transaction associations. For instance, the investors may be more interested in a rule like “When the price of stock TSMC goes up for two consecutive days, the price of stock UMC will go up with 60% probability on the third day.” This kind of the temporal association with definite temporal relationships between stocks can be envisioned as a tool for describing and forecasting of the behavior of temporal databases.

The above rule can be generated from frequent continuities [2], an inter-transaction association which correlates the definite time with each object. The problem is first introduced by Tung et al in [5], where an algorithm called FITI (First Intra Then Inter) is proposed for mining frequent continuities. FITI is a three-phase algorithm. The first phase discovers intra-transaction itemsets. The second phase transforms the original database into another database to facilitate the mining of inter-transaction associations. The third phase follows the Apriori principle to perform a level-wise mining. In order to make search quickly, FITI is devised with several hashing structures for pattern searching and generation. Similar to Apriori-like algorithms, FITI could generate a huge number of candidates and require several scans over the whole database to check which candidates are frequent. Therefore, Huang et al. introduce a projected window list (PROWL) technique [2] which enumerates new frequent continuities by concatenating frequent items in the time lists of the following time slots (called the projected window list) of an existent frequent continuity. PROWL utilizes memory for storing both vertical and horizontal formats of the database, therefore it discovers frequent continuities without candidate generation. Note that PROWL was designed to mining frequent continuity from a sequence of events instead of a sequence of eventsets.

Since inter-transaction associations break the boundaries of transactions, the number of potential con-

*The authors are with the department of Computer Science and Information Engineering, National Central University, Taiwan. Email: want@db.csie.ncu.edu.tw, chia@csie.ncu.edu.tw, kuozui@db.csie.ncu.edu.tw

tinuities and the number of rules will increase drastically. This reduces not only efficiency but also effectiveness since users have to sift through a large number of mined rules to find useful ones. Although compressed continuity (and the corresponding algorithm COCOA) [1] reduces the number of continuities, they are not the minimum set that can represent all continuities. They are simply continuities that are composed of closed frequent itemsets. Therefore, we focus on discovering **closed frequent continuities** which have no proper super-continuity with the same support in databases.

What are super-continuity and sub-continuity? Given two continuities $P = [p_1, p_2, \dots, p_u]$ and $P' = [p'_1, p'_2, \dots, p'_v]$, we say that P is a **super-continuity** of P' (i.e., P' is a **sub-continuity** of P) if and only if, for each non-* pattern p'_j ($1 \leq j \leq v$), $p'_j \subseteq p_{j+o}$ is true for some integer o . The integer o is also called the offset of P . For example, continuity $P = [AC, E, BD]$ is a super-continuity of continuity $P' = [E, B, *]$, since the pattern E (B , resp.) is a subset of the (BD , resp.) with offset 1. On the contrary, continuity $P'' = [E, B, AC]$ is not a sub-continuity of P , since P'' can not map to P with a fixed offset. It is worth mentioning that if we don't consider the offset in the continuity matching, the continuity P' will not be a sub-continuity of continuity P .

The problem of closed frequent continuity mining is similar to frequent continuity mining [2], except for the closed constraint. Mining closed frequent continuities has the same power as mining the complete set of frequent continuities, while substantially reduce redundant rules to be generated and increase the effectiveness of mining. Therefore, the problem is formulated as follows: given a minimum support level *minsup* and a maximum time window bound *maxwin*, our task is to mine all closed frequent continuities from temporal database with support greater than *minsup* and window bound less than *maxwin*.

2 The ClosedPROWL Algorithm

Similar to FITI [5] and COCOA [1], the ClosedPROWL algorithm also consists of three phases. The first phase involves the mining of closed frequent intra-transaction itemsets. The idea is based on the observation that a closed continuity is composed of only closed itemsets and don't care characters (see Theorem 4.3). Since the third phase of the algorithm requires the time lists of each intra-transaction itemset, this phase is mined using a vertical mining algorithm, CHARM [6], for closed frequent itemsets mining.

The second phase is database transformation, where it encodes each closed frequent itemset (abbreviated C.F.I.) with a unique **ID**. Next, based on the time

lists of the C.F.I together with the encoding table, we construct a recovered horizontal database.

In the third phase, we discover all closed frequent continuities from the recovered horizontal database by concatenating a frequent continuity with its closed frequent itemsets using depth first enumeration. For ease exposition, we first define the projected window list below.

DEFINITION 2.1. *Given the time list of a continuity P , $P.timeList = \{t_1, t_2, \dots, t_k\}$ in the database D , the **projected window list (PWL)** of P with offset d is defined as $P.PWL_d = \{w_1, w_2, \dots, w_k\}$, $w_i = t_i + d$ for $1 \leq i \leq k$. Note that a time slot w_i is removed from the projected list if w_i is greater than $|D|$, i.e. $w_i \leq |D|$ for all i .*

For each frequent 1-continuity P , or equivalently closed frequent itemset (C.F.I.), the mining steps are as follows:

1. Calculate the projected window list (PWL) with offset 1 from $P.timeList$. Find all frequent C.F.I. in $P.PWL_1$ by examining the recovered horizontal database.
2. Then apply *subitemset-pruning* strategy to remove unnecessary extensions.
3. For each remaining C.F.I. x , generate a new frequent continuity $P \cdot [x]$. Step 1 to 3 are applied recursively to find all frequent continuities until the size of $(P \cdot [x]).PWL_1$ becomes less than the required counts specified by *minsup* or the window of a continuity is greater than *maxwin*.
4. Finally, we apply *subcontinuity-checking* to remove non-closed frequent continuities.

Starting from any 1-continuity P , all frequent continuities having prefix P can be generated by concatenating P with a closed frequent eventset in $P.PWL$ or the don't care character without candidate generation. As with the PROWL algorithm [2] and COCOA [1], the timelists (vertical format) record the locations of a continuity, while the recovered database (horizontal format) is used for fast access to see what itemsets are frequent enough to extend current frequent continuity. What makes ClosePROWL different is Step 2 and 4, where we incorporate the property of closed continuities to reduce the search space.

Sub-itemset pruning: For two C.F.I. x and y in the project window list of a continuity P , if $Sup(P \cdot [x]) = Sup(P \cdot [y])$, the sub-itemset pruning works as following properties:

Mining Task	Phase I	Phase III	Algorithm
Continuity	Frequent Itemset	FITI-3	FITI
		PROWL	PROWL+
Compressed	Closed Frequent Itemset	FITI-3	ComFITI
		PROWL	COCOA
Closed		PROWL+Pruning	ClosedPROWL

Table 1: Comparison of various mining tasks

1. If $x \subset y$, then remove x since all extensions of $P \cdot [x]$ must not be closed.
2. If $x \supset y$, then remove y since all extensions of $P \cdot [y]$ must not be closed.
3. If $x.timelist = y.timelist$ and neither $x \subset y$ nor $x \supset y$, then remove both x and y , since all extensions of $P \cdot [x]$ and $P \cdot [y]$ must not be closed.

In order to make the pruning efficient, we devise a hash structure, PHTab (prune header table) with $PHsize$ buckets. All C.F.I.s with the same support counts are hashed into the same bucket. Each entry in the same bucket records a frequent ID x of the current continuity P , the time list of $P \cdot [x]$, and the support count of $P \cdot [x]$. The comparison of two frequent C.F.I. x and y in the projected window lists of a continuity P is restricted to the frequent IDs in the same buckets with the same support.

The sub-itemset pruning technique removes the non-closed sub-continuity of closed frequent continuities with zero offset since the pruning is invoked within a local search of a continuity. For those sub-continuities of closed frequent continuities with non-zero offset, they can still be generated in the mining process. Therefore, we need a checking step (Step 4) to remove non-closed continuities. Again, a hash structure, FCTab (frequent continuity table), is devised to facilitate efficient sub-continuity checking using the following as the hashing function:

$$(2.1) \quad bkNum = Sup(P) \% BucketSize.$$

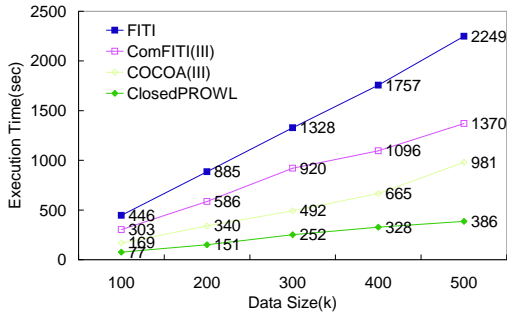
The correctness of the pruning technique and the overall algorithm can be proven by the theorems in Appendix A and B respectively.

3 Experiments

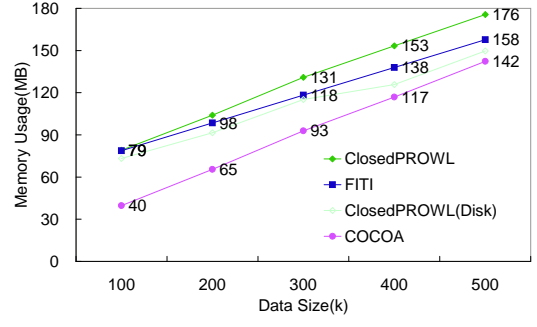
In this section, we report the performance study of the proposed algorithm on synthetic data. Since the three phases of the proposed algorithms have good correspondence with three phases of the FITI algorithm, it is possible to mine various continuities by combining various Phase Is with Phase IIIs of FITI (called FITI-3) and

PROWL. We already know the mining process of FITI. Combining frequent itemset mining with Phase III of ClosedProwl without pruning produces the same result with FITI. If we mine closed frequent itemsets at Phase I and apply FITI-3 or PROWL, we will get compressed frequent continuities. We call the algorithms ComFITI and COCOA, respectively. Finally, the closed frequent itemset mining at Phase I combined with PROWL and the pruning strategies at Phase III results the mining of ClosedPROWL for frequent closed continuities. The combinations are shown in Table 1. We compare the five algorithms using synthetic data.

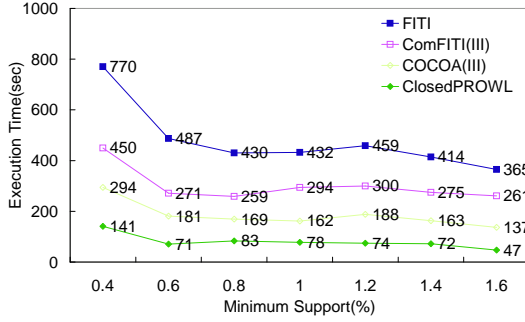
The synthetic data sets which we used for our experiments were generated using the generator described in [1]. We start by looking at the performance of ClosedPROWL with default parameter $minsup = 0.6\%$ and $maxwin = 5$. Figure 1(a) shows the scalability of the algorithms with varying database size. ClosedPROWL is faster than FITI (by a magnitude of 5 for $|D| = 500K$). The scaling with database size was linear. Therefore, the scalability of the projected window lists technique is proved. Another remarkable result is that COCOA performs better than ComFITI for the same mining task (compressed frequent continuity mining). The reason for the considerable execution time of FITI and ComFITI is that they must count the supports of all candidate continuities. The memory requirement of the algorithms with varying database size is shown in Figure 1(b). In this case, the number of frequent continuities and closed frequent continuities are 13867 and 1183 respectively. The compression rate ($\#$ of closed frequent continuities / $\#$ of frequent continuities) is about 9%. As the data size increases, the memory requirement of ClosedPROWL, COCOA and FITI increases as well. However the memory usages of FITI and ClosedPROWL are about the same at $|D| = 100K$ and the difference is only 18MB at $|D| = 500K$, with an original database of 12.2 MB. Since ClosedPROWL requires additional memory to maintain frequent continuities ($FCTab$), we modify the algorithm to disk-resident ClosedPROWL (labelled ClosedPROWL(Disk)). As illustrated in Figure 1(b), the memory requirement of the ClosedPROWL(Disk) is thus less than FITI but more



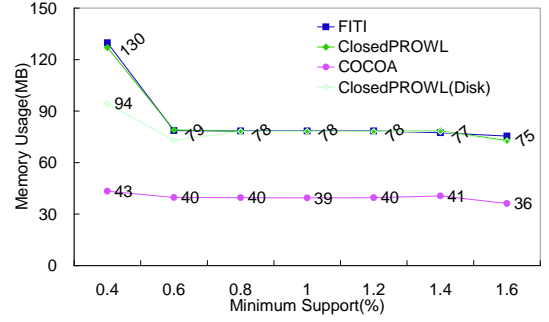
(a) Execution Time v.s. Data Size



(b) Memory Usage v.s. Data Size



(c) Execution Time v.s. $minsup$



(d) Memory Usage v.s. $minsup$

Figure 1: Performance comparison I

than COCOA for subitemset pruning (*PHTab*).

The runtime and memory usage of FITI and ClosedPROWL on the default data set with varying minimum support threshold, $minsup$, from 0.4% to 1.6% are shown in Figures 1(c) and (d). Clearly, ClosedPROWL is faster and more scalable than both FITI and ComFITI with the same memory requirements (by a magnitude of 5 and 3 for $minsup = 0.4\%$ respectively), since the number of frequent continuities grows rapidly as the $minsup$ diminishes. ClosedPROWL and ClosedPROWL(Disk) require 129MB and 94MB at the $minsup = 0.4\%$, respectively. Thus maintaining closed frequent continuities (*FCTab*) in ClosedPROWL needs 35MB main memory approximately. Meanwhile, we can observe that the pruning strategies of ClosedPROWL increase the efficiency considerably (by a magnitude of 2) through the comparison between ClosedPROWL and COCOA in Figure 1(c). In summary, projected window list technique is more efficient and more scalable than Apriori-like, FITI and ComFITI, especially when the number of frequent continuities becomes really very large.

4 Conclusion

In this paper, we propose an algorithms for the mining of closed frequent continuities. We show that the three-phase design lets the projected window list technique, which was designed for sequences of events, also applicable to general temporal databases. The proposed algorithm uses both vertical and horizontal database formats to reduce the searching time in the mining process. Therefore, there is no candidate generation and multi-pass database scans. The main reason that projected window list technique outperforms FITI/ComFITI is that it utilizes memory for fast computation. This the same reason that later algorithms for association rule mining outperform Apriori. Even so, we have demonstrated that the memory usage of our algorithms are actually more compact than the FITI/ComFITI algorithm. Furthermore, with subitemset pruning and subcontinuity checking, ClosedPROWL successfully discovered efficiently all closed continuities. For future work, maintaining and reusing old patterns for incremental mining is an emerging and important research. Furthermore, using continuities in prediction is also an interesting issue.

Acknowledgements This work is sponsored by National Science Council, Taiwan under grant NSC93-2213-E-008-023.

References

- [1] K. Y. Huang, C. H. Chang, and K.-Z. Lin. Cocoa: An efficient algorithm for mining inter-transaction associations for temporal database. In *Proceedings of 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, volume 3202 of *Lecture Notes in Computer Science*, pages 509–511. Springer, 2004.
- [2] K. Y. Huang, C. H. Chang, and K.-Z. Lin. Prowl: An efficient frequent continuity mining algorithm on event sequences. In *Proceedings of 6th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'04)*, volume 3181 of *Lecture Notes in Computer Science*, pages 351–360. Springer, 2004.
- [3] K. Y. Huang and C. H. Chang. Smca: A general model for mining synchronous periodic pattern in temporal database. *IEEE Transaction on Knowledge and Data Engineering (TKDE)*, 2005. To Appear.
- [4] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in event sequences. *Data Mining and Knowledge Discovery (DMKD)*, 1(3):259–289, 1997.
- [5] A. K. H. Tung, H. Lu, J. Han, and L. Feng. Efficient mining of intertransaction association rules. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(1):43–56, 2003.
- [6] M. J. Zaki and C. J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of 2nd SIAM International Conference on Data Mining (SIAM 02)*, pages 457–473, 2002.

Appendix A

LEMMA 4.1. Let $P = [p_1, p_2, \dots, p_w]$ and $Q = [q_1, q_2, \dots, q_w]$ be two frequent continuities and $P.timelist = Q.timelist$. For any frequent continuity U , if $P \cdot U$ is frequent, then $Q \cdot U$ is also frequent, vice versa.

THEOREM 4.1. Let $P = [p_1, p_2, \dots, p_w, p_{w+1}]$ and $Q = [p_1, p_2, \dots, p_w, p'_{w+1}]$ be two continuities. If $p_{w+1} \subset p'_{w+1}$ and $Sup(P) = Sup(Q)$, then all extensions of P must not be closed.

Proof. Since p_{w+1} is a subset of p'_{w+1} , wherever p'_{w+1} occurs, p_{w+1} occurs. Therefore, $P.timelist \supseteq Q.timelist$. Since $Sup(P) = Sup(Q)$, the equal sign holds, i.e. $P.timelist = Q.timelist$. For any extension $P \cdot U$ of P , there exists $Q \cdot U$ (Lemma 4.1), such that $Q \cdot U$ is a super-continuity of $P \cdot U$, and $(P \cdot U).timelist = P.PWL_{|U|} \cap U.timelist = Q.PWL_{|U|} \cap U.timelist = (Q \cdot U).timelist$. Therefore, $P \cdot U$ is not a closed continuity.

THEOREM 4.2. Let $P = [p_1, p_2, \dots, p_w, p_{w+1}]$ and $Q = [p_1, p_2, \dots, p_w, p'_{w+1}]$ be two continuities. If $p_{w+1} \subset p'_{w+1}$ and $Sup(P) = Sup(Q)$, then all extensions of P must not be closed.

Proof. Consider the continuity $U = [p_1, p_2, \dots, p_w, p_{w+1} \cup p'_{w+1}]$. $U.timelist = P.timelist \cap Q.timelist$. Since $P.timelist = Q.timelist$, we have $U.timelist = P.timelist = Q.timelist$. Using Theorem 4.2, all extensions of P and Q can not be closed because $Sup(U) = Sup(P) = Sup(Q)$.

Appendix B

We also prove the correctness of the ClosedPROWL algorithm below.

LEMMA 4.2. The time list of a continuity $P = [p_1, p_2, \dots, p_w]$ is $P.timelist = \bigcap_{i=1}^w p_i.PWL_{w-i}$.

We define the closure of an itemset p , denoted $c(p)$, as the smallest closed set that contains p . If p is closed, then $c(p) = p$. By definition, $Sup(p) = Sup(c(p))$ and $p.timelist = c(p).timelist$.

THEOREM 4.3. A closed continuity is composed of only closed itemsets and don't care characters.

Proof. Assume $P = [p_1, p_2, \dots, p_w]$ is a closed continuity, and some of the p_i s are composed of non-closed itemsets. Consider the continuity $CP = [c(p_1), c(p_2), \dots, c(p_w)]$, $CP.timelist = \bigcap_{i=1}^w c(p_i).PWL_{w-i} = \bigcap_{i=1}^w p_i.PWL_{w-i} = P.timelist$. Therefore, P is not a closed continuity. We thus have a contradiction to the original assumption that P is a closed continuity and thus conclude that “all closed continuities $P = [p_1, p_2, \dots, p_w]$ are composed of only closed itemsets and the don't-care characters”.

THEOREM 4.4. The ClosedPROWL algorithm generates all closed frequent continuities.

Proof. First of all, the anti-monotone property “if a continuity is not frequent, all its super-continuities must be infrequent” is sustained for closed frequent continuities. According to Theorem 4.3, the search space composed of only closed frequent itemset covers all closed frequent continuities. ClosedPROWL's search is based on a complete set enumeration space. The only branches that are pruned as those that do not have sufficient support. The sub-itemset pruning only removed non-closed continuities (Theorem 4.2). Therefore, ClosedPROWL correctly identifies all closed frequent continuities. On the other hand, sub-continuity checking remove non-closed frequent continuities. Therefore, the ClosedPROWL algorithm generates all and only closed frequent continuities.

Three Myths about Dynamic Time Warping Data Mining

Chotirat Ann Ratanamahatana

Eamonn Keogh

Department of Computer Science and Engineering

University of California, Riverside

Riverside, CA 92521

{ ratana, eamonn }@cs.ucr.edu

Abstract

The Dynamic Time Warping (DTW) distance measure is a technique that has long been known in speech recognition community. It allows a non-linear mapping of one signal to another by minimizing the distance between the two. A decade ago, DTW was introduced into Data Mining community as a utility for various tasks for time series problems including classification, clustering, and anomaly detection. The technique has flourished, particularly in the last three years, and has been applied to a variety of problems in various disciplines.

In spite of DTW's great success, there are still several persistent "myths" about it. These myths have caused confusion and led to much wasted research effort. In this work, we will dispel these myths with the most comprehensive set of time series experiments ever conducted.

Keywords

Dynamic Time Warping, Data Mining, Experimentation.

1 Introduction

In recent years, classification, clustering, and indexing of time series data have become a topic of great interest within the database/data mining community. The Euclidean distance metric has been widely used [9], in spite of its known weakness of sensitivity to distortion in time axis [6]. A decade ago, the Dynamic Time Warping (DTW) distance measure was introduced to the data mining community as a solution to this particular weakness of Euclidean distance metric [2]. This method's flexibility allows two time series that are similar but locally out of phase to align in a non-linear manner. In spite of its $O(n^2)$ time complexity, DTW is the best solution known for time series problems in a variety of domains, including bioinformatics [1], medicine [4], engineering, entertainment [22], etc.

The steady flow of research papers on data mining with DTW became a torrent after it was shown that a simple lower bound allowed DTW to be indexed with no false dismissals [6]. The lower bound requires that the two sequences being compared are of the same length, and that

the amount of warping is constrained. This work allowed practical applications of DTW, including real-time query-by-humming systems [22], indexing of historical handwriting archives [17], and indexing of motion capture data [5].

In spite of the great success of DTW in a variety of domains, there still are several persistent myths about it. These myths have caused great confusion in the literature, and led to the publication of papers that solve apparent problems that do not actually exist. The three major myths are:

Myth 1: The ability of DTW to handle sequences of different lengths is a great advantage, and therefore the simple lower bound that requires different-length sequences to be reinterpolated to equal length is of limited utility [10][19][21]. In fact, as we will show, there is no evidence in the literature to suggest this, and extensive empirical evidence presented here suggests that comparing sequences of different lengths and reinterpolating them to equal length produce no statistically significant difference in accuracy or precision/recall.

Myth 2: Constraining the warping paths is a necessary evil that we inherited from the speech processing community to make DTW tractable, and that we should find ways to speed up DTW with no (or larger) constraints[19]. In fact, the opposite is true. As we will show, the 10% constraint on warping inherited blindly from the speech processing community is actually too large for real world data mining.

Myth 3: *There is a need (and room) for improvements in the speed of DTW for data mining applications.* In fact, as we will show here, if we use a simple lower bounding technique, DTW is essentially $O(n)$ for data mining applications. At least for CPU time, we are almost certainly at the asymptotic limit for speeding up DTW.

In this paper, we dispel these DTW myths above by empirically demonstrate our findings with a comprehensive set of experiments. This work is part of an effort to redress these mistakes. In terms of number of objective datasets and size of datasets, our experiments are orders of magnitude greater than anything else in the literature. In

particular, our experiments required more than 8 billion DTW comparisons.

The rest of the paper is organized as follows. The next three sections consider each of the three myths above with a comprehensive set of experiments, testing on a wide range of both real and synthetic datasets. Section 5 gives conclusions and directions for future work. Due to space limitations, we decided to omit the datasets details and background/review of DTW (which can be found in [16]). However, their full details and actual datasets have been publicly available for free download at [8].

2 Does Comparing Sequences of Different Lengths Help or Hurt?

Many recent papers suggest that the ability of classic DTW to deal directly with sequences of different length is a great advantage; some paper titles even contain the phrase “...of different lengths” [3][13] showing their great concerns in solving this issue. These claims are surprising in that they are not supported by any empirical results in the papers in question. Furthermore, an extensive literature search through more than 500 papers dating back to the 1960’s failed to produce any theoretical or empirical results to suggest that simply making the sequences to be of the same length has any detrimental effect.

To test our claimed hypothesis that there is no significant difference in accuracies between using variable-length time series and equal-length time series in DTW calculation, we carry out an experiment as follows.

For all variable-length time series datasets (Face, Leaf, Trace, and Wordspotting – See [8] for dataset details), we compute 1-nearest-neighbor classification accuracies (leaving-one-out) using DTW for all warping window sizes (1% to 100%) in two different ways:- (1) The **4S** way; we simply reinterpolated the sequences to have the same length, and (2) By comparing the sequences directly using their original lengths.

To give the benefit of the doubt to different-length case, for each individual warping window size, we do all four possible normalizations above, and the best performing of the four options is recorded as the accuracy for the variable-length DTW calculation.

For completeness, we test over every possible warping constraint size. Note that we start the warping window size of 1% instead of 0% since 0% size is Euclidean distance metric, which is undefined when the time series are not of the same length. Also, when measuring the DTW distance between two time series of different lengths, the percentage of warping window applied is based on the length of the longer time series to ensure that we allow adequate amount of warping for each pair and deliver a fair comparison.

The variable-length datasets are then linearly reinterpolated to have the same length of the longest time series within each dataset. Then, we simply compute the classification accuracies using DTW for all warping window sizes (1% to 100%) for each dataset. The results are shown in Figure 1.

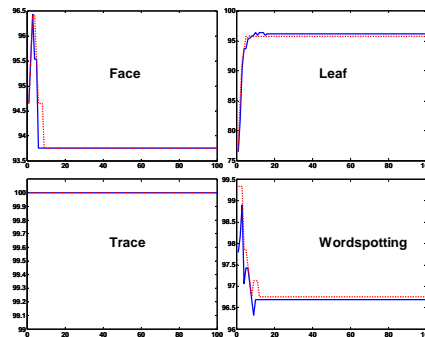


Figure 1. A comparison of the classification accuracies between variable-length (dotted lines) and the (reinterpolated) equal-length datasets (solid lines) for each warping window size (1-100%). The two options produce such similar results that in many places the lines overlap.

Note that the experiments do strongly suggest that changing the amount of warping allowed does affect the accuracy (an issue that will be discussed in depth in the next section), but over the entire range on possible warping widths, the two approaches are nearly indistinguishable. Furthermore, a two-tailed test using a significance level of 0.05 between each variable-length and equal-length pair indicates that there is no statistically significant difference between the accuracy of the two sets of experiments. An even more telling result is the following. In spite of extensive experience with DTW and an extensive effort, we were unable to create an artificial problem where reinterpolating made a significant difference in accuracy. To further reinforce our claim, we also reinterpolate the datasets to have the equal length of the shortest and averaged length of all time series within the dataset. We still achieve similar findings.

These results strongly suggest that work allowing DTW to support similarity search that does require reinterpolation, is simply solving a problem that does not exist. The often-quoted utility of DTW, such as “(DTW is useful) to measure similarity between sequences of different lengths” [21], for being able to support the comparison of sequences of different lengths is simply a myth.

3 Are Narrow Constraints Bad?

Apart from (slightly) speeding up the computation, warping window constraints were originally applied mainly to prevent pathological warping (where a relatively small section of one sequence maps to a much larger section of another). The vast majority of the data mining researchers have used a Sakoe-Chiba Band with a 10% width for the

global constraint [1][14][18]. This setting seems to be the result of historical inertia, inherited from the speech processing community, rather than some remarkable property of this particular constraint.

Some researchers believe that having wider warping window contributes to improvement in accuracy [22]. Or without realizing the great effect of the warping window size on accuracies, some applied DTW with no warping window constraints [12], or did not report the window size used in the experiments [11] (the latter case makes it particularly difficult for others to reproduce the experiment results). In [19], the authors bemoan the fact that “(4S) cannot be applied when the warping path is not constrained” and use this fact to justify introducing an alternative approach that works for the unconstrained case.

To test the effect of the warping window size to the classification accuracies, we performed an empirical experiment on all seven classification datasets. We vary the warping window size from 0% (Euclidean) to 100% (no constraint/full calculation) and record the accuracies.

Since we have shown in Section 2 that reinterpolation of time series into the same length is at least as good as (or better than) using the original variable-length time series, we linearly interpolate all variable-length datasets to have the same length of the longest time series within the dataset and measure the accuracy using the 1-nearest-neighbor with leaving-one-out classification method. The results are shown in Figure 2. As we hypothesized, wider warping constraints do not always improve the accuracy, as commonly believed [22]. More often, the accuracy peaks very early at much smaller window size (average = 4%). We also did an additional experiment, where half of the objects in the databases were randomly removed from the database iteratively. We measure the classification accuracies for each database size; as the database size decreases, the classification accuracy also declines and the peak appears at larger warping window size.

This finding suggests that warping window size adjustment does affect accuracy, and that the effect also depends on the database size. This in turn suggests that we should find the best warping window size on realistic (for the task at hand) database sizes, and not try to generalize from toy problems.

To summarize, there is no evidence to support the idea that we need to be able to support wider constraints. While it is possible that there exist some datasets somewhere that could benefit from wider constraints, we found no evidence for this in a survey of more than 500 papers on the topic. More tellingly, in spite of extensive efforts, we could not even create a *large* synthetic dataset for classification that needs more than 10% warping.

In fairness, we should note that it is only in the database/data mining community that this misconception exists. Researchers that work on real problems have long ago noted that constraining the warping helps. For example, Tomasi et al. who work with chromatographic data noted “*Unconstrained dynamic time warping was found to be too flexible for this chromatographic data set, resulting in a overcompensation of the observed shifts*” [20], or Rath & Manmatha have carefully optimized the constraints for the task of indexing historical archives [17].

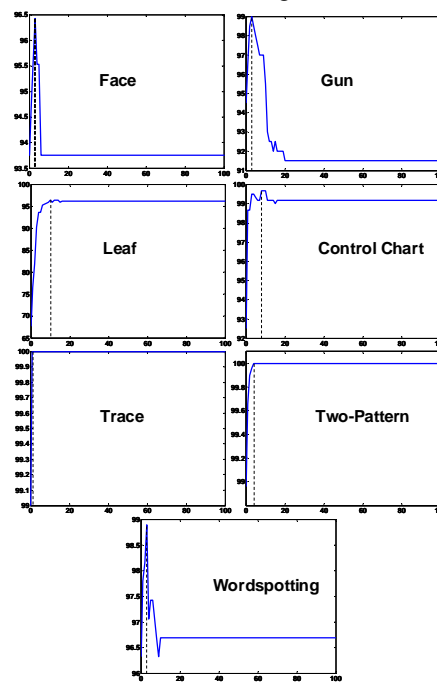


Figure 2. The classification accuracies for all warping window sizes (0% to 100%). All accuracies peak at very small window sizes.

All the evidence suggests that narrow constraints are *necessary* for accurate DTW, and the “need” to support wide (or no) constraints is just a myth.

4 Can DTW be further Speeded up?

Smaller warping windows speed up the DTW calculations simply because there is less area of the warping matrix to be searched. Prior to the introduction of lower bounding, the amount of speedup was directly proportional to the width of the warping window. For example, a nearest neighbor search with a 10% warping constraint was almost exactly twice as fast as a search done with a 20% window. However, it is important to note that with the introduction of lower bounding based on warping constraints (i.e. 4S), the speedup is now highly nonlinear in the size of the warping window. For example, a nearest neighbor search with a 10% warping constraint may be many times faster than twice a search done with a 20% window.

In spite of this, many recent papers still claim that there is a need and room for further improvement in speeding up DTW. Surprisingly, as we will show, the amortized CPU cost of DTW is essentially $O(n)$, using trivial **4S** technique.

To really understand what is going on, we will avoid measuring the efficiency of DTW when using index structures. The use of such index structures opens the possibility of implementation bias [9]; it is simply difficult to know if the claimed speedup truly reflects a clever algorithm, or simply care in choice of buffer size, caching policy, etc.

Instead, we measure the computation time of DTW for each pair of time series in terms of the amortized percentage of the warping matrix that needs to be visited for each pair of sequences in our database. This number depends only on the data itself and the usefulness of the lower bound. As a concrete example, if we are doing a one nearest neighbor search on 120 objects with a 10% warping window size, and the **4S** algorithm only needs to examine 14 sequences (pruning the rest), then the amortized cost for this calculation would be $(w * 14) / 120 = 0.12 * w$, where w is the area (in percentage) inside the warping window constraint along the diagonal (Sakoe-Chiba band). Note that 10% warping window size does not always occupy 10% of the warping matrix; it mainly depends on the length of the sequence as well (longer sequences give smaller w). In contrast, if **4S** was able to prune all but 3 objects, the amortized cost would be $(w * 3) / 120 = 0.03 * w$.

The amount of pruning we should actually expect depends on the lower bounds. For example, if we used a trivial lower bound hard-coded to zero (pointless, but perfectly legal), then line 4 of Table 1 would always be true, and we would have to do DTW for every pair of sequences in our dataset. In this case, amortized percentage of the warping matrix that needs to be accessed for each sequence in our database would exactly be the area inside the warping window. If, on the other hand, we had a “magic” lower bound that returned the true DTW distance minus some tiny epsilon, then line 4 of the Table 1 would rarely be true, and we would have to do the full DTW calculation only rarely. In this case, the amortized percentage of the warping matrix that needs to be accessed would be very close to zero. We measured the amortized cost for all our datasets, and for every possible warping window size. The results (and its 0-10% warping zoom-in) are shown in Figure 3. The results are surprising. For reasonably large datasets, simply using a good lower bound insures that we rarely have to use the full DTW calculation. In essence, we can say that DTW is effectively $O(n)$, and not $O(n^2)$, when searching large datasets.

For example, in the Gun, Trace, and 2-Pattern problems (all maximum accuracy at 3% warping), we only need to do

much less than half a percent of the $O(n^2)$ work that we would have been forced to do without lower bounding. For some of the other datasets, it may appear that we need to do a significant percentage of the CPU work. However, as we will see below, these results are pessimistic in that they reflect the small size of these datasets.

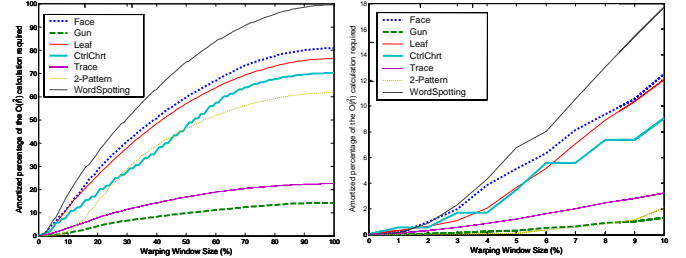


Figure 3. (left) The amortized percentage of warping matrix that needs to be accessed during the DTW calculation for each warping window size. The use of a lower bound helps prune off numerous unnecessary calculations. (right) Zoom-in of the warping range 0-10%

If the amortized cost of DTW is linear, where does the claimed improvement from recent papers come from? It is true that these approaches typically use indices, rather than sequential search, but an index must do costly random access rather than the optimized linear scans of sequential search. In order to break even in terms of disk access time, they must avoid looking at more than 10% of the data [7], but for time series where even the reduced dimensionality (i.e. the Fourier or wavelet coefficients) is usually greater than 20 [9], it is not obvious that this is possible.

Some recent papers that claim speedups credit the improved lower bounds, for example “...we present progressively tighter lower bounds... that allow our method to outperform (4S)” [19]. Indeed, it might be imagined that speedup could be obtained by having tighter lower bounds. Surprisingly, this is not true! We can see this with our simple experiment. Let us imagine that we have a wonderful lower bound, which always returns a value that is within 1% of the correct value (more concretely, a value uniformly distributed between 99% and 100% of the true DTW value). We will call this idealized lower bound *LB_Magic*. In contrast, the current best-known lower bounds typically return a value between 40% and 60% of the true value [6].

We can compare the speedup obtained by *LB_Magic* with the current best lower bound, *LB_Keogh* [6], on 1-nearest neighbor search. Note that we have to cheat for *LB_Magic* by doing the full DTW calculation then assigning it a value up to 1% smaller. We will use a warping constraint of 5%, which is about the mean value for the best accuracy (cf. Sect. 3). As before, we measured the amortized percentage of the warping matrix that needs to be accessed for each sequence in our database. Here, we use a *randomwalk* data

of length 128 data points, and vary the database size from 10 objects to 40,960 objects. Figure 4 shows the results.

Once again, the results are very surprising. The idealized LB_Magic allows a very impressive speedup; for the largest size database, it eliminates 99.997% of the CPU effort. However, the very simple lower bounding technique that has been in the literature for several years is able to eliminate 99.369% of the CPU effort! The difference is not quite so dramatic for very small datasets, say less than 160 objects. But here we can do unoptimized search in much less than a hundredth of a second. Note that we obtain similar results for other datasets.

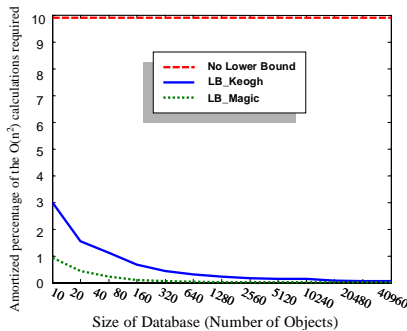


Figure 4. Amortized percentage of the warping matrix that needs to be accessed. As the size of the database increases, the amortized percentage of the warping matrix accessed becomes closer to zero.

To summarize, for problems involving a few thousand sequences or more, each with a few hundred data points, the “significant CPU cost of DTW” is simply non-issue (as for problems involving less than a few thousand sequences, we can do them in less than a second anyway).

The lesson for the data mining community from this experiment is the following; it is almost certainly pointless to attempt to speed up the CPU time for DTW by producing tighter lower bounds. Even if you could produce a magical lower bound, the difference it would make would be tiny, and completely dwarfed by minor implementation choices.

5 Conclusions and Future Work

In this work, we have pointed out and investigated some of the myths in Dynamic Time Warping measure. We empirically validated our three claims. We hope that our results will help researchers focus on more useful problems. For example, while there have been dozens of papers on speeding up DTW in the last decade, there has only been one on making it more accurate [15]. Likewise, we feel that the speed and accuracy of DTW that we have demonstrated in this work may encourage researchers to apply DTW to a wealth of new problems/domains.

6 References

- [1] Aach, J. & Church, G. (2001). Aligning gene expression time series with time warping algs. *Bioinformatics*(17), 495-508.
- [2] Berndt, D. & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. *AAAI Workshop on Knowledge Discovery in Databases*, pp. 229-248.
- [3] Bozkaya, T., Yazdatani, Z., & Ozsoyoglu, Z.M. (1997). Matching and Indexing Sequences of Different Lengths. *CIKM*
- [4] Caiani, E.G., Porta, A., Baselli, G., Turiel, M., Muzzupappa, S., Pieruzzi, F., Crema, C., Malliani, A., & Cerutti, S. (1998). Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. *IEEE Computers in Cardiology*, pp. 73-76.
- [5] Cardle, M. (2003). Music-Driven Animation. Ph.D. Thesis, Cambridge University.
- [6] Keogh, E. (2002). Exact indexing of dynamic time warping. In *28th VLDB*. Hong Kong. pp. 406-417.
- [7] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD*, pp. 151-162.
- [8] Keogh, E. & Folias, T. (2002) The UCR Time Series Data Mining Archive. [<http://www.cs.ucr.edu/~eamonn/TSDMA>]
- [9] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In the *8th ACM SIGKDD*, pp. 102-111.
- [10] Kim, S.W., Park, S., & Chu, W.W. (2004). Efficient processing of similarity search under time warping in sequence databases: an index-based approach. *Inf. Syst.* 29(5): 405-420.
- [11] Kornfield, E.M, Manmatha, R., & Allan, J. (2004). Text Alignment with Handwritten Documents. *1st Int'l workshop on Document Image Analysis for Libraris (DIAL)*, pp. 195-209.
- [12] Laaksonen, J., Hurri, J., and Oja, Erkki. (1998). Comparison of Adaptive Strategies for On-Line Character Recognition. In *proceedings of ICANN'98*, pp. 245-250.
- [13] Park, S., Chu, W, Yoon, J., and Hsu, C (2000). Efficient searches for similar subsequences of different lengths in sequence databases. In *ICDE-00*.
- [14] Rabiner, L., Rosenberg, A. & Levinson, S. (1978). Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Trans. Acoustics Speech, and Signal Proc.*, Vol. ASSP-26, pp. 575-582.
- [15] Ratanamahatana, C.A. & Keogh, E. (2004). Making Time-series Classification More Accurate Using Learned Constraints. *SDM International conference*, pp. 11-22.
- [16] Ratanamahatana, C.A. & Keogh, E. (2004). Everything You Know about Dynamic Time Warping is Wrong. *SIGKDD Workshop on Mining Temporal and Sequential Data*.
- [17] Rath, T. & Manmatha, R. (2003). Word image matching using dynamic time warping. *CVPR*, Vol. II, pp. 521-527.
- [18] Sakoe, H. & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, & Signal Proc.*, ASSP-26, 43-49.
- [19] Shou, Y., Mamoulis, N., and Cheung, D.W. Efficient Warping of Segmented Time-series, *HKU CSIS Tech rep*, TR-2004-01
- [20] Tomasi, G., van den Berg, F., & Andersson, C. (2004). Correlation Optimized Warping and DTW as Preprocessing Methods for Chromatographic Data. *J. of Chemometrics*.
- [21] Wong, T.S.F & Wong, M.H. (2003). Efficient Subsequence Matching for Sequences Databases under Time Warping. *IDEAS*.
- [22] Zhu, Y. & Shasha, D. (2003). Warping Indexes with Envelope Transforms for Query by Humming. *SIGMOD*, 181-192.

PCA without eigenvalue calculations: a case study on face recognition*

E. Kokiopoulou and Y. Saad
Comp. Sci. & Eng. Dept
University of Minnesota
{kokiopou,saad}@cs.umn.edu

Abstract

Principal component analysis (PCA) is an extensively used dimensionality reduction technique, with important applications in many fields such as pattern recognition, computer vision and statistics. It employs the eigenvectors of the covariance matrix of the data to project it on a lower dimensional subspace.

However, the requirement of PCA eigenvectors is a computational bottleneck which poses serious challenges and limits the applicability of PCA-based methods, especially for real-time computations. This paper proposes an alternative framework, relying on polynomial filtering which enables efficient implementations of PCA. We showcase the applicability of the proposed scheme on face recognition. In particular, we consider the eigenfaces methods which employ PCA. The numerical experiments reported indicate that the proposed technique competes with the PCA-based method in terms of recognition rate, while being much more efficient in terms of computational and storage cost.

Keywords Principal Component Analysis, Polynomial Filtering, Face Recognition.

1 Introduction

Principal component analysis (PCA) [5] is one of the most popular dimensionality reduction techniques. It has numerous applications in many areas such as pattern recognition, computer vision, statistics and data analysis. PCA has been successfully applied in automated face recognition [14], resulting in the so called method of *eigenfaces* introduced by Kirby and Sirovich [6], Sirovich and Kirby [12] and Turk and Pentland [10], [13]. The eigenfaces method is one of the most popular appearance-based holistic approaches (see e.g., [1], [13]) which employs PCA on the covariance matrix C , constructed by the training data.

Typical implementations of the eigenfaces method rely upon eigendecomposition of the covariance matrix. However, when the datasets are dynamic and of large scale, the applicability of the above methods is limited due to their high computational cost (which is $O(n^3)$ for dense matrices). This is even more evident in the case of real-time and adaptive algorithms (see e.g. [9]). In these cases, the eigendecomposition must be updated frequently and the time constraints are very strict. To that end, a lot of research efforts have been devoted to efficient eigenspace update schemes such as the one proposed in [4].

In this paper we propose an alternative implementation scheme which approximates directly the similarity score without computing the eigendecomposition of C or any other matrix decomposition. Denoting by A the data matrix in the input space, the new method relies on polynomial filtering, where a well defined polynomial ψ of the matrix AA^T or $A^T A$ is applied on the new face image and yields an approximation to the similarity score that is very close to the one obtained using eigendecomposition. The polynomial ψ is chosen appropriately such that it is a good approximation of the step function.

The polynomial filtering framework was applied successfully in [7] for dimensionality reduction in information retrieval. In this paper we showcase the applicability of this technique in a different context, that of face recognition. We claim that the proposed framework can be applied in any method employing PCA to estimate similarities among data vectors. Numerical experiments indicate that the proposed framework is quite close to the PCA methods in terms of recognition rate without suffering from their computational and storage limitations.

The remaining sections of this paper are organized as follows: Section 2 provides an overview of the eigenfaces method using eigenvalue decomposition. In Section 3 the eigenfaces method is interpreted in terms of

*Work supported by the Minnesota Supercomputing Institute

Singular Value Decomposition (SVD). Next, in Section 4 the implementation of face recognition using eigenfaces, via polynomial filtering is described. Finally, Section 5 provides a series of numerical results verifying the practical advantages of the proposed scheme.

2 The method of eigenfaces

2.1 Construction of the face space Suppose that a face image consists of N pixels, so it can be represented lexicographically by a vector x of dimension N . Let $\{x_i | i = 1, \dots, M\}$ be the training set of face images. The mean face is given by

$$(2.1) \quad \mu = \frac{1}{M} \sum_{i=1}^M x_i.$$

The covariance matrix of the translated training data is

$$(2.2) \quad C = \frac{1}{M} A A^\top \in R^{N \times N},$$

where $A = [\tilde{x}_1, \dots, \tilde{x}_M] \in R^{N \times M}$ is the matrix of the translated data points

$$(2.3) \quad \tilde{x}_i = x_i - \mu, \quad i = 1, \dots, M.$$

The eigenvectors u_l , $l = 1, \dots, M$ of the covariance matrix C are usually called “eigenfaces”, since they resemble faces when reshaped and illustrated in a pictorial fashion. In practice only a small number, say k , of eigenvectors corresponding to the largest eigenvalues are computed and then used for performing Principal Component Analysis (PCA) for face identification. The subspace spanned by the eigenfaces is called face space.

2.2 Face recognition using eigenfaces The face recognition procedure consists of two stages; the training stage and the recognition stage. In the training stage each face image x_i of the known individuals is projected on the face space and a k -dimensional vector P_i is obtained

$$(2.4) \quad P_i = U_k^\top (x_i - \mu), \quad i = 1, \dots, M,$$

where $U_k = [u_1, \dots, u_k]$ is the matrix with orthonormal columns, which are the eigenvectors associated with the k largest eigenvalues.

In the recognition stage, the new image $x \in R^N$ to be processed, is translated and then projected into the face space to obtain the vector

$$(2.5) \quad P_x = U_k^\top (x - \mu).$$

The distance between P_x and each face image is defined by

$$(2.6) \quad \begin{aligned} d_i^2 &= \|P_x - P_i\|_2^2 \\ &= \|P_x\|_2^2 + \|P_i\|_2^2 - 2P_x^\top P_i, \quad i = 1, \dots, M, \end{aligned}$$

where $\|\cdot\|_2$ is the Euclidean norm. Furthermore, in order to discriminate between face images and non-face images, the distance ϵ between the original image x and its reconstructed image from the face space, $x_f = U_k P_x + \mu$, is also computed:

$$(2.7) \quad \epsilon = \|x - x_f\|_2.$$

Note in passing that

$$\begin{aligned} \epsilon &= \|x - \mu - U_k P_x\|_2 \\ &= \|(x - \mu) - U_k U_k^\top (x - \mu)\|_2, \end{aligned}$$

and therefore ϵ represents simply the distance between $x - \mu$ and its orthogonal projection onto $\text{span}\{U_k\}$, i.e.,

$$\begin{aligned} (2.8) \quad \epsilon^2 &= \|(I - U_k U_k^\top)(x - \mu)\|_2^2 \\ (2.9) &= \|x - \mu\|_2^2 - \|P_x\|_2^2. \end{aligned}$$

This metric is used to decide whether or not a given image is a face.

3 Eigenfaces in terms of the SVD

In this section we interpret the above training and recognition stages in terms of the truncated singular value decomposition of A . The SVD [3] of a rectangular $N \times M$ matrix A of rank r , is defined as

$$(3.10) \quad A = U \Sigma V^\top,$$

$$(3.11) \quad U^\top U = I_N \in R^{N \times N},$$

$$(3.12) \quad V^\top V = I_M \in R^{M \times M},$$

where $U = [u_1, \dots, u_N]$ and $V = [v_1, \dots, v_M]$ are unitary matrices and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_M)$, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_M = 0$. The σ_i 's are the *singular values* of A and the u_i 's and v_i 's are respectively the *left and right singular vectors* associated with σ_i , $i = 1, \dots, r$. We define the i -th singular triplet of A as $\{u_i, \sigma_i, v_i\}$. It follows from the SVD that the matrix A can be expressed as a sum of r rank-one matrices,

$$A = \sum_{i=1}^r \sigma_i u_i v_i^\top.$$

Additionally, it is well known that

$$\min_{\text{rank}(B) \leq k} \|A - B\|_F = \|A - A_k\|_F$$

where $A_k = \sum_{i=1}^k \sigma_i u_i v_i^\top$ and $\|\cdot\|_F$ is the Frobenius norm. It is helpful for what follows to rewrite the matrix A_k as

$$(3.13) \quad A_k = U_k \Sigma_k V_k^\top,$$

where U_k (resp. V_k), consists of the first k columns of U (resp. V), and Σ_k is a diagonal matrix of size $k \times k$.

Thus, if we truncate the SVD to keep only the k largest singular triplets we obtain the closest (in a least-squares sense) approximation to A .

Observe that the matrix U_k containing the k largest left singular vectors of $\tilde{A} = \frac{1}{\sqrt{M}}A$, is exactly the matrix computed by PCA containing the largest eigenvectors of the covariance matrix. This follows from the fact that

$$C = \tilde{A}\tilde{A}^\top = U\Sigma V^\top V\Sigma^\top U^\top = U\Sigma\Sigma^\top U^\top,$$

is the eigendecomposition of the covariance matrix. Using this observation, equation (2.4) can be written in the form

$$\begin{aligned} P_i &= U_k^\top \tilde{x}_i = U_k^\top \tilde{A}e_i \\ &= U_k^\top [U_k \ U_{N-k}] \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_{M-k} \end{bmatrix} \begin{bmatrix} V_k^\top \\ V_{M-k}^\top \end{bmatrix} e_i \\ &= [I_k \ 0] \begin{bmatrix} \Sigma_k V_k^\top \\ \Sigma_{M-k} V_{M-k}^\top \end{bmatrix} e_i \\ &= \Sigma_k V_k^\top e_i, \quad i = 1, \dots, M. \end{aligned}$$

Denote by $P = \Sigma_k V_k^\top$ the matrix whose columns are the projections P_i , $i = 1, \dots, M$, of every known face image to the face space. Assuming that all vectors are normalized, the similarity measurement (2.6) among the new image x and all known images, can be equivalently computed by the similarity vector s_k ,

$$(3.14) \quad \begin{aligned} s_k &= P^\top P x = V_k \Sigma_k^\top U_k^\top (x - \mu) \\ &= \tilde{A}_k^\top (x - \mu), \end{aligned}$$

containing a similarity score between the new face image and each of the known images. Thus, the computation of the similarity vector s_k employs a rank k approximation of the translated matrix A . We discuss the assumption of normalized projected vectors in the following section.

Note also that using the SVD, equation (2.8) expresses the metric ϵ as the distance from $x - \mu$ to the space $\text{span}\{U_k\}$ of the dominant left singular space. In the sequel, we show how to approximate the similarity vector s_k in (3.14), as well as the distance ϵ in (2.8) without using eigendecompositions. The proposed scheme relies on polynomial filtering.

4 Eigenfaces using polynomial filtering

Polynomial filtering allows to closely approximate the effect of reduced rank approximation used in PCA models. Denote by $\psi(A)$ a matrix polynomial of degree d on the matrix A , i.e.,

$$\psi(A) = \xi_d A^d + \xi_{d-1} A^{d-1} + \dots + \xi_1 A + \xi_0 I.$$

Assuming that A is normal (i.e., $A^\top A = A A^\top$) and letting $A = Q\Lambda Q^\top$ be its eigendecomposition, observe

that $\psi(A) = \psi(Q\Lambda Q^\top) = Q\psi(\Lambda)Q^\top$. Therefore, the polynomial on A is translated to a polynomial on its eigenvalues. We are now ready to describe how one can use polynomial filtering to approximate the similarity vector directly, avoiding completely eigenvalue computations.

Let $\tilde{x} = x - \mu$ be the translated new image. In order to estimate the similarity measurement, we use a polynomial ψ of $\tilde{A}^\top \tilde{A}$ such that

$$(4.15) \quad \begin{aligned} s &= \psi(\tilde{A}^\top \tilde{A}) \tilde{A}^\top \tilde{x} \\ &= \psi(V\Sigma^\top \Sigma V^\top) V\Sigma^\top U^\top \tilde{x} \\ &= V\psi(\Sigma^\top \Sigma) V^\top V\Sigma^\top U^\top \tilde{x} \\ &= V\psi(\Sigma^\top \Sigma) \Sigma^\top U^\top \tilde{x}. \end{aligned}$$

Compare the last expression above with (3.14). Choosing the polynomial $\psi(t)$ appropriately will allow us to interpretate this approach as a compromise between the correlation [2] and the PCA approaches. Assume now that ψ is not restricted to being a polynomial but can be any function (even discontinuous). When $\psi(t) = 1 \ \forall x$, then $\psi(\Sigma^\top \Sigma)$ becomes the identity operator and the above scheme would be equivalent to the correlation method. On the other hand, taking ψ to be the step function

$$(4.16) \quad \psi(t) = \begin{cases} 0, & 0 \leq t \leq \sigma_k^2 \\ 1, & \sigma_k^2 \leq t \leq \sigma_1^2 \end{cases}$$

results in $\psi(\Sigma^\top \Sigma) = \begin{bmatrix} I_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ where I_k is the identity matrix of size k and $\mathbf{0}$ is a zero matrix of an appropriate size. Then, equation (4.15) may be re-written as:

$$(4.17) \quad \begin{aligned} s &= V\psi(\Sigma^\top \Sigma) \Sigma^\top U^\top \tilde{x} \\ &= [V_k \ V_{n-k}] \begin{bmatrix} \Sigma_k^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} U_k^\top \\ U_{m-k}^\top \end{bmatrix} \tilde{x} \\ &= [V_k \Sigma_k^\top \ \mathbf{0}] \begin{bmatrix} U_k^\top \\ U_{m-k}^\top \end{bmatrix} \tilde{x} \\ &= V_k \Sigma_k^\top U_k^\top \tilde{x} \\ &= \tilde{A}_k^\top \tilde{x} \end{aligned}$$

which is precisely the rank- k approximation provided in equation (3.14).

Using polynomial filtering we can also approximate the “faceness” (i.e., whether or not a given image contains a face) of an image as it is expressed by equation (2.8). Using the SVD, observe that

$$(4.18) \quad \begin{aligned} \psi(C)(x - \mu) &= \psi(\tilde{A}\tilde{A}^\top)(x - \mu) \\ &= \psi(U\Sigma V^\top V\Sigma^\top U^\top)(x - \mu) \\ &= U\psi(\Sigma\Sigma^\top)U^\top(x - \mu). \end{aligned}$$

Note that if ψ is exactly the step function (4.16), then $\|\psi(C)(x - \mu)\|_2 = \|U_k U_k^\top (x - \mu)\|_2 = \|P_x\|_2$ which would

allow to obtain ϵ from (2.8). If the polynomial ψ is an approximation of the step function, this will provide an estimate of the distance metric ϵ , needed to decide on the faceness of an image, without the availability of U or U_k .

Therefore, the approach of polynomial filtering in PCA models can give virtually the same result as eigen-decomposition, without resorting to the costly eigenvalue decomposition or any other matrix decomposition. Furthermore, the need to store additional (dense or sparse) matrices as is the case in PCA, is completely avoided as is the need to update these matrices, when the subspace used for learning changes dynamically. The selection of the cut-off point is somewhat similar to the issue of choosing the parameter k in the PCA method. However, there is a salient difference between the two: choosing a large k in PCA may render the method much more expensive, while selecting a high cut-off in polynomial filtering does not affect cost significantly.

Recall that in the computation of the similarity vector we assumed that the projected vectors P_i have unity norm. Here are two solutions to overcome this problem. Before applying the proposed scheme we normalize all input data vectors x_i . Next, we compute the similarity score and sort the samples in descending order. Then we have two options. Using the first $k \ll M$ samples, either we can employ PCA or we can use k -nearest neighbor classification. Observe that since $k \ll M$, the cost of exact PCA will be very limited, and certainly orders of magnitude smaller than PCA on the original data matrix. Similarly, applying k -nearest neighbor classification on a very small set of data points will have very limited cost. We observed empirically that the first option yields slightly better results and this is the option that we included in our experiments (Section 5) with $k = 30$.

5 Numerical results

All experiments are implemented in MATLAB 6.5 on a Xeon@2.4GHz. We use three datasets that are publically available: YALE, ORL and a subset of AR. The YALE database [1] contains 165 images of 15 individuals that include variation in both facial expression and lighting. In the preprocessing phase, each face image is closely cropped, and the size of images after the cropping phase is decreased to 112×92 . The ORL (formerly Olivetti) database [11] contains 40 individuals and 10 different images for each individual. In this case no preprocessing is done. Finally, the AR face database [8] contains 126 subjects with 4 different facial expressions for each individual.

In what follows, error rates are estimated using a

$k = 40$	ORL (%)	YALE (%)	AR (%)
$\gamma=2$	2.5	26.06	8.33
$\gamma=3$	3.5	25.45	8.53
$\gamma=4$	2.75	26.06	7.14
$\gamma=5$	3	26.06	6.15

Table 1: Error rates of the PPF method for various values of γ , on all face databases.

cross validation “leave-one-out” strategy. In order to compute the error rate with respect to a certain facial expression, the image associated with it is used as a test image. In order to recognize the test image, all images, excluding the test one, are projected to the reduced subspace. Then, the test image is projected as well and recognition is performed using a nearest neighbor rule. Denote by e_i as the number of misses counted across the subjects for a given facial expression i . Denote also by N_f the number of different facial expressions/poses associated with each individual in the database. Define $e = \frac{1}{N_f} \sum_{i=1}^{N_f} e_i$, $i = 1, \dots, N_f$. Thus, e is the mean error rate averaged across all different facial expressions. In what follows, denote by PCA the “eigenfaces” method and by PPF the polynomial filtering method.

Example 1 In the first example we investigate the behavior of the PPF method with respect to the degree of the polynomial ψ . Table 1 illustrates the error rate of PPF with respect to γ . The parameter γ affects the degree of the polynomial approximation to the step function. The higher the value of γ the higher the degree of the polynomial. Observe that in most cases the value $\gamma = 4$ seems to give the most satisfactory results. To that end, in what follows, we use $\gamma = 4$ for PPF.

Example 2 We now investigate the effect of the dimension k of the reduced space on the recognition performance of the methods. We use MATLAB’s `svd` builtin function since the matrix is dense and this way we avoid the explicit use of the matrices AA^\top or $A^\top A$. We experiment with $k = 20 : 20 : 100$ (in MATLAB notation) and measure the error rate (%) for all face databases.

Table 2 illustrates the error rate e versus the dimension k measured on the ORL, YALE and AR datasets respectively. All tables contain the corresponding time measurements t (in sec) for each method. The timings for PCA methods measure the time needed to construct the subspace (i.e., computing the eigenvectors) and perform the recognition of the test image (i.e., one step of “leave-one-out” cross validation). The timings for PPF methods measure the time needed to recognize the test data point via polynomial filtering.

Concerning the ORL database, observe that PPF

ORL	PCA		PPF	
	e	t	e	t
$k=20$	3.5	32.74	3	2.52
$k=40$	2.75	30.68	2.75	2.49
$k=60$	3.25	30.93	3.25	2.48
$k=80$	3.25	32.96	3	2.52
$k=100$	3	32.03	3	2.49

YALE	PCA		PPF	
	e	t	e	t
$k=20$	29.70	5.93	25.45	1.15
$k=40$	27.88	6.02	26.06	1.16
$k=60$	27.27	6.10	25.45	1.14
$k=80$	27.27	6.22	25.45	1.16
$k=100$	26.06	6.33	25.45	1.15

AR	PCA		PPF	
	e	t	e	t
$k=20$	8.34	82.02	6.35	5.71
$k=40$	6.75	82.02	7.34	5.71
$k=60$	6.15	83.12	7.14	5.71
$k=80$	6.15	83.67	6.75	5.70
$k=100$	5.75	83.64	6.35	5.71

Table 2: Error rates e (%) and timings t (in sec) of both methods for various values of k , on all the face databases.

competes with PCA in terms of error rate. Furthermore, the PPF method is much more efficient achieving significant speedups over its PCA counterpart. On the YALE dataset, the results are quite similar with PPF outperforming PCA not only in timings but in error rate as well. Finally, on the AR dataset, the results are similar to ORL, with the PPF methods being quite close to PCA in terms of error rate and being much more efficient in terms of computational cost.

6 Conclusion

We have described an alternative framework for implementing PCA without eigenvalue calculations. The proposed framework relies on polynomial filtering, in order to render the same effect as PCA, for dimensionality reduction. We illustrated the applicability of the proposed technique in the eigenfaces method for face recognition. The numerical experiments indicated that the new scheme has very close performance to the PCA method, while being much more efficient in terms of computational cost and storage.

7 Acknowledgments

We would like to thank prof. N. Papanikolopoulos for his valuable advice concerning face recognition issues.

References

- [1] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Trans. Pattern Analysis and Machine Intelligence, Special Issue on Face Recognition*, 19(7):711–20, July 1997.
- [2] R. Brunelli and T. Poggio. Face recognition: Features vs Templates. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(10):1042–1053, 1993.
- [3] G. H. Golub and C. Van Loan. *Matrix Computations*, 3rd edn. The John Hopkins University Press, Baltimore, 1996.
- [4] L. Hoegaerts, L. De Lathauwer, J.A.K. Suykens, and J. Vanderwalle. Efficiently updating and tracking the dominant kernel eigenspace. *16th International Symposium on Mathematical Theory of Networks and Systems*, July 5-9 2004. Belgium.
- [5] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, New York, 1986.
- [6] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12, 1990.
- [7] E. Kokiopoulou and Y. Saad. Polynomial Filtering in Latent Semantic Indexing for Information Retrieval. In *ACM-SIGIR Conference on research and development in information retrieval*, Sheffield, UK, July 25th-29th 2004.
- [8] A.M. Martinez and R. Benavente. The AR Face Database. Technical report, CVC no. 24, 1998.
- [9] S.K. Nayar, S.A. Nene, and H. Murase. Subspace Methods for Robot Vision. *IEEE Transactions on Robotics and Automation*, 12(5):750–758, October 1996.
- [10] A. Pentland, B. Moghaddam, and T. Starner. View-based and Modular Eigenspaces for Face Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.
- [11] F. Samaria and A. Harter. Parameterisation of a Stochastic Model for Human Face Identification. In *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL, December 1994.
- [12] L. Sirovich and M. Kirby. Low-dimensional Procedure for the Characterization of Human Faces. *J. Optical Soc. Am. A (JOSA A)*, 4(3):519–524, March 1987.
- [13] M. Turk and A. Pentland. Face Recognition using Eigenfaces. In *Int. Conf. on Patt. Recog.*, pages 586–591, 1991.
- [14] W. Zhao, R. Chellapa, P. Phillips, and A. Rosenfeld. Face Recognition: a Literature Survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.

Mining Top- K Itemsets over a Sliding Window Based on Zipfian Distribution

Raymond Chi-Wing Wong, Ada Wai-Chee Fu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
cwwong,adafu@cse.cuhk.edu.hk

Abstract

Frequent pattern discovery in data streams can be very useful in different applications. In time critical applications, a sliding window model is needed to discount stale data. In this paper, we adopt this model to mine the K most interesting itemsets, or to estimate the K most frequent itemsets of different sizes in a data stream. In our method, the sliding window is partitioned into buckets. We maintain the statistics of the frequency counts of the itemsets for the transactions in each bucket. We prove that our algorithm guarantees no false negatives for any data distributions. We also show that the number of false positives returned is typically small according to Zipfian Distribution. Our experiments on synthetic data show that the memory used by our method is tens of times smaller than that of a naive approach, and the false positives are negligible.

1 Introduction

Data mining processing is typically time-consuming. However, there are some recent demands on real-time data mining for unbounded data stream arriving at high speed. Examples include financial data monitoring and network monitoring. The mining process becomes much more difficult because it requires not only the handling of massive unbounded data stream but also the ability to return the results within a short time.

With limited memory storage, it is natural to devise methods to store some kinds of statistics or summary of the data stream. Until now, most research work consider all data read so far. However, in many applications, old data are less important or not relevant, compared with more recent data. There are two common approaches to deal with this issue. The first one is *aging* [11, 5], where each data is assigned a weight, with more weight for more recent data (e.g. exponential-decay model). Another approach is to use a *sliding window* [2, 4, 8, 3, 9], so that only the most recent W data elements in the data stream is considered, where W is the width of a sliding window. In this paper, we adopt the second approach.

For association rule mining, the difficult subproblem of frequent itemset discovery has been the focus of research for some time. Many motivating examples are given in [12] for the mining of frequent itemsets in data streams. A major issue with mining frequent itemsets is that user has to define a support or frequency threshold s on the resulting itemsets, and without any guidance, this is typically a wild guess. In some previous study [6, 7], it is found that, in different data sets, or even with different subsets of the same data set, the proper values of s can differ by an order of magnitude. In most previous work of data stream mining a major concern is to minimize the error of the false positive to a small fraction of s . However, if the threshold s is not appropriate in the first place, such a guarantee is quite pointless.

Therefore, it is of interest to replace the requirement of a frequency threshold to that of the simpler threshold on the amount of results. It is much easier for users to specify that say the 20 most frequent patterns should be returned. Some previous work assumed that such a threshold can be applied to itemsets of all sizes. However, there is a major pitfall with such an assumption. It is that it implies a uniform frequency threshold for itemsets of all sizes. It is obvious that small size itemsets have an intrinsic tendency to appear more often than large size itemsets. The result from this assumption is that smaller size itemsets can dominate and hide some interesting large size itemsets. The mining of closed patterns does not help much. For example, an interesting closed itemsets X of size 4 may have a frequency of 0.01, while many smaller size closed itemsets have frequencies above 0.1, and hence X cannot hope to reach the top K frequency. Therefore, some previous work has proposed to mine the K most frequent itemsets of size l , for each l that is within a range of sizes specified by user. We shall focus on this mining problem for data streams.

Let us call an itemset of size l an l -itemset. Our problem is about mining K l -itemsets with the greatest frequencies (supports) for each l up to a certain L . We shall tackle this problem for a data stream with a sliding window of size m (contains m transactions). In our approach, the sliding window is divided into n_B partitions, called *buckets*. Each

bucket corresponds to a set of transactions and we maintain the statistics for the transactions in each bucket separately. The window is slid forward one bucket at a time. When the window is advanced, the oldest bucket is discarded and a newly generated bucket is appended to the sliding window. At the same time, the candidate top K interesting itemsets are adjusted. Our method have some guarantees for the results. It gives no false negatives for any data distribution. Given a Zipfian data distribution with Zipfian parameter θ and an error parameter $\epsilon > 0$, it outputs no more than $K[(1 + \epsilon)^{1/\theta} - 1]$ false positives. The memory usage of our algorithm is bounded by $O(\frac{n_B^{1+2/\theta} m^{1/\theta} K}{\epsilon^{1/\theta}})$. From our experiment, we found that error in frequency of false positives is very small, and the proposed method can achieve memory usage that is many times less than a more naive approach.

2 Problem Definitions and Terminologies

In this section we introduce the problem definition and also other terminologies.

Problem Definition: The data stream is considered a sequence of *equisized* data buckets with s_B transactions each. The most recent n_B full buckets in the data stream is considered as the sliding window. Given two positive integers K and L . For each l , where $l \leq L$, and let the K -th highest frequency among all l -itemsets in the sliding window be $f(l)$, find all l -itemsets with frequencies greater than or equal to $f(l)$ in the sliding window. These are called the top K l -itemsets. \square

Note that in the above definition, at any time, there will be a most recent bucket B , which may or may not be full. A bucket is full when it contains s_B transactions. When a transaction T arrives at the data stream, it will be inserted into B if it is not full; otherwise, a new bucket containing only T will be created and becomes the most recent bucket B . Let $m = n_B \times s_B$. Hence the size of the sliding window is m (number of transactions). The sliding window contains buckets B_1, B_2, \dots, B_{n_B} , in chronological order, where bucket B_{n_B} represents the most recently created bucket.

In our algorithm we need to process itemsets of sizes l , $1 \leq l \leq L$. Without loss of generality let us consider size l itemsets, for a certain l . We are going to find the top K l -itemsets.

Let l -itemset denote itemset of size l . Each bucket B_i stores a list of entries (e, f) , where e is one of the top $K'_{i,l} \geq K$ l -itemsets and f is the frequency of e in the bucket.¹ We use $f_{i,e}$ to denote the frequency of e in bucket B_i . We say that $f_{i,e}$ is recorded if e is among the top $K'_{i,l}$ itemsets.

¹Note that $K'_{i,l}$ can be different for different l and will be determined by our algorithm automatically.

Therefore, each bucket stores information about the top $K'_{i,l}$ frequent itemsets.

Let $f_{i,min}$ be the frequency of the $K'_{i,l}$ -th frequent l -itemset in bucket B_i . For entry e in bucket B_i , $f_{i,e} \geq f_{i,min}$. We define $\min(e)$ and $\max(e)$. $\min(e) = \sum_{B_i \text{ and } f_{i,e} \text{ is recorded}} f_{i,e}$ and $\max(e) = \sum_{B_i \text{ and } f_{i,e} \text{ is recorded}} f_{i,e} + \sum_{B_i \text{ and } f_{i,e} \text{ is not recorded}} (f_{i,min} - 1)$.

When we sum up all recorded frequencies $f_{i,e}$ of itemset e in different buckets B_i , this value should be the least possible frequency of itemset e . However, in some buckets B_i , there may be no recorded frequencies. The itemset e may appear in those buckets. To estimate the maximum possible frequency, we assume the maximum possible frequency for itemsets e with no recorded frequency, and this frequency is $f_{i,min} - 1$ for buckets B_i . Therefore $\min(e)$ is the minimum possible frequency of itemset e in the sliding window while $\max(e)$ is the maximum possible frequency of itemset e in the sliding window. Let f_e be the frequency of e in the sliding window. Thus, $\min(e) \leq f_e \leq \max(e)$.

We define $f_{(1)} = \max_e \{\min(e)\}$, which is the greatest value of $\min(e)$ among all e . We define $M_l = \min_{B_i} \{f_{i,min}\}$, which is the minimum value of $f_{i,min}$ among all buckets. We also define $\Delta_{\max,l} = \sum_{B_i} f_{i,min} - M_l$.

3 Algorithm

Let the size of the sliding window be m (there are m transactions). There are n_B buckets in the sliding window. So, the bucket size s_B is $\lceil m/n_B \rceil$. For each full bucket we store a list of entries (e, f) . The 2 major steps of our algorithm will be introduced in this section. At the beginning of the algorithm, we process the first full bucket containing the transactions at the beginning of the data stream in Step 1. For each new bucket, we need to accumulate s_B transactions in the memory temporarily. After receiving the s_B transactions, we process the transactions with Step 2. Every time a bucket leaves the sliding window, the bucket and its entries will be removed.

There are two major parameters in our algorithm - (1) θ and (2) ϵ . (1) θ is a Zipfian parameter in a Zipfian distribution. The greater the value of θ , the greater the skewness of the distribution. The Zipfian parameter $\theta \geq 1$ is commonly used in the Zipfian distribution in previous research on data streams[12, 10]. In [12], $\theta = 1.25$; and [10] sets θ to be 1.0, 1.25 and 1.5. In our real data set, we found that the distribution is quite skew, which also corresponds to $\theta \geq 1$. (2) ϵ is an error parameter. The smaller the value of ϵ is, the more accurate the algorithm is. However, with a small value of ϵ , the memory consumption will be great. So, ϵ is a user input parameter of our algorithm. It can determine the storage and the accuracy of our algorithm. The the accuracy bound and storage bound can be found in Corollaries 1 and

2, respectively in Section 4.

The major steps of our algorithm are described as follows.

1. After receiving the first bucket of transactions at the beginning of the data stream, we do the following. Let $r_0 = \lceil n_B(n_B - 1) \left(\frac{1}{2K^\theta(1+1/\epsilon)} + \frac{n_B - 1}{m} \right)^{-1} \rceil^{1/\theta}$,² where θ is the Zipfian parameter and ϵ is an error parameter. If r_0 is greater than the number of possible itemsets, r_0 is assigned to be the number of possible itemsets.
 - (a) find top r_0 itemsets of size l
For this task, we can use an existing algorithm for mining top K itemsets (e.g. [7]).
 - (b) store the entries (e, s) of the itemsets found
2. After the first bucket, we can process other buckets B_i in the following way. We define $\max'(e)$ with the same definition of $\max(e)$ but $\max'(e)$ is evaluated with the scope of all buckets in the current sliding window except for the bucket B_i .
 - (a) find the K -th largest value of $\min(e)$ of itemset e of size l , $K_{\min,l}$, within the current bucket B_i and all previous buckets in the sliding window
 - (b) Determine the rank $r_{i,e}$ of each e in bucket B_i . Find the greatest rank $r_{i,e}$, say \tilde{r}_i , in order that $\max'(e) + f_{i,e} \geq K_{\min,l}$ and $r_{i,e} \leq r_0$. Store all entries of itemsets e of size l with $r_{i,e} \leq \tilde{r}_i$. Again we can make use of the existing algorithm in [7].
 - (c) calculate $\tilde{\Delta}_{\max,l} = \frac{f_{(1)}}{2K^\theta(1+1/\epsilon)}$. If $\Delta_{\max,l} > \tilde{\Delta}_{\max,l}$, then store the additional next top frequent itemsets in the bucket (if any) until $\Delta_{\max,l} \leq \tilde{\Delta}_{\max,l}$.³
3. We continue our process in Step 2. Whenever a bucket leaves the sliding window, we can remove the entries in that bucket and the bucket itself.
4. We output the result on demand. We find the K -th largest value of $\min(e)$ of itemset e of size l , say $K_{\min,l}$, for all buckets in the sliding window. Then, we output all itemsets e of size l with $\max(e)$ greater than or equal to $K_{\min,l}$.

Theorem 1 *For any data distribution, the proposed algorithm gives no false negatives.*

²We shall see in Section 4 that r_0 is a bound on the ranks of itemsets that we keep in all buckets.

³Storing more top frequent itemsets can lead to a smaller value of $f_{i,\min}$ and thus $\Delta_{\max,l}$.

Proof: In the algorithm, the K -th largest value of $\min(e)$ (i.e. $K_{\min,l}$) is found. In this step, we make sure that we have found K l -itemsets e where $\min(e) \geq K_{\min,l}$. Also these are at least K itemsets found in the algorithm, which have the chance to become the top K itemsets.

The possible values of frequency of an itemset e are in the range between $\min(e)$ and $\max(e)$. Hence the only other itemsets e which have the chance to become the top K itemsets are those with $\max(e) \geq K_{\min,l}$. Thus, the entries with $\max(e) \geq K_{\min,l}$ are in the output. This ensures that no top K l -itemset will be missed, for all l . \square

The above theorem shows the correctness. It is quite easy to understand all steps in our algorithm except for Step 2b and Step 2c. The purpose of Step 2b is to store as few entries as possible. Meanwhile, the accuracy can be maintained. We prune all entries e with $r_{i,e} > \tilde{r}_i$ even though the entries satisfy $\max(e) \geq K_{\min,l}$. After pruning those entries, we can save a lot of space and can still maintain the accuracy. Step 2c is to maintain the inequality $\Delta_{\max,l} \leq \tilde{\Delta}_{\max,l}$ by making $\Delta_{\max,l}$ smaller and smaller. When $\Delta_{\max,l}$ is smaller, $f_{i,\min}$ is also made to be smaller at the same time. This implicitly means that more itemsets are stored and a smaller value of $\max(e)$ which depends on $f_{i,\min}$ is calculated. When $\max(e)$ is smaller, the number of possible frequencies of each itemset in the range between $\min(e)$ and $\max(e)$ is smaller, leading to a higher accuracy of our algorithm. Thus, the number of false positives in the output can be reduced.

4 Analysis

In this section, we are going to analyze our algorithm, and show some useful properties.

We first consider the number of false positives. From our analysis, we have the following theorem.

Theorem 2 *The frequency difference between any l -itemset which is a false positive returned by the algorithm and the K -th frequent l -itemset is at most $2\tilde{\Delta}_{\max,l}$.*

Recall that $\tilde{\Delta}_{\max,l} = \frac{f_{(1)}}{2K^\theta(1+1/\epsilon)}$. The following table shows $\tilde{\Delta}_{\max,l}$ for some particular values of $f_{(1)}$, K and ϵ . In the following table, we observe that $\tilde{\Delta}_{\max,l}$ is small relative to $f_{(1)}$. By Theorem 2, The frequency difference between any l -itemset which is a false positive returned by the algorithm and the K -th frequent l -itemset is small, which can be shown in Table 1 (a).

In the remaining discussion of this section we assume that the l -itemsets in the sliding window follow the Zipfian distribution. We have derived the following theorem and corollary.

Theorem 3 *Our algorithm outputs the itemsets of ranks r*

$f_{(1)}$	K	ϵ	$\Delta_{max, l}$
1,000	20	0.5	8.33
1,000	20	1	12.50
1,000	10	1	25.00
10,000	20	1	125.00

(a)

θ	ϵ	Max. No. of False Positives
1	1	K
1	0.5	$0.5 \times K$
2	1	$0.41 \times K$
0.5	1	$3 \times K$

(b)

K	n_B	ϵ	θ	m	Max. No. of Entries
20	10	0.5	1	500,000	107,767
20	10	1	1	500,000	71,896
20	10	1	2	500,000	3,741
20	20	1	1	500,000	606,157

(c)

Table 1. Some values of the theoretical bound

Table 2.1

L	Stream Algorithm		BOMO		Ratio
	Structure	Recent Bucket	Structure	Sliding Window	
1	810K	400K	8M	40M	39.66
3	2665K	400K	8M	40M	15.66
5	4667K	400K	8M	40M	9.47
7	6867K	400K	8M	40M	6.60

Table 2.2

s_B	Stream Algorithm		BOMO		Ratio
	Structure	Recent Bucket	Structure	Sliding Window	
2K	5979K	80K	1.6M	8M	1.58
4K	5799K	160K	3.2M	16M	3.22
6K	6069K	240K	4.8M	24M	4.56
8K	5744K	320K	6.4M	32M	6.33
10K	5735K	400K	8M	40M	7.82

Table 2.3

K	Stream Algorithm		BOMO		Ratio
	Structure	Recent Bucket	Structure	Sliding Window	
1	4595K	400K	8M	40M	10.01
10	5680K	400K	8M	40M	7.89
20	5735K	400K	8M	40M	7.82
50	5769K	400K	8M	40M	7.78
100	5780K	400K	8M	40M	7.77

Table 2. Synthetic Data Set: Memory Usage (Default $\epsilon = 1$, $L = 6$, $K = 20$, $s_B = 10K$ and $n_B = 100$)

within the sliding window with the following bound.

$$r \leq K(1 + \epsilon)^{1/\theta}$$

Corollary 1 The number of false positives returned by our algorithm is no more than $K[(1 + \epsilon)^{1/\theta} - 1]$.

Table 1 (b) gives the bound of false positives for some values of θ and ϵ .

Next, we are going to analyze the storage capacity in each bucket and in the whole sliding window. Additionally, we have proved that there is a bound of the entries stored in buckets in the following theorem and corollary.

Theorem 4 Each bucket stores entries of ranks smaller than or equal to r , where

$$r \leq [n_B(n_B - 1)(\frac{1}{2K^\theta(1+\epsilon)} + \frac{n_B-1}{m})^{-1}]^{1/\theta}$$

Note that $r_0 = [n_B(n_B - 1)(\frac{1}{2K^\theta(1+\epsilon)} + \frac{n_B-1}{m})^{-1}]^{1/\theta}$.

Corollary 2 Our algorithm stores at most $n_B[n_B(n_B - 1)(\frac{1}{2K^\theta(1+\epsilon)} + \frac{n_B-1}{m})^{-1}]^{1/\theta}$ entries in all buckets. The memory required is $O(\frac{n_B^{1+2/\theta} m^{1/\theta} K}{\epsilon^{1/\theta}})$.

Proof: By Theorem 4, each bucket should store at most $[n_B(n_B - 1)(\frac{1}{2K^\theta(1+\epsilon)} + \frac{n_B-1}{m})^{-1}]^{1/\theta}$ entries. As there are n_B buckets, the total storage is at most $n_B[n_B(n_B - 1)(\frac{1}{2K^\theta(1+\epsilon)} + \frac{n_B-1}{m})^{-1}]^{1/\theta}$ entries. The memory requirement is thus $O(\frac{n_B^{1+2/\theta} m^{1/\theta} K}{\epsilon^{1/\theta}})$. \square

The above theorem shows that the memory usage of our algorithm is very small. Table 1 (c) shows the number of entries for some particular values of n_B, ϵ and θ . We observe that more buckets, a smaller value of ϵ and a smaller value of θ require more storage space.

Theorem 5 The memory usage used in our algorithm is bounded by

$O\left(n_B[n_B(n_B - 1)(\frac{1}{2K^\theta(1+\epsilon)} + \frac{n_B-1}{m})^{-1}]^{1/\theta}\right) +$
memory for the transactions stored in the most recent bucket

5 Empirical Study

The experiment was conducted with a Pentium IV 1.5GHz PC with 512MB memory on the Linux platform. We compare our algorithm with BOMO. BOMO mines the top K itemsets of at most size L in all transactions of in the sliding window. Thus, BOMO has to store all such transactions. Our algorithm and the BOMO algorithm are implemented in C/C++. The code of the BOMO algorithm is provided by [7]. We make use of the BOMO algorithm in our algorithm to obtain top K' itemsets in the bucket. Synthetic data sets are tested. We have conducted some experiments to study the memory usage, the amount of false positives and the execution time, by varying three factors in our algorithm - (1) L , the largest size of the itemsets to be mined and (2) Bucket Size.

We adopt the IBM synthetic data set[1]. The data set is generated with the following parameters (same as the parameters of [9]): 1,000 items, 3×10^6 transactions, 10 items per transaction on average, and 4 items per frequent itemset on average. We apply the same methodology as [9] to scramble the item-number mapping, in order to simulate the seasonal variations. For every five buckets, we permute 200 items. In all experiments, we set $\theta = 1$. In most previous work, θ was set greater than 1. However, from the analysis of our algorithm, the worst case for the false positives and memory usage occurs when θ is the smallest. Hence we choose a small value for the experiments. For each measurement, we have repeated the experiments 5 times and taken the average.

The experimental results of memory usage with the study of the factors of L , bucket size s_B and K are shown in Table 2. The ratio measured is the ratio of the memory usage of BOMO over that of our algorithm. The ratio shows our algorithm uses much less memory.

The experimental results of the number of false positives over the number of itemsets returned are shown in Table 3. For the number of false positives found in the experiment, we observe that the numbers in the above tables are smaller than $K[(1 + \epsilon)^{1/\theta} - 1]$ as predicted in Theorem 3. That means

Table 3.1

$L \setminus l$	1	2	3	4	5	6	7
1	0.00						
3	0.00	0.00	0.38				
5	0.00	0.00	0.38	0.35	0.74		
7	0.00	0.00	0.38	0.35	0.74	0.33	0.71

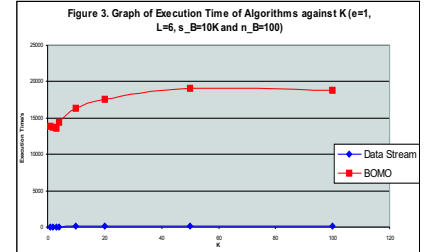
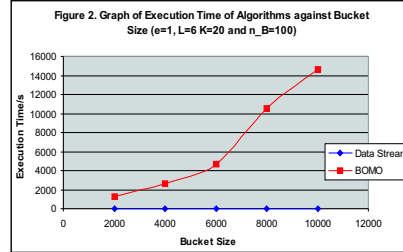
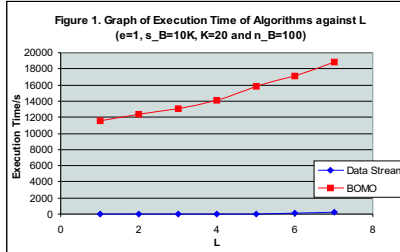
Table 3.2

$s_B \setminus l$	1	2	3	4	5	6
10K	0.00	0.00	0.31	0.29	0.67	0.23
20K	0.00	0.00	0.26	0.33	0.69	0.33
30K	0.00	0.00	0.29	0.33	0.73	0.29
40K	0.00	0.00	0.38	0.30	0.71	0.29
50K	0.00	0.00	0.38	0.35	0.74	0.33

Table 3.3

$K \setminus l$	1	2	3	4	5	6
1	0.00	0.00	0.00	0.00	0.00	0.00
10	0.00	0.09	0.09	0.33	0.23	0.63
20	0.00	0.00	0.38	0.35	0.74	0.33
50	0.00	0.00	0.35	0.38	0.39	0.82
100	0.00	0.00	0.27	0.45	0.84	0.64

Table 3. Synthetic Data Set: Fraction of False Positives (Default $\epsilon = 1$, $L = 6$, $K = 20$, $s_B = 10K$ and $n_B = 100$)



the experimental results give a verification of our analysis.

The experimental results of the execution time are shown in Figures 1, 2 and 3. We observe that our Data Stream algorithm runs much faster than BOMO algorithm. This is because the process of finding top K itemsets in our algorithm is more efficient due to a smaller data set in each bucket. Besides, the overhead of the combination of different results in different buckets is small. One more reason is that for every bucket to be processed, our data stream algorithm needs to manipulate one bucket only but BOMO requires to handle all buckets in the sliding window. Thus, our algorithm runs much faster.

Let us take a closer look at the false positives in our experiments. When we examine the frequencies of the false positives, they have actually a very small differences from the K -th frequent itemset in all cases. For example, in the experiment by varying L (the largest size of the itemsets to be mined), if $K = 20$, the actual count of the K -th frequent 4-itemset is 1733. Although there are 11 false positives in the output in Table 3, all their frequencies are greater than 1730, which means that the frequency difference is at most 3. The small frequency difference holds for all cases. The bound in Theorem 2 is only a worst-case upper bound. In practice, the count difference did not reach this bound.

6 Conclusion

In this paper, we address the problem of mining the K most frequent itemsets in a sliding window in a data stream. We propose an algorithm to estimate these K itemsets in the data stream. We prove that our algorithm gives no false negatives for any data distribution. It outputs at most $K(1 + \epsilon)^{1/\theta}$ top frequent itemsets and stores a small number of entries for the Zipfian data distribution. We have conducted experiments to show that our algorithm can manipulate the data stream efficiently and both the memory usage

and the execution time are many times smaller compared with a naive approach.

Acknowledgements We thank Y.L. Cheung for providing us the coding of BOMO. This research was supported by the RGC Earmarked Research Grant of HKSAR CUHK 4179/01E, and the Innovation and Technology Fund (ITF) in the HKSAR [ITS/069/03].

References

- [1] R. Agrawal. Ibm synthetic data generator, <http://www.almaden.ibm.com/cs/quest/syndata.html>.
- [2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS*, 2002.
- [3] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *SIGMOD*, 2003.
- [4] C.-R. Lin C.-H. Lee and M.-S. Chen. Sliding-window filtering: An efficient algorithm for incremental mining. In *Intl. Conf. on Information and Knowledge Management*, 2001.
- [5] J. H. Chang and W. S. Lee. Finding recent frequent itemsets adaptively over online data streams. In *SIGKDD*, 2003.
- [6] Y.-L. Cheung and A. W.-C. Fu. An fp-tree approach for mining n -most interesting itemsets. In *SPIE Conference on Data Mining*, 2002.
- [7] Y.-L. Cheung and A. W.-C. Fu. Mining frequent itemsets without support threshold: With and without item constraints. In *IEEE Trans. on Knowledge and Data Engineering*, to appear 2004.
- [8] M. Datar, A. Gionis, P. Indyk, and R. Motwani. "maintaining stream statistics over sliding windows". In *SIAM Journal on Computing*, 2002.
- [9] C. Giannella, J. Han, J. Pei, X. Yan, and P.S. Yu. Mining frequent patterns in data streams at multiple time granularities. In *Next Generation Data Mining*, 2003.
- [10] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD*, 1998.
- [11] A. Gilbert, Y. Kotidis, and S. Muthukrishnan. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, 2001.
- [12] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *VLDB*, 2002.

Hierarchical Document Classification Using Automatically Generated Hierarchy

Tao Li*

Shenghuo Zhu[†]

Abstract

Automated text categorization has witnessed a booming interest with the exponential growth of information and the ever-increasing needs for organizations. The underlying hierarchical structure identifies the relationships of dependence between different categories and provides valuable sources of information for categorization. Although considerable research has been conducted in the field of hierarchical document categorization, little has been done on automatic generation of topic hierarchies. In this paper, we propose the method of using linear discriminant projection to generate more meaningful intermediate levels of hierarchies in large flat sets of classes. The linear discriminant projection approach first transforms all documents onto a low-dimensional space and then clusters the categories into hierarchies accordingly. The paper also investigates the effect of using generated hierarchical structure for text classification. Our experiments show that generated hierarchies improve classification performance in most cases. A preliminary short version of the paper has appeared in [8].

1 Introduction

Many studies in document classification focus on *flat classification*, in which the predefined categories are treated individually and equally so that no structures exist to define relationships among them [10, 1]. Limitations to the flat classification approach exists in the fact that, as the Internet grows, the number of possible categories increases and the borderlines between document classes are blurred. To resolve this issue, recently several researchers have studied the use of hierarchies for text classification and obtained promising results [6, 1, 9]. However, the previous studies were mostly conducted on corpora with predefined hierarchical structures and little has been done on automatic generation of topic hierarchies.

This motivates us to address the issue of automatically building hierarchies of documents. Such studies are meaningful for the following reasons: First, manual building of hierarchies is an expensive task since the process requires domain experts to evaluate the relevance of documents to the topics. Second, there may exist document domains in which there are no natural hierarchies and even domain experts have difficulties in evaluating the semantics. Third, automatic hierarchy generation based upon document statistics may generate hierarchies that provide better statistical correlations among categories. Once such a statistically more significant hierarchy has been build, the hierarchy can be incor-

porated into various classification methods to help achieve better performance.

2 Linear Discriminant Projection Approach

The first step in hierarchy generation is to define the similarity measure between categories upon which hierarchies are built. In this section, we present the linear discriminant projection approach for inferring class relationships. Its core idea is to compare the class representatives in a low-dimensional space so that the comparison is more “meaningful”. More specifically, after finding the transformation, the similarity between classes is defined to be the distance between their centroids in the transformed spaces. The notations used through the discussion of this paper are listed in the Table 1.

Notations	Descriptions
A	document-term matrix
n	number of data points, i.e., documents
N	number of the dimensions, i.e, terms
k	number of class
S_i	covariance matrix of the i -th class
S_b	between-class scatter matrix
S_w	within-class scatter matrix
G	reduction transformation
m_i	centroid of the i -th class
m	global centroid of the training set

Table 1: Notations

2.1 Finding the Transformation Given a document-term matrix $A = (a_{ij}) \in \mathcal{R}^{n \times N}$, where each row corresponds to a document and each column corresponds to a particular term, we consider finding a linear transformation $G \in \mathcal{R}^{N \times \ell}$ ($\ell < N$) that maps each row a_i ($1 \leq i \leq n$) of A in the N -dimensional space to a row y_i in the ℓ -dimensional space. The resulting data matrix $A^L = AG \in \mathcal{R}^{n \times \ell}$ contains ℓ columns, i.e. there are ℓ features for each document in the reduced (transformed) space. It is also clear that the features in the reduced space are linear combinations of the features in the original high dimensional space, where the coefficients of the linear combinations depend on the transformation G . Linear discriminant projection tries to compute the optimal transformation matrix G such that the

*School of Computer Science, Florida International University, taoli@cs.fiu.edu.

[†]NEC Labs America, Inc., zsh@sv.nec-labs.com. Major work was completed when the author was in University of Rochester.

class structure is preserved. More details are given below.

Assume there are k classes in the data set. Suppose m_i , S_i , P_i are the mean vector, covariance matrix, and a prior probability of the i -th class, respectively, and m is the total mean. For the covariance matrix S_i for the i th class, we can decompose it as $S_i = X_i X_i^T$, where X_i has the same number of columns as the number of data points in the i -th class. Define the matrices

$$H_b = [\sqrt{P_1}(m_1 - m), \dots, \sqrt{P_k}(m_k - m)] \in \mathbb{R}^{N \times k},$$

$$H_w = [\sqrt{P_1}X_1, \dots, \sqrt{P_k}X_k] \in \mathbb{R}^{N \times n}.$$

Then the between-class scatter matrix S_b , the within-class scatter matrix S_w , and the total scatter matrix S_t are defined as follows [3]:

$$S_b = \sum_{i=1}^k P_i (m_i - m)(m_i - m)^T = H_b H_b^T,$$

$$S_w = \sum_{i=1}^k P_i S_i = H_w H_w^T.$$

In the lower-dimensional space resulting from the linear transformation G , the within-cluster and between-cluster matrices become

$$S_w^L = (G^T H_w)(G^T H_w)^T = G^T S_w G,$$

$$S_b^L = (G^T H_b)(G^T H_b)^T = G^T S_b G.$$

An optimal transformation G would maximize $\text{Trace}(S_b^L)$ and minimize $\text{Trace}(S_w^L)$. A common optimization for computing optimal G is

$$G^* = \arg \max_G \text{Trace} \left((G^T S_w G)^{-1} G^T S_b G \right).$$

The solution can be readily obtained by solving an eigenvalue decomposition problem on $S_w^{-1} S_b$, provided that the within-class scatter matrix S_w is nonsingular. Since the rank of the between-class scatter matrix is bounded above by $k-1$, there are at most $k-1$ discriminant vectors.

2.2 Extension on General Cases In general, the within-class scatter matrix S_w may be singular especially for document-term matrix where the dimension is very high. A common way to deal with it is to use generalized eigenvalue decomposition [4, 7]

Let $K = [H_b \ H_w]^T$, which is a $k+n$ by N matrix. By the generalized singular value decomposition, there exist orthogonal matrices $U \in \mathbb{R}^{k \times k}$, $V \in \mathbb{R}^{n \times n}$, and a nonsingular matrix $X \in \mathbb{R}^{N \times N}$, such that

$$(2.1) \quad \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} K X = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \\ \Sigma_2 & 0 \\ 0 & 0 \end{bmatrix},$$

where

$$\Sigma_1 = \text{diag}(\overbrace{1, \dots, 1}^r, \alpha_1, \dots, \alpha_s, \overbrace{0, \dots, 0}^{t-r-s}),$$

$$\Sigma_2 = \text{diag}(\overbrace{0, \dots, 0}^r, \beta_1, \dots, \beta_s, \overbrace{1, \dots, 1}^{t-r-s}),$$

$$t = \text{rank}(K), \quad r = t - \text{rank}(H_w^T),$$

$$s = \text{rank}(H_b) + \text{rank}(H_w) - t,$$

satisfying

$$1 > \alpha_1 \geq \dots \geq \alpha_s > 0,$$

$$0 < \beta_1 \leq \dots \leq \beta_s < 1,$$

$$\text{and} \quad \alpha_i^2 + \beta_i^2 = 1 \quad \text{for } i = 1, \dots, s.$$

From Eq. (2.1), we have

$$(X^T H_b)(X^T H_b)^T = \begin{bmatrix} \Sigma_1^T \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix},$$

$$(X^T H_w)(X^T H_w)^T = \begin{bmatrix} \Sigma_2^T \Sigma_2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Hence a natural extension of the proposed linear discriminant projection in Section 2.1 is to choose the first $q = r + s$ columns of the matrix X in Eq. (2.1) as the transformation matrix G^* .

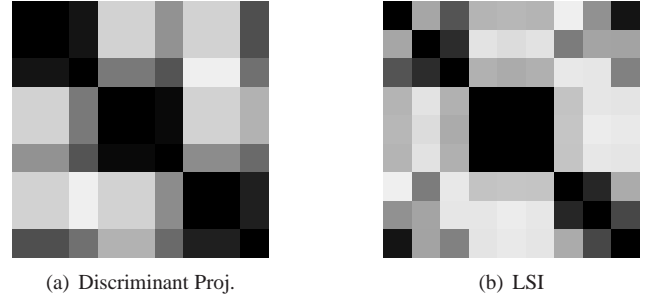


Figure 1: Document similarity. Each block represents the similarity between the corresponding row and column documents. The darker the contrast, the more similar the documents. For perfect class structure preserving, we expect three consecutive dark squares along the main diagonal.

2.3 Defining the Similarity After finding the transformation G , we define the similarity between classes to be the distance between their centroids in the transformed spaces. In other words, two categories are similar if they are “close” to each other in the transformed space. The linear discriminant projection finds the transformation that preserves the class structure by minimizing the sum of squared within-class scatter while maximizing the sum of squared between-class scatter and hence the distances in the transformed space should be able to reflect the inherent structure of the dataset.

To make it more clear on the linear discriminant projection approach, we compare the method with the well-known Latent Semantic Indexing (LSI) [2] and give a concrete example. LSI projects a document onto the latent semantic space. Although LSI has been proven to be extremely useful in various information retrieval tasks, it is not an optimal transformation for text categorization since LSI is completely unsupervised. In other words, LSI aims at optimal transformation of the original data into the lower dimensional space in terms of mean squared error but it pays no attention to the underlying class structure. Linear discriminant projection explicitly utilizes the intra-class and inter-class covariance matrices and tends to preserve the class structure.

We consider a dataset consisting of nine sentences from three different topics: user interaction, graph theory and distributed systems:

- 1(1) Human interface for user response
- 2(1) A survey of user opinion of computer system response time
- 3(1) Relation of user-perceived response time to error measurement
- 4(2) The generation of random, binary, unordered trees
- 5(2) The intersection graph of paths in trees
- 6(2) Graph Minors IV: Widths of trees and well-quasi-ordering
- 7(3) A survey of distributed shared memory system
- 8(3) RADAR: A multi-user distributed system
- 9(3) Management interface tools for distributed computer system

By removing words/terms that occur only once, we obtain the document-term matrix. Suppose that the first and second sentences in each class are used for training data. Then the transformation shown in Figure 1(a) is obtained and the plot of the LSI algorithm in Figure 1(b). The example shows that linear discriminant projection has discrimination power and is able to reflect the inherent similarity structure of the classes. Hence the distance between the centroids is a good measure for the similarity between categories.

3 Hierarchy Generation

After obtaining the similarities/distances between classes, we use the Hierarchical Agglomerative Clustering (HAC) algorithm of [5] to generate automatic topic hierarchies from a given set of flat classes. The result of hierarchical clustering is a dendrogram where similar classes are organized into hierarchies. We choose UPGMA (Unweighted Pair-Groups Method Average method), which is known to be simple, efficient and stable [5]. In UPGMA, the average distance between clusters is calculated from the distance between each point in a cluster and all other points in another cluster. The two clusters with the lowest average distance are joined together to form the new cluster.

4 Experiments on hierarchy Generation

We use a wide range of datasets in our experiments and anticipate that these data sets would provide us enough insights on automatic hierarchy generation. The datasets and their characteristics are summarized in Table 2. **20Newsgroups**

Datasets	# documents	# classes
20Newsgroups	20,000	20
WebKB	8,280	7
Industry Sector	9,637	105
Reuters-top 10	2,900	10
Reuters-2	8,000	42
K-dataset	2,340	20

Table 2: Data Sets Descriptions

dataset¹ contains about 20,000 articles evenly divided among 20 Usenet newsgroups. **WebKB** dataset contains web-pages gathered from university computer science departments. There are about 8300 documents and they are divided into seven categories: student, faculty, staff, course, project, department and other. **Industry Sector** dataset consists of company homepages classified in a hierarchy of industry sectors². **Reuters**: The Reuters-21578 Text Categorization Test collection contains documents collected from the Reuters newswire in 1987. It is a standard text categorization benchmark and contains 135 categories. In our experiments, we used two subsets of the data collection. The first one includes the ten most frequent categories among the 135 topics, which we call Reuters-top10. The second one contains the documents that have unique topics (documents that have multiple class assignments are ignored), which we call Reuters-2. **K-dataset**³ contains 2340 documents consisting news articles from Reuters news service via the Web in October 1997.

4.1 Data Preprocessing To preprocess the datasets, we remove the stop words using a standard stop list and perform the stemming operations with a Porter stemmer. All HTML tags and all header fields except subject and organization are ignored. In all our experiments, we first randomly choose 70% for hierarchy building (and later training in categorization), the remaining 30% is then used for testing. The 70% training set is further preprocessed by selecting the top 1000 words by information gain. The feature selection is done with the rainbow package⁴. All of our experiments are performed on a P4 2GHz machine with 512M memory running Linux 2.4.9-31.

¹<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html>.

²<http://www.cs.cmu.edu/~TextLearning/datasets.html>.

³<ftp://ftp.cs.umn.edu/dept/users/boley/PDDPdata/>.

⁴<http://www.cs.cmu.edu/~mccalum/bow>.

4.2 Experimental Results Figure 2 shows the hierarchies of WebKB, 20Newsgroups and Reuters-top10 built via linear discriminant projection⁵ The block in the graphs represents the similarity between the corresponding row and column document categories and the darker the more similar.

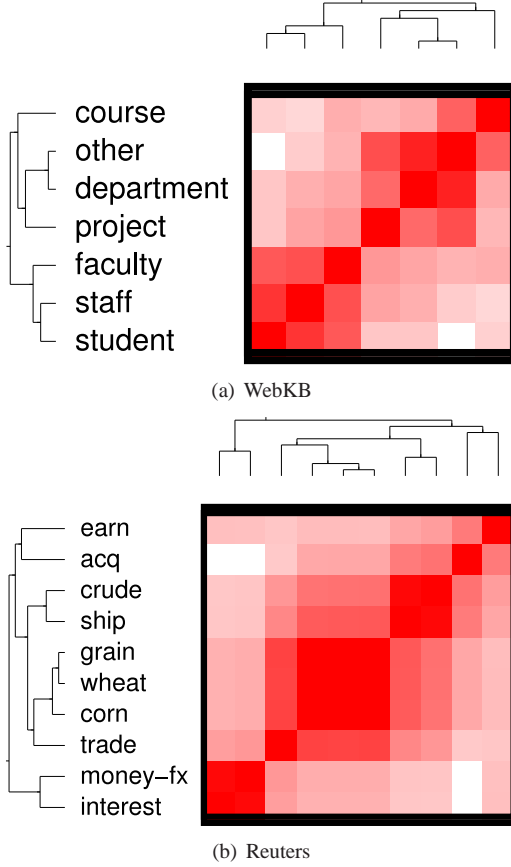


Figure 2: Hierarchies of WebKB, 20Newsgroups and Reuters-top 10 using linear discriminant projection.

We can observe from the dendrogram the semantic similarity of classes. For example, on WebKB, *faculty*, *staff* and *student* are close to each other. Note that *faculty*, *staff* and *student* are people and they are different from other datasets. Hence the linear discriminant projection approach tends to group them together. We can observe similar phenomena on 20Newsgroup and Reuters-top10. On 20Newsgroups, *talk.politics.guns* and *talk.politics.misc* are grouped together. On Reuters-top10, for example, the close pairs of classes using linear discriminant projection are: (*ship*, *crude*), (*money-fx*, *interest*), (*grain*, *wheat*) and (*earn*, *acq*).

5 Exploiting the Generated Hierarchy for Classification

In this section, we investigate the effects of exploiting the automatically generated hierarchy for classification and use

classification accuracy as the evaluation measure.

An obvious approach to utilization of the hierarchy is a top-down level-based approach that arranges the clusters in a two-level tree hierarchy and trains a classifier at each internal node. We analyze the generated dendrogram to determine the clusters that provide maximum inter-class separation and find the best grouping of classes at the top level. The dendrogram is scanned in a bottom-up fashion to find the distances at which successive clusters get merged. We clip the dendrogram at the point where the cluster merge distances begin increasing sharply. In our experiments, we clip the dendrogram when the current merge distance is at least two times larger than previous one. For example, on 20Newsgroups dataset with linear discriminant projection approach, we have 8 top-level groups as shown in Table 3. The experimental results reported here are obtained via two-level classification.

groups	members
1	<i>alt.atheism</i> , <i>talk.region.misc</i> , <i>talk.politics.guns</i> , <i>talk.politics.misc</i> , <i>talk.politics.mideas</i>
2	<i>sci.space</i> , <i>sci.med</i> <i>sci.electronic</i>
3	<i>comp.os.mswindows.misc</i> , <i>comp.sys.ibm.pc.hardware</i> , <i>comp.graphs</i> , <i>comp.sys.mac.hardware</i> , <i>comp.windows.x</i>
4	<i>rec.sport.baseball</i> , <i>rec.sport.hockey</i> , <i>rec.motorcycles</i>
5	<i>misc.forsale</i>
6	<i>soc.religion.christian</i>
7	<i>rec.autos</i>
8	<i>sci.crypt</i>

Table 3: Top level groups for 20Newsgroups via linear projection.

LIBSVM⁶ is used as our classifier. LIBSVM is a library for support vector classification and regression and supports multi-class classification. In addition, we use linear kernel in all our experiments as it gives best results on our experiments.

We first build a top-level classifier (L1 classifier) to discriminate among the top-level clusters of labels. At the second level (L2) we build classifiers within each cluster of classes. Each L2 classifier can concentrate on a smaller set of classes that confuse with each other. In practice, each classifier has to deal with a more easily separable problem, and can use an independently optimized feature set; this should lead to slight improvements in accuracy apart from the gain in training and testing speed. Table 4 gives the performance comparisons of flat classification with hierarchical classification. We observe the improved performance on all datasets.

From Table 4, we observe that Reuters-top10 has the most significant gain in accuracy using hierarchy. Figure 3 presents the accuracy comparison for each class of Reuters-top10. Shown in Figure 3, each class' accuracy is improved by using the generated hierarchy except the accuracy of

⁵Due to the space limit, we do not include the hierarchies on K-dataset, Reuters-2 and Industry sector.

⁶<http://www.csie.ntu.edu.tw/~cjlin/libsvm>

Datasets	Flat	Linear Projection	Linear Projection
		Level One	Overall
20Newsgroups	0.952	0.985	0.963
WebKB	0.791	0.860	0.804
Industry Sector	0.691	0.739	0.727
Reuters-top 10	0.826	0.963	0.921
Reuters-2	0.923	0.938	0.927
K-dataset	0.915	0.961	0.921

Table 4: Accuracy Table. The flat column gives the accuracy of flat classification, the “Level One” column shows the level one accuracy while the “Overall” column represent the overall accuracy.

trade stays unchanged. The accuracy of *corn* is improved significantly from about 7% to 60%. In flat classification, almost all the documents in *corn* class are misclassified to *grain* and *wheat* classes. Using hierarchical classification, by grouping *corn*, *grain* and *wheat* together, A second-level classifier using an independently optimized feature set can then be designed to focus on separation of the three similar classes. The performance improvement is thus obtained.

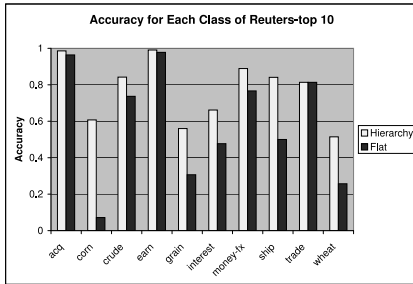


Figure 3: Accuracy comparison on each class of Reuters-top10

6 Manually-Built Hierarchy vs Generated Hierarchy

Once we have the automatic approach for hierarchy generation, it is natural to compare the generated hierarchy with the manually-built one. In this section, we illustrate their difference via three experiments on 20Newsgroups, WebKB and Reuters-top10.

Due to space limit, we only present the human-built (two-level) hierarchies for the 20Newsgroups as shown in Table 5. The manual hierarchy is generated by the authors to group the categories with strong confidence. To further understand the differences between two kinds of hierarchies, we also compare their hierarchical categorization performances, as listed in Table 6. For comparison purposes, the experimental results are based on two-level classifiers.

As you can observe from the comparisons, human-built hierarchy is purely based on “human semantics”, but not necessarily optimized for classification purpose. In all the three datasets, using the automatic generated hierarchy, the classification accuracies are slightly higher than those using human-built hierarchy. Hence, an important research

direction is to combine the automatic and manual approaches for generating both statistically significant and intuitively meaningful hierarchies.

groups	members
1	<i>talk.region.misc, talk.politics.guns, talk.politics.misc, talk.politics.mideas</i>
2	<i>sci.electronic, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware</i>
3	<i>comp.os.mswindows.misc, sci.crypt, comp.graphs, comp.windows.x</i>
4	<i>rec.sport.baseball, rec.sport.hockey</i>
5	<i>misc.forsale</i>
6	<i>alt.atheism, soc.religion.christian</i>
7	<i>rec.autos, rec.motorcycles</i>
8	<i>sci.space, sci.med</i>

Table 5: Human-generated 8 top-level groups for 20Newsgroups.

Datasets	Human-generated	Automatic Generated
20Newsgroups	(0.956, 0.954)	(0.985, 0.963)
WebKB	(0.811, 0.795)	(0.860, 0.804)
Reuters-top 10	(0.928, 0.9475)	(0.963, 0.901)

Table 6: Performance comparisons of human-generated hierarchy with automatic generated hierarchy. The entries are in the format of (level one, flat). The accuracy of automatic generated hierarchy was taken from the linear projection approach.

References

- [1] D’Alessio, S., Murray, K., Schiaffino, R., & Kershenbaum, A. (2000). The effect of using hierarchical classifiers in text categorization. *RIAO-00*.
- [2] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41, 391–407.
- [3] Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. New York: Academic Press. 2nd edition.
- [4] Howland, P., & Park, H. (2004). Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE PAMI*, 26, 995–1006.
- [5] Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice Hall.
- [6] Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. *ICML*.
- [7] Li, T., Zhu, S., & Ogihara, M. (2003a). Efficient multi-way text categorization via generalized discriminant analysis. *ACM CIKM* (pp. 317–324).
- [8] Li, T., Zhu, S., & Ogihara, M. (2003b). Topic hierarchy generation via linear discriminant projection. *ACM SIGIR* (pp. 421–422).
- [9] Sun, A., & Lim, E.-P. (2001). Hierarchical text classification and evaluation. *ICDM* (pp. 521–528).
- [10] Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. *SIGIR*.

On Clustering Binary Data

Tao Li*

Shenghuo Zhu†

Abstract

Clustering is the problem of identifying the distribution of patterns and intrinsic correlations in large data sets by partitioning the data points into similarity classes. This paper studies the problem of clustering binary data. This is the case for market basket datasets where the transactions contain items and for document datasets where the documents contain “bag of words”. The contribution of the paper is two-fold. First a new clustering model is presented. The model treats the data and features equally, based on their symmetric association relations, and explicitly describes the data assignments as well as feature assignments. An iterative alternating least-squares procedure is used for optimization. Second, a unified view of binary data clustering is presented by examining the connections among various clustering criteria.

1 Introduction

The problem of clustering data arises in many disciplines and has a wide range of applications. Intuitively, clustering is the problem of partitioning a finite set of points in a multi-dimensional space into classes (called clusters) so that (i) the points belonging to the same class are *similar* and (ii) the points belonging to different classes are *dissimilar*.

In this paper, we focus our attention on binary datasets. Binary data have been occupying a special place in the domain of data analysis. Typical applications for binary data clustering include market basket data clustering and document clustering. For market basket data, each data transaction can be represented as a binary vector where each element indicates whether or not any of the corresponding item/product was purchased. For document clustering, each document can be represented as a binary vector where each element indicates whether a given word/term was present or not.

The first contribution of the paper is the introduction of a new clustering model along with a clustering algorithm. A distinctive characteristic of the binary data is that the features (attributes) they include have the same nature as the data they intend to account for: both are binary. This characteristic implies the symmetric association relations between data and features: if the set of data points is associated to

the set of features, then the set of attributes is associated to the set of data points and vice versa. The association relation suggests a new clustering model where the data and features are treated equally. Our new clustering model, *BMD* (Binary Matrix Decomposition), explicitly describes the data assignments (assigning data points into clusters) as well as feature assignments (assigning features into clusters). The clustering problem is then presented as binary matrix decomposition, which is solved via an iterative alternating least-squares optimization procedure. The procedure simultaneously performs two tasks: data reduction (assigning data points into clusters) and feature identification (identifying features associated with each cluster). By explicitly feature assignments, *BMD* produces interpretable descriptions of the resulting clusters. In addition, by iterative feature identification, *BMD* performs an implicit adaptive feature selection at each iteration and flexibly measures the distances between data points. Therefore it works well for high-dimensional data.

The second contribution of this paper is the presentation of a unified view for binary data clustering by examining the connections among various clustering criteria. In particular, we show the equivalence among the matrix decomposition, dissimilarity coefficients, minimum description length and entropy-based approach.

2 BMD Clustering

In this section, we describe the new clustering algorithm. Section 2.1 introduces the cluster model. Section 2.2 and Section 2.3 present the optimization procedure and the refining methods, respectively. Section 2.4 gives an example to illustrate the algorithm.

2.1 The Clustering Model Suppose the dataset X has n instances, having r features each. Then X can be viewed as a subset of R^r as well as a member of $R^{n \times r}$. The cluster model is determined by two matrices: the data matrix $D_{n \times K} = (d_{ik})$ and the feature matrix $F_{r \times K} = (f_{jk})$, where K is the number of clusters.

$$d_{ik} = \begin{cases} 1 & \text{Data point } i \text{ belongs to cluster } k \\ 0 & \text{Otherwise} \end{cases}$$
$$f_{jk} = \begin{cases} 1 & \text{Attribute } j \text{ belongs to cluster } k \\ 0 & \text{Otherwise} \end{cases}$$

*School of Computer Science, Florida International University, taoli@cs.fiu.edu.

†NEC Labs America, Inc., zsh@sv.nec-labs.com. Major work was completed when the author was in University of Rochester.

The data (respectively, feature) matrix specifies the cluster memberships for the corresponding data (respectively, features).

For clustering, it is customary to assume that each data point is assigned to one and only one cluster, i.e., $\sum_{k=1}^K d_{ik} = 1$ holds for $j = 1, \dots, n$. Given representation (D, F) , basically, D denotes the cluster assignments of data points and F indicates the feature representations of clusters. The ij -th entry of DF^T is the dot product of the i -th row of D and the j -th row of F , and indicates whether the j -th feature will be present in the i -instance. Hence, DF^T can be interpreted as the approximation of the original data X . Our goal is then to find a (D, F) that minimizes the squared error between X and its approximation DF^T .

$$(2.1) \quad \underset{D, F}{\operatorname{argmin}} O = \frac{1}{2} \|X - DF^T\|_F^2,$$

where $\|X\|_F$ is the Frobenius norm of the matrix X , i.e., $\sqrt{\sum_{i,j} x_{ij}^2}$. With the formulation, we transform the data clustering problem into the computation of D and F that minimizes the criterion O .

2.2 Optimization Procedure The objective criterion can be expressed as

$$(2.2) \quad \begin{aligned} O_{D,F} &= \frac{1}{2} \|X - DF^T\|_F^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m \left(x_{ij} - \sum_{k=1}^K d_{ik} f_{kj} \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^K d_{ik} \sum_{j=1}^m (x_{ij} - f_{kj})^2 \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^K d_{ik} \sum_{j=1}^m (x_{ij} - y_{kj})^2 \\ &\quad + \frac{1}{2} \sum_{k=1}^K n_k \sum_{j=1}^m (y_{kj} - f_{kj})^2, \end{aligned}$$

where $y_{kj} = \frac{1}{n_k} \sum_{i=1}^n d_{ik} x_{ij}$ and $n_k = \sum_{i=1}^n d_{ik}$ (note that we use f_{kj} to denote the entry of F^T). The objective function can be minimized via an alternating least-squares procedure by alternatively optimizing one of D or F while fixing the other.

Given an estimate of F , new least-squares estimates of the entries of D can be determined by assigning each data point to the closest cluster as follows:

$$(2.3) \quad \hat{d}_{ik} = \begin{cases} 1 & \text{if } \sum_{j=1}^m (x_{ij} - f_{kj})^2 < \sum_{j=1}^m (x_{ij} - f_{lj})^2 \\ & \text{for } l = 1, \dots, K, l \neq k \\ 0 & \text{Otherwise} \end{cases}$$

When D is fixed, $O_{D,F}$ can be minimized with respect to F by minimizing the second part of Equation (2.2):

$$O'(F) = \frac{1}{2} \sum_{k=1}^K n_k \sum_{j=1}^m (y_{kj} - f_{kj})^2.$$

Note that y_{kj} can be thought of as the probability that the j -th feature is present in the k -th cluster. Since each f_{kj} is binary¹, i.e., either 0 or 1, $O'(F)$ is minimized by:

$$(2.4) \quad \hat{f}_{kj} = \begin{cases} 1 & \text{if } y_{kj} > 1/2 \\ 0 & \text{Otherwise} \end{cases}$$

In practice, if a feature has similar association to all clusters, then it is viewed as an outlier at the current stage.

The optimization procedure for minimizing Equation (2.2) alternates between updating D based on Equation (2.3) and assigning features using Equation (2.4). After each iteration, we compute the value of the objective criterion $O(D, F)$. If the value is decreased, we then repeat the process; otherwise, the process has arrived at a local minimum. Since the *BMD* procedure monotonically decreases the objective criterion, it converges to a local optimum. The clustering procedure is shown in Algorithm 1.

Algorithm 1 BMD: clustering procedure

Input: (data points: $X_{n \times r}$, # of classes: K)

Output: D : cluster assignment;

F : feature assignment;

begin

1. **Initialization:**

1.1 Initialize D

1.2 Compute F based on Equation (2.4)

1.3 Compute $O_0 = O(D, F)$

2. **Iteration:**

begin

2.1 Update D given F (via Equation (2.3))

2.2 Compute F given D (via Equation (2.4))

2.3 Compute the value of $O_1 = O(D, F)$;

2.4 if $O_1 < O_0$

2.4.1 $O_0 = O_1$

2.4.2 Repeat from 2.1

2.5 else

2.5.1 break; (Converges)

end

3. **Return** D, F ;

end

2.3 Refining Methods Clustering results are sensitive to initial seed points. The initialization step sets the initial values for D and F . Since D is a binary matrix and has at most one occurrence of 1 in each row, it is very sensitive to initial assignments. To overcome the sensitivity of initialization, we refine the procedure. Its idea is to use mutual information to measure the similarity between a pair

¹If the entries of F are arbitrary, then the optimization here can be performed via singular value decomposition.

of clustering results. In addition, clustering a large data set may be time-consuming. To speed up the algorithm, a small set of data points, for example, 1% of the entire data set, may be selected as a *bootstrap* data set. The clustering algorithm is first executed on the bootstrap data set. Then, the algorithm is run on the entire data set using the data assignments obtained from the bootstrap data set (instead of using random seed points).

2.4 An Example To illustrate how *BMD* works, we show an artificial example, a dataset consisting of six sentences from two clusters: user interaction and distributed systems, as shown in Figure 1.

- 1(1) An system for user response
- 2(1) A survey of user interaction
on computer response
- 3(1) Response for interaction
- 4(2) A multi-user distributed system
- 5(2) A survey of distributed computer system
- 6(2) distributed systems

Figure 1: The six example sentences. The numbers within the parentheses are the clusters: 1=user interaction, 2=distributed system.

After preprocessing, we get the dataset as in Table 1. In this example, D is a 6×2 matrix and F is a 7×2 matrix. Initially, the data points 2 and 5 are chosen as seed points, where the data point 2 is in class 1 and the data point 5 is in class 2. Initialization is then performed on the seed points to get the initial feature assignments. After *Step 1.2*, features a , b and c are positive in class 1, e and f are positive in class 2, and d and g are outliers. In other words, $F(a, 1) = F(b, 1) = F(c, 1) = 1$, $F(e, 2) = F(f, 2) = 1$, and all the other entries² of F are 0. Then *Step 2.1* assigns data points 1, 2 and 3 to class 1 and data points 4, 5 and 6 to class 2. then *Step 2.2* asserts a , b and c are positive in class 1, d , e and f are positive in class 2, and g is an outlier. In the next iteration, the objective criterion does not change. At this point the algorithm stops. The resulting clusters are: For data points, class 1 contains 1, 2, and 3 and class 2 contains 4, 5, and 6. For features, a , b and c are positive in class 1, d , e and f are positive in class 2 while g is an outlier.

We have conducted experiments on real datasets to evaluate the performance of our *BMD* algorithm and compare it with other standard clustering algorithms. Experimental results on suggest that *BMD* is a viable and competitive binary clustering algorithm. Due to space limit, we omitted the experiment details.

3 Binary Data Clustering

In this section, a unified view on binary data clustering is presented by examining the relations among various binary

data point	feature						
	a	b	c	d	e	f	g
1	1	1	0	0	1	0	0
2	1	1	1	1	0	0	1
3	1	0	1	0	0	0	0
4	0	1	0	0	1	1	0
5	0	0	0	1	1	1	1
6	0	0	0	1	0	1	0

Table 1: A bag-of-words representation of the sentences. a , b , c , d , e , f , g , correspond to the presence of *response*, *user*, *interaction*, *computer*, *system*, *distributed* and *survey*, respectively.

clustering approaches. Section 3.1 sets down the notation, Section 3.2, Section 3.3 and Section 3.4 discuss the binary dissimilarity coefficients, minimum description length, and the entropy-based approach respectively. The unified view on binary clustering is summarized in Figure 2. Note that the relations of maximum likelihood principle with the entropy-based criterion and with minimum description length (MDL) are known in machine learning literature [8].

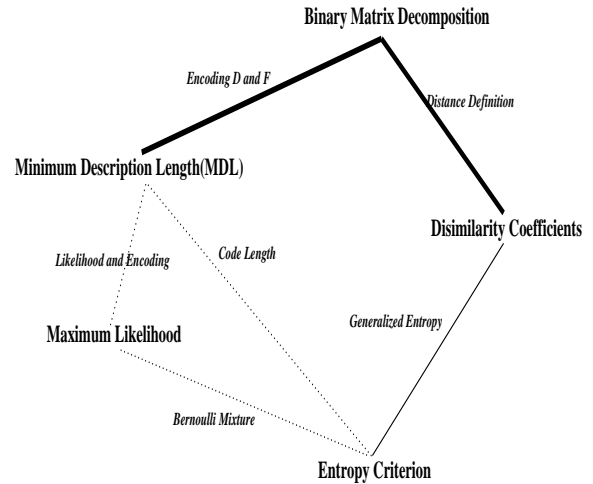


Figure 2: A Unified View on Binary Clustering. The thick lines are relations first shown in this paper, the dotted lines are well-known facts, and the thin line is first discussed in [7].

3.1 Notation We first set down some notation. Suppose that a set of n r -dimensional binary data vectors, X , represented as an $n \times r$ matrix, (x_{ij}) , is partitioned into K classes $C = (C_1, \dots, C_K)$ and we want the points within each class are *similar* to each other. We view C as a partition of the indices $\{1, \dots, n\}$. So, for all i , $1 \leq i \leq n$, and k , $1 \leq k \leq K$, we write $i \in C_k$ to mean that the i -th vector belongs to the k -th class. Let $N = nr$. For each k , $1 \leq k \leq K$, let $n_k = \|C_k\|$, $N_k = n_k r$, and for each j , $1 \leq j \leq r$, let $N_{j,k,1} = \sum_{i \in C_k} x_{ij}$ and $N_{j,k,0} = n_k - N_{j,k,1}$. Also, for each j , $1 \leq j \leq r$, let

²We use a, b, c, d, e, f, g to denote the rows of F .

$N_{j,1} = \sum_{i=1}^n x_{ij}$ and $N_{j,0} = n - N_{j,1}$. We use x_i as a point variable.

3.2 Binary Dissimilarity Coefficients A popular partition-based criterion (within-cluster) for clustering is to minimize the summation of distances/dissimilarities inside the cluster. The within-cluster criterion can be described as minimizing

$$(3.5) \quad S(C) = \sum_{k=1}^K \frac{1}{n_k} \sum_{i,i' \in C_k} \delta(x_i, x_{i'}),$$

or ³

$$(3.6) \quad S(C) = \sum_{k=1}^K \sum_{i,i' \in C_k} \delta(x_i, x_{i'}),$$

where $\delta(x_i, x_{i'})$ is the distance measure between x_i and $x_{i'}$. For binary clustering, the dissimilarity coefficients are popular measures of the distances.

3.2.1 Various Coefficients Given two binary data points, w and w' , there are four fundamental quantities that can be used to define similarity between the two [1]: $a = \|\{j \mid w_j = w'_j = 1\}\|$, $b = \|\{j \mid w_j = 1 \wedge w'_j = 0\}\|$, $c = \|\{j \mid w_j = 0 \wedge w'_j = 1\}\|$, and $d = \|\{j \mid w_j = w'_j = 0\}\|$, where $1 \leq j \leq r$. It has been shown in [1] that the presence/absence based dissimilarity measure can be generally ⁴ written as $D(a, b, c, d) = \frac{b+c}{\alpha a + b + c + \beta d}$, where $\alpha > 0$ and $\beta \geq 0$. Dissimilarity measures can be transformed into a similarity function by simple transformations such as adding 1 and inverting, dividing by 2 and subtracting from 1, etc. [6]. If the joint absence of the attribute is ignored, i.e., β is set to 0, then the binary dissimilarity measure can be generally written as $D(a, b, c, d) = \frac{b+c}{\alpha a + b + c}$, where $\alpha > 0$.

In cluster applications, the rankings based on a dissimilarity coefficient is often of more interest than the actual value of the dissimilarity coefficient. It has been shown that [1], if the paired absences are ignored in the calculation of dissimilarity values, then there is only one single dissimilarity coefficient modulo the global order equivalence: $\frac{b+c}{a+b+c}$. Thus our following discussion is based on the single dissimilarity coefficient.

3.2.2 BMD and Dissimilarity Coefficients Given representation (D, F) , basically, D denotes the assignments of data points associated into clusters and F indicates the fea-

ture representations of clusters. Observe that

$$(3.7) \quad \begin{aligned} O(D, F) &= \frac{1}{2} \|X - DF^T\|_F^2 \\ &= \frac{1}{2} \sum_{i,j} (x_{ij} - (DF^T)_{ij})^2 \\ &= \frac{1}{2} \sum_{k=1}^K \sum_{i \in C_k} \sum_j |x_{ij} - e_{kj}|^2 \\ &= \frac{1}{2} \sum_{k=1}^K \sum_{i \in C_k} d(x_i, e_k), \end{aligned}$$

where $e_k = (f_{k1}, \dots, f_{kr})$, $i = 1, \dots, K$ is the cluster “representative” of cluster C_i . Thus minimizing Equation (3.7) is the same as minimizing Equation (3.6) where the distance is defined as $d(x_i, e_k) = \sum_j |x_{ij} - (e_k)_{ij}|^2 = \sum_j |x_{ij} - (e_k)_{ij}|$ (the last equation holds since x_{ij} and $(e_k)_{ij}$ are all binary.) In fact, given two binary vectors X and Y , $\sum_i |X_i - Y_i|$ calculates their mismatches (the numerator of their dissimilarity coefficients).

3.3 Minimum Description Length Minimum Description length (MDL) aims at searching for a model that provides the most compact encoding for data transmission [10] and is conceptually similar to minimum message length (MML) [9, 2] and stochastic complexity minimization [11]. In fact, the MDL approach is a Bayesian method: the code lengths and the code structure in the coding model are equivalent to the negative log probabilities and probability structure assumptions in the Bayesian approach.

As described in Section 2, in BMD clustering, the original matrix X can be approximated by the matrix product of DF^T . Instead of encoding the elements of X alone, we then encode the model, D, F , and the data given the model, $(X|DF^T)$. The overall code length is thus expressed as

$$L(X, D, F) = L(D) + L(F) + L(X|DF^T).$$

In the Bayesian framework, $L(D)$ and $L(F)$ are negative log priors for D and F and $L(X|DF^T)$ is a negative log likelihood of W given D and F . If we assume that the prior probabilities of all the elements of D and F are uniform (i.e., $\frac{1}{2}$), then $L(D)$ and $L(F)$ are fixed given the dataset X . In other words, we need to use one bit to represent each element of D and F irrespective of the number of 1's and 0's. Hence, minimizing $L(X, D, F)$ reduces to minimizing $L(X|DF^T)$.

Use \hat{X} to denote the generated data matrix by D and F . For all i , $1 \leq i \leq n$, j , $1 \leq j \leq p$, $b \in \{0, 1\}$, and $c \in \{0, 1\}$, we consider $p(x_{ij} = b \mid \hat{x}_{ij}(D, F) = c)$, the probability of the original data $W_{ij} = b$ conditioned upon the generated data $(\hat{x})_{ij}$, via DF^T , is c . Note that

$$p(x_{ij} = b \mid \hat{x}_{ij}(D, F) = c) = \frac{N_{bc}}{N_{.c}}.$$

³Equation (3.5) computes the weighted sum using the cluster sizes.

⁴Basically, the presence/absence based dissimilarity measure satisfies a set of axioms such as non-negative, range in $[0, 1]$, rationality whose numerator and denominator are linear and symmetric, etc. [1].

Here N_{bc} is the number of elements of X which have value b where the corresponding value for \hat{X} is c , and $N_{.c}$ is the number of elements of \hat{X} which have value c . Then the code length for $L(X, D, F)$ is

$$\begin{aligned} L(X, D, F) &= - \sum_{b,c} N_{bc} \log P(x_{ij} = b \mid \hat{x}_{ij}(D, F) = c) \\ &= -np \sum_{b,c} \frac{N_{bc}}{np} \log \frac{N_{bc}}{N_{.c}} \\ &= npH(X \mid \hat{X}(D, F)) \end{aligned}$$

So minimizing the coding length is equivalent to minimizing the conditional entropy. Denote $p_{bc} = p(x_{ij} = b \mid \hat{x}_{ij}(D, F) = c)$. We wish to find the probability vectors $\mathbf{p} = (p_{00}, p_{01}, p_{10}, p_{11})$ that minimize

$$(3.8) \quad H(X \mid \hat{X}(D, F)) = - \sum_{i,j \in \{0,1\}} p_{ij} \log p_{ij}$$

Since $-p_{ij} \log p_{ij} \geq 0$, with the equality holding at $p_{ij} = 0$ or 1, the only possible probability vectors which minimize $H(X \mid \hat{X}(D, F))$ are those with $p_{ij} = 1$ for some i, j and $p_{i_1 j_1} = 0, (i_1, j_1) \neq (i, j)$. Since \hat{X} is an approximation of X , it is natural to require that p_{00} and p_{11} be close to 1 and p_{01} and p_{10} be close to 0. This is equivalent to minimizing the mismatches between X and \hat{X} , i.e., minimizing $O(D, F) = \frac{1}{2} \|X - DF^T\|_F^2$.

3.4 Entropy-Based Approach

3.4.1 Classical Entropy Criterion The classical clustering criteria [3, 4] search for a partition C that maximizes the following quantity $O(C)$:

$$\begin{aligned} (3.9) \quad O(C) &= \sum_{k=1}^K \sum_{j=1}^r \sum_{t=0}^1 \frac{N_{j,k,t}}{N} \log \frac{NN_{j,k,t}}{N_k N_{j,t}} \\ &= \sum_{k=1}^K \sum_{j=1}^r \sum_{t=0}^1 \frac{N_{j,k,t}}{N} \left(\log \frac{N_{j,k,t}}{n_k} - \log \frac{N_{j,t}}{n} \right) \\ &= \frac{1}{r} \left(\hat{H}(X) - \frac{1}{n} \sum_{k=1}^K n_k \hat{H}(C_k) \right). \end{aligned}$$

Observe that $\frac{1}{n} \sum_{k=1}^K n_k \hat{H}(C_k)$ is the entropy measure of the partition, i.e., the weighted sum of each cluster's entropy. This leads to the following criterion: Given a dataset, fix $\hat{H}(X)$, then maximizing $O(C)$ is equivalent to minimizing the expected entropy of the partition:

$$(3.10) \quad \frac{1}{n} \sum_{k=1}^K n_k \hat{H}(C_k)$$

3.4.2 Entropy and Dissimilarity Coefficients Now examine the within-cluster criterion in Equation (3.5). We

have:

$$\begin{aligned} S(C) &= \sum_{k=1}^K \frac{1}{n_k} \sum_{i,i' \in C_k} \delta(x_i, x_{i'}) \\ &= \sum_{k=1}^K \frac{1}{n_k} \sum_{i,i' \in C_k} \frac{1}{r} \sum_{j=1}^r |x_{i,j} - x_{i',j}| \\ &= \frac{1}{r} \sum_{k=1}^K \sum_{j=1}^r n_k \rho_k^{(j)} (1 - \rho_k^{(j)}). \end{aligned}$$

Here for each k , $1 \leq k \leq K$, and for each j , $1 \leq j \leq r$, $\rho_k^{(j)}$ is the probability that the j -th attribute is 1 in C_k .

Using the generalized entropy⁵ defined in [5], $H^2(Q) = -2(\sum_{i=1}^n q_i^2 - 1)$, we have

$$\begin{aligned} &\frac{1}{n} \sum_{k=1}^K n_k \hat{H}(C_k) \\ &= -\frac{1}{2n} \sum_{k=1}^K \sum_{j=1}^r n_k \left((\rho_k^{(j)})^2 + (1 - \rho_k^{(j)})^2 - 1 \right) \\ &= \frac{1}{n} \sum_{k=1}^K \sum_{j=1}^r n_k \rho_k^{(j)} (1 - \rho_k^{(j)}) = \frac{r}{n} S(C). \end{aligned}$$

References

- [1] F. B. Baulieu. Two variant axiom systems for presence/absence based dissimilarity coefficients. *Journal of Classification*, 14(1):159–170, 1997.
- [2] R. A. Baxter and J. J. Oliver. MDL and MML: similarities and differences. TR 207, Monash University, 1994.
- [3] H.-H. Bock. Probabilistic aspects in cluster analysis. In *Conceptual and Numerical Analysis of Data*, pages 12–44, 1989.
- [4] G. Celeux and G. Govaert. Clustering criteria for discrete data and latent class models. *Journal of Classification*, 8(2):157–176, 1991.
- [5] J. Havrda and F. Charvat. Quantification method of classification processes: Concept of structural a-entropy. *Kybernetika*, 3:30–35, 1967.
- [6] N. Jardine and R. Sibson. *Mathematical Taxonomy*. John Wiley & Sons, 1971.
- [7] T. Li, S. Ma, and M. Ogihara. Entropy-based criterion in categorical clustering. In ICML, 2004. 536–543.
- [8] T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.
- [9] J. J. Oliver and R. A. Baxter. MML and Bayesianism: similarities and differences. TR 206, Monash University, 1994.
- [10] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [11] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Press, Singapore, 1989.

⁵Note that $H^s(Q) = (2^{(1-s)} - 1)^{-1} (\sum_{i=1}^n q_i^s - 1)$.

Time-series Bitmaps: a Practical Visualization Tool for Working with Large Time Series Databases

Nitin Kumar Venkata Nishanth Lolla Eamonn Keogh Stefano Lonardi Chotirat Ann Ratanamahatana Li Wei

University of California - Riverside
Department of Computer Science & Engineering
Riverside, CA 92521, USA

{nkumar, vlolla, eamonn, stelo, ratana, wli}@cs.ucr.edu

Abstract

The increasing interest in time series data mining in the last decade has resulted in the introduction of a variety of similarity measures, representations, and algorithms. Surprisingly, this massive research effort has had little impact on real world applications. Real world practitioners who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. In this work, we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our system has the unique advantage that it can be embedded directly into any standard graphical user interfaces, such as Microsoft Windows, thus making deployment easier. Our approach extracts features from a time series of arbitrary length and uses information about the relative frequency of its features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection. We demonstrate the utility of our approach with a set of comprehensive experiments on real datasets from a variety of domains.

Keywords: Time Series, Chaos Game, Visualization.

1 Introduction

The increasing interest in time series data mining in the last decade has resulted the introduction of a variety of similarity measures/ representations/ definitions/ indexing techniques and algorithms (see, e.g., [1][2][3][4][9][13][14]). Surprisingly, this massive research effort has had little impact on real world applications. Examples of implemented systems are rare exceptions [17]. Cardiologists, engineers, technicians, and others who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. While it is difficult to firmly establish the reasons for the discrepancy between tool availability and practical adoption, the following reasons suggested themselves after an informal survey.

- Time series data mining tools often come with a bewildering number of parameters. It is not obvious to the practitioner how these should be set [15].
- Research tools often require (relatively) specialized hardware and/or software, rather than the ubiquitous desktop PC/Windows environment that prevails.
- Many tools have a steep learning curve, requiring the user to spend many unproductive hours learning the system before any possibility of useful work.

In this work, we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our approach extracts features from a time series of arbitrary length, and uses information about the relative frequency of these features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of these bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection.

While our system can be used as an interactive tool, it also has the unique advantage that it can be embedded directly into any standard graphical user interfaces, such as Windows, Aqua, X-windows, etc. Since users navigate through files by looking at their icons, we decided to employ the bitmap representation as the icon corresponding to each time series. Simply by glancing at the icons contained in a folder of time series files, a user can quickly identify files that require further investigation. In Figure 1, we have illustrated a simple example.

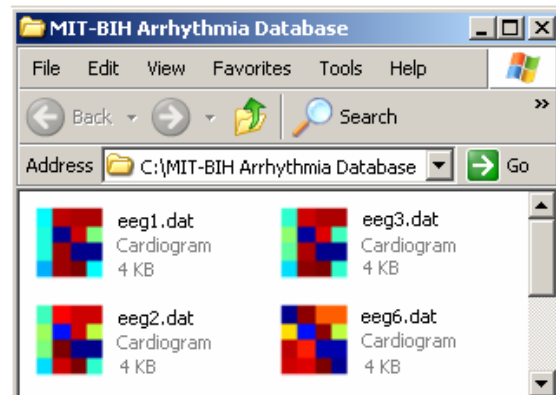


Figure 1. Four time series files represented as time series thumbnails. While they are all examples of congestive heart failure, *eeg6.dat* is from a different person to all the rest. This fact is immediately apparent from a casual inspection of the thumbnail representation.

The utility of the idea shown in Figure 1 can be further enhanced by arranging the icons within the folder by *pattern similarity*, rather than the typical choices of arranging them by *size*, *name*, *date*, etc. This can be achieved by using a simple multidimensional scaling or a self-organizing map algorithm to arrange the icons.

Unlike most visualization tools which can only be evaluated subjectively, we will perform objective evaluations on the amount of useful information contained within a time series bitmap. More precisely, we will analyze the loss of accuracy of classification /clustering/anomaly detection algorithms when the input is based solely on the information contained in the bitmap. As we will show, the experiments strongly confirm the utility of our approach.

2 Chaos Game Representations

Our visualization technique is partly inspired by an algorithm to draw fractals called the *Chaos game* [3]. The method can produce a representation of DNA sequences, in which both local and global patterns are displayed. For example, a biologist can recognize that a particular substring, say in a bacterial genome, is rarely used. This would suggest the possibility that the bacteria have evolved to avoid a particular restriction enzyme site, which means that he/she might not be easily attacked by a specific bacterio-phage.

From our point of view, the crucial observation is that the CGR representation of a sequence allows the investigation of the patterns in sequences, giving the human eye a possibility to recognize hidden structures.

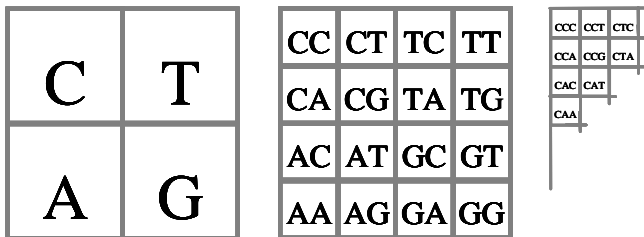


Figure 2. The quad-tree representation of a sequence over the alphabet {A,C,G,T} at different levels of resolution

We can get a hint of the potential utility of the approach if, for example, we take the first 16,000 symbols of the mitochondrial DNA sequences of four familiar species and use them to create their own file icons. Figure 3 below illustrates this. Even if we did not know these particular animals, we would have no problem recognizing that there are two pairs of highly related species being considered.

With respect to the non-genetic sequences, Joel Jeffrey noted, “The CGR algorithm produces a CGR for any sequence of letters”[9]. However, it is only defined for discrete sequences, and most time series are real valued.

This encouraged us to try a similar technique on time series data and investigate the utility of such representation on the classic data mining tasks of clustering, classification, and visualization.

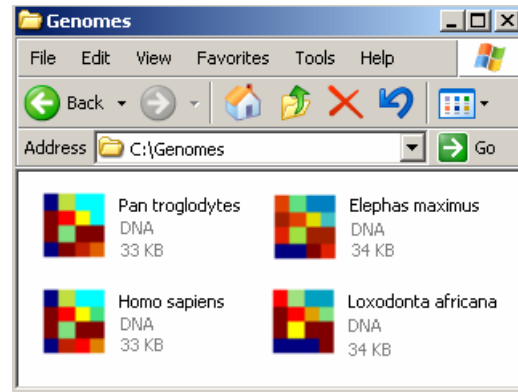


Figure 3. The gene sequences of mitochondrial DNA of four animals, used to create their own file icons using a chaos game representation. Note that *Pan troglodytes* is the familiar Chimpanzee, and *Loxodonta africana* and *Elephas maximus* are the African and Indian Elephants, respectively. The file icons show that humans and chimpanzees have similar genomes, as do the African and Indian elephants.

Since CGR involves treating a data input as an abstract string of symbols, a discretization method is necessary to transform continuous time series data into discrete domain. For this purpose, we used the Symbolic Aggregate approXimation (SAX) [18]. While there are at least 200 techniques in the literature for converting real valued time series into discrete symbols [7], the SAX technique of Lin *et. al.* [18] is unique and ideally suited for data mining. SAX is the only symbolic representation that allows the lower bounding of the distances in the original space. The ability to efficiently lower bound distances is at the heart of hundreds of indexing algorithms and data mining techniques [2][6][12][14][18][19].

The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. The mean value of each section is then calculated. By substituting each section with its mean, a reduced dimensionality piecewise constant approximation of the data is obtained. This representation is then discretized in such a manner as to produce a word with approximately equi-probable symbols. Figure 4 shows a short time series being converted into the SAX word **baabccbc**.

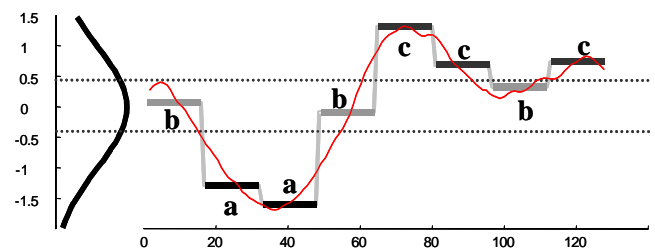


Figure 4. A real valued time series can be converted to the SAX word **baabccbc**. Note that all three possible symbols are approximately equally frequent.

The time and space complexity to convert a sequence to its SAX representation is linear in the length of the sequence. The SAX representation has been successfully used by various groups of researchers for indexing, classification, clustering [18], motif discovery [5][6][21], rule discovery, [20], visualization [17], and anomaly detection [15].

3 Time Series Bitmaps

At this point, the connection between the two “ingredients” for the time series bitmaps should be evident. We have seen in Section 2.3 that the *Chaos game* [3] bitmaps can be used to visualize discrete sequences, and we have seen in Section 2.4 that the SAX representation is a discrete time series representation that has demonstrated great utility for data mining. It is natural to consider combining these ideas.

The *Chaos game* bitmaps are defined for sequences with an alphabet size of four. It is fortuitous that DNA strings have this cardinality. SAX can produce strings on any alphabet sizes. As it happens, a cardinality of four (or three) has been reported by many authors as an excellent choice for diverse datasets on assorted problems [5][6][15][17][18] [20][21].

We need to define an initial ordering for the four SAX symbols **a**, **b**, **c**, and **d**. We use simple alphabetical ordering as shown in Figure 5.

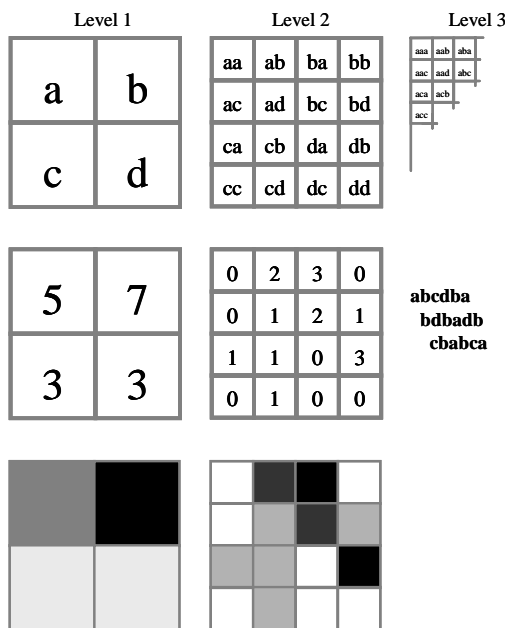


Figure 5. *Top*) The four possible SAX symbols are mapped to four quadrants of a square, and pairs, triplets, etc are recursively mapped to finer grids. *Middle*) We can extract counts of symbols from a SAX representation and record them in the grids. *Bottom*) The recorded values can be linearly mapped to colors, thus creating a square bitmap.

After converting the original raw time series into the SAX representation, we can count the frequencies of SAX “subwords” of length L , where L is the desired level of recursion. Level-1 frequencies are simply the raw counts of the four symbols. For level 2, we count pairs of subwords of size 2 (**aa**, **ab**, **ac**, etc). Note that we only count subwords taken from individual SAX words. For example, in the SAX representation in Figure 5 *middle right*, the last symbol of the first line is **a**, and the first symbol of the second word is **b**. However, we do not count this as an occurrence of **ab**.

Once the raw counts of all subwords of the desired length have been obtained and recorded in the corresponding pixel of the grid, a final step is required. Since the time series in a data collection may be of various lengths, we normalize the frequencies by dividing it by the largest value. The pixel values thus range from 0 to 1. The final step is to map these values to colors. In the example above, we mapped to grayscale, with 0 = *white*, 1 = *black*. However, it is generally recognized that grayscale is not *perceptually uniform* [22]. A color space is said to be perceptually uniform if small changes to a pixel value are approximately equally perceptible across the range of that value. For all images produced in this paper we use Matlab’s “jet” color space, which is a linearization of a large fraction of all possible colors and which is designed to be perceptually uniform.

Note that unlike the arbitrarily long, and arbitrarily shaped time series from which they were derived, for a fixed L , the bitmaps have a constant space and structure.

We do not suggest any utility in viewing a single time series bitmap. The representation is abstract, and we do not expect a user to be able to imagine the structure of time series given the bitmap. The utility of the bitmaps comes from the ability to efficiently compare and contrast them.

4 Time Series Thumbnails

A unique advantage of the time series bitmap representation is the fact that we can transparently integrate it into the user graphical interface of most standard operating systems. Since most operating systems use the ubiquitous square icon to represent a file, we can arrange for the icons for time series files to appear as their bitmap representations. Simply by glancing at the contents of a folder of time series files, a user may spot files that require further investigation, or note natural clusters in the data.

The largest possible icon size varies by operating system. All modern versions of Microsoft Windows support 32 by 32 pixels, which is large enough to support a bitmap of level 5. As we will see, level 2 or 3 seems adequate for most tasks/datasets. To augment the utility of the time series bitmaps, we can arrange for their placement on

screen to reflect their structure. Normally, file icons are arranged by one of a handful of common criteria, such as *name*, *date*, *size*, etc.

We have created a simple modification of the standard Microsoft Windows (98 or later) file browser by introducing the concept of *Cluster View*. If *Cluster View* is chosen by the user, the time series thumbnails arrange themselves by similarity. This is achieved by performing Multi-Dimensional Scaling (MDS) of the bitmaps, and projecting them into a 2 dimensional space. For aesthetic reasons, we “snap” the icons to the closest grid point.

Figure 6 displays an example of *Cluster View* in Microsoft Windows XP Operating System. In this example, the *Cluster View* is obtained for five MIT-BIH Arrhythmia Database files. It is evident in the figure that *eeg1.dat*, *eeg2.dat*, and *eeg3.dat* belong to one cluster whereas *eeg6.dat* and *eeg7.dat* belong to another. In this case, the grouping correctly reflects the fact that latter two files come from a different patient to first three.

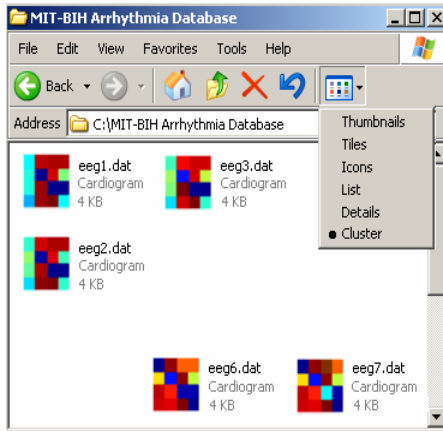


Figure 6. A snapshot of a folder containing cardiograms when its files are arranged by “Cluster” option. Five cardiograms have been grouped into two different clusters based on their similarity.

5 Experimental Evaluation

We have tested our proposed approach with a comprehensive set of experiments; most of these were omitted for the sake of brevity. We urge the interested reader to consult the full version of this paper or URL [11] for large-scale color reproductions and additional details.

We examined the UCR Time Series Archive for datasets that come in pairs. For example, in the **Buoy Sensor** dataset, there are two time series, *North Salinity* and *East Salinity*, and in the **Exchange Rate** dataset, there are two time series, *German Marc* and *Swiss Franc*. We were able to identify fifteen such pairs, from a diverse collection of time series covering the domains of finance, science, medicine, industry, etc. Although our method is able to deal with time series of different lengths, we truncated all time series to length 1,000 for visual clarity.

While the correct hierarchical clustering at the top of the tree is somewhat subjective, at the lower level of the tree, we would hope to find a single bifurcation separating each pair in the dataset. Our metric, Q , for the quality of clustering is therefore the number of such correct bifurcations divided by fifteen, the number of datasets. For a perfect clustering, $Q = 1$. Figure 7 shows the resulting dendrogram for our approach. The dendrograms for the other approaches are omitted here for brevity, but may be viewed at [11].

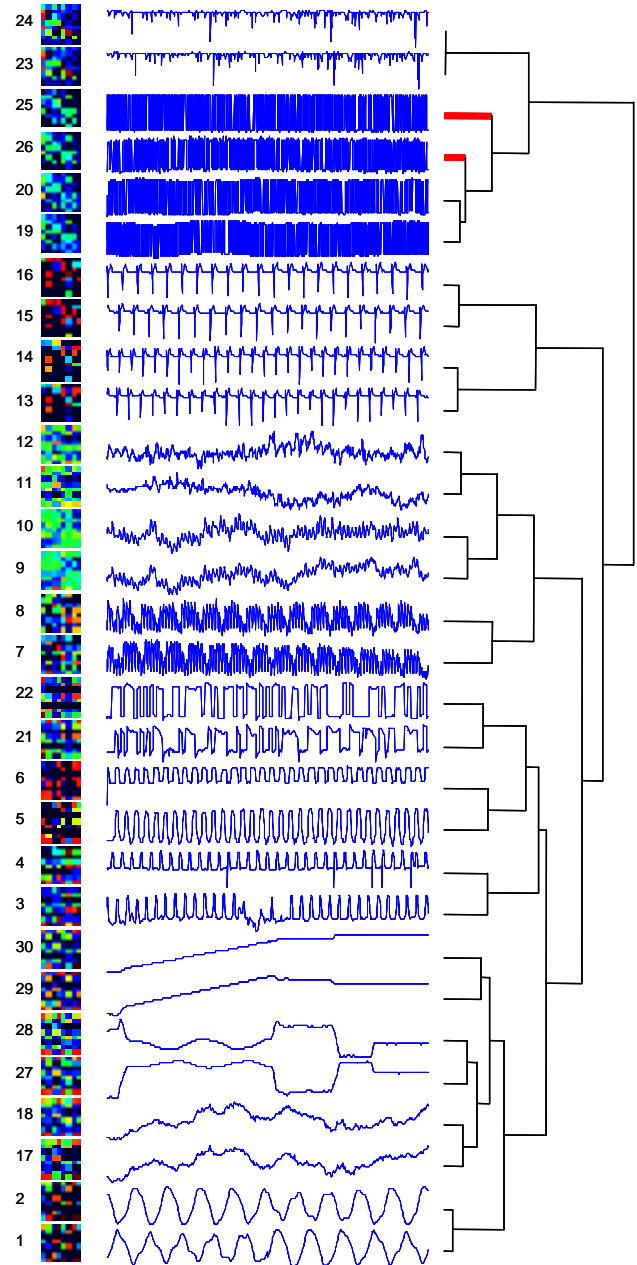


Figure 7. The clustering obtained by the time series thumbnail approach on a heterogeneous data collection. Bold lines denote incorrect subtrees. A key of the data appears in Appendix A of the full paper.

We compared to two well-known and highly referenced techniques, Markov models [8] and ARIMA models [10][22]. For each technique, we spent one hour searching over parameter choice and reported only the best performing result. To mitigate the problem of overfitting, we set the parameters on a different, but similar dataset. The results for the three approaches are given in Table 1.

Table 1. The quality of clustering obtained by the three algorithms under consideration.

Algorithm	Q
Thumbnails	0.93
Markov Model	0.46
ARMA models	0.40

6 Conclusions and Future Work.

In this work, we have introduced a new framework for visualization of time series. Our approach is unique in that it can be directly embedded into any standard GUI operating system. We demonstrated the effectiveness of our approach on a variety of tasks and domains. Future work includes an extensive user study, and investigating techniques to automatically set the system parameters.

This research was partly funded by the National Science Foundation under grant IIS-0237918.

Reproducible Results Statement: In the interests of competitive scientific inquiry, all datasets used in this work are available at the following URL [11].

References

- [1] Aach, J., & Church, G. (2001). *Aligning gene expression time series with time warping algorithms*. Bioinformatics, Volume 17, pp. 495-508.
- [2] Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). *Fast similarity search in the presence of noise, scaling, and translation in times-series databases*. In Proceedings of twenty-first International Conference on Very Large Databases, pp. 490-501.
- [3] Barnsley, M.F., & Rising, H. (1993). *Fractals Everywhere*, second edition, Academic Press.
- [4] Berndt, D., & Clifford, J. (1994). *Using dynamic time warping to find patterns in time series*, AAAI Workshop on Knowledge Discovery in Databases, pp. 229-248.
- [5] Celly, B. & Zordan, V. B. (2004). *Animated People Textures*. In proceedings of the 17th International Conference on Computer Animation and Social Agents. Geneva, Switzerland.
- [6] Chiu, B., Keogh, E., & Lonardi, S. (2003). *Probabilistic Discovery of Time Series Motifs*. In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493-498.
- [7] Daw, C. S., Finney, C.E.A., & Tracy, E.R.(2001). *Symbolic Analysis of Experimental Data*. Review of Scientific Instruments. (2002-07-22).
- [8] Ge, X., & Smyth, P. (2000). *Deformable Markov model templates for time-series pattern matching*. In proceedings of the sixth ACM SIGKDD, pp. 81-90.
- [9] Jeffrey, H.J. (1992). *Chaos Game Visualization of Sequences*. Comput. & Graphics 16, pp. 25-33.
- [10] Kalpakis, K., Gada, D., & Puttagunta, V. (2001). *Distance Measures for Effective Clustering of ARIMA Time-Series*. In the Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 273-280.
- [11] Keogh, E. www.cs.ucr.edu/~nkumar/SDM05.html
- [12] Keogh, E. (2002). *Exact indexing of dynamic time warping*. In Proceedings of the twenty-eighth International Conference on Very Large Data Bases, pp. 406-417.
- [13] Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra (2001). *Locally adaptive dimensionality reduction for indexing large time series databases*. In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 151-162.
- [14] Keogh, E. & Kasetty, S. (2002). *On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration*. In the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 102-111.
- [15] Keogh, E., Lonardi, S., & Ratanamahatana, C. (2004). *Towards Parameter-Free Data Mining*. In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- [16] Korn, F., Jagadish, H., & Faloutsos, C. (1997). *Efficiently supporting ad hoc queries in large datasets of time sequences*. In Proceedings of SIGMOD, pp. 289-300.
- [17] Lin, J., Keogh, E., Lonardi, S., Lankford, J.P. & Nystrom, D.M. (2004). *Visually Mining and Monitoring Massive Time Series*. In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining..
- [18] Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003) *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*. In proceedings of the eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.
- [19] Ratanamahatana, C.A., & Keogh, E. (2004). *Everything you know about Dynamic Time Warping is Wrong*. 3rd Workshop on Mining Temporal and Sequential Data, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- [20] Silvent, A. S., Carbay, C., Carry, P. Y. & Dojat, M. (2003). *Data, Information and Knowledge for Medical Scenario Construction*. In proceedings of the Intelligent Data Analysis In Medicine and Pharmacology Workshop.
- [21] Tanaka, Y. & Uehara, K. (2004). *Motif Discovery Algorithm from Motion Data*. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI).
- [22] Wysocki, G. (1982). *Color science: Concepts and methods, quantitative data and formulae*, 2nd edition.

Pushing Feature Selection ahead of Join

Rong She, Ke Wang, Yabo Xu
School of Computing Science, Simon Fraser University
rshe, wangk, yxu@cs.sfu.ca

Philip S. Yu
IBM T.J. Watson Research Center
psyu@us.ibm.com

ABSTRACT¹

Current approaches for feature selection on multiple data sources need to join all data in order to evaluate features against the class label, thus are not scalable and involve unnecessary information leakage. In this paper, we present a way of performing feature selection through class propagation, eliminating the need of join before feature selection. We propagate a very compact data structure that provides enough information for selecting features to each data source, thus allowing features to be evaluated locally without looking at any other information. Our experiments confirmed that our algorithm is highly scalable while effectively preserving the data privacy.

Keywords

feature selection, class propagation, scalability, data privacy, classification

1. INTRODUCTION

In scientific collaborations and business initiatives, data often reside in multiple data sources in different formats. Classification on such data is challenging, as the presence of many irrelevant features causes problems on both scalability and accuracy. Additionally, as data may be collected from different data providers, if irrelevant features are not removed before data integration, unnecessary details will be revealed to other data sources, causing problems on data privacy.

To address these concerns, feature selection is needed as it effectively reduces the data size and filters out noises, while limiting the information shared among different data sources. However, with data scattered among multiple sources and the class label exists in only one of the data sources (“class table”), current approaches for feature selection have to perform join before features can be evaluated against the class label.

Example 1.1 Consider a toy example in Figure 1.

¹ This work is supported in part by a grant from Networks of Centers of Excellence/Institute for Robotics and Intelligent Systems, a grant from Institute for Robotics and Intelligent Systems/Precarn, and a grant from Natural Sciences and Engineering Research Council of Canada.

Given “Credit Card” table and “Transaction” table, suppose we want to select features that are relevant to determine whether certain credit card is in good standing or not. A straight-forward method would join these tables on the common feature “Account No”, resulting in a joined table with 8 records and 9 attributes. Each feature can then be examined against the class attribute in the joined table. ■

Several problems arise with this approach. Firstly, as the join operation is expensive and there is blow-up in the data size, it is a waste in both time and space as many features will be removed by subsequent feature selection. Secondly, unnecessary details are revealed which violates the data privacy constraint, thus is undesirable. In addition, there are cases where learning is done on some query results defined with each user specification, e.g., to study personal behaviors, user may want to join the two example tables on persons’ names. It is thus impossible to materialize the joined result once and use it for all subsequent learning.

In this paper, we propose a way to perform feature selection without join. We observe that a feature selection algorithm is essentially a computational solution that measures the class relevance of all features. For typical relevance measures, all information required in the computation is the class distribution associated with each feature. If the class information can be propagated to all data sources, it is easy to evaluate all features locally. Thus, instead of evaluating all features in a central table, we push the class labels into each individual data source. *In other words, we push the process of feature selection ahead of the join operation.*

2. Related Works

In multi-relational learning such as [1,10,13], the target entity is an “object”, which is one entry in the target table. Features in all non-target tables are properties of the target object. However, the problem we are dealing with is conceptually different. We regard each entry in the final joined table as our target entity, i.e., classification is defined on joined instances.

A number of surveys on feature selection methods are available [3,8]. In general, it’s a process that chooses an optimal subset of features according to certain criterion

<i>Credit Card</i>	<i>Acc. No</i>	<i>Card Holder</i>	<i>Class</i>
C1	A1	Mary	Good
C2	A1	Michael	Good
C3	A2	Helen	Bad
C4	A2	John	Good

Credit Card Table

<i>Trans. No</i>	<i>Acc. No</i>	<i>Date</i>	<i>Customer</i>	<i>Type</i>	<i>Amount</i>
1	A1	02/12'02	Michael	transfer	100.00
2	A1	03/03'03	Michael	withdraw	200.00
3	A2	05/20'02	John	deposit	390.98
4	A2	11/01'03	Helen	transfer	34.00

Transaction Table

Figure 1. Example Database with Many-to-Many Relationship

[7]. Previous works on feature selection focused on developing evaluation criteria and search strategies, given one flat table with a set of features. Not much work is done when it comes to feature selection across multiple relations, other than the intuitive join approach.

Sampling is another technique that is often used for scalability. However, as it only operates on a portion of the original data, the results are only approximations. Also, it does not address the data privacy issue at all. As one work that is close to ours, the VFREL algorithm proposed in [5] made use of feature selection to reduce the dataset size before passing the flattened (joined) data to a propositional learner. At each iteration, a small sample of the flattened database is used to eliminate features that are most likely to be irrelevant. That is, they still perform join on a portion of data at each step in order to select features.

Our work differs as we eliminated the need of join before feature selection. In addition, we also address the data privacy problem while there is no such concern in their context.

The idea of information propagation across multiple relations has been explored [13]. However, they propagated IDs of target records, which may be of arbitrarily large size. And they need to do propagation in each iteration of building the classifier. We propagate class labels with a size equal to the number of classes (typically very small in classification). Once the class label is propagated to each table, all evaluation is done locally and there's never need to propagate again.

3. Algorithm Overview

Figure 2 compares our approach with the existing approach. With our framework, feature selection is done through *class propagation* where class distribution information is propagated from the class table to other tables without join. By pushing feature selection ahead, we only need to join a much smaller subset of features at a later stage, thus the algorithm is more scalable and data privacy is protected.

In order to measure the relevance of features, typical

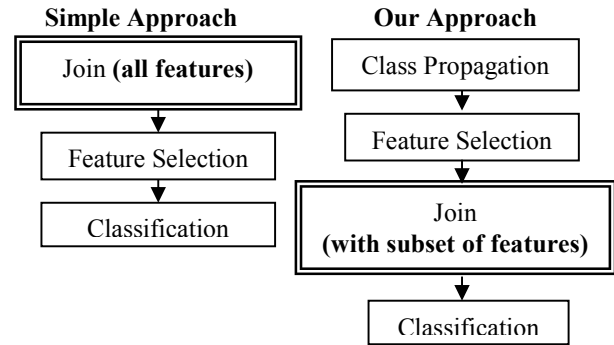


Figure 2. Work Flow Overview

measures such as *information gain* [9] and *gini index* [2] are defined based on the relative frequency of each class. Everything that is needed for calculating these measures is contained in the projection of the examined feature values and their class distribution. Such projection has been referred to as “AVC (Attribute-Value-Class label) set” in [4]. The size of such AVC set is proportional to the number of distinct values in each attribute. To obtain such AVC values, the only data structure that needs to be propagated is the class distribution information. This observation leads to the first and the core part of our algorithm, *class propagation*.

The operations after class propagation are rather standard, including feature selection, join and classification. As our focus is to study the effect of class propagation, we simply made use of some commonly-used existing methods. We do not intend to introduce a new feature selection or classification algorithm. Rather, we provide a method to perform feature selection directly on multiple data sources. Since we measure the relevance of features in the same way as it is done on the joined table, it is guaranteed the resulted feature set is exactly the same as would be produced by joining the databases. In the next section, we will focus on details of class propagation.

4. Class Propagation

To propagate class information, we maintain a data structure at each data source, named “ClsDis” (class

distribution vectors), in the form of “ $\langle \text{count}_1, \text{count}_2, \dots, \text{count}_n \rangle$ ” where n is the total number of classes. Each count in the vector represents the number of instances of the corresponding class in the joined table. Operations on such class vectors are performed on each corresponding pair of class counts. For some operator ‘ \otimes ’ and two vectors $V: \langle C_1, C_2, \dots, C_n \rangle$ and $V': \langle C'_1, C'_2, \dots, C'_n \rangle$, $V \otimes V' = \langle C_1 \otimes C'_1, C_2 \otimes C'_2, \dots, C_n \otimes C'_n \rangle$. (e.g., $\langle 1, 2 \rangle \otimes \langle 3, 4 \rangle = \langle 1 \cdot 3, 2 \cdot 4 \rangle = \langle 3, 8 \rangle$)

4.1 An example with a 2-table database

Consider our toy example database in **Figure 1**. *Credit Card* table contains the class label (credit standing “good” or “bad”). We consider the query where we need to join the tables on “Acc. No” for classification.

Step 1. Initialization

In the class table, if a tuple has class i , its class vector is initialized such that $\text{count}_i=1$ and $\text{count}_j=0$ where $j \neq i$. For non-class tables, all counts of the class vectors are initialized to 0s.

Step 2. Forward Propagation

Class propagation starts from the class table. Since we are joining on the attribute *AccNo*, for each tuple in *Transaction* table with *AccNo* = A_i , its “ClsDis” is the aggregation of class counts in *Credit Card* table with the same *AccNo*.

This results in the *Transaction* table with propagated “ClsDis” as shown after step 2 in Figure 3. Note the total class count in *Transaction* table have reflected the effect of join on both tables.

Step 3. Backward Propagation

Then we need to propagate back from *Transaction* table to *Credit Card* table, as the class counts in *Credit Card* table have not reflected the join. Consider a tuple T with *AccNo*= A_i in *Credit Card* table. T will join with all the tuples in *Transaction* having this account number. Let V be the aggregated class counts for such tuples in *Transaction*. V is also the aggregated class counts over all tuples in *Credit Card* with this *AccNo*. T is one of such tuples. We need to redistribute V among such tuples in *Credit Card* according to their shares of class counts in *Credit Card* table.

As an example, the third tuple in *Credit Card* table has “*AccNo*= A_2 ” and gets the new “ClsDis” $\langle 0, 2 \rangle$ as a result of $\langle 2, 2 \rangle * (\langle 0, 1 \rangle / \langle 1, 1 \rangle)$, since the aggregated “ClsDis” with “*AccNo*= A_2 ” is $\langle 2, 2 \rangle$ in *Transaction* table, the original class vector is $\langle 0, 1 \rangle$ in this entry of *Credit Card* table, and the aggregated class vector with this account number in *Credit Card* table is $\langle 1, 1 \rangle$. The final results are shown after step 3 in Figure 3.

After propagation, all tables contain the same aggregated class count, which is the same as if we have joined the tables. This is also why we need to propagate in a backward direction, so that the effect of join is reflected in all tables. (We omit the formal proof due to the space limit.)

4.2 General Scenarios

In general, we can deal with datasets with acyclic relationships among tables, i.e., if each table is a node and related tables (share join predicate) are connected

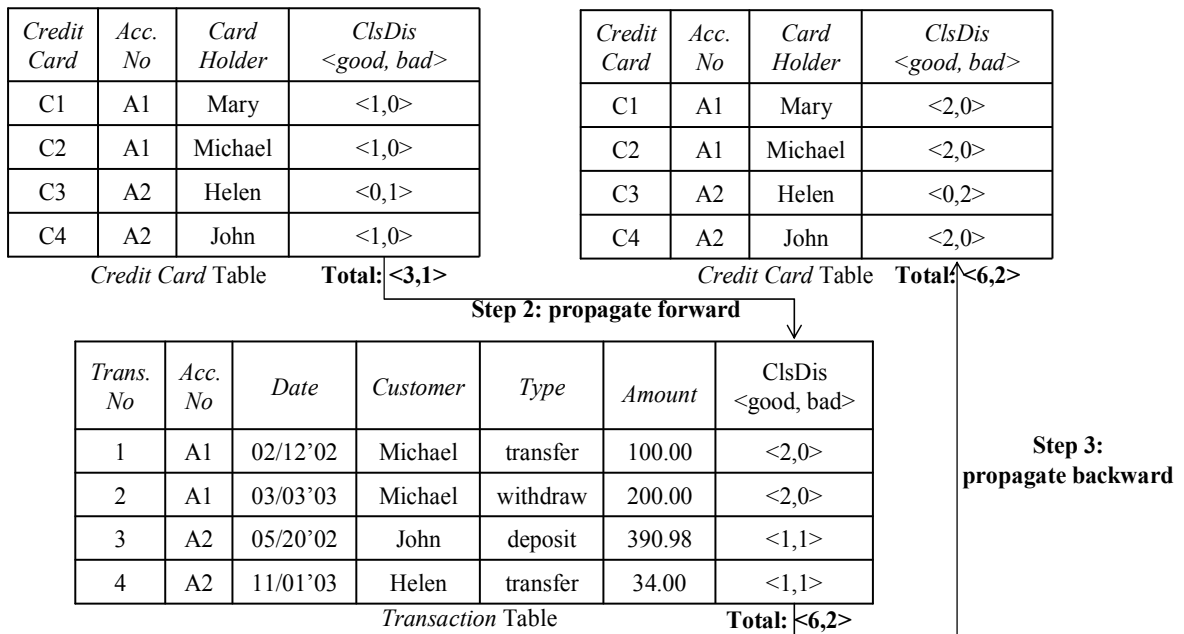


Figure 3. Class Propagation on the Example Database with 2 tables

by edges, the resulted graph should be acyclic.

Under this assumption, for cases with more than two tables and complex schemas, class vectors are propagated in the depth-first order from the class table. This process may include both forward (when propagating from table at a higher level downward) and backward (from a lower table upward) propagations. The last table contains class information aggregated from all tables. To ensure all other tables contain the same information, class vectors are then propagated back in one pass.

5. EXPERIMENTAL RESULTS

5.1 Experiment Settings

We compared the performance of feature selection on multiple tables through class propagation (*CP* algorithm) with feature selection on joined table (*Join* algorithm). Since they both return the same set of features, we only need to compare their running time. We also examined the effect of feature selection on classification.

We rank features according to *information gain*, as it is one of the most often used evaluation criteria. Then we select a top percentage of features. When join is needed, it is done as a standard database operation by using *Microsoft SQL Server*. As this software has a limit on the number of attributes in any single table (1024), when the dataset exceeds this limit, we wrote an alternative join program which is a simple implementation of the *nested loop join* [11]. For classification, we implemented *RainForest* [4] to build a decision tree classifier, since decision trees are reasonably good in performance and easy to comprehend [2,9]. *RainForest* has been shown to be a fast classifier on large scale data, where the traditional decision tree classifier C4.5 [9] can not be used when data is very large or has very high dimensions.

All implementations are written in C++. Experiments were carried out on a PC with 2GHz CPU and 500M main memory running Windows XP.

5.2 Datasets

Mondial dataset is a geographic database that contains data from multiple geographical web data sources. We obtained it in the relational format online [12]. Our classification task is to predict the religion of a country based on related information contained in multiple tables. We consider all religions which are close to Christian² as positive class and all other religions as negative class. We ignored tables that only consist of

geographic information. Finally we have 12 tables, the number of attributes ranges from 1 to 5 and all tables have less than 150 records with two exceptions (one table has 1757 records and another has 680 records). About 69% of data is negative and 31% is positive. 10 fold cross validation is used on this dataset.

Yeast Gene Regulation dataset was deduced from KDD Cup 2002 task 2 [6]. We obtained 11 tables that contain information about genes (detailed descriptions are omitted due to space limit). The biggest table contains keywords produced from abstracts that discuss related genes by using standard text processing techniques (removing stopwords, word stemming). It has 16959 records with 6043 attributes. The class table has 3018 records with the class label, which represents the effect of gene on the activity level of some hidden system in yeast (“has changes”(1%), “has controlled changes”(2%), “no changes”(97%)). Separated training and testing samples are used as provided in KDD Cup.

5.3 Experiment Results

Table 1 shows the running time on each stage of both algorithms and classification results on both datasets. Note that the step for building the classifier is exactly the same for both approaches, since we have the same joined data at this stage. Also, when 100% features are selected, i.e., there is no feature selection, both algorithms degrade to the same method.

It can be seen that our CP algorithm runs much faster than the join approach. The breakdown of the running time shows that the major gain is on the time needed for joining the data. As explained earlier, the join approach has to join a much larger dataset, taking a long time, whereas we only need a fraction of that time joining much less features. The experiments also show that our class propagation process is very fast and efficient, giving us the benefit of doing feature selection before join at very little cost.

For *Mondial* dataset, the accuracy difference is significant between 40% and 50% feature subsets, suggesting that some attributes are very helpful in identifying the class label, although they may not be ranked very high. In general, when more of data privacy is preserved (with less features revealed to other parties), classification accuracy starts to decrease. However, if the user has very strict privacy requirements, then we can only select less features to satisfy such constraint.

For *Yeast Gene* dataset, a small number of features provide very accurate classification and all other features are irrelevant. It is shown by the perfect accuracy starting from 5% feature subset. When more features are selected, it simply prolongs the running time without changing accuracy. On the other hand, the

² Armenian Orthodox, Bulgarian Orthodox, Christian, Christian Orthodox, Eastern Orthodox, Orthodox, Russian Orthodox

total running time of the *Join* approach without feature selection is less than the total time with 10% or more feature selection. This is because the effect of feature selection on classification time is offset by the time

spent for feature selection itself. However, this is not the case for our *CP* algorithm, as the time of our feature selection is much shorter and the total running time always benefits from feature selection.

Running Time (seconds)		Mondial Dataset					Yeast Gene Dataset				
		Features selected (%)					Features selected (%)				
		20	30	40	50	100	5	10	20	50	100
Join	Join	137	137	137	137	137	1099	1099	1099	1099	1099
	Feature Selection	9.1	10.6	11.7	14.7	0	1265	1587	1879	2549	0
	Building Classifier	32.1	41.2	50.1	53.5	119	50	176	370	1001	1739
	Total	178.2	188.8	198.8	205.2	256	2414	2862	3348	4649	2838
CP	Class Propagation	1.2	1.2	1.2	1.2	0	9	9	9	9	0
	Feature Selection	0.4	0.5	0.5	0.5	0	130	144	158	228	0
	Join	1	1	1	1	137	67	118	226	589	1099
	Building Classifier	32.1	41.2	50.1	53.5	119	50	176	370	1001	1739
	Total	34.7	43.9	52.8	56.2	256	256	447	763	1827	2838
Accuracy (%)		71.5	72.1	69.2	98.1	98.5	100	100	100	100	100

Table 1. Comparison of Running Time / Accuracy

6. DISCUSSION

In this paper, we present a way of selecting features in multiple data sources without join. With a clever propagation of class vectors, each local data source receives the same class distribution information as produced by a join approach. Features can then be evaluated locally. Thus the resulted feature selection scheme is highly scalable, while limiting the amount of information disclosed to other data sources.

The idea of class propagation can be used to develop more complex feature selection solutions. For example, we can incorporate the privacy constraints directly into feature evaluation by defining privacy score (from 0 to 1) for each feature. Features with higher privacy scores should have less chance to be selected. Since such score is available at each local data source, we can design some complex measure to evaluate the features based on not only the relevance to the class label, but also such privacy constraints.

7. REFERENCES

- [1] A. Atramentov, H. Leiva and V. Honavar, *A Multi-Relational Decision Tree Learning Algorithm – Implementation and Experiments*. ILP 2003.
- [2] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and Regression Trees*. Wadsworth: Belmont, 1984.
- [3] M. Dash and H. Liu, *Feature Selection for classification*. Intelligent Data Analysis – An International Journal, Elsevier, 1(3), 1997.
- [4] J. Gehrke, R. Ramakrishnan and V. Ganti, *RainForest: a Framework for Fast Decision Tree Construction of Large Datasets*. The 24th VLDB conference, 1998.
- [5] G. Hulten, P. Domingos and Y. Abe, *Mining Massive Relational Databases*, 18th International Joint Conference on AI - Workshop on Learning Statistical Models from Relational Data, Acapulco, Mexico, 2003.
- [6] KDD Cup 2002, <http://www.biostat.wisc.edu/~craven/kddcup/train.html>
- [7] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Kluwer Academic Publishers, 1998.
- [8] L. C. Molina, L. Belanche and A. Nebot, *Feature Selection Algorithms: A Survey and Experimental Evaluation*. ICDM 2002.
- [9] J.R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993
- [10] J. R. Quinlan and R. M. Cameron-Jones. *FOIL: A midterm report*. In Proc. 1993 European Conf. Machine Learning, Vienna, Austria, 1993.
- [11] R. Ramakrishnan and J. Gehrke, *Database Management Systems*. McGraw-Hill, 2003.
- [12] The Mondial Database, <http://dbis.informatik.uni-goettingen.de/Mondial/#Oracle>
- [13] X. Yi, J. Han, J. Yang, and P. Yu. *Crossmine: efficient classification across multiple database relations*. ICDE 2004.

Discarding Insignificant Rules during Impact Rule Discovery in Large, Dense Databases

Shiying Huang
CSSE, Monash University
Shiying.Huang@infotech.monash.edu.au

Geoffrey I. Webb
CSSE, Monash University
Geoff.Webb@infotech.monash.edu.au

Abstract

Considerable progress has been made on how to reduce the number of spurious exploratory rules with quantitative attributes. However, little has been done for rules with undiscritized quantitative attributes. It is argued that propositional rules can not effectively describe the interactions between quantitative and qualitative attributes. Aumann and Lindell proposed quantitative association rules to provide a better description of such relationship, together with a rule pruning techniques. Since their technique is based on the frequent itemset framework, it is not suitable for rule discovery in large, dense databases. In this paper, an efficient technique for automatically discarding insignificant rules during rule discovery is proposed, based on the OPUS search algorithm. Experiments demonstrate that the algorithm we propose can efficiently remove potentially uninteresting rules even in very large, dense databases.

Keywords

Rule discovery, impact rule, rule insignificance.

1 Introduction

It has been recognized that mining multiple models may lead to unmanageable numbers of rules. In some cases, the vast majority of the resulting rules are spurious or uninteresting. Summarization of existing rule pruning approach can be found in related works [4].

Although techniques for discovering rules from qualitative data are highly developed, there has been limited research into how best to discover rules from quantitative data. Srikant et al. [5] discretized the quantitative variables and mapped them into qualitative ones. Nevertheless, qualitative data have a lower level of measurement scale than quantitative data. Simply applying discretization may lead to information loss. [2] proposed a variant of association rule whose consequent is quantitative, and is described by its distribution instead of being discretized. They call these rules *quantitative association rules* (QAR). We follow Webb [7] by calling these rules *impact rules* instead, to distinguish them from quantitative rules as defined by Srikant et al [5].

Aumann and Lindell [2] proposed a technique for QAR pruning. However, their technique is inefficient for very dense databases. In this paper, we focus on

further developing their technique so that insignificant rules can be discarded during rule discovery in large, dense databases.

The rest of this paper is organized as follows. In section 2, we briefly describe the impact rule discovery problem settings we use throughout this paper. Section 3 presents the algorithm OPUS_IR_Filter which incorporates filtering spurious rules during rule discovery. Section 4 presents techniques for filtering insignificant impact rules. An anti-monotonic triviality filter is also proposed for improving the insignificance filter efficiency. We present and summarize our experiments in section 5, followed by conclusions in section 6.

2 Impact Rule Discovery

Exploratory rule discovery [9] seeks all models that satisfy some set of constraints. Examples include *association rule discovery* [1], *contrast set discovery* [3] and *QAR discovery*. For some of these techniques, both the antecedent and the consequent of the resulting rules are conjunctions of Boolean conditions. We use the term *propositional exploratory rule discovery* to encompass these techniques. However, Boolean conditions cannot effectively describe interactions between quantitative and qualitative variables and others. We introduce the *distributional-consequent rule (DCR) discovery*, which is designed specially to accommodate the need of discovering relations regarding quantitative variables. The influence of the antecedent on the *target* variable is described by distributional statistics. It is argued that DCR can present more useful interactions with quantitative data than can propositional rules [7, 2].

We characterize some impact rule discovery related terms as follows:

1. A , which is a conjunction of Boolean conditions, **covers** a records r , iff r satisfies all conditions in A . $Coverset(A)$ is the set of records covered by A .
2. An **impact rule** is a rule in form of $A \rightarrow target$, where the *antecedent* A is a conjunction of one or more Boolean conditions and the *target*, which is also referred to as the *consequent*, is the variable

(or combination of variables) in which we are interested. The status of the rule is the *influence* on the target of selecting the itemset records covered by antecedent A , which is described by the statistics of the target of $cover_{ser}(A)$.

3. An k -optimal impact rule discovery task is a 7-tuple: $KOIRD(D, \mathcal{C}, \mathcal{T}, \mathcal{M}, \lambda, \mathcal{I}, k)$.

D : is a nonempty set of records, which is called the database. A record is a pair $\langle c, v \rangle$, $c \subseteq \mathcal{C}$ and v is a set of values for \mathcal{T} . D is an available sample from the population \mathcal{D} .

\mathcal{C} : is a nonempty set of available Boolean conditions for impact rule antecedents, which is generated from the given data in D .

\mathcal{T} : is a nonempty set of the variables in whose distribution we are interested.

\mathcal{M} : is a set of constraints. There are two types of constraints *prunable* and *non-prunable constraints*. *Prunable constraints* are constraints that you can derive useful bounds for search space pruning and still ensures the completeness of information. Other constraints are *non-prunable constraints*.

λ : $\{X \rightarrow Y\} \times \{D\} \rightarrow \mathcal{R}$ is a function from rules and databases to values and defines a interestingness metric such that the greater the value of $\lambda(X \rightarrow Y, D)$ the greater the interestingness of this rule given the database.

\mathcal{I} : is the set of resulting impact rules satisfying all the constraints in \mathcal{M} , whose antecedents are conjunctions of conditions in \mathcal{C} . The rule consequent is the target variable \mathcal{T} .

k : is a user specified integer number denoting the number of rules in the ultimate set of solutions for this rule discovery task.

How the k -optimal constraint is enforced in rule discovery to facilitate better search space pruning is explained by Webb [7].

3 Algorithm

Aumann and Lindell [2] adopted the frequent itemset framework for AQR discovery. However, when there are numerous large itemsets, the overheads of itemset maintenance and the manipulation for frequent itemset techniques can be unwieldy. The separation of rule discovery process into two phases leads to loss of some opportunities for using filtering to improve the efficiency [6]. Impact rule discovery is based on the OPUS algorithm, and can successfully overcome these problems by performing efficient search space pruning and perform rule discovery in one phase.

OPUS_IR_Filter systematically searches through

Algorithm: OPUS_IR_Filter(Current, Available, \mathcal{M})

```

1. SoFar := {}
2. FOR EACH P in Available
  2.1 New := Current  $\cup$  P
  2.2 IF New satisfies all the prunable constraints in  $\mathcal{M}$ 
    except the nontrivial constraint THEN
    2.2.1 IF any direct subset of New has the same
      coverage as New THEN
      New  $\rightarrow$  relevant stats is a trivial rule
      Any superset of New is trivial, so do not
      access any children of this node, go to
      step 2.
    2.2.2 ELSE IF the mean of New  $\rightarrow$  relevant stats is
      significantly higher than all its direct parents
      THEN
      IF the rule satisfies all the other
      non-prunable constraints in  $\mathcal{M}$ 
      THEN record Rule to the ordered
      rule_list
    2.2.3 OPUS_IR(New, SoFar,  $\mathcal{M}$ )
    2.2.4 SoFar := SoFar  $\cup$  P
    2.2.5 END IF
  2.3 END IF
3. END FOR

```

Table 1: OPUS_IR_Filter

the condition combinations that may appear in the antecedent of an impact rule and prune the search space according to the requirements of a particular search. Depth-first search and the *branch and bound* [6] pruning technique is used for pruning the search space. Based on this structure, the memory requirement is moderate without the need to store all the frequent itemsets during the rule generation process, making it efficient for rule discovery in very large, dense databases.

Table 1 lists the pseudo code of OPUS_IR_Filter. *Current* is the antecedent of the rule currently being explored, *available* is the set of conditions that may be added to the antecedents of rules. \mathcal{M} is the set of constraints specified by the users. *Rule_list* stores the top- k optimal rules encountered. The filtering of insignificant impact rules is done at step 2.2.

4 Filtering Insignificant Rules

In order to make our demonstration easier, we contrived a fictitious database. It contains 4 attributes among which *target* is the quantitative variable in whose distribution we are interested and *num* is a numeric variable which is discretized into two ranges: greater than 10 and smaller than or equal to 10.

OPUS_IR_Filter finds 15 rules out of the fictitious database without using any filters, when searches with minimum coverage 0.3. However, by applying the filters the number of resulting rules can be greatly reduced.

tid	target	cat1	num	cat2
1	5.3	A	13	C
2	3	B	12	D
3	2	B	10	C
4	8.2	A	4	C
5	6	A	15	C
6	6.3	A	11	C
7	6.3	B	7	C
8	4.8	B	11	D
9	0	B	11	D
10	10	A	3	C

Table 2: Database: mean=5.19, variance=8.59878

4.1 Insignificant Impact Rules Aumann and Lindell defined a rule with a significantly different mean from all its parents as significant (desired). Using Aumann and Lindell’s definition, many rules whose performance isn’t significantly improved in comparison with their parents are found, which should be discarded for some discovery tasks. Some of the conditions in such rules may be negatively correlated to the consequent given the others [4].

DEFINITION 4.1. *An impact rule $A \rightarrow target$ is significant if the distribution of its target is improved at a given significance lever, in comparison with any of the target distribution of the rule $A' \rightarrow target$, where $A' \subset A$ and $|A'| = |A| - 1$.*

$$\begin{aligned} & \text{significant}(A \rightarrow target) = \\ & \forall x \in A, \text{dist}(A \rightarrow target) \gg \text{dist}(A - x \rightarrow target) \end{aligned}$$

A rule is insignificant if it is not significant.

The most important issue of implementing the insignificance filter is how exactly the term *significantly improved* is defined. We assume a context where the users seek impact rules that maximize a measure of interestingness, such as the *mean*. Equivalent techniques for minimization can be derived from our technique in a straightforward manner. In this paper, we regard that if a distribution $dist_a$ has a mean which is significantly more desirable than that of $dist_b$ at a specified significance level, then $dist_a$ is said to be *significantly improved* in comparison to $dist_b$. The most general impact rule is the rule $\emptyset \rightarrow target$.

4.1.1 Statistical Tests The χ^2 [4, 3] and Fisher exact test [9] that are both adopted to assess propositional rules significance, are not applicable for distributional-consequent rules. The standard z test is adopted by Aumann and Lindell for identifying QAR significance, which is inappropriate for small samples. To address this problem, we choose the t test instead. Furthermore, as the degree of freedom increases, the t test approaches the standard z test.

Using statistical tests to automatically discard the insignificant rules is inherently statistically unsound.

There are high risks of type-1 errors of accepting spurious or uninteresting rules, as well as type-2 errors of rejecting rules that are not spurious. However, this is not a problem of concern in our paper. Statistical soundness of such techniques can be achieved by applying the technique proposed by Webb [9] using a holdout set.

After applying the insignificance filter, only two impact rules remained as significant. The number of resulting rules goes through a decrease of near 90%.

4.2 Trivial Impact Rules Although applying a significance test during rule discovery enables successful removal of potentially uninteresting rules, this approach requires an additional pass through the database so as to obtain necessary statistics for each rule. Trivial propositional rules were defined by Webb [8]. We further develop their definition and present *trivial impact rules*, which are special cases of an insignificant impact rules. The property of triviality can speed up the identification of insignificant rules.

DEFINITION 4.2. *An impact rule $A \rightarrow target$ is trivial iff there is a rule $A' \rightarrow target$ where $A' \subset A$, and $\text{coverage}(A') = \text{coverage}(A)$.*

$$\begin{aligned} & \text{trivial}(A \rightarrow target) = \exists A' \subset A, \\ & \text{coverage}(A) = \text{coverage}(A') \end{aligned}$$

THEOREM 4.1. *“An impact rule is not trivial” is an anti-monotone constraint: if a rule $A \& B \rightarrow target$ is trivial wrt its parent rule $A \rightarrow target$, then all the rules, whose antecedent is a superset of $A \& B$, are also trivial.*

Proof. According to definition 4.2,

$$(4.1) \quad \text{coverset}(A) = \text{coverset}(A \& B).$$

For any record $r' \in D$, if

$$\begin{aligned} & r' \notin \text{coverset}(A \& B \& C) \\ (4.2) \quad & \Rightarrow r' \notin \text{coverset}(A \& B) \vee r' \notin \text{coverset}(C) \end{aligned}$$

Consider equation 4.1

$$\begin{aligned} & \Rightarrow r' \notin \text{coverset}(A) \vee r' \notin \text{coverset}(C) \\ & \Rightarrow r' \notin \text{coverset}(A \& C) \end{aligned}$$

So

$$\begin{aligned} & \forall r \notin \text{coverset}(A \& B \& C) \rightarrow r \notin \text{coverset}(A \& C) \\ (4.3) \quad & \text{coverset}(A \& C) \subseteq \text{coverset}(A \& B \& C) \end{aligned}$$

Since $A \& C$ is a subset of $A \& B \& C$,

$$(4.4) \quad \text{coverset}(A \& B \& C) \subseteq \text{coverset}(A \& C)$$

It can be concluded from 4.3 and 4.4 that

$$\text{coverset}(A \& B \& C) = \text{coverset}(A \& C)$$

The rule $A \& B \& C \rightarrow target$ is trivial w.r.t. its parent $A \& C \rightarrow target$. The theorem is proved.

It can be easily derived from theorem 4.1 that if a rule $A \rightarrow target$ is trivial, there must be a condition $x \in A$ where $\text{coverage}(A) = \text{coverage}(A - x)$. The

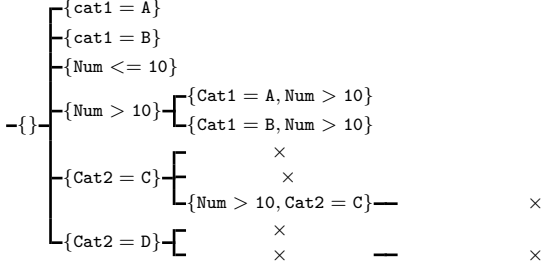


Figure 1: Pruned search space at step 2.2.1

database	rec-ords	attri-butes	condi-tions	Target
Abalone	4117	9	24	ShuckedWeight
Heart	270	13	40	MaxHeartRate
Housing	506	14	49	MEDV
German credit	1000	20	77	CreditAmount
Ipums.la.97	70187	61	1693	TotalIncome
Ipums.la.98	74954	61	1610	TotalIncome
Ipums.la.99	88443	61	1889	TotalIncome
Ticdata2000	5822	86	771	AveIncome
Census income	199523	42	522	Wage/Hour
Covtype*	581012	55	131	Evaluation

Table 3: Basic information of the databases we used

Database	Sig rules in all	Nontri rules in all	Sig rules in nontri
Abalone	173(173)	998	173
Heart	52(100)	923	54
Housing	83(288)	935	84
German credit	31(295)	738	43
Ipums.la.97	31(1000)	31	1000
Ipums.la.98	133(1000)	138	803
Ipums.la.99	297(1000)	578	507
Ticdata2000	1(1000)	564	1
Census income	30(1000)	466	42
Covtype*	316(1000)	386	866

Table 4: Comparison in number of rules

Database	impact rules	trivial Filter	sig rules	
			Insig	Both
abalone	0.29	0.57	0.75	0.74
heart	0.05	0.08	1.16	1.2
housing	0.06	0.16	1.62	1.47
german-credit	0.47	0.85	30.35	29.14
ipums.la.97	7.25	471.56	7365.23	623.52
ipums.la.98	1382.66	1551.8	1871.35	1860.31
ipums.la.99	874.2	1006.9	1886.07	1414.88
ticdata2000	1996.57	2082.1	10933.98	10808.03
census-income	873.74	1396.2	3960.84	3781.6
Covtype*	8927.16	9164.55	9640.63	9451.2

Table 5: Running time for discovering rules (in seconds)

distribution of these two rules are exactly the same, since they cover the same set of records. The triviality of rules is more powerful in its effect, since it is anti-monotone enables more effective search space pruning during rule discovery. Theorem 4.1 justifies our pruning at step 2.2.1.

Figure 1 shows the effect of pruning according to triviality in OPUS_IR_Filter search space for the fictitious database. As an example, node $\{\text{Num} > 10, \text{Cat2} = D\}$ is trivial, so the whole branch under this node should be pruned, according to theorem 4.1. After applying the triviality filter of impact rules, 6 out of the 15 rules found without using filters are removed.

5 Experimental Evaluation

We evaluate our algorithm by applying OPUS_IR_Filter to 10 databases selected from UCI repository and KDD archive, which are described in table 3. We applied 3-bin equal-frequency discretization to map all the quantitative attributes, other than the target variable, into qualitative ones. The significance level for the insignificance filter is 0.05. The program was run on a computer with PIII 933MHz processor, 1.5G memory and 4G of virtual memory, with minimum coverage and maximum number of conditions that may appear on the antecedents respectively set to 0.01 and 5 (except for *covtype*, which is set to 4).

First, we ran our program by using no filters, to find the top 1000 impact rules with highest impact. Second, the insignificance filter is applied to discover the top 1000 significant impact rules. The two sets of resulting rules were compared to find the number

of significant rules in the top 1000 impact rules. The triviality filter was then applied to find the top 1000 nontrivial impact rules, followed by comparisons to find the number of nontrivial rules in top 1000 impact rules and the number of significant rules in the top 1000 nontrivial rules. Finally, we applied both filters to find the top 1000 significant rules, and how incorporating the triviality filter can improve the efficiency is exhibited. Experimental results are in table 4 and table 5.

5.1 Result Analysis The second column of table 4 shows the number of significant rules in the top 1000 impact rules. Most databases go through a dramatic change in the resulting rules after the significance filter is applied. The number of resulting significant impact rules for *abalone*, *heart*, *housing* and *German credit* is less than 1000. The parenthesized numbers are the actual numbers of resulting significant impact rules discovered in these databases.

From column 3 and column 4 of table 4, it can be concluded that although the triviality filter can not automatically discard as many spurious impact rules as those by the significance filter, the decrease is also considerable. Notably for *ipums.la.97* only 31 rules among the top 1000 impact rules found without using any filter is nontrivial, while all the nontrivial impact rules are accepted as significant! For databases *ipums.la.98*, *ipums.la.99*, *covtype*, *ticdata2000* and *census-income*, more than 40% of the resulting impact rules are discarded as trivial.

The results justifies our argument about the efficiency of triviality filter: Applying only the triviality fil-

Database	Frequent Itemsets	CPU time(sec)
abalone	11131	0.07
heart	91213	0.11
housing	129843	0.2
german-credit	2721279	4.16
ipums.la.97	-	stop after 18462.20
ipums.la.98	-	stop after 17668.01
ipums.la.99	-	stop after 10542.40
ticdata2000	-	stop after 103.17
census-income	314908607	7448.52
covtype*	3810921	1496.76

Table 6: Results for Apriori

ter requires less CPU time, and the efficiency of insignificance filter improves when combined with the triviality filter. The triviality filter is an efficient complement for the insignificance filter.

5.2 Comparisons As is mentioned before, Aumann and Lindell’s algorithm for removing insignificant AQR uses the frequent itemset framework, which is limited in its capacity to analyze dense data by the requirement of vast amount of memory to store all the frequent itemsets and the computation to maintain those frequent itemsets during the generation procedure. It is after this stage that the significance test is performed on the set of resulting rules.

Since we failed to find QAR implementation, we compile and run Christian Borgelt’s Apriori implementation using exactly the same environment and parameter settings as for OPUS_IR.Filter. Target attributes are deleted from the databases, so that the frequent itemsets found by Christian Borgelt’s Apriori program are the antecedents of QAR discovered by Aumann and Lindell’s approach. The running time and the numbers of frequent itemsets discovered in each of the 10 databases are listed in table 6. By comparing the experimental results, Apriori cannot successfully work on databases with huge number of conditions, examples are *ipums.la.97*, *ipums.la.98*, *ipums.la.99* and *ticdata1000*, whose number of conditions all exceed 700. Apriori stops because of insufficient memory for these databases. However, OPUS_IR.Filter can be applied to the above databases successfully and efficiently. The time spent on looking for all the frequent itemsets in *german-credit* and *census-income* are much longer than that for OPUS_IR.Filter. Although for *abalone* and *covtype* the running time seems better than our approach, it should be noted that Apriori is only searching for the frequent itemsets, without performing the expensive computations and data accesses associated with calculating the statistics for the target attribute for each itemset. However, it is known to all that going through the data is one of the most disaster for efficiency. Situation gets worse as the size of database increases. Even if we do not take the time spent on itemset discovery

into account, to do significance test over all the resulting frequent itemset is inefficient, since the number of itemsets found in some of the databases exceeds 10^6 . It is safe to conclude that OPUS_IR.Filter is efficient for deriving rules from very large dense databases, for which Aumann and Lindell’s approach cannot.

6 Conclusions

Observing that there is a lack in research on distributional-consequent rule pruning, Aumann and Lindell proposed a technique for identifying potentially uninteresting rules after rule discovery. Their technique is based on the frequent itemset mechanism and is therefore inefficient for large, dense databases. Furthermore, the standard z test, which they use is not suitable for small samples. We proposed an efficient technique for removing insignificant impact rules using the student’s t test, which is a better approximation for small samples. Our algorithm is based on the OPUS framework, which enables efficient removal of insignificant rules even for large dense databases. By utilizing the anti-monotonicity of trivial rules, which is a subset of insignificant ones, more efficient search space pruning can be facilitated. The triviality filter for is provided both as an alternative and a complement to the insignificance filter. Experimental result showed that our algorithm can successfully remove potentially uninteresting impact rules, especially in very large, dense databases for which the frequent itemset approaches fail to.

References

- [1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*.
- [2] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *Knowledge Discovery and Data Mining*, pages 261–270, 1999.
- [3] S.D. Bay and M.J. Pazzani. Detecting group differences: Mining contrast sets. In *Data Mining and Knowledge Discovery*, pages 213–246, 2001.
- [4] B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *Knowledge Discovery and Data Mining*, pages 125–134, 1999.
- [5] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*.
- [6] G. I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.
- [7] G. I. Webb. Discovering associations with numeric variables. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–388. ACM Press, 2001.
- [8] G. I. Webb and Songmao Zhang. Efficient techniques for removing trivial associations in association rule discovery. In *in the proceedings of ICAIS2000*, 2002.
- [9] G.I. Webb. Statistically sound exploratory rule discovery, 2004.

SPID4.7: Discretization Using Successive Pseudo Deletion at Maximum Information Gain Boundary Points

Somnath Pal*

Himika Biswas†

Abstract

Discretization is the process of converting the continuous attributes of the database into discrete ones in order to apply some classification algorithms. This is an important problem in developing generally applicable methods in machine learning and data mining for classification and prediction. This paper introduces a new technique for discretization based on successive pseudo deletion of instances to reduce the conflicting instances, i.e., by reduction of noise in the database. Such successive pseudo deletions in the database are performed by introducing threshold points on maximum information gain boundary points of the continuous attributes. Our empirical experiments show that the state of the art algorithms for learning, such as CN2, C4.5, Naive-Bayes and RISE give improvement in performances with the discretized output from our method than outputs with other state of the art discretization algorithms.

Keywords: Machine Learning, Continuous attribute, Data pre-processing, Discretization, Classification and Prediction.

1 Introduction.

Many algorithms developed in machine learning, data mining and uncertain reasoning tasks can not handle continuous features [1], [4]. To use them on real-world data sets, continuous attributes must first be discretized into small number of distinct ranges. Also discretization provides a kind of insight into critical values in a continuous attribute. Furthermore, the response time of many classifier inducing algorithms, such as RISE [6], increase if the data is continuous.

Much work has been done in discretization of continuous valued attributes [3], [9], [11], etc. In [7] a valuable systematic review of a number of works on discretization have been carried out. It has been found there that Fayyad and Irani's algorithm based on Minimum Description Length Principle (MDLP) [9] achieved

best overall results. The same superiority of MDLP discretizer was further established in [8].

In this work we introduce a discretization method based on a novel concept called successive pseudo deletion (for noise reduction) and show that it achieves a favorably competitive performance compared to MDLP discretization method. Our experimental evaluation is carried on 20 data sets and using CN2 [5], C4.5 [14], Naive-Bayes [7], and RISE [6] algorithms which are used for classification and prediction. In §2, we present our algorithm, SPID4.7 (Selective Pseudo Iterative Deletion 4.7), that uses the new concept - pseudo deletion. Then in §3 we include details of experimental design for comparative empirical evaluation of our algorithm with MDLP discretizer. Next, in §4, we give comparative results of our discretization with the MDLP and also in-built (local) discretization methods of state of the art machine learning algorithms. This is followed by §5 where a discussion on related works for data pre-processing by discretization and SPID4.7's standing with respect to them is included. The conclusion is summarized finally in §6.

2 Discretization by Pseudo-Deletion.

Our algorithm (SPID4.7), based on successive pseudo deletion at maximum information gain boundary points works on iterative noise reduction of the database. In the next subsection we have presented the algorithm SPID4.7 using the method of successive pseudo deletion.

2.1 SPID4.7 discretization algorithm. We view the problem of discretization from a new angle. When there is no discretization of continuous attributes, the whole instance table consists of highest number of conflicting instances, i.e., instances with same set of attribute values but different class values. On the other extreme, if all continuous attributes are discretized at all possible continuous values, then there are minimum number of conflicting instances (actually no conflicting instances if there is no noise in the training data). Our successive pseudo deletion approach of discretization introduces threshold points in different continuous attributes of the database globally at maximum infor-

*Department of Computer Science & Technology, Bengal Engineering & Science University, Shibpur, Howrah - 711103, India. Email: *sp@becs.ac.in*

†Department of Computer Science & Technology, Bengal Engineering & Science University, Shibpur, Howrah - 711103, India. Email: *himika@cs.becs.ac.in*

INPUT: Set of instances (S) with attributes stored in table $attb[][]$ (where element $attb[i][j]$ is the value of the j^{th} attribute in the i^{th} instance), and table $class[]$ (where $class[i]$ stands for the class value of the i^{th} instance). Also m , user-set minimum instances constraint within two discretized ranges.

OUTPUT: Discretized instance set in table $attb[][]$.

```

begin
  for each continuous attribute in S
    insert threshold point at the boundary point having maximum information gain;
  find pseudo_deletion_count (PDEL_count) by majority vote;
  while (PDEL_count  $\neq$  0) and (all boundary points have not yet been considered) do
    begin
      for each continuous attribute
        select the boundary point which has maximum information gain and calculate PDEL_reqd;
      find the min_PDEL_reqd among the above selected points;
      if min_PDEL_reqd < PDEL_count
        then begin
          accept the selected boundary point as threshold point;
          PDEL_count  $\leftarrow$  min_PDEL_reqd;
          remove boundary point(s) violating  $m$  on both sides of selected threshold point;
        end;
      else begin
        for each continuous attribute
          select the boundary point which has minimum information gain and calculate PDEL_reqd;
        find the max_PDEL_reqd among the above selected points;
        if max_PDEL_reqd > PDEL_count then reject the point(s) with max_PDEL_reqd;
        else
          reject the point(s) with max_PDEL_reqd among the maximum gain points of each continuous attribute;
        end; /* of else */
      end; /* of while */
    end.

```

Table 1:
SPID4.7 algorithm.

mation gain points one by one such that the noise in the database gets reduced with introduction of each threshold point.

Any discretization algorithm has got two parts: (1) (threshold point) selection criterion and (2) stopping criterion. The threshold point selection criterion in SPID4.7 are mainly made to reduce noise at successive steps conditional on information gain (or entropy). The stopping criterion of SPID4.7 are simple: either the noise in the data base is zero or the noise is minimum on the path traversed by the greedy algorithm and there is no more boundary point [9] left to be considered as probable threshold point.

Based on the above, binary threshold points are added to all continuous attributes at the point where information gain of the each attribute is maximum among its boundary points. Then if there are conflicting instances we determine how many instances are to be deleted (not actually deleted) to reduce the conflict in the data set according to majority voting.

After incorporating binary threshold points to all continuous attributes, in case there are conflicting in-

stances, we choose the maximum gain point for each continuous attribute and then temporarily incorporate a threshold point there. Then we calculate the number of instances to be deleted (pseudo deletion required), according to majority voting. If the minimum pseudo deletion required among these selected points is less than the previous pseudo deletion required then the point is accepted as a threshold point. Otherwise, if the maximum pseudo deletion required among the minimum gain points of all continuous attributes is greater than the previous pseudo deletion required then that (those) boundary point(s), where pseudo deletion required is equal to the maximum, is (are) rejected, i.e., not considered as boundary point(s) any more. However, if the maximum pseudo deletion required among minimum gain points is less than or equal to the previous pseudo deletion required then we reject the maximum pseudo deletion required point(s) among the maximum gain points of each continuous attribute. If a boundary point is accepted as a threshold point then the previous pseudo deletion required is replaced by the pseudo deletion required calculated after accepting the

Data set	# E	# A	# CL	M.V.
anneal(ann)	798	6C, 14D	6	yes
australian(aus)	690	6C, 8D	2	no
credit(cre)	690	6C, 9D	2	yes
dermatology(der)	330	1C, 33D	6	yes
echocardio(ech)	132	8C, 3D	3	yes
ecoli(eco)	336	7C, 0D	8	no
glass(gla)	214	9C, 0D	7	no
hungary(hun)	294	5C, 8D	5	yes
heart-statlog(sta)	270	5C, 8D	2	no
switzerland(swi)	123	4C, 8D	5	yes
horse-colic(hor)	300	7C, 15D	2	yes
imports-85(imp)	201	15C, 10D	7	yes
iris(iri)	150	4C, 0D	3	no
liver-disorder(liv)	345	6C, 0D	2	no
machine(mac)	209	7C, 0D	30	no
newthyroid(thy)	214	5C, 0D	3	no
pima(pim)	768	8C, 0D	2	no
vehicle(veh)	94	18C, 0D	4	no
wine(win)	178	13C, 0D	3	no
wisconsin(wis)	699	9C, 0D	2	yes

Table 2:

Characteristics of Data sets.
E - Examples, A - Attributes, CL - Classes, M.V. - Missing Values.
C - #Continuous Attributes, D - #Discrete Attributes.

selected boundary point. The process is repeated until either the pseudo deletion required becomes zero (i.e., there is no noise in the data set) or there is no boundary point left to be examined.

Some authors like [14] used a constraint on the minimum number of instances in each of the ranges, which means that any given range may include a mixture of class values. In SPID4.7, this minimum instance(s) constraint between ranges is called m and it is an user-set value. When a boundary point is selected as a threshold point, if there are any other unselected boundary point(s) violating the minimum instance(s) constraint at either side of it then the unselected boundary point(s) is (are) rejected. The complete algorithm of SPID4.7 is shown in Table 1.

3 Experimental Design.

To empirically evaluate the performance of SPID4.7 algorithm, experiments are performed on 20 real-world data sets drawn from the University of California at Irvine data repository [2]. The characteristics of the data sets are shown in Table 2.

For empirical evaluation of SPID4.7 we have chosen four well known state of the art classification algorithms: CN2 [5], C4.5 [14], Naive-Bayes [7], and RISE [6]. Missing values for both MDLP and SPID4.7 algorithms are replaced by most frequent value of the attribute.

Each of CN2, C4.5, Naive-Bayes and RISE was

Data Set	in-built discretizer <i>acc. \pm s.d.</i>	MDLP <i>acc. \pm s.d.</i>	SPID4.7 <i>acc. \pm s.d.</i>
ann	86.59 \pm 4.39	92.83 \pm 2.44	91.67 \pm 2.96 ⁶
aus	81.79 \pm 4.70	81.06 \pm 3.97	81.32 \pm 4.18 ⁶
cre	82.10 \pm 4.37	79.78 \pm 4.61	80.57 \pm 4.75 ⁵
der	86.53 \pm 7.02	90.83 \pm 4.64	90.66 \pm 4.37 ⁶
ech	62.06 \pm 13.03	60.91 \pm 13.22	61.51 \pm 11.83 ⁶
eco	78.99 \pm 6.28	79.87 \pm 6.17	73.99 \pm 6.81 ¹
gla	66.12 \pm 8.85	67.07 \pm 8.66	68.19 \pm 9.54 ⁶
hun	62.81 \pm 9.45	64.90 \pm 8.57	64.71 \pm 7.75 ⁶
sta	77.28 \pm 8.66	76.39 \pm 6.61	76.17 \pm 7.55 ⁶
swi	36.30 \pm 12.50	33.39 \pm 10.64	38.00 \pm 13.20 ⁴
hor	75.06 \pm 8.14	71.26 \pm 7.93	71.20 \pm 8.94 ⁶
imp	75.80 \pm 9.64	74.99 \pm 8.84	71.11 \pm 10.15 ³
iri	93.33 \pm 5.65	94.39 \pm 4.68	96.66 \pm 4.06 ²
liv	66.60 \pm 7.59	67.59 \pm 7.26	69.57 \pm 7.42 ⁵
mac	43.45 \pm 11.47	38.65 \pm 11.32	47.97 \pm 11.89 ¹
thy	94.40 \pm 5.38	95.60 \pm 4.34	95.42 \pm 3.88 ⁶
pim	74.20 \pm 4.58	71.98 \pm 5.73	72.95 \pm 5.42 ⁶
veh	60.20 \pm 17.37	72.99 \pm 12.27	72.41 \pm 12.83 ⁶
win	93.25 \pm 5.61	94.28 \pm 6.28	96.85 \pm 4.77 ¹
wis	94.25 \pm 2.61	93.76 \pm 3.02	94.56 \pm 2.76 ⁵

Table 3:

Accuracy Comparisons Using CN2 algorithm.

Empirical Results: acc.=average accuracy and s.d.=standard deviation.

Superscripts denote confidence levels comparing MDLP and SPID4.7: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Data Set	in-built discretizer <i>acc. \pm s.d.</i>	MDLP <i>acc. \pm s.d.</i>	SPID4.7 <i>acc. \pm s.d.</i>
ann	93.96 \pm 2.58	92.76 \pm 2.66	92.98 \pm 2.52 ⁶
aus	83.95 \pm 4.30	86.20 \pm 3.66	85.61 \pm 4.05 ⁶
cre	85.04 \pm 4.02	85.85 \pm 4.54	85.76 \pm 3.79 ⁶
der	92.11 \pm 5.69	91.99 \pm 5.68	92.11 \pm 5.69 ⁶
ech	64.28 \pm 13.35	65.12 \pm 11.86	68.62 \pm 11.64 ⁵
eco	81.84 \pm 4.82	80.65 \pm 6.11	82.45 \pm 6.06 ⁵
gla	66.06 \pm 8.19	73.67 \pm 8.16	72.24 \pm 8.20 ⁶
hun	62.08 \pm 8.42	64.98 \pm 8.62	61.05 \pm 10.85 ⁴
sta	77.80 \pm 8.04	78.91 \pm 7.33	80.24 \pm 7.42 ⁶
swi	34.43 \pm 15.19	36.24 \pm 12.60	40.95 \pm 12.81 ⁴
hor	83.74 \pm 6.03	84.19 \pm 5.93	82.13 \pm 7.13 ⁵
imp	78.67 \pm 9.87	77.68 \pm 9.85	78.68 \pm 9.31 ⁶
iri	95.59 \pm 5.09	95.46 \pm 5.40	96.66 \pm 4.06 ⁶
liv	66.73 \pm 6.96	69.22 \pm 7.67	67.37 \pm 7.57 ⁶
mac	45.23 \pm 11.71	43.84 \pm 11.44	47.09 \pm 11.64 ⁵
thy	93.46 \pm 5.13	94.68 \pm 5.01	94.21 \pm 4.79 ⁶
pim	74.56 \pm 5.06	75.37 \pm 5.13	76.26 \pm 5.44 ⁶
veh	63.50 \pm 14.09	68.06 \pm 15.51	68.35 \pm 16.07 ⁶
win	93.49 \pm 5.53	96.38 \pm 4.47	96.30 \pm 4.47 ⁶
wis	95.41 \pm 2.26	96.07 \pm 2.29	95.39 \pm 2.71 ⁵

Table 4:

Accuracy Comparisons Using C4.5 algorithm.

Empirical Results: acc.=average accuracy and s.d.=standard deviation.

Superscripts denote confidence levels comparing MDLP and SPID4.7: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Data Set	in-built discretizer <i>acc. \pm s.d.</i>	MDLP <i>acc. \pm s.d.</i>	SPID4.7 <i>acc. \pm s.d.</i>
ann	93.86 \pm 2.60	92.96 \pm 2.91	93.38 \pm 3.04 ⁶
aus	73.88 \pm 5.62	85.42 \pm 3.76	86.41 \pm 3.83 ⁶
cre	74.03 \pm 5.32	86.29 \pm 3.90	86.87 \pm 3.90 ⁶
der	97.52 \pm 2.07	98.42 \pm 1.94	98.48 \pm 1.84 ⁶
ech	57.10 \pm 11.98	73.55 \pm 10.39	76.46 \pm 11.90 ⁶
eco	66.49 \pm 9.05	83.86 \pm 5.79	86.84 \pm 5.09 ¹
gla	40.93 \pm 9.83	74.70 \pm 8.23	76.61 \pm 7.36 ⁶
hun	59.54 \pm 9.04	67.08 \pm 9.17	68.31 \pm 9.61 ⁶
sta	70.74 \pm 8.55	83.56 \pm 7.52	83.56 \pm 7.63 ⁶
swi	24.09 \pm 14.04	43.10 \pm 12.66	45.67 \pm 12.78 ⁶
hor	77.27 \pm 7.01	78.87 \pm 6.13	79.07 \pm 6.16 ⁶
imp	73.84 \pm 9.76	81.47 \pm 7.85	70.94 \pm 9.23 ¹
iri	91.60 \pm 5.30	94.53 \pm 4.93	95.07 \pm 4.58 ⁶
liv	58.02 \pm 7.61	66.32 \pm 7.77	70.34 \pm 8.22 ²
mac	63.06 \pm 9.24	47.68 \pm 11.12	53.94 \pm 10.32 ¹
thy	91.81 \pm 6.91	95.71 \pm 4.06	98.14 \pm 3.06 ¹
pim	65.52 \pm 4.39	77.13 \pm 4.45	77.94 \pm 3.90 ⁶
veh	44.40 \pm 13.98	57.18 \pm 14.83	57.40 \pm 13.88 ⁶
win	70.60 \pm 9.38	99.33 \pm 2.12	99.22 \pm 2.23 ⁶
wis	97.40 \pm 1.85	97.14 \pm 2.13	97.25 \pm 2.06 ⁶

Table 5:

Accuracy Comparisons Using Naive-Bayes algorithm.
Empirical Results: acc.=average accuracy and s.d.=standard deviation.
Superscripts denote confidence levels comparing MDLP and SPID4.7: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Data Set	in-built discretizer <i>acc. \pm s.d.</i>	MDLP <i>acc. \pm s.d.</i>	SPID4.7 <i>acc. \pm s.d.</i>
ann	98.67 \pm 1.29	93.93 \pm 2.81	94.79 \pm 2.32 ⁵
aus	83.13 \pm 4.45	85.42 \pm 4.13	86.46 \pm 3.74 ⁵
cre	83.16 \pm 4.34	85.94 \pm 3.79	87.01 \pm 3.45 ⁵
der	95.94 \pm 3.35	95.39 \pm 3.69	96.00 \pm 3.50 ⁶
ech	71.09 \pm 11.75	71.96 \pm 9.99	72.86 \pm 10.34 ⁶
eco	83.86 \pm 5.51	83.62 \pm 4.96	83.92 \pm 5.87 ⁶
gla	72.48 \pm 8.27	78.78 \pm 7.68	79.79 \pm 7.44 ⁶
hun	63.68 \pm 7.83	65.22 \pm 8.74	67.15 \pm 9.73 ⁶
sta	81.56 \pm 7.57	78.74 \pm 8.57	79.41 \pm 8.29 ⁶
swi	33.64 \pm 14.96	39.99 \pm 12.39	47.60 \pm 13.50 ¹
hor	84.73 \pm 6.08	82.40 \pm 6.83	83.40 \pm 6.78 ⁶
imp	75.20 \pm 8.99	83.25 \pm 9.03	80.07 \pm 10.04 ⁵
iri	96.00 \pm 5.16	94.67 \pm 4.99	96.80 \pm 3.83 ³
liv	62.46 \pm 8.31	70.26 \pm 7.95	71.67 \pm 7.44 ⁶
mac	57.14 \pm 10.28	49.38 \pm 10.16	57.21 \pm 9.77 ¹
thy	95.54 \pm 4.96	95.89 \pm 4.02	97.39 \pm 3.51 ⁴
pim	72.39 \pm 4.81	74.84 \pm 4.44	75.86 \pm 4.85 ⁶
veh	58.31 \pm 15.40	70.78 \pm 16.21	73.84 \pm 14.02 ⁶
win	96.75 \pm 4.79	96.97 \pm 4.37	99.56 \pm 1.87 ¹
wis	96.51 \pm 1.86	96.02 \pm 2.31	95.02 \pm 2.67 ⁴

Table 6:

Accuracy Comparisons Using RISE algorithm.
Empirical Results: acc.=average accuracy and s.d.=standard deviation.
Superscripts denote confidence levels comparing MDLP and SPID4.7: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

run on each of the original data set with continuous values discretized by respective in-built local discretizer while inducing rules. On the otherhand, the original data sets are first discretized using SPID4.7 and the discretized data was run using CN2, C4.5, Naive-Bayes and RISE. Similarly, for comparisons, the original data sets are discretized using MDLP [9] algorithm and the discretized data were run using the same four algorithms. The accuracy of the induced rules were calculated using 10-fold cross-validation method. Such 10-fold cross-validation experiments were repeated 5 times.

4 Results and Empirical Comparisons.

Table 3 shows empirical results for CN2 algorithm run on its in-built discretizer, with data discretized by MDLP and SPID4.7. Similarly Table 4, Table 5 and Table 6 shows empirical results with C4.5, Naive-Bayes and RISE algorithms respectively in the same format. Although Tables 3-6, showing the accuracies obtained in individual data sets, are interesting in themselves, some interesting features of the results are summarized in Table 7. The results of Table 7 can be interpreted as follows.

Mean: The pre-processed discretized data from SPID4.7 generates higher mean across all domains for all four algorithms (CN2, C4.5, Naive-Bayes, and RISE) compared to their in-built (or locally) discretized data and MDLP discretization algorithm.

Number of wins: An obvious test is simply to compare the number of data sets where discretized output by SPID4.7 achieved higher accuracy compared to MDLP and in-built discretized output. As we can see from Table 7, in case of CN2, its in-built discretizer produces better accuracies for 8 data sets, whereas discretized output of SPID4.7 produces better accuracies for 12 data sets. Again SPID4.7 produces better accuracies for 11 data sets, whereas MDLP produces better accuracies in 9 data sets. Similarly, for C4.5, SPID4.7 wins over its in-built discretizer in 15 out of 20 data sets (1 draw) and over MDLP in 11 out of 20. In case of Naive-Bayes, SPID4.7 wins over its in-built discretizer in 16 out of 20 data sets and over MDLP in 17 out of 20 (1 draw). For RISE, SPID4.7 wins over its in-built discretizer in 16 out of 20 data sets and over MDLP in 18 out of 20.

Significant Win: A better alternative compared to the above is to count only those data sets where the difference was significant at a confidence level of 95% or higher (see Tables 3-6, where superscripts show the results of one-tailed paired t-test performed with MDLP and SPID4.7). This still shows improved results in favor of SPID4.7 discretization for all algorithms except for C4.5 algorithm, where the results of MDLP and SPID4.7 are the same.

Wilcoxon Signed-rank test: In our case, the result of the signed-rank test support that SPID4.7 produces better discretized output than CN2 with confidence of 81.71%. In case of C4.5, SPID4.7 produces better discretization compared to MDLP discretization with confidence 78.52% and in both cases of Naive-Bayes and RISE, SPID4.7 generates better outputs with confidence 99.97%.

Criteria	in-built discretizer	MDLP	SPID4.7
Algorithm CN2			
Mean	73.06	75.13	75.77
in-built(W-D-L)	—	10 – 0 – 10	12 – 0 – 8
MDLP(W-D-L)	10 – 0 – 10	—	11 – 0 – 9
Significant Win	—	2	4
Wilcoxon Test	—	—	89.71%
Algorithm C4.5			
Mean	74.41	77.87	78.22
in-built(W-D-L)	—	14 – 0 – 6	15 – 1 – 4
MDLP(W-D-L)	6 – 0 – 14	—	11 – 0 – 9
Significant Win	—	1	1
Wilcoxon Test	—	—	78.52%
Algorithm Naive-Bayes			
Mean	68.68	79.22	80.10
in-built(W-D-L)	—	17 – 0 – 3	16 – 0 – 4
MDLP(W-D-L)	3 – 0 – 17	—	17 – 1 – 2
Significant Win	—	1	4
Wilcoxon Test	—	—	99.97%
Algorithm RISE			
Mean	75.91	79.67	81.29
in-built(W-D-L)	—	12 – 0 – 8	16 – 0 – 4
MDLP(W-D-L)	8 – 0 – 12	—	18 – 0 – 2
Significant Win	—	1	5
Wilcoxon Test	—	—	99.97%

Table 7:

Summary of Accuracy Results.

W=Win, D=Draw, L=Loss. X(W-D-L) under the column Y means win-draw-loss of Y compared to X.

Significance Test and Wilcoxon Signed-Rank Test compare MDLP with SPID4.7.

5 Related Works.

SPID4.7, presented in this paper, is a supervised and global method of discretization; whereas MDLP [9] discretization is a supervised and local method of discretization. The ChiMerge system [11], is another local method that provides a statistically justified heuristic method for supervised discretization. Another statistical discretization method Khiops [3], based on chi-square statistics, have recently been published. There, an empirical comparative study of a number of discretization methods based on Naive-Bayes algorithm on 15 data sets was carried out. In that study, Equal Frequency method was ranked higher than ChiMerge and Equal Width, whereas Khiops along with MDLP discretizer was ranked above Equal Frequency method. From the empirical study carried out in the §4 of this paper, we see that SPID4.7 compares favorably with MDLP discretizer.

6 Conclusion.

We have presented a discretization algorithm based on a new concept – successive pseudo iterative deletion at

maximum information gain boundary points, that can be used for generating pre-processed data for state of the art machine learning algorithms for data mining tasks. Empirical evaluation of our algorithm SPID4.7 has shown that discretized data generated by it is better than state of the art discretization algorithm MDLP, which has been ranked very high as a discretizer along with Khiops [3] in a recent study.

It is a well-accepted opinion that there is no one superior discretization method that will give best results across all domains. However, it has been observed in [10], that discretization methods based on conditional entropy, perform very well. The success of SPID4.7 can possibly be attributed to the fact that its threshold point selection criteria are conditional (successive noise reduction in database) on maximum information gain boundary points. Furthermore, unlike MDLP and ChiMerge, SPID4.7 is a global discretization algorithm, which may have contributed to its better performance than the others. However, we feel that there is scope for further improvement of SPID4.7.

Future work on this paper involves time complexity analysis of SPID4.7. Other direction of future research includes the applicability of SPID4.7 on large databases. Finally, SPID4.7 can be made available on request to the second author.

Acknowledgement

The authors are grateful to D. Chakraborty for implementation of Naive-Bayes algorithm and A. M. Ghosh for many fruitful discussions on this work. The authors are also grateful to P. Clark, J. R. Quinlan and P. Domingos for making the source codes of CN2, C4.5 and RISE systems respectively available for this work.

References

- [1] C. Apte and S. Hong, *Predicting equity returns from security data*, Advances in Knowledge Discovery and Data Mining, AAAI Press and the MIT press, chapter 22, pp. 541–569, 1996.
- [2] C. L. Blake and C. J. Merz, *UCI repository of machine learning databases (Machine-readable data repository)*, (<http://www.ics.uci.edu/mllearn/MLRepository.html>) Department of Information and Computer Science, University of California, Irvine, 1999.
- [3] M. Boule, *Khiops: A statistical discretization method of continuous attributes*, Machine Learning, 55 (2004), pp. 53–69.
- [4] J. Cendrowska, *PRISM: An algorithm for inducing modular rules*, International Journal for Man-Machine Studies, 27 (1987), pp. 349–370.
- [5] P. Clark and R. Boswell, *Rule Induction with CN2: some recent improvements*, Machine Learning: Proceedings of the Fifth European Conference, Berlin, pp. 151–163, 1991.
- [6] P. Domingos, *Unifying instance-based and rule-based induction*, Machine Learning, 3 (1996), pp. 139–168.
- [7] J. Dougherty, R. Kohavi, and M. Sahami, *Supervised and unsupervised discretization of continuous features*, in Proceedings of the Twelfth International Conference on Machine Learning, A. Friedlitz and S. Russell, eds., Morgan Kaufmann, San Francisco, pp. 194–202, 1995.
- [8] T. Elomaa and J. Rousu, *General and efficient multisplitting of numerical attributes*, Machine Learning, 36/3 (1999), pp. 1–49.
- [9] U. M. Fayyad and K. B. Irani, *Multi-interval discretization of continuous-valued attributes for classification learning*, in Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp. 1022–1027, 1993.
- [10] J. W. Grzymala-Busse, *Discretization of numerical attributes*, in Handbook of Data Mining and Knowledge Discovery, W. Klösgen and J. M. Zytkow, eds., Oxford University Press, pp. 218–225, 2002.
- [11] R. Kerber, *ChiMerge: Discretization of numeric attributes*, in Proceedings of the Tenth National Conference on Artificial Intelligence, MIT Press, pp. 123–128, 1992.
- [12] J. R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann, 1993.

Iterative Mining for Rules with Constrained Antecedents

Zheng Sun*

Philip S. Yu†

Xiang-Yang Li‡

Abstract

In this study we discuss the following association rule mining problem: for a user-defined set \mathcal{A} of items, the objective is to compute all association rules (satisfying suitable support and confidence thresholds) *induced* by \mathcal{A} , where an association rule is said to be induced by \mathcal{A} if its antecedent (*i.e.*, LHS) is a subset of \mathcal{A} while the consequent (*i.e.*, RHS) contains no items in \mathcal{A} . In particular, we are interested in a multi-step scenario where in each step \mathcal{A} is incremented by one item and all association rules induced by the updated \mathcal{A} are to be computed. We propose an efficient iterative algorithm that can exploit mining information gained in previous steps to efficiently answer subsequent queries.

1 Introduction

The problem of discovering association rules is to compute, given a set of items $\mathcal{I} = \{I_1, I_2, \dots, I_n\}$ and a set \mathcal{T} of transactions each of which is a subset of \mathcal{I} , all association rules in the form of $X \Rightarrow Y$, where $X, Y \subset \mathcal{I}$ and $X \cap Y = \emptyset$. Here X is called the *antecedent* (or the LHS) of the rule, while Y is the *consequent* (or the RHS). The merit of the rule $r : X \Rightarrow Y$ may be measured by its *support* $\text{Supp}(r)$, the fraction of transactions containing both X and Y out of all transactions in \mathcal{T} , and *confidence* $\text{Conf}(r)$, the fraction of transactions containing Y that also contain X . Usually, we are interested in those *good rules* that satisfy some user-defined support and confidence thresholds, Supp_{\min} and Conf_{\min} .

Traditionally, association rule discovery is conducted in an *offline* manner, and the rules being sought are *general* ones without any constraint on either the antecedent or the consequent part. In this paper we study the problem of iterative mining with constrained antecedents. Each query of our problem is specified with Supp_{\min} , Conf_{\min} , and a set $\mathcal{A} \subset \mathcal{I}$ of *antecedent items* selected by the user. Let $\mathcal{C} = \mathcal{I} \setminus \mathcal{A}$ be the set of *consequent items*. The objective is to find association rules *induced* by \mathcal{A} , where a rule $r : X \Rightarrow Y$ is said to be induced by \mathcal{A} if $X \subseteq \mathcal{A}$ and $Y \subseteq \mathcal{C}$. In particular, we are interested in a multi-step scenario where in each step \mathcal{A} is incremented by one item and all association rules induced by the updated \mathcal{A} are to be computed. Therefore, it is critical to efficiently handle the previous mining results as new antecedent items are introduced.

To facilitate efficient computation of association rules in the query phase, we use an *adjacency lattice* (see [2]), denoted by $\mathcal{L}_{\text{itemset}}$, to store frequent itemsets with respect to a minimum support picked such that the entire adjacency lattice can be stored in the main memory.

Several different forms of constrained mining have been discussed in previous literature. Srikant *et al.* [8] considered the problem of computing association rules under constraints that are boolean expressions over the presence or absence of items. Bayardo *et al.* [4] described an algorithm that exploits user-defined constraints on support, confidence, and predicative advantage. Lakshmanan *et al.* [6] studied a rich class of anti-monotone and succinct constraints for itemsets and provided mining algorithms that achieve a significant degree of pruning for these constraints. Pei *et al.* [7] studied several types of constraints that are difficult to be handled using traditional techniques. Cong and Liu [5] provided an efficient technique to utilize previous mining information to compute frequent itemsets when the minimum support constraint is relaxed.

Aggarwal and Yu [2] proposed an online mining technique that can efficiently answer queries with different minimum support constraints using a pre-computed adjacency lattice of itemsets. Later Aggarwal *et al.* [1] extended this technique to mining profile association rules with quantitative attributes. Thomas *et al.* [9] studied interactive rule mining with constraint relaxations. In a sense, our iterative mining method complements the existing works by providing a more complete mechanism for iterative mining, with constraints defined on support, confidence, and antecedent/consequent.

2 Preliminaries

Let $\mathcal{A} = \{I_{i_1}, I_{i_2}, \dots, I_{i_k}\}$ be a set of items. For a given \mathcal{A} , an itemset X is called an *antecedent itemset* if $X \subseteq \mathcal{A}$. We use \mathcal{X} to denote the set of all *frequent antecedent itemsets* with supports no less than Supp_{\min} . An itemset Z is called a *rule itemset* if $Z \setminus \mathcal{A} \neq \emptyset$; such an itemset uniquely corresponds to a rule $Z \cap \mathcal{A} \Rightarrow Z \setminus \mathcal{A}$ (without considering any support and confidence thresholds) induced by \mathcal{A} . We call $Z \cap \mathcal{A}$ the *associated antecedent itemset* of Z . In this paper, we assume that the goal is to compute all rule itemsets corresponding to good rules (including those with null antecedents). With minor modifications, our algorithm can also be used to compute rules with some other properties.

We define the following lexicographic order $O(\cdot)$ on items: i) $O(I_j) < O(I_m)$ for any $I_j \in \mathcal{A}$ and $I_m \in \mathcal{C}$; ii) $O(I_{i_j}) < O(I_{i_m})$ if $I_{i_j}, I_{i_m} \in \mathcal{A}$ and $j < m$;

*Hong Kong Baptist University, sunz@comp.hkbu.edu.hk. The author was supported in part by Grant FRG/03-04/II-21.

†IBM T.J. Watson Research Center, psyu@us.ibm.com.

‡Illinois Institute of Technology, xli@cs.iit.edu. The author was supported in part by NSF under Grant CCR-0311174.

iii) $O(I_j) < O(I_m)$ if $I_j, I_m \in \mathcal{C}$ and $j < m$. We say that an item I_j is *ordered before* (after, respectively) I_k if $O(I_j)$ is less than (more than, respectively) $O(I_k)$.

For any itemset Z and any $I_m \in \mathcal{C}$, if $Z' = Z \cup \{I_m\}$ is a good rule itemset and $O(I_m) > O(I_j)$ for all $I_j \in Z$, we say that Z' is a *consequent extension* of Z . We use $\mathcal{E}_{con}(Z)$ to denote the set of all consequent extensions of Z . We define *antecedent extensions* $\mathcal{E}_{ant}(Z)$ analogously. We say that I_m is the *generating item* of Z' and denote it by $I_{gen}(Z')$. We use $\mathcal{I}_{gen}(Z)$ to denote the set of generating items of all itemsets in a set \mathcal{Z} of itemsets.

For the purpose of comparison, we first describe an Apriori-like offline algorithm, FindRulesOffline (see Algorithm 1), that computes all rule itemsets induced by \mathcal{A} assuming that the entire \mathcal{A} is given at once. This algorithm recursively calls Procedure FindRulesForItemset (see Procedure 1), which computes for a given itemset Z all rules that are extensions of Z .

Algorithm 1 FindRulesOffline(\mathcal{A})

- 1: $\mathcal{R} \leftarrow \emptyset$; $Z \leftarrow \emptyset$.
 - 2: $\mathcal{I}_{cc}(Z) \leftarrow \mathcal{C}$; $\mathcal{I}_{ca}(Z) \leftarrow \mathcal{A}$.
 - 3: FindRulesForItemset(Z).
-

Procedure 1 FindRulesForItemset(Z)

- 1: **for all** $I_t \in \mathcal{I}_{ca}(Z)$ **do**
 - 2: **if** $\text{Supp}(Z \cup \{I_t\}) \geq \text{Supp}_{min}$ **and** $O(I_t) > O(I_m)$ for all $I_m \in Z$ **then**
 - 3: $\mathcal{I}_{ca}(Z \cup \{I_t\}) \leftarrow \mathcal{I}_{gen}(\mathcal{E}_{ant}(Z))$; $\mathcal{I}_{cc}(Z \cup \{I_t\}) \leftarrow \mathcal{C}$.
 - 4: FindRulesForItemset($Z \cup \{I_t\}$).
 - 5: **for all** $I_t \in \mathcal{I}_{cc}(Z)$ **do**
 - 6: **if** $\text{Supp}(Z \cup \{I_t\}) \geq \max\{\text{Supp}_{min}, \text{Conf}_{min} \cdot \text{Supp}(X)\}$ **and** $O(I_t) > O(I_m)$ for all $I_m \in Z$ **then**
 - 7: add itemset $Z \cup \{I_t\}$ into \mathcal{R} .
 - 8: $\mathcal{I}_{ca}(Z \cup \{I_t\}) \leftarrow \emptyset$; $\mathcal{I}_{cc}(Z \cup \{I_t\}) \leftarrow \mathcal{I}_{gen}(\mathcal{E}_{con}(Z))$.
 - 9: FindRulesForItemset($Z \cup \{I_t\}$).
-

Similar to Apriori Algorithm, for any (antecedent or rule) itemset Z and any $Z' \in \mathcal{E}_{ant}(Z) \cup \mathcal{E}_{con}(Z)$, we let the *candidate consequent extension list* $\mathcal{I}_{cc}(Z')$ of Z' be the list $\mathcal{I}_{gen}(\mathcal{E}_{con}(Z))$ of all generating items of $\mathcal{E}_{con}(Z)$. We construct the list $\mathcal{I}_{ca}(Z')$ of *candidate antecedent extensions* in a similar manner, with the exception that for any rule itemset Z' , $\mathcal{I}_{ca}(Z')$ is always empty.

Next, we suppose that antecedent items are added into \mathcal{A} one at a time. Let $\mathcal{A}_k = \{I_{i_1}, I_{i_2}, \dots, I_{i_k}\}$ be the set of antecedent items in Step k , and let \mathcal{R}_k be the list of all good rules itemsets induced by \mathcal{A}_k . In Step $(k+1)$ a new antecedent item I_d is added, and therefore $\mathcal{A}_{k+1} = \mathcal{A}_k \cup \{I_d\}$.

We first examine each rule itemset $X \cup Y$ (corresponding to rule $X \Rightarrow Y$) in \mathcal{R}_k after I_d becomes an antecedent item. There are three cases:

- Type A.** $I_d \notin Y$: $X \cup Y$ is intact, as it corresponds to the same rule with support and confidence unchanged.
- Type B.** $\{I_d\} \subset Y$: $X \cup Y$ now corresponds to a different (but still good) rule $X \cup \{I_d\} \Rightarrow Y \setminus \{I_d\}$.
- Type C.** $\{I_d\} = Y$: $X \cup Y$ is no longer a good rule

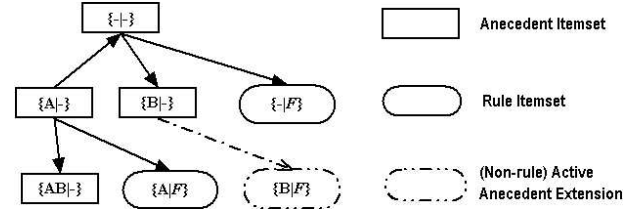


Figure 1: Active antecedent itemsets with respect to item F.

itemset as now the corresponding rule $X \cup \{I_d\} \Rightarrow \emptyset$ is no longer meaningful.

To construct \mathcal{R}_{k+1} from \mathcal{R}_k , we first need to delete all Type C rule itemsets from \mathcal{R}_k . Further, we need to compute all rules in the form of $X \cup \{I_d\} \Rightarrow Y$, where $X \subseteq \mathcal{A}_k$ and $Y \cap \mathcal{A}_{k+1} = \emptyset$. If the corresponding rule itemset $X \cup Y \cup \{I_d\}$ is not in \mathcal{R}_k (i.e., rule $X \Rightarrow Y \cup \{I_d\}$ is not a good rule discovered in Step k), it shall be added into \mathcal{R}_k . We call these rule itemsets *Type D itemsets*. The resulting set of rule itemsets is \mathcal{R}_{k+1} .

To better utilize mining information gained in previous steps, we maintain a *local rule itemset tree* \mathcal{T}_{rule} , which is a lexicographic tree (according to the lexicographic order defined in the previous section) and contains all frequent antecedent itemsets as well as all good rule itemsets. Naturally, there is a one-to-one mapping between nodes in \mathcal{T}_{rule} and nodes in $\mathcal{L}_{itemset}$. There are two types of nodes in \mathcal{T}_{rule} , *antecedent nodes* and *rule nodes*, corresponding to antecedent itemsets and rule itemsets respectively. For any $Z \in \mathcal{T}_{rule}$, we use $\mathcal{T}_{rule}(Z)$ to denote the subtree of \mathcal{T}_{rule} rooted at Z .

3 Terminologies and Data Structures

Support- and confidence-constrained antecedent itemsets Let \mathcal{X}_k be the set of frequent antecedent itemsets with respect to \mathcal{A}_k . Any $X \in \mathcal{X}_k$ is said to be *support-constrained* if $\text{Conf}_{min} \cdot \text{Supp}(X) \leq \text{Supp}_{min}$; otherwise, it is said to be *confidence-constrained*. It is easy to see that, if X is support-constrained, there will be no Type D rule itemset in \mathcal{R}_{k+1} that is a descendent of X . This is because in Step k a rule itemset Z associated with X can only be disqualified (from being a good rule) due to low support and hence it will remain a “bad rule” even after the introduction of a new antecedent item I_d .

Active antecedent extensions A confidence-constrained $X \in \mathcal{X}_k$ is said to be *active with respect to* a consequent item I_m if $X \cup \{I_m\}$ is frequent. For each I_m , we maintain an *active antecedent extension list* $\mathcal{X}_{act}(I_m)$ containing all $X \cup \{I_m\}$ such that X is active with respect to I_m .

As a running example throughout the paper, we assume that we have $\mathcal{I} = \{A, B, \dots, M, N\}$ (in this exact lexicographic order). After two steps, items A and B are specified as antecedent items, and the next item to be specified as an antecedent item is F (i.e., $I_d = F$). In the following discussion as well as in the figures, we

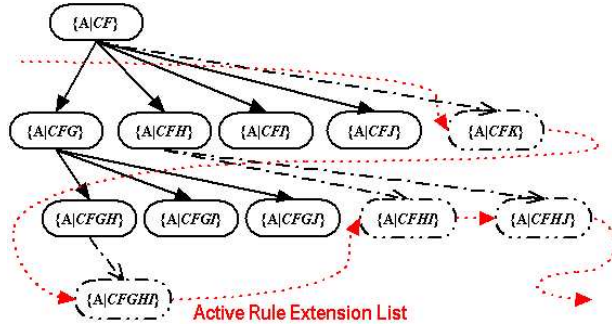


Figure 2: Recursive active rule extensions for rule itemset $\{A|CF\}$.

use $\{A_1 A_2 \dots A_s | C_1 C_2 \dots C_t\}$ to represent an itemset corresponding to rule $\{A_1, A_2, \dots, A_s\} \Rightarrow \{C_1, C_2, \dots, C_t\}$. We use “-” to denote that the antecedent (or consequent) part of an itemset is empty. Figure 1 shows that for consequent item F, there are three active antecedent extensions: itemsets $\{-|F\}$ and $\{A|F\}$ are good rule itemsets, while itemset $\{B|F\}$ does not have enough confidence to be a good rule itemset (but it still has a support no less than Supp_{min}).

It is important to note that there is no additional computation required to acquire such information, as for each frequent antecedent itemset X and each consequent item I_m , we need to check $\text{Supp}(X \cup \{I_m\})$ in order to determine whether $X \Rightarrow \{I_m\}$ is a good rule.

Active rule extensions A good rule itemset Z associated with a confidence-constrained antecedent itemset X is said to be *active with respect to* a consequent item I_m if it is *judged* that the $Z' = Z \cup \{I_m\}$ is frequent (with respect to Supp_{min}) and yet is not a good rule itemset. We call the list containing all such Z' the *active rule extension list of Z* , and denote it by $\mathcal{E}_{act}(Z)$. For example, rule itemset $\{A|CFH\}$ has two active rule extensions, $\{A|CFHI\}$ and $\{A|CFHJ\}$, as shown in Figure 2. If Z is an antecedent itemset or a rule itemset associated with a support-constrained antecedent, its active rule extension list is defined to be empty.

We shall emphasize here that the $\mathcal{E}_{act}(Z)$ may not contain all $Z \cup \{I_m\}$ that are frequent but not good. For example, since itemset $\{A|CFK\}$ is not a good rule itemset (see Figure 2), in previous steps we would not have attempted to extend $\{A|CFH\}$ (or any of the descendant rule itemsets of $\{A|CF\}$ other than $\{A|CFK\}$ itself) by item K, as K is even not in the candidate consequent extension list of $\{A|CFH\}$. Therefore, we have no prior information regarding the support of itemset $\{A|CFHK\}$; it may or may not be a frequent itemset.

However, it is easy to see that, if Z is a good rule item and yet $Z \cup \{I_m\}$ is a frequent but not a good rule itemset, either Z is active with respect to I_m , or there exists an ancestor Z'' of Z such that Z'' is active with respect to I_m . This property can be used to identify all potential Type D rule itemsets in the event of an $I_m \in Z$ switching from a consequent item to an antecedent item.

To facilitate finding these rule itemsets in an efficient manner, we recursively merge the active rule extension lists into a single list \mathcal{E}_{act} . For any good rule itemset Z , we define the *recursive active rule extension list* $\mathcal{E}_{act}^r(Z)$ of Z as the following: $\mathcal{E}_{act}^r(Z) = \mathcal{E}_{act}(Z) \cup (\bigcup_{k=1}^t \mathcal{E}_{act}^r(Z \cup \{I_{j_k}\}))$. Here $Z \cup \{I_{j_1}\}, Z \cup \{I_{j_2}\}, \dots, Z \cup \{I_{j_t}\}$ are the frequent extensions of Z such that $j_1 < j_2 < \dots < j_t$. Therefore, $\mathcal{E}_{act}^r(Z)$ contains all active rule extensions of Z as well as those of rule itemsets extending Z (by one or multiple consequent items). Figure 2 shows the recursive active rule extension list for rule itemset $\{A|CF\}$. \mathcal{E}_{act} is then defined to be $\mathcal{E}_{act}^r(\emptyset)$, which contains all active rule extensions.

4 Iterative Mining: The Algorithm

In this section, we describe an iterative mining algorithm that will update \mathcal{T}_{rule} every time a new antecedent item is specified. The goal is to manipulate \mathcal{T}_{rule} in such an efficient manner that the total computation time of iterative mining is close to the time required if all items in \mathcal{A} are known at once.

We first give the description of the main loop of our algorithm (see Algorithm 2).

Algorithm 2 FindRulesOnline(I_d)

```

1:  $\mathcal{A}_{k+1} \leftarrow \mathcal{A}_k \cup \{I_d\}$ ;  $\mathcal{R}_{k+1} \leftarrow \mathcal{R}_k$ ;  $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_k$ .
2: for all  $X' \in \mathcal{X}_{act}(I_d)$  do
3:    $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_{k+1} \cup \{X'\}$ .
4:    $X \leftarrow X' \setminus \{I_d\}$ ;  $\mathcal{I}_{cc}(X') \leftarrow \mathcal{I}_{gen}(\mathcal{E}_{con}(X) \cup \mathcal{E}_{act}(X))$ .
5:   if  $X'$  was already a good rule itemset in Step  $k$  then
6:     change  $X'$  to an antecedent itemset.
7:     ComputeSubTree( $X', I_d$ ).
8:   else
9:      $\mathcal{I}_{ca}(X') \leftarrow \emptyset$ .
10:    FindRulesForItemset( $X'$ ).

```

Notice that, in Step $(k+1)$ we only need to search for good rule itemsets associated with an antecedent $X' = X \cup \{I_d\}$ for each $X \in \mathcal{X}_k$. If X is supported-constrained, there will be no Type D rule itemsets underneath X , meaning that no new good rule itemsets associated with X' will be found. Therefore, we can skip the subtree $\mathcal{T}_{rule}(X')$ of \mathcal{T}_{rule} rooted at X' entirely.

Moreover, for a confidence-constrained X , we need to search for good rule itemsets associated with $X' = X \cup \{I_d\}$ only if $\text{Supp}(X') \geq \text{Supp}_{min}$, meaning that X was active with respect to I_d in the previous step, when I_d was a consequent item. Therefore, we just need to go through $\mathcal{X}_{act}(I_d)$, and add each $X' \in \mathcal{X}_{act}(I_d)$ into \mathcal{X}_{k+1} , as indicated in Line 3 of Algorithm 2.

Among these new antecedent itemsets, some were not good rule itemsets in the previous step. For each of these antecedent itemsets X' , to build the subtree $\mathcal{T}_{rule}(X')$ rooted at X' , we can only use Procedure 1 (as indicated in Line 10 of Algorithm 2) as there is no prior information to utilize. The remaining new antecedent itemsets are the Type C ones; they were converted from good rule itemsets found in the previous step. For each such X' , it would be wasteful to compute

To illustrate how Procedure 2 works, we use itemset $X' = \{\mathbf{AF}|\cdot\}$ (which was $\{\mathbf{AF}\}$ in the previous step) as an example. The new antecedent item F divides all consequent items into two groups; Group 1 contains those ordered before F in the previous step (*i.e.*, C , D , and E), and Group 2 contains those ordered after F (*i.e.*, G , H , \dots , N). In turn, the subtree $\mathcal{T}_{rule}(X')$ is divided into two partial trees: the left partial tree, which contains all the immediate children (as well as their descendants) that extend $\{\mathbf{AF}|\cdot\}$ by items in Group 1, and the right partial tree, which contains all immediate children (as well as their descendants) that extend $\{\mathbf{AF}|\cdot\}$ by items in Group 2. In the following, we show how to compute the two partial trees.

```

1: ExpandPartialSubTree( $Z, \mathcal{E}_{act}^r(Z), \emptyset$ ).
2: if  $Z$  is an antecedent node then
3:    $i_{min} \leftarrow 0$ .
4: else
5:    $I_m \leftarrow I_{gen}(Z); i_{min} \leftarrow m$ .
6: for all  $I_t \in \mathcal{C}_{k+1}$  such that  $i_{min} < t < d$  do
7:   if  $\text{Supp}(Z \cup \{I_t\}) \geq \max\{\text{Supp}_{min}, \text{Conf}_{min} \cdot \text{Supp}(X)\}$ 
8:     then
9:       if  $Z \cup \{I_t\}$  was already a good rule itemset in Step  $k$ 
10:        then
11:          change the parent of  $Z \cup \{I_t\}$  from  $Z \setminus \{I_d\} \cup \{I_t\}$  to  $Z$ .
12:        ComputeSubTree( $Z \cup \{I_t\}, I_d$ ).
13:   else
14:     add  $Z \cup \{I_t\}$  as a new child of  $Z$ .
15:    $\mathcal{I}_{cc}(Z \cup \{I_t\}) \leftarrow \mathcal{I}_{gen}(\mathcal{E}_{con}(Z)); \mathcal{I}_{ca}(Z \cup \{I_t\}) \leftarrow \emptyset$ .
16:   FindRulesForItemset( $Z \cup \{I_t\}$ ).

```

We may find Type D children of $\{\mathbf{AF|}-\}$, *e.g.*, $\{\mathbf{AF|E}\}$. Before **F** becomes an antecedent item, $\{\mathbf{AF|E}\}$ was $\{\mathbf{A|EF}\}$ (see Figure 3), who did not satisfy the minimum confidence constraint. We may also find Type B children of $\{\mathbf{AF|}-\}$, *e.g.*, $\{\mathbf{AF|C}\}$, which was $\{\mathbf{A|CF}\}$ in the previous step. However, at that time $\{\mathbf{A|CF}\}$ was a child of $\{\mathbf{A|C}\}$. Now itemset $\{\mathbf{A|CF}\}$ becomes $\{\mathbf{AF|C}\}$ and thus should be changed to become a child of $\{\mathbf{AF|}-\}$ to maintain the lexicographic order, as indicated in Line 9 of Procedure 2.

is a Type B one, however, we recursively call Procedure 2, as indicated in Line 10 of Procedure 2, so that we can “reuse” the descendants of Z' already in \mathcal{T}_{rule} .

```

1: if  $\mathcal{I}_{cc}(Z) = \emptyset$  then
2:   if  $L_{act} = \emptyset$  then
3:     return.
4:    $Z' \leftarrow$  the first active rule extension in  $L_{act}$ .
5:    $Z'' \leftarrow$  the parent node of  $Z'$ .
6:   if  $Z'' \neq Z$  then
7:      $Z \leftarrow Z''; \mathcal{I}_{cc}(Z) \leftarrow \emptyset$ .
8:    $L'_{act} \leftarrow \emptyset$ .
9:   remove from  $L_{act}$  all active rule extensions that extend  $Z$ 
   and add them into  $L'_{act}$ .
10:   $L_{can} \leftarrow \mathcal{I}_{gen}(L'_{act}) \cup \mathcal{I}_{cc}(Z); L_{new} \leftarrow \emptyset$ .
11:  for all  $I_t \in L_{can}$  such that  $t > d$  do
12:    if  $\text{Conf}(Z \cup \{I_t\}) \geq \text{Conf}_{min}$  (after  $I_d$  becomes an
    antecedent item) then
13:       $\mathcal{E}_{con}(Z) \leftarrow \mathcal{E}_{con}(Z) \cup \{Z \cup \{I_t\}\}$ .
14:       $L_{new} \leftarrow L_{new} \cup \{Z \cup \{I_t\}\}$ .
15:  for all  $Z' \in \mathcal{E}_{con}(Z)$  ordered by generating item do
16:    if  $Z' \in L_{new}$  then
17:       $\mathcal{I}_{cc}(Z') \leftarrow \mathcal{I}_{gen}(\mathcal{E}_{con}(Z)); \mathcal{I}_{ca}(Z') \leftarrow \emptyset$ .
18:      FindRulesForItemset( $Z'$ ).
19:  else
20:     $\mathcal{I}_{cc}(Z') \leftarrow \mathcal{I}_{gen}(L_{new})$ .
21:    ExpandPartialSubTree( $Z', L_{act}$ ).

```

554

check each of them to see if it now satisfies the minimum confidence constraint.

Further, for any Type B itemset Z' in the subtree $\mathcal{T}_{rule}(Z)$, we should try to extend Z' if either Z' itself has active rule extension(s), or Z' has new “younger siblings.” For example, if $Z' = \{AF|CH\}$, we should try to extend Z' by: i) items I and J, as previously $\{A|CFH\}$ was active with respect to I and J; and ii) item K, as it is the generating item of $\{AF|CK\}$, a younger sibling of Z' . (Here we assume that the confidence of $\{AF|CK\}$, originally $\{A|CFK\}$, grows enough to make it a good rule itemset.) The trick is, however, that we should avoid revisiting each $Z' \in \mathcal{T}_{rule}(Z)$ to see if it is “expandable,” which could be costly if $\mathcal{T}_{rule}(Z)$ contains a large number of nodes (in case we use relatively relaxed thresholds for support and confidence).

In Procedure 3, we add all immediate children of Z in a list L_{new} (see Line 14). We use the generating items of all itemsets in L_{new} as the candidate consequent extensions $\mathcal{I}_{cc}(Z')$ for each existing child Z' of Z , and recursively call Procedure 3 to revisit Z' so that we can expand the subtree rooted at Z' . Therefore, if a revisited Z' is found to have no candidate consequent extensions, we do not need to revisit any of its existing children. We can skip $\mathcal{T}_{rule}(Z')$ entirely if Z' does not have any recursive active rule extensions (*i.e.*, $\mathcal{E}_{act}^r(Z') = \emptyset$); or, we go directly to the first recursive active rule extension of Z' (see Lines 1-7 of Procedure 3). For any Type D itemset Z' found, we invoke Procedure 1 to compute the subtree $\mathcal{T}_{rule}(Z')$, as indicated in Line 18 of Procedure 3.

5 Experimental Results

To conduct experiments, we generated using the method first proposed by Agrawal and Srikant [3] a synthetic data set T20.I8.D100K containing 100,000 transactions. We generated using a minimum support of 0.15% an adjacency lattice with 968,820 nodes. We compare Algorithm 2 and Algorithm 1 using a number of combinations of support and confidence thresholds. For each combination, we ran 30 different test cases, each with a distinct set \mathcal{A} of 300 antecedent items (out of a total of 1000 items) randomly picked, and recorded the running time averaged over the 30 test cases.

Figure 4 illustrates the “competitive ratio” of Algorithm 2, which is defined to be the ratio of its average running time to that of Algorithm 1. Here for Algorithm 1 the running time is defined to be the time for answering a *single* query with all 300 antecedent items known in advance. For Algorithm 2, the running time is the total time for finishing 300 queries, each of which was performed after one antecedent item was added. Figure 4 shows that the performance of our online algorithm is very close to that of the offline algorithm. Further, for Algorithm 2, answering a single query is almost instantaneous. This demonstrates the advantage of pre-

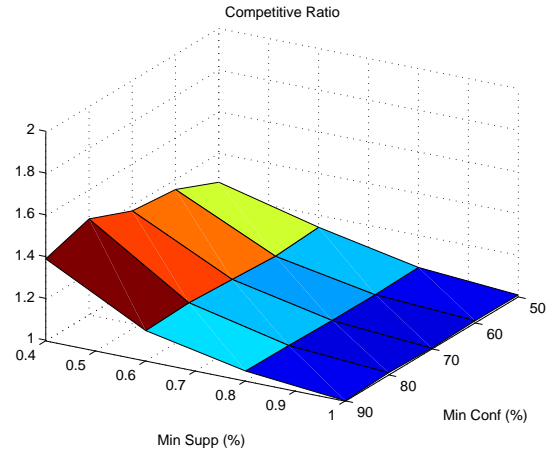


Figure 4: Competitive ratio.

computing an adjacency lattice for mining association rules with constrained antecedents.

6 Conclusion and Future Works

In this paper we provided a novel approach for online mining of association rules with constrained antecedents and in multiple steps. By carefully maintaining a tree of rules as well as some auxiliary data structures, our algorithm performs nearly as well as the offline algorithm, as shown by the preliminary experimental results. For future works, we plan to study how this algorithm can be effectively combined with other iterative mining algorithms to answer queries with both tightened/relaxed support and confidence constraints, as well as increased/decreased antecedent set. Further, we are interested in studying an alternative approach that does not require the pre-computation of an adjacency lattice.

References

- [1] C. C. Aggarwal, Z. Sun, and P. S. Yu, *Fast algorithms for online generation of profile association rules*, TKDE, 14 (2002), pp. 1017–1028.
- [2] C. C. Aggarwal and P. S. Yu, *A new approach to online generation of association rules*, TKDE, 13 (2001), pp. 527–540.
- [3] R. Agrawal and R. Srikant, *Fast algorithms for mining association rules*, VLDB'94, pp. 487–499.
- [4] R. Bayardo, R. A. Jr., and D. Gunopulos, *Constraint-based rule mining in large, dense databases*, ICDE'99, pp. 188–197.
- [5] G. Cong and B. Liu, *Speed-up iterative frequent itemset mining with constraint changes*, ICDE'02, pp. 107–114.
- [6] L. V. S. Lakshmanan, R. T. Ng, J. Han, and A. Pang, *Optimization of constrained frequent set queries with 2-variable constraints*, SIGMOD'99, pp. 157–168.
- [7] J. Pei, J. Han, and L. V. S. Lakshmanan, *Mining frequent item sets with convertible constraints*, ICDE'01, pp. 433–442.
- [8] R. Srikant, Q. Vu, and R. Agrawal, *Mining association rules with item constraints*, KDD'97, pp. 67–73.
- [9] S. Thomas and S. Chakravarthy, *Incremental mining of constrained associations*, HiPC'00, pp. 547–558.

Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach

Al Mamunur Rashid

George Karypis

John Riedl*

Abstract

Recommender systems have been shown to help users find items of interest from among a large pool of potentially interesting items. *Influence* is a measure of the effect of a user on the recommendations from a recommender system. Influence is a powerful tool for understanding the workings of a recommender system. Experiments show that users have widely varying degrees of influence in ratings-based recommender systems. Proposed influence measures have been algorithm-specific, which limits their generality and comparability. We propose an algorithm-independent definition of influence that can be applied to any ratings-based recommender system. We show experimentally that influence may be effectively estimated using simple, inexpensive metrics.

1 Introduction

Sociologists have long tried to characterize the influence of a person in a social network of many people [1]. Identifying the influential people can bring twin advantages to those who study group dynamics: (1) The influential people can be directly studied, yielding insight since their choices may be predictive of group choices; or (2) The influential people may be influenced to change the behavior of the group. Many social networks are formed and maintained through informal, qualitative, and unobserved interactions. Capturing data about these interactions is difficult, and the act of capturing those data may change the social interactions themselves.

Collaborative Filtering (CF) recommender systems [2, 3, 4] base their decisions on the opinions of users. In contrast to other social networks, recommender systems capture interactions that are *formal*, *quantitative*, and *observed*. The social network can be analyzed directly through data already captured in the computer system.

Past research has demonstrated that analyzing the social network can provide leverage in influencing the group [5]. The analysis performed in these studies is based on a deep investigation of the characteristics of one particular recommender algorithm, the well-known user-user nearest neighbor algorithm [2]. Careful analysis of this type has many advantages, but one key disadvantage: it is tied closely to the details of the algorithm. In principle, similar techniques could

be applied to other algorithms, but doing so would be laborious, and the resulting influence measure only applies to algorithms that work precisely according to the details of the analysis. Since many commercial operators tweak the operation of the recommender in many ways to fit the needs of their business, this analysis may not apply in practice. Further, the resulting measures of influence would be unlikely to be comparable between different algorithms, since they have been produced through very different techniques.

A key goal of the present research is to identify a measure of influence for recommender systems that is applicable to any ratings-based recommender system, independent of the particulars of the algorithm. Such a measure would allow for consistent, black-box analysis of influence.

2 Related Work

2.1 Recommender Systems. Resnick, et al. [2] introduced an automatic collaborative filtering algorithm based on a k-nearest neighbors (kNN) algorithm among users; this algorithm is now called *user-user* CF. The *user-user* algorithm we use in this paper is a version of the original kNN algorithm, tuned to achieve best known performance. Sarwar et al. [4] proposed an alternative kNN CF algorithm based on similarity among items. This variant is often called *item-item* CF. Breese et al. [3] have divided a number of CF algorithms into two classes: memory-based algorithms and model-based algorithms. Over the years many other algorithms were proposed including ones based on SVD, clustering, Bayesian Networks [3]. We focus on the *user-user* and *item-item* algorithms in this paper because they are the most common in existing systems.

2.2 Social Networks and Influence. A Social network is a form of graph delineating relationships and interactions among individuals. Finding the important nodes in such graphs has been an object of interest to sociologists for a long time. One proposed measure for importance is *centrality* [1]. Two examples of “centrality” measures are “degree centrality”, which treats high degree nodes as important, and “distance central-

*Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN-55455, {arashid, karypis, riedl}@cs.umn.edu

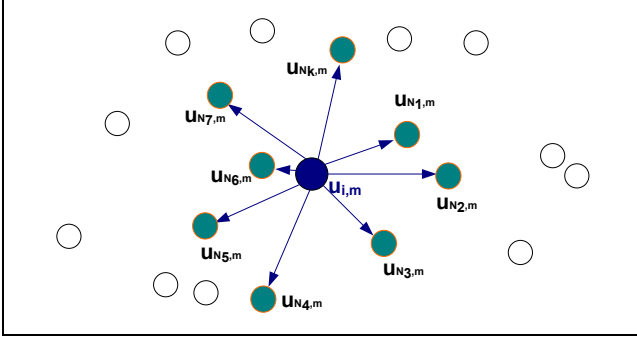


Figure 1: Showing the notion of *in-links* for the k closest neighbors of u_i . Here, prediction is being computed for the (user, item) pair, (u_i, m) .

ity”, which treats nodes with short paths to many other nodes as important [1]. Kleinberg’s HITS [6], and Brin and Page’s PageRank [7] algorithms for ordering nodes in a graph of web are based on social network principles.

Domingos et al. [5] have studied the problem of choosing influential users for marketers who wish to attract attention to their products. They show that selecting the right set of users for a marketing campaign can make a big difference. Kempe et al. [8] focus on a collection of models widely studied in social networks, as well as the models in [5], under the categories: Linear Threshold Models, and Independent Cascade Models.

Our research also investigates influence in social networks. Like Domingos et al. we focus on networks in recommender systems. We extend their research to general measures of influence that are independent of the particular recommender algorithm being used.

3 Defining Influential Users in CF Systems

We first discuss the data used in this project, then analyze a popular CF algorithm to understand a possible formation process of influential users, and then try different ways to set the definition.

3.1 The Data. We have used a publicly available dataset from *www.grouplens.org*. The dataset is a fraction of the usage data drawn from MovieLens (*www.movielens.org*), a CF-based online movie recommendation system. It contains 6,040 users, 3,593 movies, and about one million ratings on a 5-star scale. Each user has rated at least 20 movies in the dataset. We have partitioned this data into training and test sets by a random 80%/20% split.

3.2 The User-User Algorithm. The most widely cited and arguably the most commonly used CF algo-

rithm in research is a kNN-based algorithm. In this scheme the users’ preference data is represented in a $n \times m$ user-item matrix for a system with n users and m items, where the (i, j) -th entry of this matrix stands for the user u_i ’s rating on item j , or null, depending on whether the user u_i has rated the item j , or not, respectively. The *user-user* algorithm can be thought of working in two stages. In the first stage, similarities between every pair of users are computed and are stored as a model. Although many different formulations are possible for similarity weight calculations, the *GroupLens* [2] proposed mechanism is the Pearson correlation coefficient. Accordingly, the similarity weight between two users, u_i , and u_j is measured by equation 3.1:

$$(3.1) W_{ij} = \frac{\sum_{k \in I} (R_{ik} - \bar{R}_i)(R_{jk} - \bar{R}_j)}{\sqrt{\sum_{k \in I} (R_{ik} - \bar{R}_i)^2 \sum_{k \in I} (R_{jk} - \bar{R}_j)^2}}$$

where I is the set of items rated by both of the users, R_{ik} is user u_i ’s rating on item k , and \bar{R}_i is the average rating of u_i . Using this similarity metric, the next step, prediction generation, is carried out as follows. Prediction on item a for user u_i is computed by picking k nearest users who have also rated item a , and by applying a weighted average of deviations from the selected users’ means:

$$(3.2) P_{ia} = \bar{R}_i + \frac{\sum_{u=1}^k (R_{ua} - \bar{R}_u) W_{iu}}{\sum_{u=1}^k W_{iu}}$$

3.3 Some Plausible Influence Metrics Based on Prior Work. We can now propose several influence metrics. One type of metric is motivated by *targeted marketing*. Another type of metric exploits connections between users based on similarity.

3.3.1 Expected Lift in Profit: Network Values. This approach, as outlined in [5], is based on the goal of *targeted marketing*. In this scheme, users who can yield the most *expected lift in profit* by making a cascading adoption of a product happen, are considered as influential users. Domingos et al. [5] have applied this idea on a recommendation system dataset based on the *user-user* CF algorithm described in the last section.

The probabilistic model in [5] is based on the *Markov Random Fields*, which requires the neighbors be symmetric; i.e., two users are neighbors to each other if one of them is a neighbor to the other. The authors mention that in a kNN-based CF system, this might not hold. Again, *ELP_Network_Value* is tied to a particular product; more specifically, it is specific to a set of features of the product being marketed. Translating this issue into the RS domain, *ELP_Network_Values*

are specific to particular genre vectors. Thus a user's *ELP_Network_Value* will differ for movies with different genre vectors.

3.3.2 Network Structure: Similarity Links. By closely observing the process of neighbor-selection, we notice some network structure that could facilitate in forming a definition for influential users. Figure 1 demonstrates a situation where the system is computing a prediction on item m for user u_i . In order to do so, it selects top k neighbors who also have rated the item m . Now we can imagine directed edges from u_i towards each of the k neighbors.

Equations 3.3 and 3.4 show the updated *authority* and *hub* equations. In order to consider the fact that all the links may not of same weight, we have incorporated a weight term similar to [9] to the basic HITS [6] equations. Here the conditional probability, $p(i|j)$ refers to the degree of user u_j 's presence indicating user u_i 's presence.

$$(3.3) \quad a(i) = \sum_{j \rightarrow i} p(i|j)h(j)W_{ij}$$

$$(3.4) \quad h(i) = \sum_{i \rightarrow j} p(j|i)a(j)W_{ij}$$

We can use this modified *authority* to represent influence.

The drawback of this scheme of influence, however, is algorithm dependence: the network structure captured here is very much algorithm-specific; and, for other algorithms, the structure might not be as apparent. In order to derive a definition that is generic enough, yet simple, we use the *Hide-one-User* approach discussed next. The fundamental concept with this approach is figuring out which user causes the largest cumulative change of prediction in the system.

4 Algorithm-Independent Influence

These metrics define influence as the amount of effect a user has over others via the predictions they receive. One way to observe this effect is to exclude a user and measure the net changes in predictions caused by the removal.

The idea: Let U be the set of available users in the system, M_U be the model built with the preference data of this set of users. We call NPD_{u_i} (*Number of Prediction-Differences*) as the number of times the following expression holds true:

$$|P_{ja}(M_U) - P_{ja}(M_{U-\{u_i\}})| \geq \delta, \forall j \neq i$$

Here, $P_{ja}(M_U)$ is the prediction on item a for the user u_j using the model M_U , δ is a threshold that can be

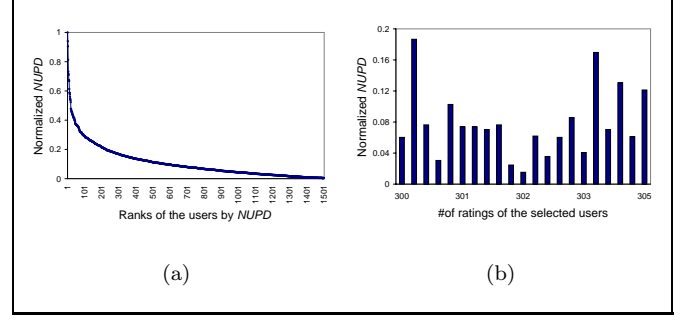


Figure 2: (a) Distribution of influence. (b) *NUPD* values of a group of 20 users who have rated almost the same number of items.

tied with the smallest prediction change perceivable to the users via the available user interface. As an example, smallest prediction change a MovieLens user would notice is 0.5 or a half-star. In essence, the expression for NPD_{u_i} says how many times the predictions would change beyond some threshold if we build the model without the user u_i . NPD_{u_i} is the influence level of user u_i . There is a problem with NPD_{u_i} : if the group of users, who get affected by u_i 's removal, need predictions on many items, u_i could exhibit possessing a large NPD_{u_i} . To overcome this problem, we propose another version of this definition and call it $NUPD_{u_i}$. $NUPD_{u_i}$ counts the number of unique users whose predictions' got changed by at least the threshold amount as we keep the i -th user out during model-building.

As is evident from the definition of $NUPD_{u_i}$, it is equally applicable to any CF algorithm, provided that we have the historical data to compute it from.

Notice that a straightforward computation of *NUPDs* can become very expensive; if we are to compute *NUPD* online or in a regular basis, we need to find a cheaper way. Section 6 details such an endeavor.

4.1 The Nature of Influence. Figure 2(a) shows normalized *NUPD* values of the top 1500 influential users and highlights the fact that only a handful of the users possess high influence. This is true for both *authority* and *NUPD* measures. The shapes demonstrate the power-law or a Zipf-like distribution. A similar shape is reported in [5] for *ELP_Network_Values*.

Note that the correlation between *authority* and *NUPD* is 0.96.

5 Building a Predictive Model

As stated before, *NUPD* suffers from a drawback: the computation is quite time consuming. In order to circumvent this limitation, we seek a predictive model that can provide users' influence levels on the fly while

maintaining good accuracy.

Although the correlation coefficient between *NUPD* and the number of ratings is 0.75, figure 2(b) shows that the amount of influence can vary widely between users who have rated approximately the same number of movies. This suggests we look for a model that can account for factors not captured by the number of ratings.

In the following section we compile a list of qualitative factors that seem to affect influence levels.

5.1 Qualitative Factors

Number of ratings: This is the most immediate factor one would possibly come along with. If a user rates more items, she has a greater chance to be close to many users. Moreover, such a user can be useful to many users who are looking for recommendations for a wide variety of items.

Degree of agreement with others: This measure attempts to estimate on average how much a user agrees to the average opinion of others: $1/k \sum_{a=1}^k |R_{ia} - \bar{R}_a|$. This expression computes the extent to which the user u_i 's ratings are swayed from each of the corresponding item's average rating.

Rarity of the rated items: This is a measure very similar to that of the Inverse Document Frequency (IDF), which penalizes frequent items, as they are considered to have little discriminating power: $1/k \sum_{j \in I_{u_i}} 1/freq(j)$; where, I_{u_i} is the set of items that user u_i has rated.

Standard deviation in one's rating: This amounts to the degree a user's ratings deviate from her rating-average. The implication is that a higher standard deviation contributes a greater value through the term, $(R_{ik} - \bar{R}_i)$ in equation 3.1.

Degree of similarity with top neighbors: This is the average similarity weight of the top k neighbors of a user u_i : $1/k \sum_{j=1}^k W_{ij}$. This factor can be associated with two opposing implications: users having higher values from this expression might be able to exert more effect to be influential; whereas, a user might be easier to replace if she is very similar to a number of other users.

Aggregated popularity of the rated items: If the sum of the popularities of the rated items is high enough, the user has a greater chance to have overlapped items with many users.

*Aggregated MoviePopularity*Entropy:* Entropy of a movie simply indicates the dispersion of the ratings it received. Multiplying this with the popularity of the movie gives a measure that tries to balance between popularity and variance.

5.2 The Regression Model

We chose to use SVM Regression (SVR) for our modeling. SVMs follow the *Structural Risk Minimization Principle* which seeks to minimize an upper bound on the generalization error rather than the principle used in most of the learning machines: *Empirical Risk Minimization Principle*—minimizing the training error. Hence, SVMs have been showing better generalization in many results. Although most of the practical usages for SVMs used to be in classification problems, SVMs have been extended to solve non-linear regression problems, mostly because of the introduction of the ε -insensitive loss function [11]; and the resulting regression method called ε -SVR.

We have tried various kernel functions to perform the non-linear mapping from the input space to the feature space. However, the radial basis function (RBF) produced the best regression result. In order to select the values of the parameters, C and ε , a cross-validation approach was carried out.

We have randomly selected 2416 users (40% of the total) and partitioned them into training and test sets by a 8:2 split. libsvm[10] was used to generate regression models using the following: the seven factors outlined before as predictors (independent variables), an RBF kernel, ε -SVR, and the parameters, C and ε . The model gave a squared correlation coefficient of 0.94. Figure 3 shows the prediction performance by plotting predicted *NUPDs* against the corresponding actual *NUPDs* taken from the test-set. A five-fold cross validation was carried out to ensure the results' validity. Table 1 has the regression results as well as a few statistics of the actual *NUPD* values in the test set, averaged over the five folds.

6 Influence in an Item-based Algorithm

We now turn to how the influence picture looks when using another prediction algorithm in order to see how algorithm-dependent our measures are.

The item-item Algorithm. The kNN based CF algorithm proposed in [4] is different in many ways than the user-based algorithm we have addressed so far. The algorithm first builds the model by computing item-item similarities. [4] proposed adjusted cosine measure for estimating the similarity between two items i , and j :

$$s_{i,j} = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_u)^2 \sum_{u \in U} (R_{u,j} - \bar{R}_u)^2}}$$

Prediction for the (user, item) pair, (u, i) is computed as: $\sum_{all_similar_items, N} (s_{i,N} * R_{u,N}) / \sum (|s_{i,N}|)$.

We could not employ *authority* on this algorithm, as it is not quite straightforward to establish di-

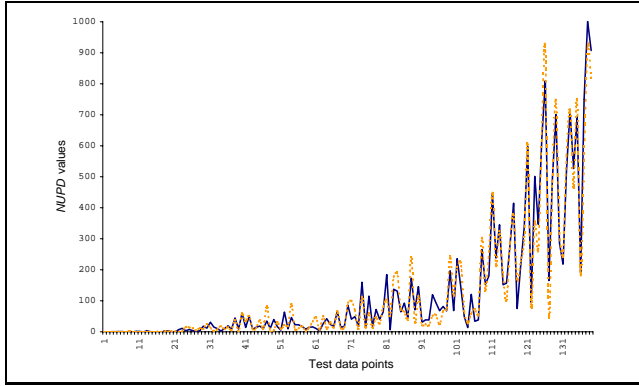


Figure 3: Performance of SVM regression for *NUPD* on *user-user* algorithm. The dotted line shows the actual values; whereas, the continuous line represents the predicted values.

rect edges between users. We could not compute *ELP_Network_Values* on this algorithm either, since *ELP_Network_Values* involve the notion of how *neighbors* affect a user, and corresponding probability computations based on this. However, applying *NUPD* by *Hide-one-User* method was easy. We have estimated *NUPDs* for the same set of users we have selected for the user-based approach. Modeling with ϵ -SVR gave a very good performance: squared correlation coefficient was 0.989.

7 Conclusion

In this paper, we have continued the investigation into influence in recommenders begun in [5]. We have shown that how many opinions a user expresses is an important component of influence, but not the whole story. We have defined several plausible influence metrics and shown that in general, they correlate strongly.

We believe our proposed metric, *NUPD*, is explainable both to researchers and operators of recommender systems. *NUPD* is also algorithm independent—it applies to any recommender system algorithm that makes predictions. *NUPD* is computationally inefficient. However, we have demonstrated how to build dataset- and algorithm-specific regression models that allow for the rapid, accurate estimation of a user’s influence.

Much remains to be done. Research is needed to understand how the *role* of influence changes it. For instance, when influence is used to help retailers sell products it may have very different characteristics than when it is used to encourage community members to contribute opinions. Another rich area of research is in interfaces for communicating influence to community

members. The interface is likely to impact both the interpretation of influence and its effectiveness in changing behavior.

References

- [1] S. Wasserman, K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, (1994).
- [2] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, *GroupLens: An open architecture for collaborative filtering of netnews*, in Proceedings of CSCW 1994, ACM SIG Computer Supported Cooperative Work, 1994.
- [3] J. S. Breese, D. Heckerman and C. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, in Proceedings of the Fourteenth Annual Conference on Uncertainty in AI, July 1998.
- [4] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, *Item-based collaborative filtering recommendation algorithms*, in Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong, May 2001.
- [5] P. Domingos and M. Richardson, *Mining the Network Value of Customers*, Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 2001. ACM Press, pp. 57–66.
- [6] L. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM, 46, 1999.
- [7] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: Bringing order to the web*, Technical Report, Stanford University, Stanford, CA. 1998.
- [8] D. Kempe, J. Kleinberg, and Tardos, *Maximizing the spread of influence through a social network*, in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington DC, 2003, pp. 137–146.
- [9] K. Wang, and M. Y. T. Su, *Item Selection by “Hub-Authority” Profit Ranking*, in SIGKDD ’02, Canada.
- [10] C. C. Chang, and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001.
- [11] V. N. Vapnik, *The Nature of Statistical Learning Theory*, New York, Springer-Verlag, 1995.

Table 1: Regression results on both CF algorithms

		<i>User-User</i>	<i>Item-Item</i>
Regression Performance	MAE	15.26	30.6
	Sq. corr. coeff.	0.94	0.99
	MSE	1036	2252.6
NUPD Statistics in Test Set	Avg.	81.57	405.6
	Min	0	0
	Max	980	2487
	StdDev	123.25	454.6

Mining Unconnected Patterns in Workflows

Gianluigi Greco*

Antonella Guzzo†

Giuseppe Manco†

Domenico Sacca†

Abstract

This paper investigates the problem of mining unconnected patterns in workflows and presents for its solution two algorithms, both adapting the Apriori approach to the graphical structure of workflows. The first one is a straightforward extension of the level-wise style of Apriori whereas the second one introduces sophisticated graphical analysis of the frequencies of workflow instances. The experiments show that graphical analysis improves the performance of pattern mining by dramatically pruning the search space of candidate patterns.

1 Introduction

A workflow is a partial or total automation of a business process, in which a collection of *activities* must be executed by humans or machines, according to certain procedural rules. There is a growing body of proposals aiming at enhancing the technology of Workflow Management Systems (WfMS) with advanced mechanisms for monitoring and diagnosing workflow executions. In this perspective, data mining techniques have proved to be very effective [3, 2, 7] in helping the administrator: Indeed, they can be exploited to look at all the previous instantiations (collected into *log* files in any commercial system), in order to extract unexpected and useful knowledge about the process, and in order to take the appropriate decisions in the executions of further coming instances. In [5], we show that both a workflow and its instances of execution have a natural representation as a directed graph, in which nodes represent activities, and edges the relationships between such activities. Hence, we concentrate on the problem of discovering frequent connected patterns of execution (short: FCPD), i.e., frequent connected subgraphs.

In this paper, we extend the approach in [5] and study the problem of discovering correlations among general patterns of execution in a workflow. In particular, we focus our attention on unconnected patterns, which are arbitrary subsets of connected patterns ex-

hibiting no explicit dependency relationship. Thus, we assume that a set \mathcal{P} of frequent \mathcal{F} -patterns is given and we are interested in discovering whether any of the subsets of \mathcal{P} is frequent as well. This problem, called *frequent unconnected patterns* discovery (short: FUPD), occurs very often in practical scenarios and is crucial for the identification of the critical subprocesses that lead with high probability to (un)desired final configurations.

It is worth noting that FUPD has a trivial solution consisting in the application of a level-wise algorithm (in the *a-priori* style) which combines all the unconnected patterns in \mathcal{P} and then checks for their frequency. However, this approach would not benefit from the peculiarities of the workflow graph that can be profitably used for pruning the search space. We show how the structure of the workflow together with some elementary information such as the frequency of occurrences of elementary activities suffices for pruning the search space and for deriving an efficient and practically fast algorithm.

2 Workflow Model and Problem Formulation

A *workflow schema* \mathcal{WS} is a tuple $\langle A, E, a_0, A_F \rangle$, where A is a finite set of *activities*, $E \subseteq (A - A_F) \times (A - \{a_0\})$ is an acyclic relation of precedences among activities, $a_0 \in A$ is the starting activity, and $A_F \subseteq A$ is the set of final activities. Moreover, A is partitioned into $A_{in}^\vee \cup A_{in}^\wedge \cup A_{out}^\vee \cup A_{out}^\wedge \cup A_{out}^\otimes$. An activity in A_{in}^\wedge (an *and-join* node) acts as synchronizer, for it can be executed only after all its predecessors are completed, whereas an activity in A_{in}^\vee (an *or-join* node) can start as soon as at least one of its predecessors has been completed. Moreover, once finished, an activity $a \in A_{out}^\wedge$ (*and-fork*) activates all its outgoing activities, $a \in A_{out}^\vee$ (*or-fork*) activates some of the outgoing activities, while $a \in A_{out}^\otimes$ (*xor-fork*) activates exactly one outgoing activity. The tuple $\langle A, E \rangle$ is referred to as the *control graph* of \mathcal{WS} .

A workflow schema has a quite natural graphical representation, by means of a directed acyclic graph. Figure 1 represents an example schema \mathcal{WS} where $A_{in}^\wedge = \{d, c, b, f, e, n, q\}$ and $A_{in}^\vee = \{a, g, l, i, h, m, o, p\}$, while $A_{out}^\otimes = \{a, h\}$, $A_{out}^\wedge = \{l, i, g, e, f, m, n, o, q\}$, $A_{out}^\vee = \{b, c, d\}$, and $A_F = \{p\}$.

The enactment of a workflow gives rise to an instance, i.e., to a proper subgraph of the schema which

*Department of Mathematics, University of Calabria, Via Bucci 30B, I87036, Rende, Italy. Email ggreco@mat.unical.it.

†ICAR-CNR, National Research Council, Via Bucci 41C, I87036, Rende, Italy. Email [{guzzo,manco,sacca}@icar.cnr.it">guzzo,manco,sacca}@icar.cnr.it](mailto).

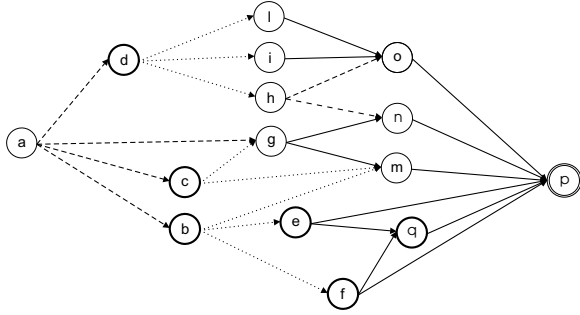
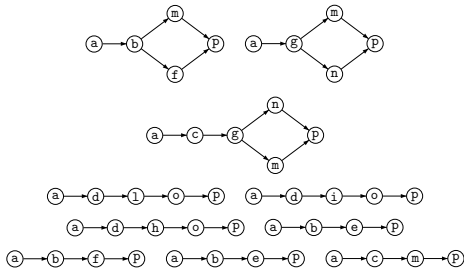


Figure 1: An example of workflow schema.

is derived satisfying the constraints imposed by the instances included. Formally, let \mathcal{WS} be a workflow schema. Any connected subgraph $I = \langle A_I, E_I \rangle$ of the control flow graph is an *instance* of \mathcal{WS} (denoted as $\mathcal{WS} \models I$) if the following conditions hold:

- (i) $a_0 \in A_I$, and $A_I \cap A_F \neq \emptyset$;
- (ii) for each $a \in A_I$, $|\{b \mid (b, a) \in E_I\}| > 0$;
- (iii) for each $a \in A_I \cap A_{in}^\wedge$, $\{b \mid (b, a) \in E\} \subseteq A_I$;
- (iv) for each $a \in A_I \cap A_{out}^\wedge$, $\{b \in A_{in}^\vee \mid (a, b) \in E\} \subseteq A_I$;
- (v) for each $a \in A_I \cap A_{out}^\otimes$, $|\{b \mid (a, b) \in E_I\}| \leq 1$ and $|\{b \mid (a, b) \in E_I\}| = 1$ if $\{b \in A_{in}^\vee \mid (a, b) \in E\} \neq \emptyset$.

For example, with reference to the schema of Figure 1, the following are example instances:



We assume that each instance is properly stored by the workflow management system in the log file, which can be seen as a set $\mathcal{F} = \{I_1, \dots, I_n\}$ such that $\mathcal{WS} \models I_i$, for each $1 \leq i \leq n$. Among the instances of \mathcal{F} we are interested in discovering the most frequent patterns of execution as next defined. A graph $p = \langle A_p, E_p \rangle \subseteq \mathcal{WS}$ is a \mathcal{F} -*pattern* (cf. $\mathcal{F} \models p$) if there exists $I = \langle A_I, E_I \rangle \in \mathcal{F}$ such that $A_p \subseteq A_I$ and p is the subgraph of I induced by the nodes in A_p .

Let $supp(p) = |\{I \mid \{I\} \models p \wedge I \in \mathcal{F}\}|/|\mathcal{F}|$, be the *support* of a \mathcal{F} -pattern p . Then, given a real number $minSupp$, we consider the following two problems on workflows: (i) FCPD [5], i.e., finding all the connected patterns whose support is greater than $minSupp$; and (ii) FUPD, i.e., finding all the subsets of connected patterns whose support is greater than $minSupp$.

As an example, let us consider the control graph of Figure 1, the instances described above and $minSupp = 0.3$. Then, the patterns



are frequent connected patterns. Also, notice that none of the nodes l, i, h is frequent, whereas the subgraph $p = p_1 \cup p_2$ is frequent (and hence $\{p_1, p_2\}$ is a frequent unconnected pattern).

3 Mining Unconnected Patterns

In this paper we shall deal with an efficient solution for FUPD by assuming that the set $C(\mathcal{F})$ of all the frequent (w.r.t. $minSupp$) connected patterns in the set of instances \mathcal{F} has been already computed. Then, let q be a not-necessarily connected component of \mathcal{WS} with frequency $f(q)$ and p be a connected component with frequency $f(p)$ such that q and p are unconnected. Our aim is to compute as efficiently as possible the number of instances in \mathcal{F} executing both the components p and q , denoted by $f_p(q)$.

The problem has a straightforward solution consisting in the application of a level-wise algorithm (in the *Apriori* style [6]) which combines all the unconnected patterns in \mathcal{P} and then checks for their frequency. Indeed, the algorithm *ws-unconnected-find* shown in Figure 2 implements the above solution. Given an unconnected pattern p , we say that p is a *starting* pattern if it contains the starting activity of the workflow schema; otherwise, it is said a *terminating* pattern. Rather than computing all the possible unconnected patterns, we limit on starting patterns and we show how the space of all the connected starting patterns forms a lower semi-lattice that can be profitably explored in a bottom-up fashion. In fact, given two starting patterns r and p we say that r directly precedes p , denoted by $r \prec p$, if there exist a terminating pattern q such that $r = p \cup q$. It is not difficult to see that starting patterns can be constructed by means of a chain over the \prec relation. Such an approach is, in fact, exploited by the algorithm in Figure 2 that computes all the frequent starting patterns, by generating at each step k the patterns made of k distinct unconnected patterns.

The algorithm exploits the procedures *InitializeStructures* and *UpdateCandidateList*, which optimize candidate generation by tracking, for each pattern p , all the patterns which have a non-null intersection with p (and hence can be discarded when generating candidates). Moreover, notice that each pattern r generated at step k is also equipped with two sets *starting*(r) and *terminating*(r), which store the starting and terminating patterns respectively that have been used for generating r . Finally, the function

```

Input: A workflow schema  $\mathcal{WS}$ , a set  $\mathcal{F}$  of instances of  $\mathcal{WS}$ , the minimal support  $\text{minSupp}$ , the set  $C(\mathcal{F})$  of frequent connected  $\mathcal{F}$ -patterns.
Output: A set of frequent unconnected  $\mathcal{F}$ -patterns.
Method: Perform the following steps:
1 InitializeStructures();
2  $L_0 := \{p \mid p \in C(\mathcal{F}), a_0 \in p\}$ ; ***frequent connected starting patterns
3  $k := 0$ ,  $R := L_0$ ;  $C' := C(\mathcal{F}) - L_0$ ;
4 repeat
5    $U := \text{UpdateCandidateList}(L_k)$ 
6    $L_{k+1} := \text{ComputeFrequentPatterns}(U)$ ;
7    $R := R \cup L_{k+1}$ ;
8   forall  $r \in U - L_{k+1}$  do begin
9      $p := \text{starting}(r)$ ;
10    forall  $p' \in L_{k+1}$  s.t.  $p \subset p'$  do
11       $\text{discarded}(p') := \text{discarded}(p') \cup \{\text{terminating}(r)\}$ ;
12    end
13 until  $L_{k+1} = \emptyset$ ;
14 return  $R$ ;

Procedure InitializeStructures;
IS1 forall  $p \in C(\mathcal{F})$  do
IS2    $\text{discarded}(p) := \{q \mid q \in C(\mathcal{F}), p \cap q \neq \emptyset\}$ ;

Function ComputeFrequentPatterns( $U$ : set of candidates): set of frequent patterns;
CFP1 return  $\{r \mid r \in U, \text{supp}(r) > \text{minSupp}\}$ ;

Function UpdateCandidateList( $L_k$ : set of frequent patterns): set of candidate patterns;
UCL1  $U := \emptyset$ 
UCL2 forall  $p \in L_k$  do ***starting pattern
UCL3   forall  $q \in C' - \text{discarded}(p)$  do begin ***terminating pattern
UCL4      $r := p \cup q$ ;  $\text{starting}(r) = p$ ;  $\text{terminating}(r) = q$ ;
UCL5      $\text{discarded}(r) := \text{discarded}(p) \cup \text{discarded}(q)$ ;
UCL6      $U := U \cup \{r\}$ ;
UCL7   end;
UCL8 return  $U$ ;

```

Figure 2: **Algorithm** *ws-unconnected-find*

`ComputeFrequentPatterns` is invoked for filtering the candidates which frequently occur in \mathcal{F} .

A larger amount of pruning of the search space identified by *ws-unconnected-find* can be achieved by exploiting the peculiarities of the workflow graph. Our idea is to exploit the structure and the information regarding the frequency of each activity in order to identify, before their actual testing w.r.t. the logs, those patterns which are necessarily (un)frequent. We show how some proper data structures and algorithms can be used for effectively identifying a suitable lower bound and an upper bound for $f_p(q)$, denoted by $l_p(q)$ and $u_p(q)$ respectively, without requiring access to the log. The key of our approach is the basic situation where p and q are patterns each one made of a single activity.

Computing Frequency Bounds for Activities.

First of all, let us denote by $f : A \cup E \mapsto \mathbf{N}$ the function mapping either an activity a or an arc e to the number of instances in $\mathcal{F} = \{I_1, \dots, I_n\}$ executing it. Given an activity $a \in A$, let G_a be the subgraph of the control flow of \mathcal{WS} induced by all the nodes b such that there is a path from b to a in \mathcal{WS} . The starting point of our approach is to approximate for each node b in G_a , the value $f_a(b)$ by computing a lower bound $l_a(b)$ and an upper bound $u_a(b)$.

In order to derive the aforementioned bounds, we first determine a topological sort $\langle a = b_1, b_2, \dots, b_k \rangle$ of the nodes in G_a of \mathcal{WS} . Then we proceed as shown in Figure 3. In the step 1, the lower and upper bounds

of the activity a are fixed to the known value $f(a)$, determined through G_a . Then, each node b_i in G_a is processed according to the topological sort. In step 4, the set of all the activities $C(b_i)$ that can be reached by means of an edge starting in b_i and that are in G_a is computed. Step 5 is responsible for computing the upper bound $u_a(b_i)$, whereas steps 6–10 are responsible for computing the lower bound $l_a(b_i)$. Intuitively, the upper bound $u_a(b_i)$ can be computed by optimistically assuming that each arc outgoing from b_i is in some path reaching c . This justifies the formula of step 5.

```

Input: A workflow schema  $\mathcal{WS}$ , a set of instances  $\mathcal{F}$ , and an activity  $a$ .
Output: for each node  $b \in G_a$ , the values  $l_a(b)$  and  $u_a(b)$ .
Method: Perform the following steps:
1  $l_a(a) := f(a)$ ;  $u_a(a) := f(a)$ ;
2 let  $\langle a = b_1, b_2, \dots, b_k \rangle$  be the topological sort of the nodes in  $G_a$ ;
3 forall  $i = 2..k$  do begin
4    $C(b_i) := \{b \mid (b_i, b) \in E \wedge b \in G_a\}$ ;
5    $u_a(b_i) := \min(f(b_i), f(a), U)$ , with  $U = \sum_{e=(b_i, c) \mid c \in C(b_i)} \min(f(e), u_a(c))$ .
6   if  $b_i \in A_{out}^\circ \cup A_{out}^\wedge$  then
7      $l_a(b_i) := \min(f(b_i), \max(L_1^\wedge, L_1^\vee))$ , with
        $L_1^\wedge = \max_{c_j \in C(b_i) \cap A_{in}^\wedge} \{l_a(c_j)\}$ , and
        $L_1^\vee = \max_{c_j \in C(b_i) \cap A_{in}^\vee} \{\max(0, l_a(c_j) - \sum_{e=(d, c_j) \in \mathcal{WS} \wedge d \neq b_i} f(e))\}$ 
8   else case of  $b_i \in A_{out}^\circ$ 
9      $l_a(b_i) := \min(f(b_i), L_2^\circ + L_2^\vee)$ , with
        $L_2^\circ = \sum_{c_j \in C(b_i) \cap A_{in}^\wedge} \{l_a(c_j)\}$ , and
        $L_2^\vee = \sum_{c_j \in C(b_i) \cap A_{in}^\vee} \{\max(0, l_a(c_j) - \sum_{e=(d, c_j) \in \mathcal{WS} \wedge d \neq b_i} f(e))\}$ 
10  end
11 end
12 forall  $(b, c) \in G_a$  do begin
13   if  $b \in A_{out}^\wedge$  and  $c \in A_{in}^\vee$  then
14      $l_a(c) := \max(l_a(c), l_a(b))$ 
15 endfor

```

Figure 3: The *compute_frequency_bounds* algorithm

Concerning $l_a(b_i)$, observe that each node $c_j \in C(b_i)$ is executed with a by at least $l_a(c_j)$ instances. Therefore, we need to know how many of the instances executing b_i contribute to $l_a(c_j)$. Two cases arise: either (i) $b_i \in A_{out}^\vee \cup A_{out}^\wedge$, so the nodes connected to b_i may occur simultaneously within an instance, or (ii) $b_i \in A_{out}^\circ$, then all c_j are executed exclusively from each other. This explains why in the first alternative L_1^\wedge and L_1^\vee are computed by maximizing the contribution of each c_j , whereas in the second alternative the single contributions are summated. Finally, observe that when $c_j \in A_{in}^\vee$, it may be not the case that all of the $l_a(c_j)$ instances execute b_i , thus requiring to differentiate the formulas for L_1^\vee and L_1^\wedge (and, in the same way, for L_2^\vee and L_2^\wedge).

Observe that the final step in the algorithm possibly finds tighter lower bounds by exploiting the fact that, given two nodes b and c in G_a if $(b, c) \in \mathcal{WS}$, b is an and-fork node and c is an or-join node, then $l_a(c) \leq l_a(b)$ the activity b is executed each time the activity c is.

As an example, let us consider the graph G_m induced by node m in Figure 1. The inferred topological sort is $\langle m, g, b, c, a \rangle$. Hence, by applying *compute_frequency_bounds*($\mathcal{WS}, \mathcal{F}, m$) where \mathcal{F} is the

set of instances described in section 2, we obtain the following bounds:

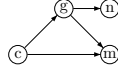
$$\begin{array}{llll} l_m(g) = 2, & u_m(g) = 2, & l_m(b) = 1, & u_m(b) = 1, \\ l_m(c) = 1, & u_m(c) = 2, & l_m(a) = 3, & u_m(a) = 4 \end{array}$$

According to these bounds, it is easy to see that $m \cup a$ is a frequent unconnected pattern, whereas $m \cup b$ is not (even though b and m are frequent patterns).

Computing Frequency Bounds for Patterns.

Let us now turn to the more general problem of approximating the value of $f_p(b)$, for any pattern p and any activity b . To this aim we simply reuse the technique described in the previous section with some adaptations. Let $INBORDER(p)$ denote the set of the activities in p having incoming arcs from $\mathcal{WS} - p$. Let $\mathcal{WS}(p)$ be the workflow schema derived from \mathcal{WS} by adding a new and-join node, say a_p (corresponding to the component p), and by adding an arc from each node b in $INBORDER(p)$ to a_p . Next, set $f(a_p) = f(p)$, and $f(e) = f(p)$ for each $e = (b, a_p) \in E$. Then, the function $compute_frequency_bounds(\mathcal{WS}, \mathcal{F}, p)$ is defined as $compute_frequency_bounds(\mathcal{WS}(p), \mathcal{F}, a_p)$.

As an example, let us consider the pattern p , structured as shown below:



According to the workflow schema shown in Figure 1, $INBORDER(p) = \{c, g\}$ (indeed, both nodes have incoming arcs from nodes which are not in p). In order to compute frequency bounds for p , we connect both g and c with a new dummy node a_p . Thus, we obtain $G_{a_p} = G_c \cup G_g$ and hence lower and upper bounds for each node w.r.t. a_p can be computed.

Improving *ws-unconnected-find*. We can now face the more general problem. Let q be a general component of \mathcal{WS} with frequency $f(q)$ and p be a connected component with frequency $f(p)$ such that q and p are unconnected. A lower bound and an upper bound of $f_p(q)$ are as follows:

- $l_p(q) = \max(0, \max_{b \in q} \{l_p(b) - (f(b) - f(q))\})$
- $u_p(q) = \min(f(q), \sum_{b \in OUTBORDER(q)} u_p(b))$.

Here, $OUTBORDER(p)$ refers to all the nodes in q having outgoing arcs in $\mathcal{WS} - q$. The intuition behind the above formulas is the following. The value $u_p(q)$ is obtained by taking into account the contribution of each node b of q from which there is a path to a node in p . However we may exclude in the upper bound computation all internal nodes of q (i.e., those not in $OUTBORDER(p)$) as they are always executed together with at least one node in $OUTBORDER(p)$. Concerning the computation of $l_p(q)$, observe that there are at least $l_p(b)$ instances executing

$b \in q$ and p . Hence, as $f(b) \geq f(q)$, there are at least $l_p(b) - (f(b) - f(q))$ instances connecting q and p and executing b . It turns out that a suitable lower bound is provided by the node exhibiting the maximum such value.

Generalized upper and lower bounds can be finally used for pruning the search space of the *ws-unconnected-find* algorithm. In fact, if for any two patterns p and q such that $u_p(q) < minSupp$, then it is always the case that p and q never occur frequently together. Conversely, if $l_p(q) \geq minSupp$ then p and q can be combined into a pattern that is frequent as well. Thus, the algorithm *ws-disconnected-find* can be optimized by suitably adapting the procedures `InitializeStructures`, `UpdateCandidateList` and `ComputeFrequentPatterns` as shown in Figure 4.

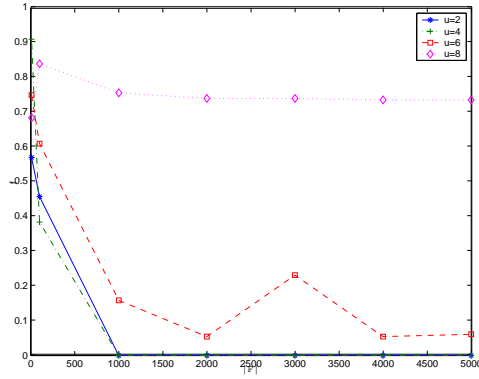
Procedure InitializeStructures;	
IS1	forall $p \in C(\mathcal{F})$ do begin
IS2	discarded(p) := $\{q \mid q \in C(\mathcal{F}), p \cap q \neq \emptyset\}$;
IS3	$(l_p, u_p) := compute_frequency_bounds(\mathcal{WS}, \mathcal{F}, p)$
IS4	end;
Function ComputeFrequentPatterns(U : set of candidates): set of frequent patterns;	
CFP1	$LF := \{r \mid r \in U, l_{terminating(r)}(starting(r)) \geq minSupp\}$;
CFP2	$LU := \{r \mid r \in U, u_{terminating(r)}(starting(r)) < minSupp\}$;
CFP3	return $LF \cup \{r \mid r \in U - (LF \cup LU), r \text{ is frequent w.r.t. } \mathcal{F}\}$;
Procedure UpdateCandidateList(L_k : set of frequent patterns): set of candidates	
UCL1	$U := \emptyset$;
UCL2	forall $p \in L_k$ do
UCL3	forall $q \in C' - discarded(p)$ do begin
UCL4	$r := p \cup q$; $starting(r) = p$; $terminating(r) = q$;
UCL5	$discarded(r) := discarded(p) \cup discarded(q)$;
UCL6	$l_q(p) = \max(0, \max_{b \in p} \{l_q(b) - (f(b) - f(p))\})$;
UCL7	$u_q(p) = \min(f(p), \sum_{b \in OUTBORDER(q)} u_p(b))$;
UCL8	$U := U \cup \{r\}$;
UCL9	end;
UCL10	return U ;

Figure 4: Optimizations to *ws-unconnected-find*

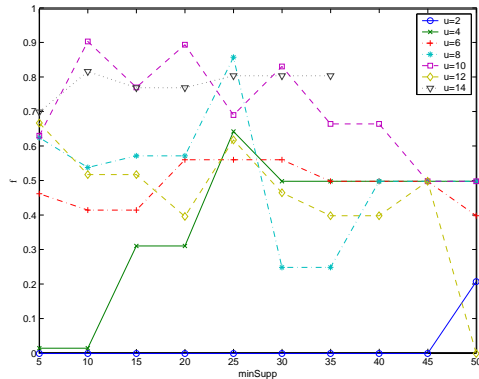
4 Experiments and Discussion

In this section we evaluate whether the computation of upper and lower bounds avoids the generation of unnecessary candidate patterns to check for frequency against the log data. In our experiments, we use synthesized data, in which both the workflow schema and the instances are artificially generated. The generation can be tuned according to: i) the size of \mathcal{F} , ii) the average number d of frequent connected patterns to use in the generation of frequent unconnected patterns, and iii) the average number u of frequent patterns to exploit in the generation of unfrequent unconnected patterns. The details of data generation are described in [4].

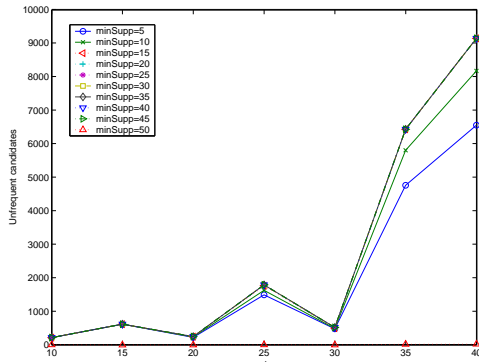
We evaluated the ratio $f = n_{cc}/n_{cp}$ between the number n_{cc} of candidate patterns checked against the logs and the total number n_{cp} of candidate patterns. Low values of f represent a higher pruning capability. Figure 5(a) shows the behavior of f for $d = 10$, $minSupp = 5\%$ and increasing values of \mathcal{F} and u . As



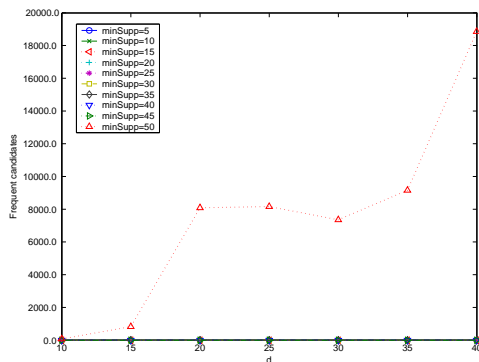
(a) Ratio f , increasing values of $|\mathcal{F}|$ and u .



(b) Ratio f , increasing values of minSupp and u .



(c) Pruning by upper bound.



(d) Pruning by lower bound.

we can see, f is quite low, except when $u = 8$.

Figure 5(b) exhibits the ratio f for increasing values of minSupp and u , when $|\mathcal{F}| = 1.000$ and $d = 15$. Peaks within the graphs are mainly due to the fact that we are mining unconnected components: at low support values, patterns are mined as frequent connected (i.e., the frequency of paths connecting the components is greater than the given threshold). As soon as support threshold increases, frequencies of paths tend to decrease, and hence a higher number of unconnected frequent patterns is detected by the algorithm.

More in general, upper bounds are better in pruning, whereas lower bounds are quite effective at high values of minSupp (which guarantee several disconnections among frequent patterns). This is shown by Figures 5(c) and 5(d), where the number of pruned unfrequent and frequent patterns is shown for increasing values of minSupp and d , with u fixed to 2 and \mathcal{F} to 1.000.

To conclude, the graph-theoretic approach developed in this paper exhibits a significant pruning capability which can be profitably applied to mining unconnected patterns in general constrained graphs, such as workflows. As a direction of future research, it would be interesting to extend the proposed approach to richer workflow models, in which complex constraints (such as time constraints, pre/post-conditions, and rules for exception handling) can be expressed.

References

- [1] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *EDBT'98*, pages 469–483, 1998.
- [2] U. Dayal, M. Hsu, and R. Ladin. Business process coordination: State of the art, trends and open issues. In *VLDB'01*, pages 3–13, 2001.
- [3] M. Gillmann, W. Wonner, and G. Weikum. Workflow management with service quality guarantee. In *SIGMOD'02*, pages 228–239, 2002.
- [4] G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining unconnected patterns in workflows. Technical Report 5, ICAR-CNR, 2004.
- [5] G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining and Reasoning on Workflows. *IEEE Trans. on Data and Knowledge Engineering*, 2005. to appear.
- [6] A. Inokuchi, T. Washi, and H. Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In *PKDD'00*, pages 13–23, 2000.
- [7] P. Koksai, S.N. Arpinar, and A. Dogac. Workflow history management. *SIGMOD Record*, 27(1):67–75, 1998.
- [8] W.M.P. van der Aalst, and others. Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering*, 47(3):237–267, 2003.

Figure 5: Performance Graphs.

The Best Nurturers in Computer Science Research

Bharath Kumar M.

mbk@csa.iisc.ernet.in

Dept. of Computer Science
and Automation,
Indian Institute of Science

Y. N. Srikant

srikant@csa.iisc.ernet.in

Dept. of Computer Science
and Automation,
Indian Institute of Science

Abstract

The paper presents a heuristic for mining nurturers in temporally organized collaboration networks: people who facilitate the growth and success of the young ones. Specifically, this heuristic is applied to the computer science bibliographic data to find the best nurturers in computer science research. The measure of success is parameterized, and the paper demonstrates experiments and results with publication count as a success metric. Rather than just the nurturer's success, the heuristic captures the influence he has had in the independent success of the relatively young in the network. These results can hence be a useful resource to graduate students and post-doctoral candidates. Interestingly, there is a recognizable deviation between the rankings of the most successful researchers and the best nurturers, which although is obvious from a social perspective has not been statistically demonstrated.

Keywords: Social Network Analysis, Bibliometrics, Temporal Data Mining.

1 Introduction

Consider a student Arjun, who has finished his undergraduate degree in Computer Science, and is seeking a PhD degree followed by a successful career in Computer Science research. How does he choose his research advisor? He has the following options with him:

1. Look up the rankings of various universities [1], and apply to any "reasonably good" professor in any of the top universities.

Does working with any reasonably good professor at a top university ensure that Arjun gets the training to pursue a successful research career?

2. Look up the web sites that present the most successful researchers, based on the number of publications [2] or the citations they have received [3] [4].

Arjun can then do his own analysis and find out how many of these researchers are active at the current date. He wants to ensure he does not work with a professor who's past his prime; or neglect a young and upcoming professor.

But still, does working with a top professor, who's

known for his research, imply Arjun will learn how to do good research and in due course have a successful research career?

3. Get word-of-mouth information on the social aspects of working with a particular advisor.

Arjun can talk to an advisor's past and current students, get their feedback, attribute a certain trust to what each one says, and then decide. How many people will Arjun ask? How much will he trust each individual feedback?

For Arjun, it is more important to seek a professor who will **nurture** him to become a good researcher: one who will teach him how best to do research that ends up in good publications, one who will bootstrap him into a good research network, where he hops onto a successful research career path on his own. Although being with a good researcher or in a top school does help, there is no guarantee of being nurtured. A good researcher may not be a good nurturer, and getting into a top school does not always ensure a good research career.

Arjun would benefit if:

- there is a way to summarize the nurturing ability of a researcher by mining the performance of people he nurtured, and thereby compare one nurturer with another.
- there is a way to find out the best nurturers in a given period of time.
- there is a way to find out researchers who have nurtured people to publish many papers, to obtain many citations for their papers or in a given area of research.

This paper presents a *Nurturer-Finder* heuristic that Arjun can use. When Arjun chooses to work with any of these people, he is assured that he is not just choosing them for their research prowess, but for the positive experiences people like himself had in the past. It may turn out that the nurturers also happen to be successful researchers themselves, as the results show.

2 The Nurturer-Finder's Design Principles

While it may be argued that nurturing may even happen inside the confines of a classroom, through well-written

books, or that a true nurturer may just stay behind the scenes and not even be a co-author; mining among associations in bibliographic databases remains the best context to scalably look for nurturers in research:

- Publishing is the defacto standard for evaluating good research.
- The art of scientific reporting is best taught “hands on”. Senior collaborators typically give direction on the most important aspects of the innovation, provide appropriate feedback on its capabilities and limitations, and contrast the innovation with other progress in the area.
- People who have contributed towards a research project often end up as co-authors in the subsequent publication.
- Bibliographic databases are well documented, and are already used for extensive analysis of the impact of research.

However, all publications may not have a nurturer-nurtured pair; often, publications have “almost equals” as co-authors. Hence, the heuristic must not stray in its analysis, and report any co-author pair as a nurturer-nurtured pair. In contrast, no co-author pair can be neglected, since every collaboration can potentially be a context of nurturing.

The nurturer-finder heuristic is inspired by the concept of *gurudakshina* known from ancient Indian traditions. After finishing his education, a student (*shishya*) pays tribute to his teacher (*guru*) for the knowledge he was bestowed. On the same light, whenever a person achieves some success (through a publication), he attributes a part of that success to his “gurus” proportionate to their nurturing influence on him. The gurus with the highest *gurudakshina* are the best nurturers.

The design principles are elucidated as follows:

1. The effect of nurturing manifests in the **post-associative period**.

Any amount of success a person may have with his nurturer, it is still not indicative whether he has been successfully nurtured. The nurturing is true and complete, when he tastes success “on his own” in the absence of his nurturer. This period is hence termed as post-associative, and is used as the context to decide the extent of the nurturing.

2. The more **self-made** a person is, the less he attributes his success to his past associates.

People who have seen success on their own, without associating with too many people, especially early in their career, can be termed as *self-made*. They are the self-motivated people, who probably were not nurtured at all by someone else. It is fair that these people attribute less of their success to their past associates.

3. The success achieved by a person at any time is considered to be influenced by all his past associates. However it is **tributed** to only those who do not have a direct pay-off in the current collaboration.

While contributing towards a publication, an author may be acting upon the influence he’s had from many of his past and current associates. However, all the current associates (the co-authors) in the publication still have their own pay-offs from it. So, the tribute for one’s success is only given away to past associates who have helped influence him to be successful in a current venture without a motive of their own.

4. The tribute is appropriated among the past associates in proportion to their estimated **nurturing influence** on the person.

Nurturing happens most when a person is still young in his career - and the people who associated with him earlier are more important (in terms of a nurturing influence) than the ones he associates with later in his career. This can be termed as the *strength of early association*. As an aside - while the strength of early association of a person with his nurturer will be high, the reverse need not hold, since the nurturer is expected to be already relatively mature in his career.

A person need not have been nurtured equally by all people he had good early associations with. The ones who nurtured him more are most likely those who were termed to have a good *nurturing ability* by other people as well.

Thus, an associate’s nurturing influence on a person is proportional to the strength of early association with this person and the associate’s own nurturing ability. The tribute can then be appropriated to each past associate in accordance to the proportion of their nurturing influence.

The above principles guide the design of the Nurturer-Finder heuristic, which works based on the following outline. Publications are processed in temporal sequence, at some granularity, either grouped by years or by months.

1. As every person publishes, his strength of early association with his associates, and their nurturing influence on him are tracked.

2. Every time he achieves a certain success from a publication, it is tributed to his past associates for influencing him in his “formative” years, in accordance to their nurturing influence. The more self-made a person is, the less is his tribute.

3. Every person collects the tribute he gets from others.

4. The person with the highest tribute is the best nurturer. People can also be sorted on the tributes they have, to arrange them in non-increasing order of their nurturing abilities.

3 The Formulation of the Nurturer-Finder Heuristic

The heuristic is abstractly formulated, allowing for reuse in domains outside of bibliographic databases.

A publication is an instance of a collaboration, and happens at a certain discrete instant in time. The bibliographic database is termed as the set of *collaborations*.

A collaboration *c* has the following properties,

$associates_c$, the set of people involved in the collaboration c .

$time_c$, the time at which the collaboration happened.

sig_c , the quantifier representing the significance of the collaboration, which could be equal to 1, the impact factor of the conference or journal where it was published, or the number of citations the publication has received.

Each associate p in a collaboration gets a certain significance measure to himself: his share of success. In the model used here, the success is equally shared among the associates.

$$sig_c^p = \begin{cases} \frac{sig_c}{|associates_c|} & \text{if } p \in associates_c \\ 0 & \text{if } p \notin associates_c \end{cases}$$

Other models, for instance, can give importance to the position of the author's name in the list, while deciding the significance of each associate.

The set of all collaborations that have happened till time t , is given by,

$$collabs^t = \{c \in collabs | time_c < t\}$$

The set of all people involved in all collaborations till time t is represented by,

$$people^t = \bigcup \{associates_c : c \in collabs^t\}$$

The cumulative significance of each person until time t is represented by,

$$cum-sig_p^t = \sum_{c \in collabs^t} sig_c^p$$

A measure of the degree of association a person q had in the significance a person p achieved during a collaboration c is given by,

$$q_{assoc}_p^c = sig_c^p * \frac{sig_c^q}{sig_c}$$

The $\frac{sig_c^q}{sig_c}$ factor is indicative of q 's involvement in c . Higher q 's involvement, higher is his association with p 's significance.

The **early association** q had with p , until time t is representative of the successful collaboration p had with q early in his career.

$$q_{early-assoc}_p^t = \sum_{c \in collabs^t} \left(\frac{q_{assoc}_p^c}{cum-sig_p^{time_c}} \right)$$

A measure of how self-made a person is, is also useful - to determine his independence on his associates for his success. This measure also considers the *earliness* of his *self-establishment*. The intuition being that, a person who gets independent success later in his career, but after collaborating with people early on, is not as self-made as a person who

was independent right from the start. It is likely that a self-made person was not nurtured by too many people at all, and hence he must attribute less of his success to his 'mentors'.

$$self-estab_p^t = p_{early-assoc}_p^t$$

The nurturing influence a person q has had on p , (where $p \neq q$) until time t is given by

$$q_{ni}_p^t = \sum_{c \in collabs^t} \left(\frac{q_{assoc}_p^c * (nship_q^{time_c})^\alpha}{cum-sig_p^{time_c}} \right)$$

The term $nship_p^t$, which is detailed later, is indicative of the nurturing ability of a person p until time t .

Most people have a relatively constant research output every year. Their cumulative significance would thus grow linearly. However, the nurtureship of a person, which is made up by collecting tributes from collaborators, can grow faster than the cumulative significance. In that event, when a person p collaborates with another person q who has a large nurtureship a little late in p 's career, he still concedes a large nurturing influence to q . This may sideline the earlier nurturers of p . To manage this effect the parameter α is introduced, which is used to control the dominance of nurtureship over cumulative significance. For smaller values of α , the earliness factor dominates the nurturing influence. Increasing the value of α makes the people with higher nurtureship "richer" at the cost of the others. In the experiments reported in the paper, α was hand-engineered to 0.5, for satisfactory results.

$p_{ni}_p^t$ is not defined. A person does not nurture himself.

The tribute given away to past associates everytime a person p achieves a certain significance through a collaboration c , is given by

$$trib_c^p = sig_c^p * \left(1 - \frac{self-estab_p^{time_c}}{cum-sig_p^{time_c}} \right)$$

The tribute a person p gives to an associate q , (where $p \neq q$), because of achieving a certain significance through a collaboration c is given by

$$q_{trib}_c^p = \begin{cases} \frac{trib_c^p * q_{ni}_p^{time_c}}{\sum_{r \in (people^{time_c} - p)} r_{ni}_p^{time_c}} & \text{if } q \notin associates_c \\ 0 & \text{if } q \in associates_c \end{cases}$$

The tribute is thus appropriated proportionate to the nurturing influence.

The $nship_p^t$ of a person is the cumulative sum of the tributes collected by p from other associates until time t . The term $nship_p^t$ is used to represent the nurturing ability of a person right after time t , inclusive of the collaborations

that happened in that time instant. This is incrementally calculated.

$$nship_p^{t'} = nship_p^t + \sum_{\substack{c \in collabs; \\ time_c = t}} \sum_{q \in associates_c} ptrib_c^q$$

and $nship_p^0 = 1$

Thus, the best nurturer is one who has the highest $nship_p^{t'}$ where t is the current time.

The total tribute a person p gives to an associate q until time t is represented by,

$$qtrib_p^t = \sum_{c \in collabs^t} qtrib_c^p$$

This is used to present a drill down of the nurtureship of each person, showing the extent of tribute each of their nurtured give them. $ptrib_p^t = 1.0$ This accounts for the default value of $nship_p^0$.

4 Some Experiments on the DBLP Database

The Digital Bibliography and Library Project (DBLP) [2] provides digital information on major computer science journals and publications, and indexes more than 520000 articles. Citations are also available for a subset of the articles indexed. The DBL-browser offers an interface to access the compressed database containing the article information. The Nurturer-Finder heuristic was applied on the DBLP in two sets of experiments to find nurturers for publications count, and for citations. For lack of space, this paper only reports a few top results based on publication count. A more detailed account of these experiments, some special handling needed for some outliers and a method to yield nurturers in time slices is discussed in a technical report [6].

The algorithm was implemented using Java, and used the DBL Browser libraries [5] for accessing the publication records. The algorithm is incremental in nature, and parses each publication in the database exactly once. Every time a publication is processed, all past associates of every co-author are processed, to be assigned tributes.

α is chosen as 0.5 in the following experiments. A discussion on the choice of α is considered in the technical report [6].

4.1 Nurturing for Publication Count

To compute nurturers and their nurtured based on publication count, every entry in the DBLP is assigned a significance of 1, and an author's significance for participation is $\frac{1}{|associates|}$. This metric in itself is not semantically very accurate due to the disparity in quality among the journals and conferences indexed by the DBLP, but acts as a good first measure nevertheless.

The table 1 lists the top few nurturers, their nurtureship value and the people who were 'nurtured' by them, and the tribute each of them gave away to the nurturer. These nurtured people are those who co-authored with the nurturers early in their careers, and then went on to be prolific on their own as authors, even in the absence of their nurturers. Only people who gave away tributes greater than or equal to the value 5 are listed. A person may appear as "nurtured" by more than one nurturer, if he gave away reasonably big tributes to all of them.

4.1.1 Interpreting the results

- The heuristic attempts to recognize the social trait of nurturing through statistical analysis, and hence acceptance of the validity of the findings is possible only by common perception of readers conversant with the who's who of the computer science research community.
- While it is questionable whether there exists a strict nurturer-nurtured distinction in the results, if the border is blurred to mean a nurturing influence, which can be mutual too at times, the results become easier to digest.
- The list of nurturers, on its own, has successful researchers. The authors found this phenomenon most interesting because the calculation of nurtureship does not take into account any publication of the nurturer himself, and considers only post-associative success of people who co-authored with them early in their career.
- The results also suggest the ability of these people to sight talent: people who would later end up doing very well on their own. Good nurturers are also good talent sighters.

5 Discussion on related work

Barabasi et. al in [7] show the existence of preferential attachment during addition of new nodes into the collaboration network.

"For a new author, that appears for the first time on a publication, preferential attachment has a simple meaning: it is more likely that the first paper will be co-authored with somebody that already has a large number of co-authors (links) than with somebody less connected. As a result "old" authors with more links will increase their number of co-authors at a higher rate than those with fewer links."

Does this imply that the best nurturers are simply the best collaborators? When Barabasi et. al. consider the addition of new nodes, they do not track the longevity and success achieved by that new node in the collaboration network. While good collaborators may be the context for addition of newer nodes, they need not be contexts where people who perform well in the long term may be added. To answer this, a new set of experiments were conducted to identify the best collaborators. Barabasi's

Nurturer	Val	Nurturer	Val	Nurturer	Val	Nurturer	Val
Nurtured		Nurtured		Nurtured		Nurtured	
1) Jeffrey D. Ullman	144	5) John E. Hopcroft	97	9) Zvi Galil	83	15) Grzegorz Rozenberg	75
Henry F. Korth	8	Jeffrey D. Ullman	24	Moti Yung	10	Dirk Vermeir	7
Yehoshua Sagiv	8	Robert Endre Tarjan	14	David Eppstein	7	Robert Meersman	6
Fereidoon Sadri	7	Richard Cole	12	Kunsoo Park	7	16) Richard J. Lipton	75
Alberto O. Mendelzon	6	Steven Fortune	5	Nimrod Megiddo	6	Dan Boneh	8
Sam Toueg	6	Joachim von zur Gathen	5	Dany Breslauer	5	Lawrence Snyder	7
Ravi Sethi	5	Gordon T. Wilfong	5	10) Christos H. Papadimitriou	81	David P. Dobkin	5
David Maier	5	6) Robert Endre Tarjan	95	Joseph S. B. Mitchell	10	17) John H. Reif	74
Joan Feigenbaum	5	Thomas Lengauer	11	Paris C. Kanellakis	6	Paul G. Spirakis	17
2) Zohar Manna	126	Haim Kaplan	6	John N. Tsitsiklis	5	Sanguthevar Rajasekaran	8
Martin Abadi	23	Jeffery Westbrook	6	Mihalis Yannakakis	5	Philip N. Klein	7
Amir Pnueli	21	Andrew V. Goldberg	6	11) Ronald L. Rivest	80	Sandeep Sen	6
Adi Shamir	15	David R. Cheriton	5	Robert E. Schapire	10	18) Adi Shamir	74
Nachum Dershowitz	11	7) Ugo Montanari	90	Avrim Blum	9	Uriel Feige	16
Shmuel Katz	6	Roberto Gorrieri	7	Benny Chor	5	Amos Fiat	9
Thomas A. Henzinger	6	Andrea Corradini	7	Jon Doyle	5	Eli Biham	8
Jean Vuillemin	5	Francesca Rossi	6	Sally A. Goldman	5	Yossi Matias	5
Luca de Alfaro	5	Vladimiro Sassone	6	12) Kurt Mehlhorn	78	Moshe Tennenholtz	5
Ashok K. Chandra	5	Alberto Martelli	6	Michael Kaufmann	11	19) Jacob A. Abraham	73
3) Albert R. Meyer	113	Pierpaolo Degano	5	Majid Sarrafzadeh	6	Prithviraj Banerjee	18
Joseph Y. Halpern	38	Giorgio Levi	5	Norbert Blum	5	Kaushik Roy	11
John C. Mitchell	11	8) C. V. Ramamoorthy	88	13) John Mylopoulos	77	Abhijit Chatterjee	7
Nancy A. Lynch	7	Benjamin W. Wah	11	James P. Delgrande	10	W. Kent Fuchs	5
David Harel	7	Vijay K. Garg	9	Hector J. Levesque	7	20) Leonidas J. Guibas	71
4) Michael Stonebraker	106	K. Mani Chandy	9	Nick Roussopoulos	6	John Hersherberger	9
Marti A. Hearst	8	Jaideep Srivastava	9	Alexander Borgida	5	Jack Snoeyink	6
Michael J. Carey	7	K. H. Kim	8	14) Amir Pnueli	76	Andrew M. Odlyzko	6
Akhil Kumar	7	Shashi Shekhar	7	Dennis Shasha	9	21) Oscar H. Ibarra	69
Timos K. Sellis	6	Wei-Tek Tsai	5	David Harel	5	Tao Jiang	15
Sunita Sarawagi	5	Atul Prakash	5	Doron Peled	5	Louis E. Rosier	6
Joseph M. Hellerstein	5			Oded Maler	5	Shlomo Moran	5
Margo I. Seltzer	5					Hui Wang	5

Table 1: Publication Count: Top nurturers and their nurtured

experiments consider only the degree of a node to qualify the best collaborators. Here, the good collaborators were said to be those who collaborated frequently with other good collaborators. This is similar to page rank computation [8], although the weights were computed iteratively year by year. It also differed from the nurturer-finder in that, there was no consideration for earliness, and post-associative significance.

The rankings for top collaborators showed changes when compared to the top nurturers, although the correlation with top collaborators was better than the correlation with the top authors. This suggests that the trait of nurturing is perhaps in some way related to the trait of collaborating. Looking at this the other way, it could also indicate that young people, the new entrants in the network have a preference for good collaborators. Good collaborators typically have good social networks which come in handy for the new.

Further, the weighted tribute graph can be analysed for transitivity and hence discover ‘nurturing’ neighborhoods. It is useful to mention [9], where Newman evaluates several social network measures on scientific coauthorship networks. The connectedness of a scientist is measured based on his reachability on a weighted collaboration graph.

The above mentioned references and [10] can be classified as means to infer different roles played by people in collaboration networks. The current work on nurturers can also be grouped alongside.

References

- [1] *USNews*, <http://www.usnews.com>
- [2] *DBLP*, <http://www.informatik.uni-trier.de/~ley/db/>
- [3] *ISI Highly Cited*, <http://www.isihighlycited.com>
- [4] *Most cited authors in Computer Science*, <http://citeseer.ist.psu.edu/mostcited.html>
- [5] *DBL Browser*, <http://dbis.uni-trier.de/DBL-Browser/>
- [6] Bharath Kumar M., Y. N. Srikant. *The Best Nurturers in Computer Science Research*. CSA, IISc Technical Report, 2004, <http://archive.csa.iisc.ernet.in/TR/2004/10/>
- [7] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. *Evolution of the social network of scientific collaboration*. Physica A, 311(3–4):590–614, 2002.
- [8] S. Brin, L. Page, R. Motwani, and T. Winograd. *The page rank citation ranking: Bringing over to the web*. Tech. Rep. 1999-66, Stanford Digital Libraries Working Paper, 1999, <http://dbpubs.stanford.edu:8090/pub/1999-66>.
- [9] M. E. J. Newman. *Who is the best connected scientist? a study of scientific coauthorship networks*. Physics Review, E64, 2001
- [10] J. Kleinberg. *Authoritative sources in a hyperlinked environment*. Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

Knowledge Discovery from Heterogeneous Dynamic Systems using Change-Point Correlations

Tsuyoshi Idé*

Keisuke Inoue*

October 4, 2004

Abstract

Most of the stream mining techniques presented so far have primarily paid attention to discovering association rules by direct comparison between time-series data sets. However, their utility is very limited for heterogeneous systems, where time series of various types (discrete, continuous, oscillatory, noisy, etc.) act dynamically in a strongly correlated manner. In this paper, we introduce a new nonlinear transformation, singular spectrum transformation (SST), to address the problem of knowledge discovery of causal relationships from a set of time series. SST is a transformation that transforms a time series into the probability density function that represents a chance to observe some particular change. For an automobile data set, we demonstrate that SST enables us to discover a hidden and useful dependency between variables.

Keywords: time-series, change-point detection, singular-spectrum analysis, hidden dependency

1 Introduction

The frontiers of data mining research are being extended to include knowledge discovery from nontraditional data types such as statically [7] and dynamically [6] structured data. However, little attention has been paid to heterogeneous dynamic systems, where time series of various types (discrete, continuous, oscillatory, noisy, etc.) act dynamically in a strongly correlated manner.

Generally, in strongly correlated dynamic systems, the behavior of the whole system can often be extremely complicated even if the mechanism of correlation between each pair of variables is relatively simple. Therefore knowledge discovery in such systems can be far more difficult than expected. For instance, in an automobile, the individual states of the variables such as engine RPM (revolutions per minutes), engaged gear, fuel flow rate, throttle position (TP) sensor, and air intake oxygen density have almost an infinite number of combinations depending on the environment around

the car and human actions. Therefore, it is generally impossible to find a rule like “if variables x_1, x_2, \dots have a certain combination of values, then the system would be faulty.”

In this paper, we address the issue of discovering causal dependencies hidden deep within the heterogeneous time-series data. We assume that we are not provided with detailed prior knowledge of dependencies. In addition, we assume that each variable exhibits sudden and steep changes in a heterogeneous manner so that traditional approaches that attempt to separate trend and noise components are difficult to use.

Note that this problem setting is different from traditional stream mining. An implicit assumption of Das et al. [2], which is known as a seminal work in this field, was that subsequences of individual variables can be clustered into one of a small numbers of patterns. Except for parts of the data which may exhibit relatively simple behaviors, the utility of their approach is very limited in heterogeneous dynamic systems. Also, Keogh-Lin-Truppel [9] recently pointed out that it is theoretically questionable whether or not one can fit an arbitrarily chosen subsequence into one of the patterns.

In this paper, we tackle this issue by introducing a new nonlinear transformation, singular spectrum transformation (SST) for a set of time series data. SST is a transformation that converts an original time series into a new time series based on change-point scores. The resultant time series can be interpreted as the probability distribution that some change occurs. Since change points in a mechanical system are expected to be caused by a well-defined mechanism, if the score simultaneously has a high value at some time for two different variables, then a causal relationship is likely between them.

The essence of our idea is illustrated in Fig. 1, where two artificially generated heterogeneous data sets (see Subsection 3.2 for details) and their SSTs are shown. While it is difficult to infer any dependency between the two original variables, SST clearly reveals a hidden dependency between them in terms of synchronization of their change points. Note that the results in Figs. 1 (b) and (d) were obtained using a common algorithm

*IBM Research, Tokyo Research Laboratory. E-mail: {goodidea, inoue}@jp.ibm.com.

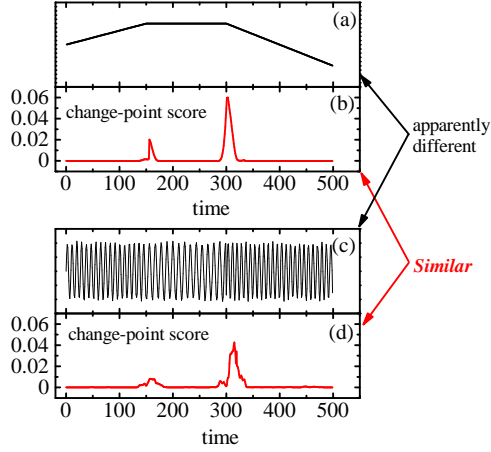


Figure 1: Example of SST in a heterogeneous system. Original time-series in (a) and (c) are transformed into change-point scores in (b) and (d), so that a hidden similarity is revealed.

and a common parameter set. Therefore, we see that, by performing SST, the problem of data mining in heterogeneous systems can be reduced to mining in homogeneous systems without using any detailed knowledge on the behavior of data. To the best of the authors' knowledge, this is the first work that uses change-point correlation in the context of knowledge discovery from dynamic systems with strongly-correlated and heterogeneous natures.

2 Change-point detection

2.1 Extraction of past patterns. Consider a time series $\mathcal{T} = \{x(1), x(2), \dots, x(t), \dots\}$ and its consecutive subsequence with length w as $\{x(t-w), \dots, x(t-2), x(t-1)\}$. We define a column vector corresponding to this subsequence as

$$\mathbf{s}(t-1) = (x(t-w), \dots, x(t-1))^T,$$

where the superscript T represents transpose. We construct a matrix, which is often called a Hankel matrix, using column vectors of this kind as

$$H(t) = [\mathbf{s}(t-n), \dots, \mathbf{s}(t-2), \mathbf{s}(t-1)].$$

We call this $w \times n$ matrix a trajectory matrix at t , following Moskvina-Zhigljavsky [11]. By definition, the trajectory matrix is defined over $w+n-1$ elements from $x(t-1)$ to $x(t-w-n+1)$. We denote $w+n-1$ as W . We illustrate the setting in Fig. 2.

The trajectory matrix $H(t)$ can be viewed as a record that contains various change patterns within the range of the past W points under the length constraint

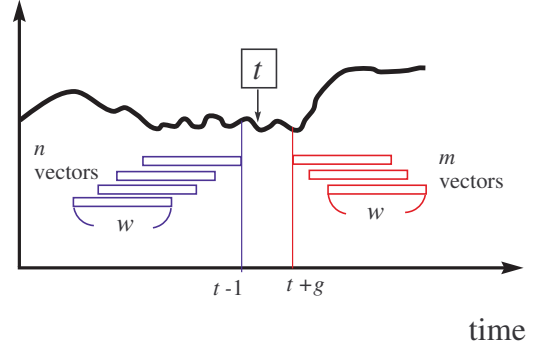


Figure 2: Summary of parameters used in SST. From m and n subsequences at both sides of a time point (t), representative patterns are calculated.

w . Now let us extract the representative patterns from $H(t)$. We write a representative pattern as \mathbf{u} . It is natural to suppose that this is expressed as a linear combination of $\mathbf{s}(t_j)$ s:

$$\mathbf{u} = c \sum_{i=1}^n v_i \mathbf{s}(t-i),$$

where c is a normalization constant to satisfy $\mathbf{u}^T \mathbf{u} = 1$. If we define an n -dimensional vector as $\mathbf{v} = (v_1, \dots, v_n)^T$, this equation is simply expressed as $\mathbf{u} = cH(t)\mathbf{v}$. We determine the representative by majority voting among the observed patterns. In particular, we want the direction that produces the strongest constructive interference between \mathbf{ss} . Mathematically, this direction will be found as

$$(2.1) \quad \mathbf{v}(t) \equiv \arg \max_{\tilde{\mathbf{v}}} \|H(t)\tilde{\mathbf{v}}\|^2,$$

where we impose a constraint of $\mathbf{v}^T \mathbf{v} = 1$. Introducing a Lagrange multiplier λ , this equation is reduced to

$$\frac{\partial}{\partial \tilde{\mathbf{v}}} [\tilde{\mathbf{v}}^T H(t)^T H(t) \tilde{\mathbf{v}} - \lambda \tilde{\mathbf{v}}^T \tilde{\mathbf{v}}] = 0.$$

From this, we immediately see that \mathbf{v} is the normalized solution of an eigenvalue equation

$$H(t)^T H(t) \mathbf{v} = \lambda \mathbf{v}.$$

Also, \mathbf{u} is the normalized solution of the eigenvalue equation of $H(t)H(t)^T$, i.e.

$$(2.2) \quad H(t)H(t)^T \mathbf{u} = \lambda \mathbf{u}.$$

These results show that the representative pattern \mathbf{u} and its coefficient vector \mathbf{v} are the left and right singular

vectors of $H(t)$, respectively. The singular value is equal to $\sqrt{\lambda}$.

Let us denote the singular values and the left singular vectors as $\{(\sigma_1, \mathbf{u}_1), (\sigma_2, \mathbf{u}_2), \dots, (\sigma_l, \mathbf{u}_l)\}$ in descending order of the singular values. The parameter l represents the number of representative patterns under consideration. The greater the singular value is, the more dominant the corresponding pattern is. If a singular value (≥ 0) is small, then the corresponding pattern can be considered to be a noise component.

As described above, the method to find the dominant components using singular value decomposition (SVD) on the Hankel matrices is called singular spectrum analysis.¹

2.2 Extraction of the current pattern. On the future side of the trajectory matrix, we again take a column vector with length w as

$$\mathbf{r}(t+g) = (x(t+g), \dots, x(t+g+w-1))^T.$$

This is the same as $\mathbf{s}(t+g+w-1)$, but we introduce this new notation to represent a symmetry between both sides of t . We again define a Hankel matrix, which we will call a test matrix at t , using m \mathbf{r} s

$$G(t) = [\mathbf{r}(t+g), \mathbf{r}(t+g+1), \dots, \mathbf{r}(t+g+m-1)].$$

As in Eq. (2.2), the present representative pattern is given by the solution of

$$(2.3) \quad G(t)G(t)^T \mathbf{u} = \mu \mathbf{u}.$$

We call the normalized largest eigenvector the test vector, and represent it as $\beta(t)$.

2.3 Change-point score. We have obtained the past representative patterns $\{\mathbf{u}_i | i = 1, \dots, l\}$ and the present representative pattern as $\beta(t)$. Let us define an anomaly metric using these patterns. If $\beta(t)$ is sufficiently similar to some of the frequent patterns, it should be on the hyperplane spanned by $\{\mathbf{u}_i | i = 1, \dots, l\}$. Otherwise, $\beta(t)$ would be directed outside of the hyperplane.

To quantitatively evaluate how far $\beta(t)$ is from the hyperplane, let us define a matrix U_l as

$$U_l = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_l].$$

Using this matrix, the normalized projection of $\beta(t)$ onto the hyperplane is given by

$$\alpha(t) \equiv \frac{U_l^T \beta(t)}{\|U_l^T \beta(t)\|}.$$

¹While it is called ‘‘spectrum analysis,’’ we should emphasize that it has nothing to do with the classical Fourier analysis.

Now we can define the change-point score as

$$(2.4) \quad z(t) \equiv 1 - \alpha(t)^T \beta(t).$$

By definition, this quantity is limited to the range from zero to 1. It is small when there is little change compared to the past patterns and large when the present pattern is quite different from the past patterns.

3 Singular spectrum transformation

3.1 Definition. As discussed, the change-point score can be defined at arbitrary t by calculating representative patterns for both the trajectory and test matrices. This can be viewed also as a transformation from an original time-series \mathcal{T} to a new time-series \mathcal{T}_c , i.e.

$$\mathcal{T} \rightarrow \mathcal{T}_c(w, l, g, m, n).$$

We define this transformation as the singular spectrum transformation (SST). As expressed in the parenthesis, there are five major parameters in SST. This transformation defines a nonlinear transformation in that it does not satisfy the principle of superposition. Hereafter, the integrated area of the transformed time-series is assumed to be normalized to one. Under this condition, the transformed time-series is interpreted as the probability density that some change occurs at time t .

The occurrence of a change-point should be independent of any apparent variety such as discrete, continuous, noisy, oscillatory, etc. Thus, one may think of the new time-series $\mathcal{T}_c(w, l, g, m, n)$ as the signs of causality hidden behind the apparent variation of the original time series: If the similarity between a pair of variables is high for a set of SST series, then some dependency between them is strongly suggested. SST can be a powerful tool to discover hidden dependencies among variables.

3.2 Example. An example of SST is shown in Fig. 1. The time series (a) was generated using three linear functions with slopes of $1/300$, 0 , and $-1/200$. The other time-series (c) was generated using a sine function $x(t) = \sin(2\pi t/\lambda)$, for $\lambda = \sqrt{80}$, $\sqrt{120}$, and $\sqrt{70}$. In (c), we also added random fluctuations to the amplitude and the periods of up to $\pm 7.5\%$ and $\pm 0.5\%$, respectively, to simulate fluctuations in realistic observations. For both data sets, the change points are located at $t = 150$ and 300 . The results of SST in Figs. 1 (b) and (d) was calculated with $w = -g = m = n = 20$ and $l = 3$. In spite of the apparent differences in the original data, we see that SST strikingly reveals the similarities without any ad hoc tuning for individual time series. It is evident that existing methods such as differentiation [5] and wavelet-based approaches [8] fail to detect the change points if a common parameter set is used for both sets.

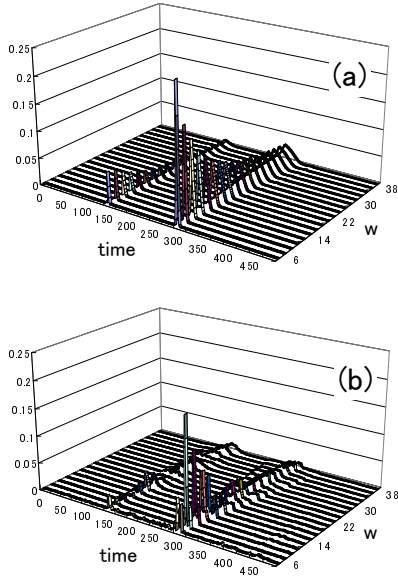


Figure 3: The dependence of SST on w for (a) the linear function and for (b) the oscillatory function shown in Fig. 1 (a) and (c), respectively.

The dependence on w is of particular interest in SST. We calculated SST as a function of w under $w = -g = m = n$ and $l = 3$. The results are shown in Fig. 3. It is surprising that the essential features remain unchanged over a very wide range of w , $6 \lesssim w \lesssim 40$, while the widths of the major features become broader as w increases. This robustness is quite suitable for heterogeneous systems.

4 Experiment

4.1 Data set. The goal of this experiment is to identify the pair of variables that correlates the most in terms of causality, without using any prior knowledge of the variables. The data set used in this section was generated by a specialized simulator for the power train control module of a vehicle, and was taken for one minute with a sampling interval of 0.1 sec. It includes fuel flow rate (x_1), engaged gear (x_2), vehicle speed (x_3), engine RPM (x_4), and manifold absolute pressure (x_5). Figure 4 (a) shows all five of these time-series. For visibility, the signals from x_1 to x_4 are shifted vertically in the figure.

4.2 Comparison between raw and SST time-series. SST was done with the parameters $w = m = n = -g = 25$ (2.5 sec) and $l=2$ for the five time series. Since SVD is not invariant with respect to translation of the origin of the column vectors in the matrix, we

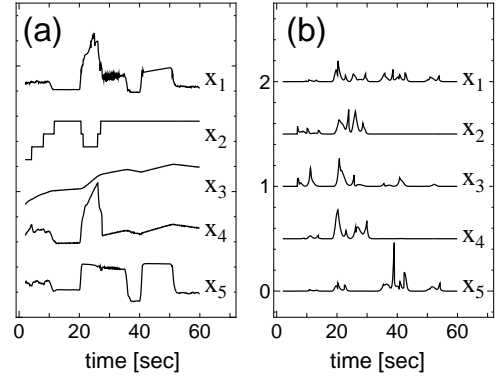


Figure 4: (a) Time-series data from an automobile and (b) the resulting SST series.

standardized the time series so that each of the averages is three times the standard deviation. The result is shown in Fig. 4 (b).

By comparing (a) with (b), we see that each feature in Fig. 4 (b) corresponds to a change in the original data. Interestingly, the SST series of x_2 and x_4 exhibit some similarity in terms of the synchronization of the change-points, in spite of the fact that they seem to behave totally differently in the original series. Similarly, x_1 and x_3 seem to have some in common in Fig. 4 (b) while the original data are quite different. This result demonstrates that SST can make the variables of different types be comparable with each other. In other words, SST converts a heterogeneous system into a different “homogeneous” system.

4.3 Visualization via MDS. To compare the inter-dependencies of the variables, we used the classical solution of multidimensional scaling (MDS) [10]. For the definition of the distance matrices, we took the L_1 and L_2 distances for SST and the raw time series, respectively. Each of the time series was normalized in advance so that $\int dt x(t)^2 = 1$ or $\int dt z(t) = 1$ holds. To remove the unwanted effects of noisy fluctuations of the signals, we performed Gaussian convolution with the standard deviation of 1.5 seconds before computing the distance matrix for SST.

The results of MDS are shown in Fig. 5. Since the definition of the distance metrics are not common in the raw and SST cases, only the relative locations within each plot are meaningful. In Fig. 5 (a), the variables x_1 , x_4 , and x_5 can be attributed to one cluster. We see that they are actually similar in shape in Fig. 4 (a). Similarly, the variables x_2 and x_3 form the other cluster due to the similarity in their increasing trends in Fig. 4 (a).

On the other hand, the two clusters collapse in Fig. 5 (b). Specifically, the closest pair is x_2 and x_4 . This is very interesting because they have totally different trends in the original sequences shown in Fig. 4 (a). This result is due to the synchronizations of the change points in both data sets. In reality, the variables x_2 and x_4 are the engaged gear and the engine RPM, respectively. The close dependency of x_2 and x_4 corresponds to “the value of engine RPM increased after shifting to a lower gear.” It is worth noting that we could discover a part of the causal relationships without using any prior knowledge. This result demonstrates that SST can reveal the signs of causality hidden deep inside of the heterogeneous correlated systems.

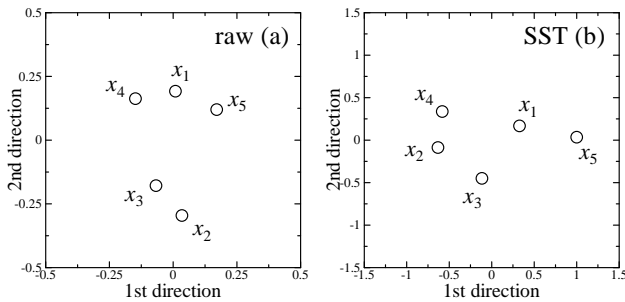


Figure 5: Two dimensional MDS plot for (a) raw data, and (b) SST data.

5 Related work and our contribution

The problem of change detection has been studied for a long time, and various methods such as CUSUM (cumulated summation) [1], wavelet analysis [8], inflection point search [5], and Gaussian mixtures [13] have been proposed. These existing methods, however, are not applicable to our task without using ad hoc tuning for individual signals. Similarly, time-series correlation methods based on these techniques in a few application domains [5, 4, 12] are inapplicable to our task.

Moskvina-Zhigljavsky [11] used the singular spectrum analysis technique [3] for change detection, based on SVD of the Hankel matrix. Mathematically, SVD can be performed for almost any kind of matrix. Thus, the method can be applicable to various sorts of time series without any ad hoc tuning. Our contribution is to have defined the problem of knowledge discovery from heterogeneous dynamic systems and to have proved that their method is one of the most suitable solutions for this problem. Theoretically, our contribution is to have adopted a dimensionless definition of the score, and to have given an algorithm that is pseudo-invariant with respect to time inversion. In other words, our algorithm is invariant with respect to $t \rightarrow -t$ for $l = 1$ and $m = n$.

Acknowledgements

The authors thank W. Nathaniel Mills III for fruitful discussions. T.I. thanks Akihiro Inokuchi for providing valuable information on stream mining.

References

- [1] M. Basseville and I. Nikiforov. *Detection of Abrupt Changes*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.
- [2] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proc. the Fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998.
- [3] M. Ghil, M. R. Allen, M. D. Dettinger, K. Ide, D. Kondrashov, M. E. Mann, A. W. Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Yiou. Advanced spectral methods for climatic time series. *Reviews of Geophysics*, 40:1–41, 2002.
- [4] H. Guo, J. Crossman, Y. Murphey, and M. Coleman. Automotive signal diagnostics using wavelets and machine learning. *IEEE Trans. Vehicular Technology*, 49:1650–1662, 2000.
- [5] S. Hirano and S. Tsumoto. Mining similar temporal patterns in long time-series data and its application to medicine. In *Proc. 2002 IEEE International Conference on Data Mining*, pp. 219–226, 2002.
- [6] T. Idé and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *Proc. the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 440–449, 2004.
- [7] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.
- [8] S. Kadambe and G. Boudreaux-Bartels. Application of the wavelet transform for pitch detection of speech signals. *IEEE Trans. Information Theory*, 38:917–924, 1992.
- [9] E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Proc. IEEE International Conference on Data Mining*. IEEE, 2003.
- [10] K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. Academic Press, 1980.
- [11] V. Moskvina and A. Zhigljavsky. An algorithm based on singular spectrum analysis for change-point detection. *Communications in Statistics—Simulation and Computation*, 32(4):319–352, 2003.
- [12] M. Thottan and C. Ji. Anomaly detection in IP networks. *IEEE Trans. Signal Processing*, 51(8):2191–2204, 2003.
- [13] K. Yamanishi and J. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proc. the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 676–681, 2002.

Building Decision Trees on Records Linked through Key References

Ke Wang*

Yabo Xu†

Philip S. Yu‡

Rong She§

Abstract

We consider the classification problem where the data is given by a collection of tables related by a hierarchical structure of key references and class labels contained in the root table. Each parent table represents a many-to-many relationship type among its child tables. Such data are frequently found in relational databases, data warehouses, XML data, and biological databases. One solution is joining all tables into a universal table based on the recorded relationships, but it suffers from a significant blowup caused by many-to-many relationships. Another solution is treating the problem as relational learning, at the cost of increased complexity and degraded performance. We propose a novel method that builds exactly the same decision tree classifier as built from the joined table, but not the blowup required in the traditional approach.

1 Introduction

Classification has been identified as an important problem in data mining. A *training set* consists of examples with the class label. Each example represents a real world entity described by several attributes. The objective is to build a *classifier* for determining the class label of unclassified records. This problem has a wide range of applications, including fraud detection, finding buyers, medical diagnosis. Most works focus on identically structured records stored in a single table. However, real life data are stored in differently structured tables that are semantically linked via different types of relationships, as illustrated by the following example.

EXAMPLE 1.1. Suppose that we like to classify the class label “Student satisfaction” in the table T(teaches), where T contains many-to-many “teaches”

relationships among professors P (100 records), students S (10,000 records) and courses C (100 records), by containing their *primary keys* as *foreign keys*. Each record in T references *exactly one* record from each of P, S and C tables. This problem is conceptually equivalent to the classification on the joined table by substituting foreign key references with actual records. However, the joined table suffers from a significant blowup due to the many-to-many “teaches” relationship: if each student takes 4 courses, the joined table would contain 40,000 records, in which each S record is replicated 4 times, and each C and P record is replicated 400 times. ■

In this example, the central table contains the class labels and many-to-many relationships among other tables through simultaneous foreign key references to them. We call the classification problem for such databases the *star join classification*. Relational databases designed by the ER model [3] have this structure, so are the star databases widely adopted in the OLAP environment [2], such as the TPC-H benchmark [10]. Foreign key references are also widely used in the DTD of XML documents in the form of nested Element structures, and in biological databases that cross-reference multiple sources.

The above example illustrates two points. First, the star join classification has an equivalent single table representation, namely, the joined table, thus, a classic solution based on the joined table. Recognizing this fact would enable the performance (i.e., accuracy) guarantee of classic solutions. Second, the many-to-many relationships cause a blowup in the joined table and a significant overhead to the subsequent classifier construction. Thus, obtaining a classic solution by applying a classic method to the joined table is not scalable.

We propose a scalable decision tree algorithm for the star join classification. Our insight is that the operation at each decision tree node only depends on the class frequency of attribute values, not the availability of the joined table. Thus, we can propagate the class label to all tables T , and at a decision tree

*Simon Fraser University, wangk@cs.sfu.ca

†Simon Fraser University, yxu@cs.sfu.ca

‡IBM T. J. Watson Research Center, psyu@us.ibm.com

§Simon Fraser University, rshe@cs.sfu.ca

node we split each table T *individually* to get exactly the same set of records at each child node as if the joined table was split. In this method, for each record in T , all its occurrences in the joined table are now represented by a single occurrence plus the class count matrix representing the aggregated class count of those occurrences. Therefore, we are able to compute the same split attribute as using the joined table. The strategy for efficiency is *counting* occurrences instead of *duplicating* records.

In Section 2, we define the problem and review related works. In Section 3 we present our approach. Finally, we conclude the paper.

2 Problems and Related Works

2.1 Star join classification

A *star database* [2] is a rooted tree of tables. T is the *fact table* (the parent) of a *dimension table* A (a child) if T contains the primary key $A.ID$ of A as a foreign key. In this paper, we consider the classification where the class label is contained in the root table. Each example consists of the records involved in a relationship in the root table. In other words, each example is represented by a record in the joined table obtained using foreign key references. There is an one-to-one correspondence between the records in the root table and the records in the joined table.

DEFINITION 2.1. (STAR JOIN CLASSIFICATION)

Given a star database in which the root table contains the class label, build a classification model with a cost measured by the size of the star database (not by the size of the joined table). ■

2.2 Decision tree based classification

We consider the decision tree based classification. A classic description of decision tree construction can be found in [1, 7]. The decision tree is built in two phases. In the *top-down construction phase*, shown in Algorithm 1, the table is recursively partitioned until each partition consists of entirely or dominantly examples from one class. At each node n of the decision tree, there are two steps:

Find the split attribute (Line 4). This step selects the split attribute and criterion to maximize the skewness of class distribution. The essential information needed is the count matrix, or termed as *AVC sets* (for Attribute-Value-Class) [5]. The AVC set for an attribute Att at a node n , denoted $AVC_n(Att)$, contains the class frequency for each distinct value v in Att . For

Algorithm 1 Top-down decision tree construction

BuildTree(n)

Input: node n

Output: decision tree rooted at node n

```

1: if the stopping criterion is met then
2:   return;
3: end if;
4:  $Crit$  = the split criterion at  $n$ ;
5:  $T$  = the database at  $n$ ;
6: create child nodes  $n_1$  and  $n_2$ ;
7: SPLIT( $n, T, Crit$ );
8: BuildTree( $n_1$ );
9: BuildTree( $n_2$ );

```

a numerical attribute, $AVC_n(Att)$ is sorted in the order of values of Att . As in [9, 5], we consider the binary split criterion where the table at a node n is split between two child nodes n_1 and n_2 . The detail can be found in [9, 5].

Partition the database (Line 7). This step splits the table at the parent node n between the child nodes n_1 and n_2 according to the split criterion. It reads the table at n and writes the tables at n_1 and n_2 to disk. In the same scan, it also collects $AVC_{n_i}(Att)$. $AVC_{n_i}(Att)$ has a size proportional to the number of distinct values in Att , not the number of records. For the depth-first construction of the decision tree, the memory is only required to hold the AVC sets for the two child nodes being created.

In the *bottom-up pruning phase*, some subtrees or branches are pruned to ensure that the model does not over-fit the training set. This phase uses only the class frequency at leaf nodes, not data records. Since our algorithm produces the same class frequency as in the decision tree built from the joined table, this phase makes no difference to our algorithm.

2.3 Related work

CrossMine [11] presented a scalable relational learning. We consider the star join classification, which has a classic solution based on the joined table, therefore, the performance guarantee of the well researched decision tree classification. [11] propagates the IDs of target records along the search path to avoid expensive joins. We propagate the class label, instead of IDs of target records. The former has a size equal to the number of distinct classes, whereas the latter does not have a pre-determined size. We search the split attribute from all tables, whereas [11] restricts the search of predicates to “active” tables related to the rule. We consider disk-resident databases whereas [11] considers

Fact Table T

T_ID	A_ID	B_ID	Att1	Att2	C1	C2
0	0	0	10	Blue	1	0
1	0	1	534	Red	0	1
2	1	0	43	Blue	1	0
3	2	0	62	Black	0	1

A_ID	Att3	Att4	C1	C2
0	54	Large	1	1
1	23	Large	1	0
2	43	Small	0	1

Dimension Table A

B_ID	Att5	Att6	Att7	C1	C2
0	X	42	L	2	1
1	Y	42	M	0	1

Dimension Table B

Figure 1. The running example

in-memory databases.

Scalable decision tree construction for a single table has been studied in the database field [5, 4, 6, 8, 9]. The focus of these works is scaling up the construction for databases stored on disk. These methods, no matter how scalable, suffer from the initial blowup of the joined table. This remark applies to the sampling approach [4] where the full joined table needs to be examined to build the *exact* decision tree. Sampling methods that do not examine the full joined table are not guaranteed to produce the exact decision tree.

3 Our Approach

Our algorithm, *Star-DT*, has two main ideas that make it scalable. Before the decision tree is constructed, it propagates the class label from the root table to all tables so that no join is needed to get the class label afterwards. At each decision tree node, it selects the split attribute by evaluating attributes within their own tables without join, and splits individual tables, instead of the joined table, to ensure that each table contains exactly the same information over its attributes as contained in the joined table. Below, we explain these ideas in details.

3.1 Propagating the class label

Before constructing the decision tree, we first propagate the class information to all tables. For each table, the new columns C_1, \dots, C_k , one for each class C_i , store the class count. For the root table, $C_i = 1$ if the record has the class C_i , and $C_i = 0$ otherwise. Recursively,

we propagate C_i from a fact table T to a dimension table A as follows: C_i for each record $A_ID = j$ in A is equal to $SUM(C_i)$ over the records in T that have the foreign key value $A_ID = j$. The records with the all-zero class count are removed from A . This propagation ensures that, for each record t in a table, C_i is equal to the sum of C_i of the records in the joined table that agree with t on the attributes of t .

EXAMPLE 3.1. (THE RUNNING EXAMPLE) In Figure 1, A and B are dimension tables of T . A_ID and B_ID are primary keys of A and B and foreign keys in T . $Att1$, $Att3$, $Att6$ are numerical attributes and others are categorical attributes. B contains the new columns C_1 and C_2 . For the record with the primary key value $B_ID = 0$, $C_1 = 2$ and $C_2 = 1$ because three records in T reference this record, two having the class C_1 and one having the class C_2 . ■

Subsequently, at each decision tree node, we find the split attribute and criterion by computing AVC sets using the class count matrix C_1, \dots, C_k in each table, and split each table by “propagating” the split criterion. We explain these steps below.

3.2 Finding the split attribute

Let $Join(n)$ denote the partition at a decision tree node n of the joined table. Note that we will not construct $Join(n)$. Let $Star(n)$ denote the partition at n of the star database. The following theorem shows that each table T in $Star(n)$ is a projection of $Join(n)$ with duplicates represented by a single record and an aggregated class count matrix. Therefore, if we compute the AVC sets using C_1, \dots, C_k in the table T , we have exactly the same result as computed using $Join(n)$. No join is needed.

THEOREM 3.1. Consider a table T in $Star(n)$. Let $Schema(T)$ be the set of attributes in T (excluding the new class columns). T is equal to

```
SELECT Schema(T), SUM(C1), ..., SUM(Ck)
FROM Join(n)
GROUP BY Schema(T) ■
```

The proof of Theorem 3.1 for the decision tree root follows from our class propagation. For other nodes, the proof follows from our splitting of $Star(n)$ among child nodes.

3.3 Splitting a table

Suppose that a node n is split into two child nodes n_1 and n_2 . We like to obtain $Star(n_i)$ by splitting

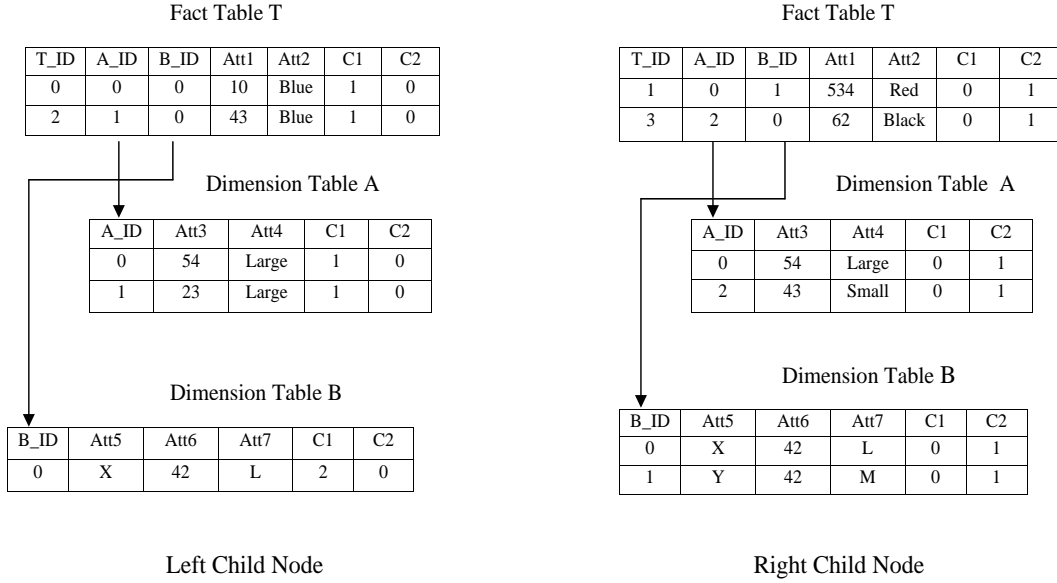


Figure 2. Splitting tables by $Att1 < 52.5$

every table in $Star(n)$ between the two child nodes. While splitting a table, the AVC sets at the child nodes are collected in the same scan. First, we split the “seed” table containing the split attribute as usual. Recursively, we propagate the splitting to dimension tables and fact tables based on foreign key references. The propagation assumes that $Star(n)$ is stored on disk.

3.3.1 The fact-to-dimension propagation Suppose that we know how to split a table T in $Star(n)$. To propagate the splitting to a dimension table A of T , while splitting T we “index” the splitting of foreign key values on A_ID as described below.

DEFINITION 3.1. (F2D INDEX) Consider a decision tree node n . Let A and T be tables in $Star(n)$ and A be a dimension table of T . Suppose that we know how to split T . For each record $A_ID = i$ in A and each child node c of n , the *Fact-to-Dimension index* ($F2D$ index) for A contains an entry denoted $F2D_A(i, c)$. $F2D_A(i, c)$ stores the aggregated class count matrix over all records in T that reference the record $A_ID = i$ and split to the child node c . ■

To create the $F2D_A$ index, while splitting T , if a record t splits to a child node c , we increment $F2D_A(i, c)$ by $\langle t[C_1], \dots, t[C_k] \rangle$, where i is the foreign key value on A_ID in t . Intuitively, $F2D_A(i, c)$ gives the portion of the class count matrix in the record $A_ID = i$ that splits to the child node c . We use this information to split the records in A .

EXAMPLE 3.2. Consider Figure 1 again. Suppose that T is split by the split criterion $Att1 < 52.5$, the midpoint between 43 and 62. Figure 2 shows the split T . Consider splitting the record $B_ID = 0$ in B . In T , the records $T_ID = 0$ and $T_ID = 2$ reference the record $B_ID = 0$ and split to the left child node. So, the record $B_ID = 0$ splits to the left child node with the aggregated class count matrix of the records $T_ID = 0$ and $T_ID = 2$, i.e., $\langle 2, 0 \rangle$ (for C_1, C_2 in that order). Similarly, the record $B_ID = 0$ splits to the right child node with the class count matrix $\langle 0, 1 \rangle$. The effect is that the record $B_ID = 0$ splits its class count matrix $\langle 2, 1 \rangle$ between the two child nodes. ■

The F2D method. Assume that $F2D_A$ is available (thus, the fact table of A was split). We split A as follows. For each record r in A with the primary key value i , if $F2D_A(i, c)$ is not all-zero for a child node c :

- r splits to the child node c with the class count matrix C_1, \dots, C_k given by $F2D_A(i, c)$;
- for a (unsplit) dimension table B of A (if any), increment $F2D_B(j, c)$ by $F2D_A(i, c)$, where j is the foreign key value on B_ID in r .

3.3.2 The dimension-to-fact propagation To propagate the splitting from a dimension table A to its fact table T , while splitting A we index the splitting of its primary key values as described below.

DEFINITION 3.2. (D2F INDEX) Consider a decision tree node n . Let A be a dimension table in $Star(n)$.

Suppose that we know how to split A . For each record $A_ID = i$ in A , the *Dimension-to-Fact index* ($D2F$ index) for A contains an entry denoted $D2F_A(i)$. $D2F_A(i)$ stores the child node to which the record $A_ID = i$ in A splits. ■

To create the $D2F_A$ index, while splitting A , for each record that has the primary key value i and splits to a child node c , set $D2F_A(i) = c$. Intuitively, $D2F_A(i)$ contains the child node for all the records in the fact table of A that contain the foreign key value $A_ID = i$.

The D2F method. Assume that $D2F_A$ is available (thus, A has the fact table). We split the fact table T of A as follows. For each record t in T having the foreign key value i on A_ID and the primary key value i' , let c be the child node given by $D2F_A(i)$:

- t splits to the child node c ;
- if T has an unsplit dimension table B : increment $F2D_B(j, c)$ by $\langle t[C_1], \dots, t[C_k] \rangle$, where j is the foreign key value on B_ID in t ,
- if T has an unsplit fact table (only if T is not the root table), set $D2F_T(i')$ to c .

EXAMPLE 3.3. Suppose that the split criterion at the root node is $Att3 < 33$, i.e., the midpoint of 23 and 43 in A . First, we split A using the split criterion, and build $D2F_A$. Thus, the record $A_ID = 1$ splits to the left child node, and the records $A_ID = 0$ and $A_ID = 2$ split to the right child node. And, $D2F_A(0) = D2F_A(2) = Right$ and $D2F_A(1) = Left$.

Next, we split T using the $D2F_A$ method, and build $F2D_B$: for each record t in T , we look up $D2F_A$ by $t[A_ID]$ to find the child node c for t , and increment $F2D_B(t[B_ID], c)$ by $\langle t[C_1], \dots, t[C_k] \rangle$. Thus, the third record in T splits to the left child node because $t[A_ID] = 1$ and the other three records in T split to the right child node. One can verify that $F2D_B(0, Left) = \langle 1, 0 \rangle$, $F2D_B(1, Left) = \langle 0, 0 \rangle$, $F2D_B(0, Right) = \langle 1, 1 \rangle$, and $F2D_B(1, Right) = \langle 0, 1 \rangle$.

Finally, we split B using the $F2D_B$ method. We omit this part because it is similar to Example 3.2. ■

It is worth noting that for both $F2D_A$ and $D2F_A$, A must be a dimension table. This implies that we never create an index that has a size proportional to the cardinality of the root table. Therefore, even if we keep all indexes (at a decision tree node) in memory, they still use less space than the hash table for joining attribute lists in Sprint [9].

3.4 Analysis

The dominating work at each iteration is propagating the splitting among the tables T in $Star(n)$ for splitting the current decision tree node n . This work essentially performs the matching as required for computing $Join(n)$. However, it does not materialize the join result, but only computes the class count matrix for each distinct record in the projection $\prod_T(Join(n))$ (Theorem 3.1). Thus, while $Join(n)$ duplicates each record in T as many times as it occurs in $Join(n)$, as is typically the case because each fact table represents a many-to-many relationship type among its dimension tables, $Star(n)$ simply keeps a count for each class label to represent the duplicates. This counting strategy speeds up the data scan at each decision tree node. Due to the space limitation, we have to report experimental results elsewhere.

References

- [1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Wadsworth: Belmont, 1984.
- [2] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1):65–74, 1997.
- [3] P. P.-S. Chen. The entity relationship model - towards a unified view of data. *ACM TODS*, 1(1):9–36, March 1976.
- [4] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-Y. Loh. Boat - optimistic decision tree construction. In *SIGMOD*, 1999.
- [5] J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest - a framework for fast decision tree construction of large datasets. In *VLDB*, 1998.
- [6] M. Mehta, R. Agrawal, and J. Rissanen. Sliq: a fast scalable classifier for data mining. In *EDBT*, 1996.
- [7] R. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [8] R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and pruning. In *VLDB*, 1998.
- [9] J. Shafer, R. Agrawal, and M. Mehta. Sprint: A scalable parallel classifier for data mining. In *VLDB*, 1996.
- [10] TPC. Tpc benchmark h standard specification. In <http://www.tpc.org/tpch/spec/tpch2.0.0.pdf>.
- [11] X. Yi, J. Han, J. Yang, and P. Yu. Crossmine: efficient classification across multiple database relations. In *ICDE*, 2004.

Efficient Allocation of Marketing Resources using Dynamic Programming

Giuliano Tirenni*
tir@zurich.ibm.com

Abderrahim Labbi
abl@zurich.ibm.com

André Elisseeff
ael@zurich.ibm.com

Cèsar Berrospi
ceb@zurich.ibm.com

Abstract

In this paper we address the following question: how to estimate a Markov Decision Process modeling the dynamics of customer relationships. Once the model is estimated, we discuss how to efficiently allocate marketing resources and instruments in order to maximize the long-term value generated by customers in a given future time horizon using dynamic programming. Our methodology allows us both to predict and to optimize the future value generated by customers. We show our approach using a case study involving a major European airline.

1 Introduction

In the last years there has been an increasing interest in the allocation of marketing resources both in the marketing (e.g. [17], [6], [19]) and in the data mining (e.g. [16], [13], [4]) communities. There is common agreement that marketing initiatives should be evaluated by measuring their impact on the customer lifetime value [9], i.e. the long-term value generated by a relationship with a customer.

In this paper we discuss how to model the dynamic relationships between the customers and the company using Markov Decision Processes [15] and how to allocate efficiently the marketing budget by finding marketing actions that maximize the long-term value generated by the customers.

We model the customer behavior in time taking into account the marketing actions executed by the company. Customers are segmented into a finite number of states, then the transition probabilities from one state to another and the expected rewards generated when applying a given marketing action in a state are estimated from the transactional data. Once all the states, the transition probabilities, and the expected rewards are known, it is possible to find the marketing policy which maximizes the expected long-term value generated by the relationship with the customers. A *marketing policy* is a mapping from customer states to marketing actions.

The use of dynamic programming techniques to

maximize customer future long-term value and the concept of Markov Decision Processes itself originated from the catalog industry in the 1950s [8]. A main issue which has not been addressed in the literature is the estimation of robust Markov Decision Processes modeling the customer relationship and the long-term effects of marketing actions. To the best of our knowledge, with the exception of [18] where customer states are built using a supervised clustering algorithm, all the models found in the literature (e.g. [1], [5], [14], [3]) assume a given state representation *ad hoc*, without providing any theoretical justification. Most of the models are based on the recency, the frequency, and the monetary value (RFM) segmentation. While the RFM segmentation is very popular in the marketing practice [11], there is not a theoretical motivation that justifies its use in modeling customer dynamics. As discussed in [18], the issue of estimating robust Markov Decision Processes is relevant because a non-reliable model can lead to a non-optimal policy, which in some cases could even perform worse than the historical, i.e. the current, policy.

The remainder of the paper is organized as follows. In section 2 we briefly review Markov Decision Processes and dynamic programming. In sections 3 and 4 we describe how to estimate robust Markov Decision Processes from the customer transactional data. In section 5 we describe a case study. Finally, the conclusions follow in section 6.

2 Markov Decision Processes

A Markov Decision Process (MDP) [15] can be defined as a set of decision epochs $\mathcal{T} \subset \mathcal{N}$, a finite set of states S , a finite set of actions A , a transition probability $p_t(s'|s, a)$ modeling the probability of moving from state $s \in S$ to state $s' \in S$ if action $a \in A$ is applied at decision epoch $t \in \mathcal{T}$, and a reward function $r_t(s, a)$ modeling the expected reward obtained in state $s \in S$ if action $a \in A$ is applied at decision epoch $t \in \mathcal{T}$.

If the transition probabilities and the rewards do not depend on the decision epoch, the process is said to be *stationary*. We consider stationary Markov Decision Processes when modeling the dynamics of customer behavior.

*Computer Science Department, IBM Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland.

A deterministic¹ *policy* π defines, for each decision epoch $t \in \mathcal{T}$, a mapping from states to actions. The expected *value* of state s at time step i , given a policy π , and a finite horizon of length T , is defined as

$$(2.1) \quad \nu_T^\pi(s_i) = \mathbb{E}_{s_i}^\pi \left[\sum_{t=i}^{T-1} r_t(s_t, a_t) + r_T(s_T) \right],$$

r_t , s_t , and a_t are, respectively, the expected reward, the state, and the action executed at time step t . $r_T(s_T)$ is the terminal reward obtained at the last epoch T which depends only on the state s_T .

The optimal policy is defined as the policy maximizing the long-term expected value (2.1) in each state and can be found using backward induction [15].

3 Estimation of the MDP

We assume that customers are segmented into different *states* and that customer transactional data is available. For each customer, the transactional data consists of a sequence of events. Each event is defined by a triple composed of a state s , an action a , and a reward r . The next event is defined by the triple s', a', r' where s' is the state resulting from applying a to s and so on. Each customer has an associated sequence of events defined as episode.

Given transactional data D , the state and action spaces are obtained by considering respectively all the states and all the actions that appear in D . In order to estimate the transition probabilities we can simply use the maximum likelihood estimator:

$$(3.2) \quad p(s'|s, a) = \frac{\#(s'|s, a)}{\#(s, a)}$$

where $\#(s'|s, a)$ is the total number of transitions from s to s' if action a is applied and $\#(s, a)$ is the total number of actions a applied to state s . If the quantity $\#(s, a)$ is zero, then the above equation is not defined. Moreover if the quantity $\#(s'|s, a) = 0$ then $p(s'|s, a) = 0$. As we are estimating the transition probabilities from a limited sample of data, we can assume that the absence of particular transitions does not necessarily imply that the real probabilities are undefined or null. To address these two issues we adopt a *Bayesian* approach incorporating the prior transition probability $\hat{p}_{s'|s, a}$ into equation 3.2. This leads to the following estimator [12]:

$$(3.3) \quad p(s'|s, a) = \frac{\#(s'|s, a) + m_1 \hat{p}_{s'|s, a}}{\#(s, a) + m_1}.$$

¹Deterministic policies are a special case of stochastic policies, which associate to each state a probability distribution on the actions.

The quantity m_1 can be interpreted as the number of instances following the prior probability that are injected into the data set D , m_1 acts therefore as a weight defining the relative importance of the prior probability with respect to the probability estimated from the data.

There are two possibilities to model the prior transition probability $\hat{p}_{s'|s, a}$: a) adopt a *state-driven* approach, emphasizing the role of the origin state, or b) adopt an *action-driven* approach, emphasizing the role of the action.

In the first case, the prior is modeled as follows

$$\hat{p}_{s'|s, a} = p(s'|s) = \frac{\#(s'|s) + m_2 \hat{p}_{s'|s}}{\#(s) + m_2},$$

where $\#(s)$ is the number of times state s appears in the data set D and $\#(s'|s)$ is the number of times a transition from state s to state s' is observed. Finally, the nested prior $\hat{p}_{s'|s}$ is estimated as follows

$$\hat{p}_{s'|s} = p(s') = \frac{\#(s') + m_3 \hat{p}}{\sum_{s \in S} \#(s) + m_3}.$$

If we set equiprobable state probabilities, then the prior \hat{p} becomes

$$\hat{p} = \frac{1}{|S|},$$

where $|S|$ is the cardinality, i.e. the number of different states, of the set S . If we set $m_3 = |S|$, we obtain the Laplace estimator:

$$\hat{p}_{s'|s} = p(s') = \frac{\#(s') + 1}{\sum_{s \in S} \#(s) + |S|},$$

we use this estimator to model the prior $\hat{p}_{s'|s}$.

In the action-driven approach, the prior is modeled as follows

$$\hat{p}_{s'|s, a} = p(s'|a) = \frac{\#(s'|a) + m_2 \hat{p}_{s'|a}}{\#(a) + m_2},$$

where $\#(a)$ is the number of times action a appears in the data set D and $\#(s'|a)$ is the number of times a transition to state s' is due to the execution of action a . The nested prior is equal to $\hat{p}_{s'|a} = p(s')$.

The expected reward $r(s, a)$ if action a is applied to state s can be estimated as follows

$$(3.4) \quad r(s, a) = \frac{\sum_{(s, a) \in D} r(s, a)}{\#(s, a)},$$

where $r(s, a)$ is the reward observed in the data when action a is applied to state s . If the quantity $\#(s, a)$ is zero, because action a has never been applied to state s ,

we can estimate the expected reward considering either a state-driven approach or an action-driven approach. The state-driven estimate is

$$r(s, a) = r(s) = \frac{\sum_{(s) \in D} r(s, a)}{\#(s)},$$

while the action-driven estimate is

$$r(s, a) = r(a) = \frac{\sum_{(a) \in D} r(s, a)}{\#(a)}.$$

3.1 Estimation of the historical policy We define as *historical policy* the policy used by the company when targeting customers. The knowledge of the historical policy allows us to simulate the customer dynamics. We learn from the available transactional data a stochastic policy, assuming that it is *stationary*², and estimate the probability of executing action a in state s as follows:

$$\pi(a|s) = \frac{\#(a|s) + m\hat{\pi}_{a|s}}{\#(s) + m},$$

where $\#(a|s)$ are the number of events (i.e. transactions) with state s and action a and $\#(s)$ are the number of events with state s . The quantity $\hat{\pi}_{a|s}$ is the prior probability of executing action a in state s . We define the prior probability using the Laplace estimator as follows

$$\hat{\pi}_{a|s} = p(a) = \frac{\#(a) + 1}{\sum_{a \in A} \#(a) + |A|},$$

where $\#(a)$ is the total number of actions of type a in all the events and $|A|$ is the number of available actions.

4 Model selection

Several parameters influence the estimation of a Markov Decision Process modeling customer relationships, such as the segmentation used to define the customer states, the state-driven or action-driven approach to estimate the transition probabilities and the rewards, the length of the time horizon in the training data, etc. The definition of the state space will probably have the highest impact on the performance of the model because the transition probabilities, the rewards, and the historical policy are estimated based on the states encountered in the training data.

Assuming there is a finite list of possible models $\mathcal{M}_1, \dots, \mathcal{M}_n$, corresponding to different choices of the parameters, we use *cross-validation* [7] to select the model with the best accuracy in predicting the long-term value generated by customers.

²This assumption is realistic if the company is not using any multistage decision model to target the customer base.

Feature	Description
rectrip	elapsed time since last purchase
frequitrip3	number of transactions in the last 3 months
frequitrip12	number of transactions in the last 12 months
value3	value generated in the last 3 months
value3camp	value generated from responding to campaigns in the last 3 months
value12	value generated in the last 12 months
value12camp	value generated from responding to campaigns in the last 12 months
miles3	miles flown in the last 3 months
miles12	miles flown in the last 12 months
longevity	number of days since first transaction

Figure 1: Customer features.

5 Case study

We apply our methodology to the customers of a major European airline and estimate the Markov Decision Process modeling the relationships with the company in order to efficiently allocate the marketing resources.

5.1 Data We use transactional data of customers for a period length of two years (2002, 2003). We do not segment customers into a finite number of states, as this is part of the model definition. At this stage we represent each customer with the set of numeric features defined in Figure 1.

After removing the outliers³, we randomly extract 10,000 customers. The customers are assigned randomly to two independent sets of size 5,000: the *validation set* used in the model selection phase and the *evaluation set* used to predict the future long-term value by simulating the historical and the optimal policy.

5.2 Defining customer states In order to build a MDP we need to discretize the high-dimensional feature space into a finite number of states. We propose a list of segmentation criteria which can be divided into two categories: a) business-based segments, and b) statistical-based segments.

The business-based segments are obtained by using recency, frequency, and monetary value. Each segmentation criterion can have several parameters. The segments are defined as follows.

- *RFM*(n) Scores the customers according to recency, frequency, and monetary value, then divides the so ranked customers into n segments of equal

³We removed marketing actions which have been applied very rarely and customers whose cumulative value is larger than the 99% percentile.

size.

- $ABC(a, b, c)$ Scores the customers according to a value feature, e.g. *value3*, and generates three segments by assigning the first $a\%$ to segment *A*, the next $b\%$ to segment *B*, and the remaining $c\%$ to segment *C*.
- $VD(a, b, c)$ The Value-Defectors (VD) segmentation performs $ABC(a, b, c)$ segmentation both on a value feature and on a loyalty index⁴, 9 segments are then obtained (e.g. *AA*, *AB*, *AC*, etc.).
- $RV(a, b, c)$ Recency-Value performs $ABC(a, b, c)$ segmentation both on a recency feature and on a value feature, there are 9 possible segments.

The following statistical-based segments use all the features defined in Figure 1.

- $Trees(n)$ Regression Trees [2] are used for supervised clustering. A regression tree is trained on an independent data set to predict the immediate reward of each customer. The leaves of the tree correspond to the segments. The parameter n indicates the number of leaves in the training set obtained by acting on the parameters of the algorithm. This is a supervised clustering technique as the leaves are built specifically to minimize the standard deviation of the reward.
- $SOM(n, m)$ Self-Organizing Maps [10] allow us to map the high-dimensional feature space into a two-dimensional rectangular $n \times m$ grid. The features are normalized and Euclidian distance is used.
- $K - means(n)$ K-means clustering [7] finds n clusters which are the centers minimizing the total within-cluster variance. The features are normalized and Euclidian distance is used.

5.3 Model selection We perform model selection using cross-validation. Each model defined in Figure 2 is tested in the case of state-driven and action-driven approach using the validation set.

We compare the mean absolute errors of each model using the state-driven and action-driven approaches. As shown in Figure 3, the state-driven approach outperforms the action-driven approach for each segmentation.

Therefore we focus on the state-driven approach in the remainder of the paper. The best model is the regression tree with 10 leaves (#19), followed by ABC (#10), the regression tree (#20), and ABC (#9).

⁴The loyalty index is a function of the frequency and the longevity of a customers and has been used by IBM in different CRM projects as a measure of the loyalty of customers.

#	Segment	Used features
1	RFM (10)	rectrip,freqtrip3,value3
2	RFM (20)	rectrip,freqtrip3,value3
3	RFM (30)	rectrip,freqtrip3,value3
4	RFM (10)	rectrip,freqtrip12,value12
5	RFM (20)	rectrip,freqtrip12,value12
6	RFM (30)	rectrip,freqtrip12,value12
7	ABC (10,10,80)	value3
8	ABC (10,20,70)	value3
9	ABC (10,10,80)	value12
10	ABC (10,20,70)	value12
11	VD (10,10,80)	value3,freqtrip3,rectrip
12	VD (10,20,70)	value3,freqtrip3,rectrip
13	VD (10,10,80)	value12,freqtrip12,rectrip
14	VD (10,20,70)	value12,freqtrip12,rectrip
15	RV (10,10,80)	value3,rectrip
16	RV (10,20,70)	value3,rectrip
17	RV (10,10,80)	value12,rectrip
18	RV (10,20,70)	value12,rectrip
19	Trees (10)	all
20	Trees (29)	all
21	K-means (10)	all
22	K-means (15)	all
23	K-means (20)	all
24	K-means (30)	all
25	SOM (3 × 3)	all
26	SOM (3 × 5)	all
27	SOM (4 × 5)	all

Figure 2: List of the used segmentations.

5.4 Simulation and optimization of customer dynamics We use an independent data set (evaluation set) to train the MDP using the tree-based segmentation #19. Then we estimate the historical policy and simulate the Markov Decision Process in order to predict the future value obtained by following the historical policy.

In order to maximize the long-term value, we apply backward induction [15] and find the optimal policy⁵. Figure 4 compares the expected value per state obtained using the historical and the optimal marketing policy, the time horizon is set to 12 months.

6 Conclusions

In this paper we provide a rigorous methodology for the estimation of Markov Decision Processes modeling the dynamic relationship between the customers and the company. Although the use of Markov Decision Processes and dynamic programming is not new in the

⁵For confidentiality reasons we cannot show the historical policy and the optimal policy in terms of marketing actions undertaken by the company.

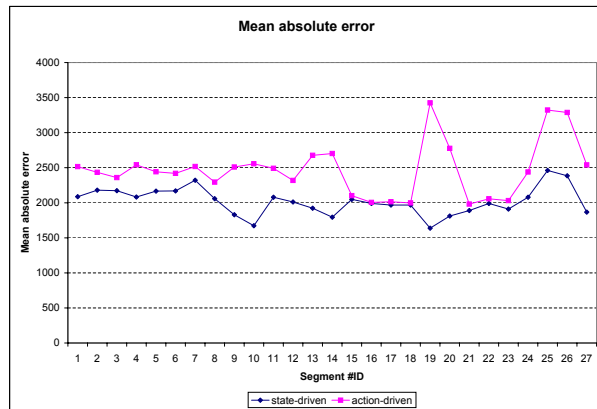


Figure 3: Comparison of the mean absolute error based on the prediction of the value generated in 12 months, for action-driven and state-driven approach.

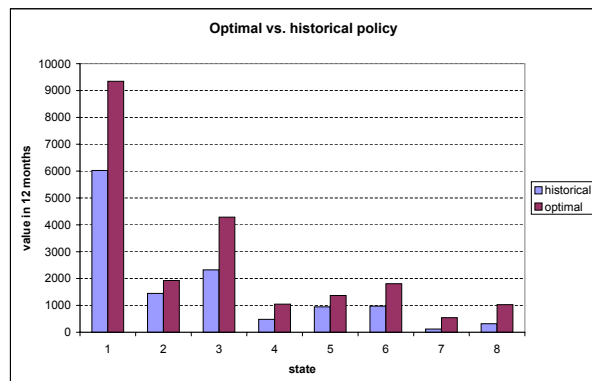


Figure 4: Comparison of the long-term value generated in 12 months when using the optimal and the historical policy.

literature, the issue of estimating robust models from customer relationship transactional data has not been addressed so far.

Using cross-validation for model selection, we are able to build a reliable Markov Decision Process taking into account the impact that the uncertainty in the parameters of the model has on the performance measure, i.e. the mean absolute error on the long-term value prediction. Our methodology enables efficient allocation of marketing resources by optimizing the long-term return on investment using the optimal marketing policy.

References

[1] G. R. Bitran and S. V. Mondschein. Mailing decisions

in the catalog sales industry. *Management Science*, 42(9):1364–1381, September 1996.

- [2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Int. Group, California, USA, 1984.
- [3] W. K. Ching, M. K. Ng, K. K. Wong, and E. Altman. Customer lifetime value: stochastic optimization approach. *Journal of the Operational Research Society*, 55:860–868, 2004.
- [4] J. H. Drew, D. R. Mani, A. L. Betz, and P. Datta. Targeting customers with statistical and data-mining techniques. *Journal of Service Research*, 3(3):205–219, 2001.
- [5] F. Gönül and M. Z. Shi. Optimal mailing of catalogs: A new methodology using estimable structural dynamic programming models. *Management Science*, 44(9):1249–1262, September 1998.
- [6] S. Gupta, D. R. Lehmann, and J. A. Stuart. Valuing Customers. *Journal of Marketing Research*, 41(1):7–18, 2004.
- [7] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.
- [8] R. A. Howard. Comments on the origin and application of Markov Decision Processes. *Operations Research*, 50(1):100–102, 2002.
- [9] D. Jain and S. S. Singh. Customer lifetime value research in marketing: A review and future directions. *Journal of Interactive Marketing*, 16(2):34–46, 2002.
- [10] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 2 edition, 1997.
- [11] P. Kotler. *Marketing Management*. Prentice-Hall, 10 edition, 2000.
- [12] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [13] E. Pednault, N. Abe, B. Zadrozny, H. Wang, W. Fan, and C. Apte. Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2002.
- [14] P. Pfeifer and R. Carraway. Modeling customer relationships as Markov Chains. *Journal of Interactive Marketing*, 14(2):43–55, 2000.
- [15] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.
- [16] S. Rosset, E. Neumann, U. Eick, N. Vatnik, and S. Idan. Lifetime Value Models for Decision Support. *Data Mining and Knowledge Discovery Journal*, 7:321–339, 2003.
- [17] R. Rust, K. Lemon, and V. Zeithalm. Return on marketing: Using customer equity to focus marketing strategy. *Journal of Marketing*, 68:109–127, 2004.
- [18] D. I. Simester, P. Sun, and J. N. Tsitsiklis. Dynamic catalog mailing policies. Submitted, March 2003; revised May 2004.
- [19] G. Tirenni. *Allocation of Marketing Resources to Optimize Customer Equity*. PhD thesis, University of St. Gallen, Switzerland, 2004.

Near-Neighbor Search in Pattern Distance Spaces

Haixun Wang Chang-Shing Perng Philip S. Yu
 IBM Thomas J. Watson Research Center, Hawthorne, NY 10532
 {haixun,perng,psyu}@us.ibm.com

Abstract

In this paper, we study the near-neighbor problem based on *pattern similarity*, a new type of similarity which conventional distance metrics such as L_p norm cannot model effectively. The problem, however, is important to many applications. For example, in DNA microarray analysis, the expression levels of two closely related genes may rise and fall under different external conditions or at different time. Although the magnitude of their expression levels may not be close, the patterns they exhibit over the time or under different conditions can be very similar. In this paper, we measure the distance between two objects by pattern similarity, i.e., whether the two objects exhibit a synchronous pattern of rise and fall under different conditions. We then present an efficient algorithm for near-neighbor search based on pattern similarity, and we perform tests on several real and synthetic data sets to show its effectiveness.

1 Introduction

The efficiency of near neighbor search to a large extent depends on the distance function in use [3]. More importantly, the distance function also determines the meaning of similarity and the meaning of the near-neighbor search. In this paper, we address a new type of similarity for near-neighbor search.

DNA microarray analysis Finding near neighbors based on subspace pattern similarity is important to many applications [1, 9, 7, 8]. Table 1 shows a small portion of the Yeast expression data, where entry d_{ij} represents the expression level of gene i under condition j (or at time j). Investigations show that more often than not, several genes contribute to a disease, which motivates researchers to identify genes whose expression levels rise and fall synchronously under different conditions or over a period of time, that is, whether they exhibit fluctuation of a similar shape when conditions change.

As shown in Table 1, the expression levels of three genes, VPS8, CYS3, and EFB1, rise and fall coherently under three different external conditions t_1, t_3 and t_5 . We can also measure the expression levels of genes at fixed time intervals. In this case, assume t_1, t_2, \dots, t_5 in Table 1

	t_1	t_2	t_3	t_4	t_5	t_6	\dots
VPS8	401	281	120	275	298	210	
SSA1	401	292	109	580	238	289	
SP07	<u>228</u>	<u>290</u>	285	<u>148</u>	224	231	
EFB1	318	280	37	277	215	99	
MDM10	10	10	<u>266</u>	<u>328</u>	101	<u>186</u>	
CYS3	322	288	41	278	219	231	
DEP1	317	<u>272</u>	<u>334</u>	232	<u>192</u>	110	
NTG1	329	296	33	274	228	129	
\vdots							

Table 1: Expression data of Yeast genes

represent arbitrarily spaced points on the time axis. We find the expression levels of three genes, SP07, MDM10, and DEP1, manifest a coherent pattern with fixed time shift.

Given a new gene, biologists are interested in finding every gene whose expression levels under a certain set of conditions rise and fall coherently with those of the new gene, as such discovery may reveal connections in gene regulatory networks [1]. Clearly, this pattern similarity cannot be captured by distance functions such as Euclidean even if they are applied in the related subspaces.

In this paper, we extend the concept of near-neighbor to the above situation. We say genes VPS8, CYS3, and EFB1 are near-neighbors in the subspace defined by conditions $\{t_1, t_3, t_5\}$, and the time series expression levels of genes SP07, MDM10, and DEP1 are near-neighbors from time t_1, t_2 and t_3 .

An even more interesting and challenging type of near-neighbor query is the following. We are given the expression levels of a new gene. This new gene might be related to any gene in the database as long as both of them exhibit a pattern in some subspace or at some time offset. The dimensionality of the subspace, or the length of the time period, is often an indicator of the degree of their closeness, that is, the more columns the pattern spans, the closer the relation between the two genes.

In this paper we focus on pattern based similarity described above. Traditional distance functions, such as the Euclidean norm, cannot measure pattern similarity. We pro-

pose an efficient method to perform near-neighbor search by pattern similarity. Traditional spatial access methods for nearest neighbor search cannot be used for pattern similarity matching because they depend on metric distance functions satisfying the triangular inequality. Experiments show our method is effective and efficient, and it outperforms alternative algorithms (based on an adaptation of the R-Tree index) by an order of magnitude.

2 NN Search by Subspace Pattern Similarity

In this section, we propose an index structure called *P-Index* (pattern index) to support fast pattern matching and near-neighbor search. A similar structure was used to support sequence matching [8].

2.1 An Overview We represent each object $u \in \mathcal{D}$ as a sequence of (column, value) pairs. For each suffix of the sequence, we derive a *base-column aligned suffix* and insert it into a trie.

The trie supports matching of patterns defined on a column set composed of a continuous sequence of columns, $\mathcal{S} = \{c_i, c_{i+1}, \dots, c_{i+k}\}$. To find patterns in any subspace efficiently, we create P-index on top of the trie.

The trie is employed as an intermediary structure to facilitate the building of the P-index. It embodies a compact index to all the distinct, non-empty, base-column aligned objects in \mathcal{D} . Various approaches to build tries or suffix trees in linear time have been developed. Ukkonen [6], for instance, developed a linear-time, on-line suffix tree construction algorithm. We do not address the details of building tries in this paper.

2.2 The Trie Structure We first introduce a sequential representation of the data, and then use an example to demonstrate the process of constructing the P-index.

Let \mathcal{D} be a dataset in multidimensional space $\mathcal{A} = \{c_1, c_2, \dots, c_n\}$. Unless the dimensions are already in an ordered domain (for example, time), we create an arbitrary order among the dimensions, that is, we assume $c_1 \prec c_2 \prec \dots \prec c_n$ is a total order. We represent each object $u \in \mathcal{D}$ as a sequence of (column, value) pairs, that is:

$$u = (c_1, u_1), (c_2, u_2), \dots, (c_n, u_n)$$

A suffix of u starting with column c_i is denoted by:

$$(c_i, u_i), (c_{i+1}, u_{i+1}), \dots, (c_n, u_n)$$

where $1 \leq i \leq n$. Using the first column in each suffix as its *base* column, we derive a *base-column aligned suffix* by subtracting the value of the base (first column) from each column value in the suffix. We use $f(u, i)$ to denote u 's base-column aligned suffix that begins with the i th column:

$$(2.1) f(u, i) = (c_i, 0), (c_{i+1}, u_{i+1} - u_i), \dots, (c_n, u_n - u_i)$$

or, if the columns are numerical (e.g. time), we have:

$$(2.2) f(u, i) = (c_{i+1} - c_i, u_{i+1} - u_i), \dots, (c_n - c_i, u_n - u_i)$$

We then insert each base-column aligned suffix $f(u, i)$ into a trie. In the following, we use an example to demonstrate the process.

EXAMPLE 1. Let database \mathcal{D} be composed of the following 2 objects defined in space $\mathcal{A} = \{c_1, c_2, c_3, c_4, c_5\}$.

obj	c_1	c_2	c_3	c_4	c_5
#1	3	0	4	2	0
#2	4	1	5	3	6

We represent each object by a sequence of (column, value) pairs. For instance, object #1 in \mathcal{D} can be represented by

$$(c_1, 3), (c_2, 0), (c_3, 4), (c_4, 2), (c_5, 0)$$

We use the first column in the sequence as its *base* column, and derive a *base-column aligned suffix* by subtracting the value of the base column from each value in the suffix:

$$(c_1, 0), (c_2, -3), (c_3, 1), (c_4, -1), (c_5, -3)$$

We do the same to each suffix (of length ≥ 2) of the object. Table 2 shows all the base-column aligned suffixes derived from the two objects.

$f(u, i)$, where $u \in \{\#1, \#2\}$ and $i = 1, \dots, 4$				
$(c_1, 0)$	$(c_2, -3)$	$(c_3, 1)$	$(c_4, -1)$	$(c_5, -3)$
	$(c_2, 0)$	$(c_3, 4)$	$(c_4, 2)$	$(c_5, 0)$
		$(c_3, 0)$	$(c_4, -2)$	$(c_5, -4)$
			$(c_4, 0)$	$(c_5, -2)$
$(c_1, 0)$	$(c_2, -3)$	$(c_3, 1)$	$(c_4, -1)$	$(c_5, 2)$
	$(c_2, 0)$	$(c_3, 4)$	$(c_4, 2)$	$(c_5, 5)$
		$(c_3, 0)$	$(c_4, -2)$	$(c_5, 1)$
			$(c_4, 0)$	$(c_5, 3)$

Table 2: Sequences and suffixes derived from \mathcal{D}

We insert the base-column aligned suffixes into a trie. Figure 1 demonstrates the insertion of sequence:

$$f(\#1, 1) = (c_1, 0), (c_2, -3), (c_3, 1), (c_4, -1), (c_5, -3)$$

Each leaf node n in the trie maintains an *object list*, L_n . If the insertion of $f(\#1, 1)$ leads to node x , which is under arc $(e, -3)$, we append 1 (object #1), to object list L_x .

2.3 Building P-Index over a Trie The trie enables us to find near-neighbors of a query object $q = (c_1, v_1), \dots, (c_n, v_n)$ in a given subspace S , provided S is defined by a set of *continuous* columns, i.e., $S =$

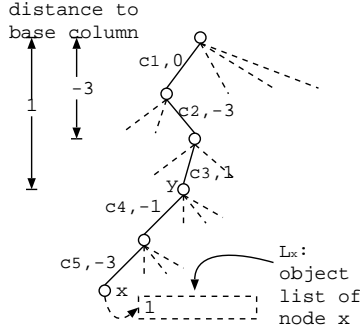


Figure 1: Insertion of sequence $f(u, 1) = (c_1, 0), (c_2, -3), (c_3, 1), (c_4, -1), (c_5, -3)$. The id of the object is appended to x 's object list L_x .

$\{c_i, c_{i+1}, \dots, c_{i+k}\}$. If $\epsilon = 0$, all we need to do is following path $(c_i, 0), (c_{i+1}, v_{i+1} - v_i), \dots, (c_{i+k}, v_{i+k} - v_i)$ in the trie, and when we reach a certain node x at the end of the path, we return objects in the object lists of those leaf nodes that are x 's descendants (including x if x is a leaf node). If $\epsilon > 0$, we may need to traverse multiple paths at each level.

Input: T : a trie built on \mathcal{D}
 S : a subspace defined by a continuous column set $\{c_i, c_{i+1}, \dots, c_k\}$
 $q = (c_1, v_1), \dots, (c_n, v_n)$: a query object
 ϵ : pattern threshold

Output: near-neighbors of q in subspace S

$n \leftarrow \text{root of } T$;
 $\text{search}(n, S)$;

Function $\text{search}(x, S)$
if $S = \emptyset$ **then**
 output the descendants of x ;
else
 assume $S = \{c_j, c_{j+1}, \dots, c_k\}$;
 for x 's child node y under edge labeled (c_j, v)
 where $v \in [(v_j - v_i) - \epsilon, (v_j - v_i) + \epsilon]$ **do**
 $\text{search}(y, \{c_{j+1}, \dots, c_k\})$;

Algorithm 1: NN Search in a given subspace defined by a continuous column set

Algorithm 1 is a formal description of the above process. It finds all objects whose value difference between column c_j and c_i is within region $(v_j - v_i) \pm \epsilon$, where $j = i, i+1, \dots, i+k$. Hence the correctness of the algorithm follows.

Algorithm 1, however, only finds near-neighbors in a given subspace defined by a set of continuous columns. In the algorithm, at each step j , we can only go directly to node under edge (c_{j+1}, \cdot) . To find a descendent node under edge (c_k, \cdot) , where $k > j$, requires us to traverse the subtree

under the current node, which is time-consuming. The P-index, described below, allows us to 'jump' directly to nodes under (c_k, \cdot) , where $k > j$. This enables us to efficiently find near-neighbors in any given subspace, and furthermore, near-neighbors in any subspace whose dimensionality is larger than a given threshold requires additional index structures.

We use the following steps to build the P-index on top of a trie. First, after all sequences are inserted, we assign to each node x a pair of labels, $\langle n_x, s_x \rangle$, where n_x is the prefix-order of node x in the trie (starting from 0, which is assigned to the root node), and s_x is the number of x 's descendent nodes. Next, we create pattern-distance links for each $(col, dist)$ pair, where $col \in \mathcal{A}$, $dist \in \{-\xi + 1, \dots, \xi - 1\}$, and ξ is the number of distinct column values¹. The links are constructed by a depth-first walk of the suffix trie. When we encounter a node x under arc $(col, dist)$, we append x 's label $\langle n_x, s_x \rangle$ to the pattern-distance link for pair $(col, dist)$. Thus, a pattern link is composed of nodes that have the same distance from their base columns (root node).

The labeling scheme and the pattern-distance links have the following property.

THEOREM 2.1. *P-Index Property*

1. if node x and y are labeled $\langle n_x, s_x \rangle$ and $\langle n_y, s_y \rangle$ respectively, and $n_x < n_y \leq n_x + s_x$, then y is a descendent node of x ;
2. nodes in any pattern-distance links are ordered by their prefix-order number; and
3. for any node x , x 's descendants in any pattern-distance link are contiguous in that link.

Proof. 1) and 2) are due to the labeling scheme which is based on depth-first traversal. For 3), note that if nodes u, \dots, v, \dots, w are in a pattern-distance link (in that order), and u, v are descendants of x , we have $n_x < n_u < n_v < n_w \leq n_x + s_x$, which means v is also a descendent of x .

The above properties enable us to use range queries to find descendants of a given node in a given pattern-distance link.

Algorithm 2 summarizes the index construction procedure. The P-Index is composed of two major parts: i) arrays of $\langle n_x, s_x \rangle$ pairs for pattern-distance links; and ii) leaf nodes' object lists.

The time complexity of building the P-index is $O(|\mathcal{D}| |\mathcal{A}|)$. The Ukkonen algorithm [6] builds suffix tree in linear time. The construction of the trie for pattern-distance indexing is less time consuming because the length of the indexed subsequences is constrained by $|\mathcal{A}|$. Thus, it can be constructed by a brute-force algorithm [4] in linear time.

¹ ξ is also regarded as a discretization parameter, or the number of bins the numerical values are discretized into.

Input: \mathcal{D} : objects in multi-dimensional space \mathcal{A}

Output: P-Index of \mathcal{D}

```

for each  $u \in \mathcal{D}$  do
  insert  $f(u, i), 1 \leq i < |\mathcal{A}|$  into a trie;
  (Eq 2.1)
for each node  $x$  encountered in a depth-first traversal
  of the trie do
  label node  $x$  by  $\langle n_x, s_x \rangle$ ;
  let  $(c, d)$  be the arc that points to  $x$ ;
  append  $\langle n_x, s_x \rangle$  to pattern-distance link  $(c, d)$ ;

```

Algorithm 2: Index Construction

The space taken by the P-Index is linearly proportional to the data size. Since each node appears once and only once in the pattern links, the total number of entries in Part I equals the total number of nodes in the trie, or $O(|\mathcal{D}||\mathcal{A}|^2)$ in the worst case (if none of the nodes are shared by any subsequences). On the other hand, there are exactly $|\mathcal{D}|(|\mathcal{A}| - 1)$ object ids stored in Part II. Thus, the space is linearly proportional to the data size $|\mathcal{D}|$.

2.4 Near-Neighbor Search in a Given Subspace In this section, we find near-neighbors in a given subspace using the P-index. For instance, assume we have a query object q :

$$q = (a, 3), (c, 7), (e, 2)$$

Using the first column of q as the base column, we get²:

$$(a, 0), (c, 4), (e, -1)$$

We start with the pattern link of $(a, 0)$, which contains only one node. Let us assume its label is $\langle 20, 180 \rangle$, meaning sequences starting with column a are indexed by nodes from 20 to 200. Next, we consult pattern-distance link $(c, 4)$, which contains all the c nodes that are 4 units away from their base column (root node). However, we are only interested in those nodes that are descendents of $(a, 0)$. According to the property of pattern-distance links, those descendents are contiguous in the pattern-distance link and their prefix-order numbers are inside range $[20, 200]$. Since the nodes in the buffer are organized in ascending order of their prefix-order numbers, the search is carried out as a range query in log time. Suppose we find three nodes, $u = \langle 42, 9 \rangle$, $v = \langle 88, 11 \rangle$, and $w = \langle 102, 18 \rangle$, in that range. Then, we consult the next pattern-distance link $(e, -1)$ and repeat the process for each of the three nodes. Assume node x is a descendent of node u , node y a descendent of node v , and no nodes in pattern link of $(e, -1)$ are descendents of node w .

²If the columns are numerical, we get $(c - a, 4), (e - a, -1)$.

We now have matched all the columns in S , and the object lists of nodes x, y and their descendents contain offsets for the query.

Algorithm 3 outlines the searching of near-neighbors in a given subspace (defined by an arbitrary set of columns). Here, we have demonstrated the purpose of having the pattern-distance links. It enables us to 'jump' directly to the next relevant column in the given subspace, while in traditional suffix trie we can only follow the tree branches. As a result, the tree structure is not needed in the searching, since the pattern-distance links already contain the complete information for pattern matching.

Input: q : a query object, S : a given subspace
 ϵ : pattern threshold

Output: q 's near-neighbors in subspace S

let $(c_1, v_1), \dots, (c_{|S|}, v_{|S|})$ be q 's projection on S ;
 $x \leftarrow$ the node under arc $(c_1, 0)$;
 $search(x, 2)$;

```

Function  $search(x, i)$ 
if  $i \leq |S|$  then
  for pattern link  $I$  of  $(c_i, v)$ , where  $v \in [v_i - v_1 - \epsilon, v_i - v_1 + \epsilon]$  do
    /* perform a binary search on  $I$  */
    for all node  $r \in I$  and  $n_r \in [n_x, n_x + s_x]$  do
       $search(r, i + 1)$ ;
    end
  end
else
  output objects in  $L_x, x = v_s, \dots, v_m$ 
end

```

Algorithm 3: Pattern Matching

3 Experiments

We tested P-Index with both synthetic and real life data setson a Linux machine with a 700 MHz CPU and 256 MB main memory.

Gene Expression Data Gene expression data are being generated by DNA chips and other micro-array techniques. The data set is presented as a matrix. Each row corresponds to a gene and each column represents a condition under which the gene is developed. Each entry represents the relative abundance of the mRNA of a gene under a specific condition. The yeast micro-array is a $2,884 \times 17$ matrix (2,884 genes under 17 conditions) [5]. The mouse cDNA array is a $10,934 \times 49$ matrix (10,934 genes under 49 conditions) [2] and it is pre-processed in the same way.

Synthetic Data We generate random integers from a uniform distribution in the range of 1 to ξ . Let $|\mathcal{D}|$ be

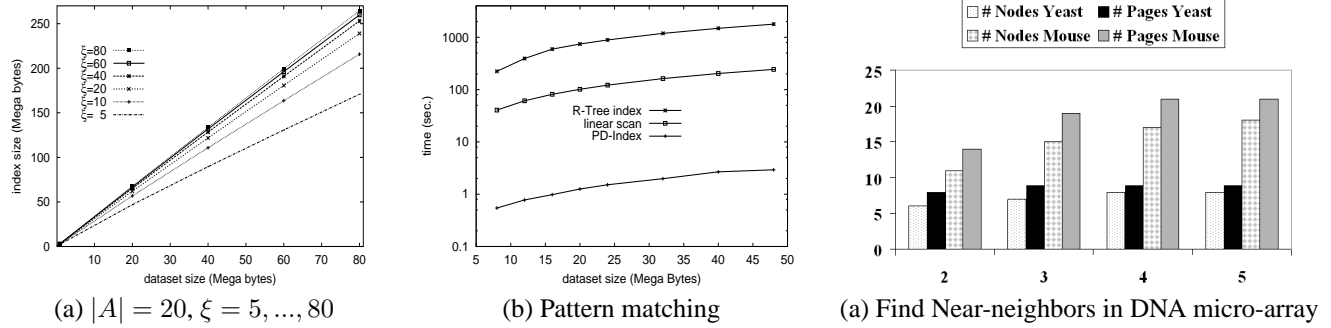


Figure 2: Performance.

the number of objects in the dataset and $|A|$ the number of dimensions. The total data size is $4|D||A|$ bytes.

3.1 Space Analysis The space requirement of the pattern-distance index is linearly proportional to the data size (Figure 2). In Figure 2(a), we fix the dimensionality of the data at 20 and change ξ , the discretization granularity, from 5 to 80. It shows that ξ has little impact on the index size when the data size is small. When the data size increases, the growth of the trie slows down as each trie node is shared by more objects (this is more obvious for smaller ξ in Figure 2(a)).

3.2 Time Analysis We compare the algorithms presented in this paper with two alternative approaches, i) brute force linear scan, and ii) R-Tree family indices. The linear scan approach for near-neighbor search is straightforward to implement. The R-Tree, however, indexes values not patterns. To support queries based on pattern similarity, we create an extra dimension $c_{ij} = c_i - c_j$ for every two dimensions c_i and c_j .

The query time presented in Figure 2(b) indicates that P-Index scales much better than the two alternative approaches for pattern matching in given subspaces. The comparisons are carried out on synthetic datasets of dimensionality $|A| = 40$ and discretization level $\xi = 20$. Each time, a subspace is designated by randomly selecting 4 dimensions, and random query objects are generated in the subspace.

To further analyze the impact of different query forms on the performance, we base our comparisons on number of disk accesses. First, we ask random queries against yeast and mouse DNA micro-array data in subspaces of dimensionality ranging from 2 to 5. The selected dimensions are evenly separated. For instance, we select dimension set $\{c_1, c_{13}, c_{25}, c_{37}, c_{49}\}$ in a mouse cDNA array that has a total of 49 conditions. Figure 2(c) shows the average number of node accesses and disk accesses. Since P-Index offers increased selectivity for longer queries, it is robust as the dimensionality of the given subspace becomes larger.

4 Conclusion

We identify the need of finding near-neighbors under subspace pattern similarity, a new type of similarity not captured by Euclidean, Manhattan, etc., but essential to a wide range of applications, including DNA microarray analysis. Two objects are similar if they manifest a coherent pattern of rise and fall in an arbitrary subspace, or over a certain time period with time shifting. We propose P-Index, which maps objects to sequences and index them using a tree structure. Experimental results show that P-Index achieves orders of magnitude speedup over alternative algorithms based on naive indexing and linear scan.

References

- [1] Y. Cheng and G. Church. Biclustering of expression data. In *Proc. of 8th International Conference on Intelligent System for Molecular Biology*, 2000.
- [2] R. Miki et al. Delineating developmental and metabolic pathways in vivo by expression profiling using the riken set of 18,816 full-length enriched mouse cDNA arrays. In *Proceedings of National Academy of Sciences*, 98, pages 2199–2204, 2001.
- [3] Piotr Indyk. On approximate nearest neighbors in non-euclidean spaces. In *IEEE Symposium on Foundations of Computer Science*, pages 148–155, 1998.
- [4] E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, April 1976.
- [5] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Church. Yeast micro data set. In <http://arep.med.harvard.edu/biclustering/yeast.matrix>, 2000.
- [6] E. Ukkonen. Constructing suffix-trees on-line in linear time. *Algorithms, Software, Architecture: Information Processing*, pages 484–92, 1992.
- [7] Haixun Wang, Fang Chu, Wei Fan, Philip S. Yu, and Jian Pei. A fast algorithm for subspace clustering by pattern similarity. In *16th International Conference on Scientific and Statistical Database Management (SSDBM)*, 2004.
- [8] Haixun Wang, Chang-Shing Perng, Wei Fan, Sanghyun Park, and Philip S. Yu. Indexing weighted sequences in large databases. In *ICDE*, 2003.
- [9] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *SIGMOD*, 2002.

An algorithm for lattice-structured subspace clusters

Haiyun Bian

bianh@ececs.uc.edu

University of Cincinnati, OH 45221

Raj Bhatnagar

raj@ececs.uc.edu

University of Cincinnati, OH 45221

Abstract

Most of the current subspace clustering methods find non-overlapping clusters of similar objects. We present a new subspace clustering algorithm that has two valuable capabilities not available in most of the current methods. First, it finds possibly overlapping subspace clusters; and second, it can discover clusters that preserve some user-specified interesting properties. Our methods build a lattice of these subspace clusters and it further enables discovery of meaningful linkage-chains in terms of objects or attributes among distant clusters. Our algorithm can be applied to graph data, social network data, and biological data to obtain a better understanding of the substructures inherent in the data space.

1 Introduction

Clustering seeks to identify homogeneous groups of objects based on the values of their attributes. We view a data space as a table in which rows correspond to individual data objects and columns correspond to attributes. When the number of attributes (dimensions) becomes large, similarity defined on the whole attribute-set becomes low for almost any subset of objects, which makes finding full-dimensional clusters difficult. Dimension reduction techniques such as principle component analysis make the final interpretation of the clusters very difficult. *Subspace clustering* is a good solution for finding clusters in such data spaces, by defining similarity only on a selected subset of attributes (regional similarity) for a set of clustered objects. Each subspace cluster is a group of similar objects within its own subset of dimensions. Several methods have been proposed recently for discovering interesting subspace-clusters [1, 2, 3, 4, 5, 6].

Most of the existing subspace clustering methods find non-overlapping clusters, that is, each object belongs to at most one subspace cluster. However, finding overlapping subspace clusters is useful and necessary in many applications for the following reasons.

Multi-domain functionality

Each subspace cluster signifies some functional

	a	b	c	d	e	f
a	1	1	1	0	0	1
b	1	1	1	0	0	0
c	1	1	1	1	1	1
d	0	0	1	1	1	1
e	0	0	1	1	1	1
f	1	0	1	1	1	1

Table 1: Pairwise multi-functionality

	a_1	a_2	a_3	a_4	a_5
a	1	1	1	0	0
b	1	1	1	0	0
c	1	1	1	1	1
d	0	0	1	1	1
e	0	0	1	1	1

Table 2: Object-attribute multi-functionality

group of objects in the domain. It is very common that some objects participate in multiple functional groups and hence should belong to multiple subspace clusters. For example, a particular gene may play active role in multiple biological processes. We show two different examples situations here. The first is in the form of pairwise similarity between a set of objects as shown in table 1. The same set of objects form the row and the column labels in the table, as is the case for a graph incidence matrix. An entry of ‘1’ in the table indicates the objects of the pair are similar to each other. The table shows two clearly defined clusters, one with objects $\{a, b, c\}$, and the other with objects $\{c, d, e, f\}$. c is a multi-functional object in the sense that it belongs to more than one subspace cluster.

The second example is in the form of a table of object-attribute pairs as shown in table 2. Clusters in this type of data are defined by the shared object-attribute values. The example shown in the table 2 has two clusters, one with objects $\{a, b, c\}$ in subspace $\{a_1, a_2, a_3\}$, and the other with object $\{c, d, e\}$ in subspace $\{a_3, a_4, a_5\}$. Object c occurs in both the clusters

which are within different subspaces, and attribute a_3 is common between the two subspaces.

Connection between distantly related objects

A good criterion for connection path between two objects can be defined at the cluster level instead of the single object level. the connections among the individual objects. Consider the example in table 1, if we are trying to find a good connection between b and f , two paths satisfy the shortest path criterion, path $b \rightarrow a \rightarrow f$ and path $b \rightarrow c \rightarrow f$. However, since $\{a, b, c\}$ and $\{c, d, e, f\}$ are clusters, the connection $a \rightarrow f$ is more likely to be a case of spurious connectivity, and the preference should go to $b \rightarrow c \rightarrow f$. defined. We show in this paper that the cluster level connections are best captured by the lattice built from all the dense regions found in the data.

Our Contribution We present in this paper a new subspace clustering algorithm that can find overlapping subspace clusters satisfying different constraints. A lattice structure is built from the subspace clusters, and connective relationships among the objects and attributes of a cluster can be revealed by the lattice. Our algorithm is capable of dealing with similarity measures as well as some kinds of discrepancy as the metric for clustering objects together. Also, our algorithm simultaneously clusters on both the objects space and the attributes space.

2 Related Research

Subspace clustering in *CLIQUE* [1] is a density-based method which partitions the data space into non-overlapping rectangular units. A search is performed to find dense units within all possible subspaces. The pruning strategy is based on the fact that “if a k -dimensional region is dense, then all the $k-1$ dimensional regions that it contains must also be dense”. *ENCLUS* [4] follows the density based idea, but uses entropy as a heuristic to prune away uninteresting subspaces. *PROCLUS* [2] is a variation of the k -medoid algorithm, and it returns a partition of the data points into clusters together with the set of dimensions on which each cluster is correlated. Bi-clustering algorithms [5, 6] are proposed to meet the requirements in bioinformatics field where we need to find sets of genes showing similarity under a set of conditions. Greedy search is used in [5] to find the subsets of genes and subsets of conditions for which a measure called *mean square residue score* is minimized, and the algorithm returns non-overlapping bi-clusters. Work presented in *FLOC*[6] is a further improvement on the bi-clustering idea by allowing the finding of multiple bi-clusters simultaneously.

3 Our Approach

3.1 Problem Description A data space \mathcal{DS} is given by a set of attributes $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ and a population of objects $\mathcal{O} = \{o_1, o_2, \dots, o_m\}$. Each o_i has a value for each of the attributes in \mathcal{A} . A subspace cluster c of the data space \mathcal{DS} is defined as $\langle \mathcal{O}_i, \mathcal{A}_i \rangle$, where $\mathcal{O}_i \subseteq \mathcal{O}$ and $\mathcal{A}_i \subseteq \mathcal{A}$. We call \mathcal{O}_i and \mathcal{A}_i the object set and the attribute set of the subspace cluster respectively. The symbol d_{ij} denotes the j^{th} attribute of the i^{th} object. The following definitions, used in our algorithm, are defined on 0/1 binary valued data spaces. **DEFINITION 3.1.** A subspace cluster $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ on 0/1 valued data space is a **dense region** if o_{ij} are ‘1’ for all $o_i \in \mathcal{O}_i$ and $a_j \in \mathcal{A}_i$.

DEFINITION 3.2. A dense region $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ is a **maximal dense region** if all regions that are proper supersets of c are not dense.

A region c_i is said to be proper superset of c_j if either $\mathcal{O}_j \subset \mathcal{O}_i$, or $\mathcal{A}_j \subset \mathcal{A}_i$, or both. That is, a dense region is maximal if and only if adding any object (attribute) will force the enlarged region to contain ‘0’ entries. We have proved that our definition of maximal dense region is equivalent to the **concept**’s definition in formal concept analysis [9]. (The proof is omitted due to space limitation.) So all the dense regions form a lattice structure based on the following order: for two dense regions $c_1 = \langle \mathcal{O}_1, \mathcal{A}_1 \rangle$ and $c_2 = \langle \mathcal{O}_2, \mathcal{A}_2 \rangle$, we define $c_2 \prec c_1$ if $\mathcal{O}_2 \subset \mathcal{O}_1$. c_2 is called a **child** of c_1 , and c_1 is called a **parent** of c_2 . c_1 is called the **cover (direct parent)** of c_2 if $c_2 \prec c_1$, and for all c_i : $c_2 \prec c_i$ ($c_i \neq c_1$ and $c_i \neq c_2$), we have $c_1 \prec c_i$; correspondingly, c_2 is called the **direct child** of c_1 .

The **prime operator** ‘ \prime ’ is defined as: $\mathcal{O}'_i := \{a \in \mathcal{A} | d_{oa} = 1 \text{ for all } o \in \mathcal{O}_i\}$, and $\mathcal{A}'_i := \{o \in \mathcal{O} | d_{oa} = 1 \text{ for all } a \in \mathcal{A}_i\}$. Then a maximal dense region $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ is a pair where $\mathcal{O}'_i = \mathcal{A}_i$ and $\mathcal{A}'_i = \mathcal{O}_i$. We call maximal dense region as dense region in the remaining discussion. The lattice built from dense regions is constructed using the following rules:

- Each node in the lattice is a dense region;
- If node c_1 is a cover for node c_2 , put c_1 above c_2 and add an edge between them;
- The lattice top is $c_{top} = \langle \mathcal{O}'', \mathcal{O}' \rangle$, and lattice bottom is $c_{bottom} = \langle \mathcal{A}', \mathcal{A}'' \rangle$.

We define a **path** in a lattice to be a contiguous sequence of edges between nodes. Two nodes are said to be **connected** if there exists a path between them.

3.2 Algorithm Outline Our method has three main phases. Phase-0 transforms a multi-valued data space to a binary data space. In phase-1, dense regions are found

and the lattice is built. In phase-2, connections are searched through the lattice. Algorithms for building concept lattices find all concepts without any pruning and therefore have a high complexity. Our definition of dense region is mathematically equivalent to a formal concept [9], the semantics of **dense regions** can be used as pruning tool to discover subspaces with different semantics for the clusters. That is, a combination of the binary transformation (phase-0), pruning strategy (phase-1) and search strategy (phase-2) produce clusters satisfying different \mathcal{P} -property. Formally, \mathcal{P} -cluster is defined as the follows:

DEFINITION 3.3. Let $\mathcal{B}(\mathcal{DS})$ denote set of all dense regions in data space \mathcal{DS} . A subspace cluster $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ on data space \mathcal{DS} is called a **\mathcal{P} -cluster** if it satisfies some property \mathcal{P} . Here \mathcal{P} is defined as a function: $\mathcal{P}: \mathcal{B}(\mathcal{DS}) \rightarrow \{0, 1\}$

3.3 Phase-0: To Binary Data For multiple valued data sets, a function F is applied to the original data to convert it into binary valued data. The following are some example F functions and are similar to the ideas of *scale* defined in the formal concept analysis [9], but here they have different semantics interpretations. Choice of F significantly determines the \mathcal{P} property of the clusters found by the algorithm. For similarity-based clusters, we define F as a function that maps each attribute value o_{ij} into a 0/1 vector: $F: o_{ij} \rightarrow \{(0|1)^*\}$

Choice of F from *Equality functions* can find clusters in which objects have same values for the attributes, and when F is from *range functions* we find clusters in which objects have values in certain ranges.

We have used another F' function which can be used to find clusters of objects with discrepant attribute values. We define discrepant F' as functions that map each attribute value pair into 0/1: $F': o_{ij} \times o_{kj} \rightarrow \{0, 1\}$. An example where this may be useful is the situation where negatively correlated genes provide insights into the structures of some biochemical pathways [10].

We define $F'(o_{ij}, o_{kj}) = 1$ if $o_{ij} \neq o_{kj}$ or they are in different ranges for real valued attributes. Clusters found satisfy the \mathcal{P} property that all objects in every cluster are different for all the attributes of the clusters.

3.4 Phase-1: Pruned Lattices Objective of this phase is to build the lattice of dense regions using the binary data space (from phase-0). Many exhaustive algorithms are available to find all concepts (=clusters) in formal contexts [8, 9]. These algorithms have high complexity because the number of possible concepts increases exponentially with the number of objects. Our algorithm is motivated by the concept search algorithms, but uses an additional **Pruning** feature.

We show that the proposed pruning methods can restrict the search space effectively.

3.4.1 Size Pruning We consider the following two types of size constraints: minimum number of objects and minimum number of attributes in a cluster. Algorithm 1 given below finds all dense regions that have at least s objects. It is based on the fact that the object

Algorithm 1

Let L be the list of candidate dense regions, and initially empty;

foreach $a_i \in A$

foreach $S_i = \langle \mathcal{O}_i, \mathcal{A}_i \rangle \in L$

$O \leftarrow a_i' \cap \mathcal{O}_i$

if $|O| > s$ and $O \notin L$

add $L \leftarrow L \cup \langle O, \mathcal{A}_i \cup a_i \rangle$

else if $|O| > n$ and $\langle O, \mathcal{A}_j \rangle \in L$

$\langle O, \mathcal{A}_j \rangle \leftarrow \langle O, \mathcal{A}_j \cup \mathcal{A}_i \cup a_i \rangle$

set \mathcal{O}_i of any dense region c must be intersection of all $(a_i)' \in \mathcal{A}_i$, where $\mathcal{A}_i \subset A$. It finds all the dense regions that have at least s objects, and the lattice structure is built after the dense regions are found.

3.4.2 Effectiveness of Size Pruning We tested the size constraint on congressional voting data set from the UCI repository [11]. This data set includes votes for 435 congressmen on 16 issues. Each attribute has three possible values: 'y', 'n', and '?'. Using algorithm 1 without any size constraint, there are 227032 maximal dense regions (concepts) in this data. Figure 1 shows the number of dense regions found and pruned for different size constraints on the object set. With the increasing threshold on the number of objects, the number of dense regions found decreases significantly and the number of pruned dense regions increases. The most effective region for the size constraint pruning is [0,20].

3.4.3 Semantic Pruning We can also set semantic constraints for pruning, and only those dense regions that satisfy the constraints will be found. Dense regions in binary valued data spaces have simpler semantics: regions of all 1's or regions of all 0's. For multiple valued data spaces, dense regions can have much richer semantics. All combinations of values of different attributes can constitute dense regions, while not all of them maybe semantically meaningful.

3.4.4 Effectiveness of Semantic Pruning We tested the semantic constraints on the same data set. The 16 issues are grouped into five categories, with each category having 4-8 issues. The semantics constraint is set as: a dense region is interesting when all the con-

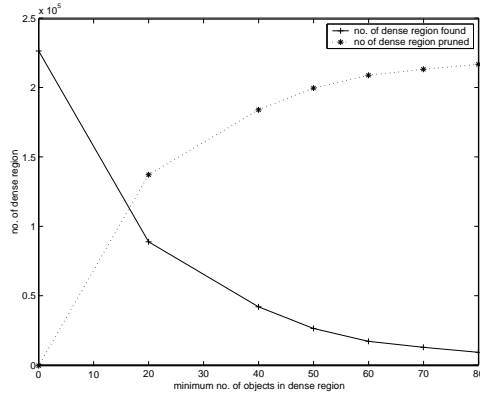


Figure 1: Size Constraint pruning: number of dense regions found and pruned

gressmen in this region agree with each other on at least $\alpha\%$ of the issues in any of the five categories. The results show that the number of dense regions found decreases quadratically with the increase of α , and the number of pruned regions increases quadratically.

3.5 Phase-2: Search for Connections and Combinations All dense regions found are not equally interesting. Paths among nodes of the lattice and mergers of some dense regions (leading to non-dense clusters) may have valuable semantic content. We consider the following two types of cluster shapes. The first type is a symmetric matrix, such as the graph incidence matrix. The second type is a non-symmetric matrix, resulting in a rectangular shape in the dataspace.

3.5.1 Symmetric-reflective Matrix Many of the symmetric-reflective matrices, such as graph incidence matrix and pairwise similarity data, have a commonality that the object set is the same as the attribute set. The lattice of dense regions built from this type of data has a lot of special characteristics which can be used to facilitate the mining process. For clarity purpose, we call graph vertex as vertex, and dense region in the lattice as node.

DEFINITION 3.4. A dense region $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ is called **square dense region** if $|\mathcal{O}_i| = |\mathcal{A}_i|$.

DEFINITION 3.5. A dense region $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ is called **core** if $\mathcal{O}_i = \mathcal{A}_i$.

A core in graph data corresponds to a clique, an important subgraph. They are used to define closely connected objects, such as human communities, gene functional groups.

Observations: All square dense regions $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ in lattice of symmetric and reflective matrices data

that satisfy $\mathcal{O}_i \cap \mathcal{A}_i \neq \phi$ are cores. The lattice of dense regions for symmetric and reflective matrices is symmetric w.r.t the cores. That is, for any path in the lattice starting from a core to the lattice top, there is a corresponding path starting from the same core to the lattice bottom, with every node $c = \langle \mathcal{O}_i, \mathcal{A}_i \rangle$ in the first path having a corresponding node $c = \langle \mathcal{A}_i, \mathcal{O}_i \rangle$ in the second path. So, for a lattice of symmetric and reflective matrix data, only half of the lattice is needed to encode all the connectivity information. This can help us design more efficient algorithms for building the lattices.

Find dense regions of various p -properties:

DEFINITION 3.6. A region is called α -dense if the percentage of 1 entries in the submatrix exceeds some threshold α .

DEFINITION 3.7. For a dense region c , we define $\mathcal{L}_c = \langle \mathcal{O}_{L_c}, \mathcal{A}_{L_c} \rangle$, where \mathcal{O}_{L_c} is the union of the object sets of direct parents of c , and \mathcal{A}_{L_c} is the union of the attribute sets of all the direct parents of c .

Observation For any dense region c , any two objects in \mathcal{O}_{L_c} are at most 1-hop from each other. Here, one hop means one intermediate vertex.

More generally, if we go more than one level up the direct-parent relationships, the maximum distance between any two objects in the resulting combined cluster is at most $(n + m - 1)$ -hops from each other, where n and m are the number of cover operations taken at two branches.

It is inefficient to search for α -dense regions starting from every node. A better way is to start from the cores, and form larger clusters by taking the union of the object sets of the covers of the cores and the resultant α -dense clusters have an upper bound on the maximum hops between any pair of objects within each cluster, and also density maximality.

Connection between cores: The lattice structure reveals important information about connections between cores. We consider the following three types of relationships for a pair of cores, assuming the lattice top $c_{top} = \langle \mathcal{O}, \phi \rangle$.

Type I: If all paths between two cores pass the lattice top, then these cores are disconnected.

Type II: If two cores have at least one common parent, there is at least one multi-functional object for the two cores.

Type III: If two cores are connected by at least one path, but share no parent except c_{top} , then there are some objects outside both the cores that provide the connection between the cores. These

objects can be easily identified by following the *cover* and *direct child* links.

3.5.2 Non-symmetric Matrix Most of the clusters in object-attribute datasets are non-symmetric, that is, they are rectangular shaped submatrices of the data space. So cores do not exist but the square dense regions still exist. A size constraint on both the object space and the attribute space will turn in a much smaller set of dense regions that may act as the cores, which we call q -cores. After the q -cores are identified, similar strategies can be used to find α -dense regions between the q -cores. Connections between the q -cores can also be categorized into similar three types but the search needs to be performed within both halves of the lattice.

4 Algorithm Evaluation

We have designed a synthetic data generator which is an updated version of the one presented in [7] by allowing one object to be in multiple clusters. Our first test used the number of objects as the control variable. Each attribute has a real value in $[0, 100]$ interval and is transformed into a binary equivalent by using a range function (the F function in phase-0) into ten equal length intervals. For a data set with 20 dimensions and five non-overlapping dense regions of five dimensions each, the running time increases linearly with the number of objects. Addition of 10% objects as random noise still discovers all the dense regions with the same time complexity. The same test was repeated with overlapping dense regions. Subspaces overlapped in both, the objects and the attributes. There were four dense regions with two overlapping pairs, each pair sharing two attributes and 40% objects. The running time is larger than that for the non-overlapping dense regions but still increases only linearly.

We tested the scalability of the algorithm as the number of attributes is increased from 20 to 100. All the datasets had 10,000 objects and two overlapping dense regions in 5-dimensional subspaces. The results show that the algorithm time grows quadratically with the number of attributes. We also tested the effect of size constraint on the running time of the algorithm. As intuitively expected, the smaller the size constraint, the karger is the number of clusters found and more time is taken by the algorithm.

5 Conclusions

We have presented a new subspace clustering algorithm that can find possibly overlapping subspace clusters in the data. A lattice is built from all the dense regions found, from which other clusters of interesting properties can be constructed and paths among clusters

can be identified. Our algorithm provides a very good interpretation of the relationships among overlapping subspace clusters. It is the first attempt that defines subspace clusters as combinations of cores in a lattice according to some partial ordering. This provides tremendous amount of control and information about the structural properties of the subspaces that may be constructed with the lattice of subspace cores. Many extensions of this research are possible. Algorithms to find dense regions that satisfy both the object set size constraint and attribute set size constraint can be designed, which may help in making the algorithm in phase-1 more efficient.

References

- [1] Rakesh Agrawal and Johannes Gehrke and Dimitrios Gunopulos and Prabhakar Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*, ACM SIGMOD Proceedings, 1998, pp. 94-105.
- [2] Charu C. Aggarwal and Joel L. Wolf and Philip S. Yu and Cecilia Procopiuc and Jong Soo Park, *Fast algorithms for projected clustering*, SIGMOD Conference Proceedings, 1999, pp. 61-72.
- [3] C. Aggarwal and C. Procopiuc and J. Wolf and P. Yu and J. Park, *A Framework for Finding Projected Clusters in High Dimensional Spaces*, ACM SIGMOD International Conference on Management of Data Proceedings, 1999.
- [4] Chun Hung Cheng and Ada Wai-Chee Fu and Yi Zhang, *Entropy-based Subspace Clustering for Mining Numerical Data*, Knowledge Discovery and Data Mining, 1999, pp. 84-93.
- [5] Yizong Cheng and George M. Church, *Biclustering of Expression Data*, ISMB Proceedings, 2000, pp. 93-103.
- [6] Jiong Yang and Wei Wang and Haixun Wang and Philip S. Yu, *delta-cluster: Capturing Subspace Correlation in a Large Data Set*, ICDE Proceedings, 2002.
- [7] Mphamed Zait and Hammou Messatfa, *A comparative study of clustering methods*, Future Generation Computer Systems, 13 (1997), pp. 149-159.
- [8] Christian Lindig, *Fast concept analysis*, Working with Conceptual Structures - Contributions to ICCS, 2000.
- [9] B. Ganter and R. Wille, *Formal concept analysis: mathematical foundations*, Springer, Heidelberg, 1999.
- [10] Dhillon IS, Marcotte EM, Roshan U, *Diametrical clustering for identifying anti-correlated gene clusters*, Bioinformatics, 19(2003), pp. 1612-1619.
- [11] C.L. Blake and C.J. Merz, = *UCI Repository of machine learning databases* <http://www.ics.uci.edu/~mllearn/MLRepository.html>, University of California, Irvine, Department of ICS, 1998.

CBS: A New Classification Method by Using Sequential Patterns

Vincent S. M. Tseng Chao-Hui Lee

Dept. Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
Email: tsengsm@mail.ncku.edu.tw

Abstract - Data classification is an important topic in data mining field due to the wide applications. A number of related methods have been proposed based on the well-known learning models like decision tree or neural network. However, these kinds of classification methods may not perform well in mining time sequence datasets like time-series gene expression data. In this paper, we propose a new data mining method, namely *Classify-By-Sequence (CBS)*, for classifying large time-series datasets. The main methodology of CBS method is to integrate the sequential pattern mining with the probabilistic induction such that the inherent sequential patterns can be extracted efficiently and the classification task be done more accurately. Meanwhile, CBS method has the merit of simplicity in implementation. Through experimental evaluation, the CBS method is shown to outperform other methods greatly in the classification accuracy.

Keywords: Classification, Sequential Pattern, Time-series data, Data Mining.

1. Introduction

In recent years, many data mining techniques emerged in various research topics like association rules, sequential patterns, clustering and classification [1, 2, 4, 11, 12]. These techniques are also widely used in different application fields. Most of the existing data mining methods are designed for solving some specific problem independently. In the other hand, some few compound methods integrate two or more kinds of data mining techniques to solve complex problems. For example, *Classification By Association rules (CBA)* [8] and other integrated methods like the SPF Classifier in [7] are of this kind. These compound methods can effectively utilize the advantages of each individual mining technique to improve the overall performance in data mining tasks. For example, the CBA [8] method was reported to deliver higher accuracy than traditional classification method like C4.5 [11]. Hence, it is a promising direction to integrate different kinds of data mining methods to form a new methodology for solving complex data mining problems.

Among the rich data mining problems, classification modeling and prediction is an important one due to the wide applications. Numerous classification methods have been proposed [3] [4] [5] [6] [8] [9] [11], including the most popular ones like decision tree, neural network and support vector machine. The goal of a classification method is to build up a model that can relate the label (class) to the values of attributes for the data instances. Hence, the label of a new data instance can be predicted correctly by using the built model. However, few studies explore the issue of using integrated data mining methods to classify datasets with time sequences, e.g., the time-series gene expression datasets.

One of the few related work is the one by Lesh *et al.* [7], which combined the sequential patterns mining and tradition classification method like C4.5 to classify sequence datasets. In this paper, we present a new method named *Classify-By-Sequence (CBS)*, which integrates the sequential pattern mining with the probabilistic induction such that the inherent sequential patterns can be extracted and used for classification efficiently. The experimental results show that the CBS method achieves higher accuracy in classification than other methods.

2. Problem Definition

Consider a large database that stores numerous data records. Each data record is consisted of a data sequence and associated with a class label. The main problem is to build a classifier based on the information in the database such that the class of a new data instance can be predicted correctly. Let D_i represents all time-series data instances belonging to class i . So, the database is represented as $D = \{D_1, D_2, D_3, D_4, \dots, D_N\}$ assuming there are totally N classes in the database. For each class, data set D_i consists of time-series data instances in the form of $\{a_1, a_2, a_3, \dots, a_n\}$, where a_n represents the value at n th time point. To simplify this study, it is assumed that the value at each time point is discretized and transformed into categorical values in advance.

Our goal is to discover the *Classifiable Sequential Patterns (CSP)* rules that can be used as the classification rules for the classifier. The CSP rule is in the form of $SP_i \rightarrow C_m$, where SP_i is a sequence like $a_2 \rightarrow a_3 \rightarrow a_7 \dots$, and

C_m is some class m . We will describe in details how to discover these CSP rules in next section.

3. The CBS Method

With the basic idea of CBS as described previously, we propose two variations of CBS methods, namely CBS_ALL and CBS_CLASS, for discovering the CSP rules. After the CSP rules are discovered, it is easy to do the class prediction by using the induction approach. Hence, we will focus on the procedures of mining the CSP rules in the following.

3.1 CBS_ALL Method

The concept of this method is similar to the sequential patterns mining with the add-in of probabilistic induction. First, we extract all classifiable sequences from the database by using some sequential patterns mining. In addition to finding the frequent sequences, we further give the *classify-score* for these sequences. CBS_ALL algorithm considers the *class support* and *transaction support* at the same time. We adopt an Apriori-like procedure for implementing the support counting task. Figure 1 shows the CBS_ALL algorithm in details. Thus, we can get a number of CSP rules. Each CSP rule contains the classification information and we use all CSP rules to establish a classifier. Figure 2 illustrates the policy of the classifier in using all CSP rules to classify a time-series data correctly.

Notice that this algorithm deals with the categorical time-series data only since it is assumed that all original datum are transformed into categorical values as mentioned in Section 2. In CBS_ALL algorithm, we extract large-1 items as CSP_1 , which are further used to generate SP_2 (candidate large-2 Sequential Patterns). In counting the support, if SP is a subsequence of a transaction sequence data, $SP.seq_sup$ (i.e., support of the sequence) will be incremented. Another important procedure for producing classify-score of a sequences is counting the support of each class. In our design, both SP and CSP are given a *class_sup* array, respectively. In counting sequence support, the class support is counted at the same time. If the SP is with the class label C_x , then $SP.class_sup[x]$ will also be incremented. After the whole dataset has been counted for SP, we prune SP into CSP with their *seq_sup* and *class_sup*[]. In performing the pruning, a rule is reserved if at least one class support is larger than the minimum rule support; otherwise it is eliminated since the sequential pattern distributes in too many classes and is unqualified as a rule. Then, the procedure loops back to the candidate generation procedure. The algorithm generates all CSP iteratively

until no more SP meets the requirements of *min_seq_sup* and *min_class_sup* thresholds.

```

CBS _ ALL (Dataset D, min_seq_s up, min_rule_s up)
{
    CSP1 = {large 1 - items}
    for(i = 2; CSPi ≠ ∅; i++) do
        SPi = gen _ candidateS P(CSPi-1);
        for each data d ∈ D do
            SPs = SPi ∩ subseq( d);
            for each sp ∈ SPs do
                sp.seq_ sup ++;
                sp.class_ sup[d.class ] ++;
            end
        end
        CSPi = {sp/sp ∈ SPi, sp.seq_ sup ≥ min_seq_s up
        and ∃ w let sp.class_ sup[ w] / sp.seq_ sup ≥ min_rule_s up}
    end
    CSP = ∪i CSPi
}

```

Figure 1. CBS_ALL algorithm

For the classification part, we use a scoring method to determine the class label for a newly given data instance with the data sequence by utilizing the CSP rules as described above. If the subsequence of a sequence x equals to the SP of CSP y , we call that the sequence x matches CSP y . Figure 2 shows the procedure in details.

First, we find out all CSP with sequences matching the subsequences of the sequence in the new data instance. We use the class support and sequence support to calculate the score for each selected CSP. In this way, we obtain the scores for each class by their induced probability. Finally, the new data instance is assigned with the class of the highest score. Although this classification task is done through a simple score counting process, the algorithm takes into account two important factors – the length of CSP and the subsequence patterns for the matched CSP. As an easy example, assume a CSP A is the subsequence of CSP B. Consequently, if a new sequence contains CSP B, it must also contain CSP A. This means that we may incur the problem of repeated counts in calculating some scores. To resolve this problem, it seems that we should remove all matched sequences like CSP A. However, this problem is eliminated by considering another factor, namely the size of the matched CSP. By our scoring policy, the CSP is weighted according to their sequence length. In this way, for example, the CSP with length 5 get higher weights than the CSP with length 1. In fact, we have tried other measures for the weighting, like the product of length and the original score, but they did not produce better results. Hence, we use the subsequence relation as weight for the CSP.

```

Class_of_sequence (sequence x)
{
    M =  $\phi$ ;
    for each  $csp_i \in CSP$  do
        if  $csp_i.sp \in subseq(x)$ 
            M.add( $csp_i$ );
        end
    score_array[] = new array[class_set(D).count];
    for each  $csp_m \in M$  do
        for each  $c_n \in class\_set(D)$  do
            score_array[n] +=
             $csp_m.support / csp_m.class\_sup[n]$ ;
        end
    end
    k = index_of (Max{score_array[]});
    return k;
}

```

Figure 2. CBS_ALL classifier

```

class_of_sequence(sequence x)
{
    total_score[] = new array[class_count(D)];
    for each  $csp_i \in CSP$  do
        total_score[ $csp_i.class$ ] +=  $csp_i.sp.length$ ;
    end
    score_array[] = new array[class_count(D)];
    for each  $csp_i \in CSP$  do
        if  $csp_i \in subseq(x)$ 
            for each  $c_m \in belong\_class\_set(csp_i)$ 
                score_array[m] +=
                 $csp_i.sp.length / total\_score[m]$ ;
            end
        end if
    end
    k = index_of (Max{score_array[]});
    return  $c_k$ 
}

```

Figure 3. CBS_CLASS algorithm

3.2 CBS_CLASS Method

This method separates the database into groups by class labels. Different from CBS_ALL, the classifiable sequences are extracted from each class group, respectively, rather than from the whole dataset. Similarly, the classifier is built by using the extracted sequences. The concept of this method is to focus on the features of each class group in generating the classification rules. Figure 3 shows the detail of the CBS_CLASS algorithm.

Different from CBS_ALL algorithm, only one parameter, the minimum support, is needed in CBS_CLASS. The procedure FindSP in Figure 3 adopts an Apriori-like approach for mining sequential patterns. After the sequential patterns are extracted from each class, we can use these sequential patterns to classify a new sequence data instance directly. Figure 4 shows the classifier algorithm for CBS_CLASS, which also uses the

scoring approach to classify new data instance, but there are no for sequence support score and class support score in CBS_CLASS. The procedure of classifier is similar to that of CBS_ALL, with some differences in the score counting for each CSP. In CBS_CLASS classifier algorithm, we take the sequence length as the CSP score. Then, we normalize the total scores of each class into the same value base such that the maximum score for a new sequence in each class is 1.

Intuitively, the CBS-CLASS method is more effective on sequence feature mining. It not only eliminates the factor of data quantity imbalance between classes but also extracts the real sequential patterns for each class sequence data. In contrast, the CSPs of CBS_ALL are frequent sequences in terms of the whole dataset. Hence, they may not be deterministic features for classification.

```

CBS_CLASS(Dataset D, min_sup)
{
    for each  $c_i \in class\_set(D)$  do
         $D_i = class\_dataset(D, c_i)$ ;
         $CSP_i = FindSP(D_i, min\_sup)$ 
    end
    FindSP(Dataset D, min_sup)
    {
         $SP_1 = \{large\ 1-items\}$ 
        for(  $i = 2; SP_{i-1} \neq \phi; i++$ ) do
             $SP_C_i = gen\_candidateSP(SP_{i-1})$ ;
            for each data  $d \in D$  do
                 $SP_s = SP_C_i \cap subseq(d)$ ;
                for each  $sp \in SP_s$  do
                     $sp.sup++$ ;
                end
            end
             $SP_i = \{sp \mid sp \in SP_s, sp.sup \geq min\_sup\}$ 
        end
        return  $\bigcup_i SP_i$ 
    }
}

```

Figure 4. CBS_CLASS classifier

4. Experimental Evaluation

In the following, we describe the experimental results in evaluating the performance of CBS-ALL, CBS-CLASS and other method by using the simulated time-series datasets.

4.1 Synthetic Data Generator

We design a synthetic data generator that can generate time-series datasets with different properties based on the parameters as listed in Table 1. The default value for each parameter in the following experiments is also given in Table 1.

4.2 Comparisons of CBS_ALL and CBS_CLASS

Figure 5 shows the result of the first experimental, in which the parameter *pattern_len* is varied from 3 to 7 with other parameters as set in Table 1. Both of the inner test

and outer test results are given for the CBS_ALL and CBS_CLASS algorithms. We can see CBS_CLASS performs much better than CBS_ALL in classification accuracy. This result is due to the fact that CBS_CLASS can extract more precise sequential patterns than CBS_ALL for performing the classification.

Figure 6 shows the result of the second experiment in which the parameter *pattern_count* is varied from 3 to 7. It is observed that the accuracy of both algorithms goes down when the number of hidden pattern is increased. This is because that, with more hidden patterns, the time-series data in each class becomes more diverse and hard to extract. This result matches our inference. We also get the similar observation that CBS_CLASS achieves much higher accuracy than CBS_ALL. This indicates that CBS_CLASS is more stable than CBS_ALL when the dataset is more complex in terms of the hidden patterns.

The above observations support our intuitive induction that CBS_CLASS is better than CBS_ALL method for sequence classification due to its property in isolating each class data during the mining of CSP rules. From algorithm viewpoint, the CBS_CLASS method makes the data mining procedure more stable and powerful by processing each class data set separately.

Table1. Parameters for the synthetic data generator.

Parameter	Description	Value
<i>seq_len</i>	Number of items in each time sequence	10
<i>pattern_len</i>	Length of hidden sequential patterns	5
<i>value_level</i>	The discretized level of value	100
<i>seq_count</i>	Number of time sequences	5000
<i>class_count</i>	Number of classes	10
<i>pattern_count</i>	Number of hidden sequential pattern	5
<i>skew_ratio</i>	The degree of skew in quantity of classes	0

4.3 Comparisons with SPF Classifier

In this experiment, we compare the performance of CBS_CLASS with the SPF-classifier (Sequential Pattern Feature classifier) proposed in [7]. We define the *improvement ratio* $P(l)$ and *average improvement ratio* $AVG(P)$ as follows:

$$P(l) = \frac{Accuracy(CBS_CLASS(l)) - Accuracy(SPF(l))}{Accuracy(SPF(l))}$$

l : *pattern_len*

$$AVG(P) = \frac{\sum_{l=p_0}^{p_n} P(l)}{n}$$

Figure 7 shows the result of the first experiment. CBS_CLASS presents about 25% improvement over SPF-classifier in average under different settings of *pattern_len* as shown in Table 2. Figure 8 shows the performance of

both methods by varying *pattern_count*. SPF-classifier delivers stable accuracy over different *pattern_count*, while CBS_CLASS presents lower accuracy with *pattern_count* increased. But CBS_CLASS outperform SPF in accuracy with about 24% of improvement in average for the outer test results.

Finally, we investigate the impact of varying parameter *skew_ratio*, which controls the degree of skew in distribution of sequence classes. Figure 9 shows that both algorithms perform stable and even slightly better with *skew_ratio* increased. This shows that CBS_CLASS and SPF are insensitive to the distribution of sequence classes.

From the above experimental results, it is concluded that CBS_CLASS delivers much better accuracy than SPF. Hence, the CBS_CLASS method is verified to be promising in resolving the time-series data classification problem.

Table 2. improvement ratio for Figure 7.

	3	4	5	6	7	AVG
<i>improvement ratio(%)</i>	2.63	23.92	28.74	38.68	33.07	25.41

5. Concluding Remarks

We have presented a new method named CBS with two variations for classifying large time-series datasets. Through experimental evaluation, it is shown that CBS can classify time-series data with good accuracy by utilizing sequential patterns hidden in the datasets. It is shown that CBS_CLASS always outperforms CBS_ALL since the former builds up the classification model separately for each class. In comparisons with SPF-Classifier, CBS_CLASS presents much higher accuracy under varied kinds of datasets although it is not as stable as SPF-Classifier. Meanwhile, CBS has the advantage that it is easy to implement with excellent execution performance. Hence, we believe CBS is a promising method for classifying time-series data in large scale.

In the future, we will extend the CBS method such that it can handle the time-series dataset with numerical values. Meanwhile, we will consider the problem of skewed data distribution and missing values. Another important research issue will be the effective pruning of produced CSP rules with the aim to improve the efficiency in the process of class prediction.

References

- [1] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, *Proc. of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, September 1994.

- [2] R. Agrawal and R. Srikant, Mining Sequential Patterns, *Proc. of the 11th Int'l Conference on Data Engineering*, Taiwan, March 1995.
- [3] K. Ali, S. Manganaris, R. Srikant, Partial Classification using Association Rules, *Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining*, Newport Beach, California, August, 1997.
- [4] R. J. Bayardo Jr. Brute-Force Mining of High-Confidence Classification Rules. *Proc. of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 123-126, 1997.
- [5] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, *Wadsworth Int. Group*, Belmont, California, USA, 1984
- [6] U. M. Fayyad and K. B. Irani, Multi-interval discretization of continuous valued attributes for classification learning. In R. Bajcsy (Ed.), *Proc. of the 13 International Joint Conference on Artificial Intelligence*, pp. 1022-1027, Morgan Kaufmann, 1993.
- [7] N. Lesh, M. J. Zaki, M. Ogihara, Mining features for Sequence Classification, *5th ACM SIGKDD International*

- Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 342-246, San Diego, CA, August 1999.
- [8] B. Liu, W. Hsu, Y. Ma, Integrating Classification and Association Rule Mining, *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98, full paper)*, New York, USA, 1998.
- [9] B. Liu, W. Hsu and S. Chen, Using General Impressions to Analyze Discovered Classification Rules, *Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97, full paper)*, pp. 31-36, August 14-17, Newport Beach, California, USA, 1997.
- [10] M. Mehta, R. Agrawal and J. Rissanen, SLIQ: A Fast Scalable Classifier for Data Mining, *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, March 1996.
- [11] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1992.
- [12] M. J. Zaki, Efficient Enumeration of Frequent Sequences, *7th International Conference on Information and Knowledge Management*, pp. 68-75, Washington DC, November 1998.

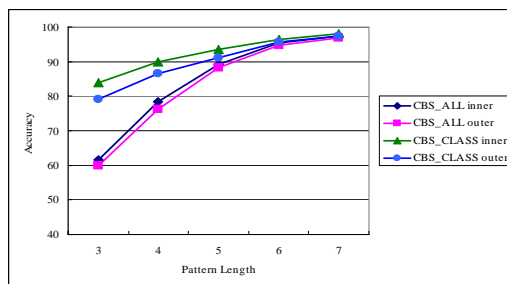


Figure 5. Comparative results by varying *pattern_len*.

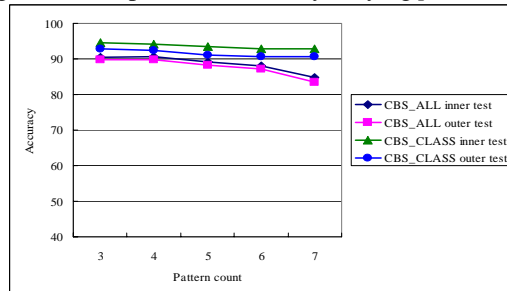


Figure 6. Comparative results by varying *pattern_count*.

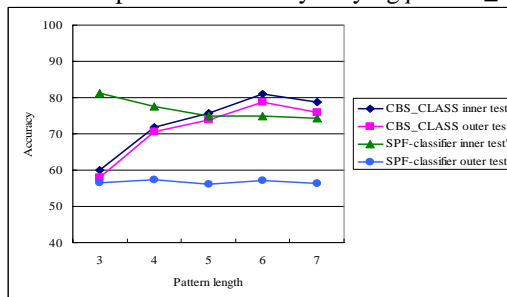


Figure 7. Comparative results by varying *pattern_len*.

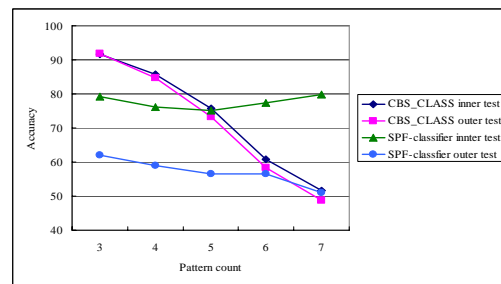


Figure 8. Comparative results by varying *pattern_count*.

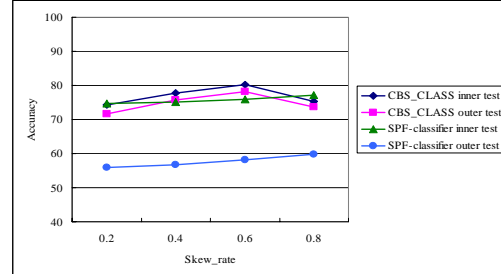


Figure 9. Comparative results by varying *data_skew*

SeqIndex: Indexing Sequences by Sequential Pattern Analysis*

Hong Cheng

Xifeng Yan

Jiawei Han

Department of Computer Science
University of Illinois at Urbana-Champaign
{hcheng3, xyan, hanj}@cs.uiuc.edu

Abstract

In this paper, we study the issues related to the design and construction of high-performance sequence index structures in large sequence databases. To build effective indices, a novel method, called SeqIndex, is proposed, in which the selection of indices is based on the analysis of discriminative, frequent sequential patterns mined from large sequence databases. Such an analysis leads to the construction of compact and effective indexing structures. Furthermore, we eliminate the requirement of setting an optimal support threshold beforehand, which is difficult for users to provide in practice. The discriminative, frequent pattern based indexing method is proven very effective based on our performance study.

1 Introduction

Sequential pattern mining is an important and active research theme in data mining [3, 9, 11, 4], with broad applications. However, with the diversity of searching and mining requests and daunting size of datasets, it is often too slow to perform mining on-the-fly but it is impossible to mine and store all the possible results beforehand.

A powerful but long-lasting alternative to this mining dilemma is to build a good sequence index structure which may serve as a performance booster for a large variety of search and mining requests. A sequence indexing structure will not only make data mining more flexible and efficient but also facilitate search and query processing in sequence databases.

Given a sequence database, there are in general two kinds of indexing structures that can be constructed for subsequence search: *consecutive* vs. *non-consecutive* subsequence indices. There are a number of *consec-*

utive sequence indexing methods developed for time-series data [1, 2, 5, 7, 12] and DNA sequences [8, 6]. Euclidean distance, dynamic time warping (DTW) and discrete fourier transform (DFT) are commonly used tools for indexing and similarity search in time-series data. DNAs is another kind of consecutive sequence. Since it is consecutive, suffix-tree and multi-resolution string (MSR) are developed for DNA indexing.

To the best of our knowledge, there is no *non-consecutive* sequence indexing method studied before. Nonconsecutive sequence mining and matching is important in many applications, such as customer transaction analysis, various event logs, sensor data flows, and video data. Indexing non-consecutive sequence poses great challenges to research. First, there exist an explosive number of subsequences due to the combination of gaps and symbols. Second, such an indexing mechanism cannot be built on top of the existing consecutive sequence indexing mechanisms. For example, models like suffix-tree or those based on time-series transformation, such as DFT and DTW, are no longer valid because sequences under our examination are not consecutive any more.

In this paper, we propose a new indexing methodology, called *discriminative, frequent sequential pattern-based (DFP) indexing*, which selects the best indexing features, based on frequent sequential pattern mining and discriminative feature selection, for effective indexing of sequential data. The study extends the recent work on graph indexing by Yan et al. [10] and examines the applicability of DFP in sequence databases.

Although the general framework of using frequent patterns as indexing features has been exposed in graph indexing [10], it is not obvious whether this framework can be successfully applied to sequences. In fact, there exists an inverted index based method to do sequence indexing. For each item, there is an id list associated with it. To process a query, just intersect the id lists. We call this algorithm ItemIndex. In this work, we compare ItemIndex and SeqIndex from multiple angles to explore the boundaries of these two algorithms.

* The work was supported in part by the U.S. National Science Foundation IIS-02-09199, IIS-03-08215. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

The remainder of the paper is organized as follows. Section 2 introduces the basic concepts related to sequence indexing and some notations used throughout the paper. Section 3 presents the algorithm to determine the size-increasing support function. Section 4 formulates the SeqIndex algorithm. We report and analyze our performance study in Section 5 and conclude our study in Section 6.

2 Preliminary Concepts

Let $I = \{i_1, i_2, \dots, i_k\}$ be a set of all items. A sequence $s = \langle i_{j_1}, i_{j_2}, \dots, i_{j_n} \rangle$ ($i_j \in I$) is an ordered list. We adopt this sequence definition to simplify the description of our indexing model. A sequence $\alpha = \langle a_1, a_2, \dots, a_m \rangle$ is a *sub-sequence* of another sequence $\beta = \langle b_1, b_2, \dots, b_n \rangle$, denoted as $\alpha \sqsubseteq \beta$ (if $\alpha \neq \beta$, written as $\alpha \subset \beta$), if and only if $\exists j_1, j_2, \dots, j_m$, such that $1 \leq j_1 < j_2 < \dots < j_m \leq n$ and $a_1 = b_{j_1}, a_2 = b_{j_2}, \dots$, and $a_m = b_{j_m}$. We also call β a *super-sequence* of α , and β *contains* α .

A sequence database, $D = \{s_1, s_2, \dots, s_n\}$, is a set of sequences. The *support* of a sequence α in a sequence database D is the number of sequences in D which contain α , $\text{support}(\alpha) = |\{s | s \in D \text{ and } \alpha \sqsubseteq s\}|$. Given a minimum support threshold, min_sup , a sequence is **frequent** if its support is no less than min_sup . The set of **frequent sequential pattern**, FS , includes all the frequent sequences.

DEFINITION 2.1. (Subsequence Search) *Given a sequence database $D = \{s_1, \dots, s_n\}$ and a query sequence q , it returns the answer set $D_q = \{s_i | s_i \in D, q \sqsubseteq s_i\}$.*

In sequence query processing, the major concern is query response time, which is composed of the time of searching, the time of fetching the candidate set from the disk and the cost to check the candidates. We want to minimize the search time since a query could have a lot of subsequences in the index. If searching in the index structure is very inefficient, a large amount of time would be wasted on intersecting some id lists which do not shrink the candidate answer set a lot. So we need an efficient algorithm to search in the “right” direction. In addition, the I/O part also plays an important role.

3 Determine the Size-increasing Support Function

The first step is to mine frequent patterns from sequence database. As pointed out in [10], the purpose of *size-increasing support constraint* is to make the mining tractable. Meanwhile, the overall index quality may be optimized. This constraint is effective in keeping a compact index size while providing high-quality indices. An interesting question is how to determine the

size-increasing support function in a systematic way? Instead of using heuristic functions, a novel solution is to use the discriminative ratio as a guide to select the appropriate threshold. It becomes a data mining problem. In this setting, we not only need to mine frequent sequential patterns with min_sup but also have to decide what min_sup is.

Let $\psi(l)$ be the size-increasing support function. The automated setting of $\psi(l)$ is as follows. Given a discriminative threshold γ_{min} and a sequence database D , we first set the support of all length-1 patterns to be 1. Then we determine $\psi(l)$ in a level wise manner. When we decide for length $k \geq 2$, we set $\psi(k) = \psi(k-1)$. Under this support, we mine a set of frequent length- k patterns. For every length- k pattern x , we calculate its discriminative ratio with respect to patterns that have already been selected,

$$(3.1) \quad \gamma = \frac{|\bigcap_{\varphi: f_{\varphi_i} \sqsubseteq x} D_{f_{\varphi_i}}|}{|D_x|}$$

If $\gamma \geq \gamma_{\text{min}}$, we say this pattern is discriminative. Let S_k be the set of length- k frequent patterns. Suppose the lowest support and the highest support in S_k is t_0 and t_h respectively. For every possible support value t , $t_0 \leq t \leq t_h$, we may calculate the number of patterns in S_k whose support is above t and of these patterns, how many of them have discriminative ratio great than γ_{min} .

Eventually we get a cut point t^* where p percentage of discriminative patterns are retained. $\psi(k)$ is then set at t^* . Using the above process, a user need not set the optimal support threshold any more.

We call the algorithm, shown in Figure 1, **AutoSupport**. AutoSupport only needs two parameters set by users, a discriminative ratio and a percentage cutoff. It will automatically adjust the support function according to the data distribution and the percentage cutoff. AutoSupport increases the support function to the extent where p percentage of discriminative patterns remain. This can reduce the number of patterns, especially those non-discriminative patterns substantially.

4 SeqIndex: A Sequence Indexing Algorithm

In this section, we present the SeqIndex algorithm. The algorithm can be divided into four steps: (1) discriminative feature selection, (2) index construction, (3) search, and (4) verification.

We first mine the frequent patterns, and use a discriminative ratio γ_{min} to filter out those redundant patterns. The output is a set of discriminative sequential patterns that will be used as indexing features. After that, we construct an *index tree* T , which is a prefix tree, to store and retrieve those features.

Algorithm AutoSupport

Input: A sequence database D ,
discriminative threshold γ_{min} ,
a percentage cutoff p ,
maximum subsequence length L .
Output: $\psi(k)$, size-increasing support function.

```

1:  $\psi(1) = 1$ ;
2: for length  $k$  from 2 to  $L$ 
3:    $\psi(k) = \psi(k-1)$ ;
4:   do
5:     Mine the frequent length- $k$  patterns
       with  $\psi(k)$ ;
6:     for each length- $k$  pattern, calculate  $\gamma$ ;
7:     Calculate pattern distribution under
       different support  $t$ ;
8:     Find a support  $t^*$  that  $p\%$  of all
       discriminative patterns remain;
9:      $\psi(k) = t^*$ ;
10:  end for
11: return  $\psi(k)$ ;
```

Figure 1: Determine Size-increasing Support Function

The most important part is the search algorithm, since an efficient search algorithm can improve the query processing time substantially. We discuss it in the following.

4.1 Search. Given a query q , SeqIndex enumerates all its subsequences within the maximum subsequence length L and searches them in the index tree. For those subsequences found in the index tree, SeqIndex intersects their id lists to get a set of candidate sequences.

There could be a large number of subsequences contained in a given query q . To optimize the search process, we should try to reduce the number of subsequences that need to be checked.

Two optimization techniques [10] are studied extensively in SeqIndex to reduce the search space. One is Apriori pruning and the other is maximum discriminative sequential patterns. With these two optimization techniques in consideration, we propose an efficient algorithm for searching the index tree efficiently. We traverse the index tree in a depth first search manner. At each node p , we check the sequence from the root to p and label it as a candidate if it is a subsequence of query q . Then we visit its child nodes recursively. The reason we just label it but not intersect its id list immediately is that we want to check whether there is a maximum discriminative sequential pattern. If there is, it is unnecessary to intersect the id list of node p . On

Algorithm DFS Search

Input: A sequence database D ,
Index tree T , Query q ,
maximum subsequence length L .
Output: Candidate set C_q .

```

1: Let  $C_q = D$ ;
2: DFS Traverse index tree  $T$  {
3:   Check the sequence from the root to node  $p$ 
4:   if (it is a subsequence of  $q$ )
5:     Label it and visit its child node;
6:   else
7:     Skip  $p$  and its subtree;
8:   Once arriving at a leaf node OR
   before skipping a subtree, do
9:     Find the deepest labelled node  $p'$ 
       along this path;
10:     $C_q = C_q \cap D_{p'}$ ;
11:    if ( $|C_q| < minCanSize$ )
12:      Early termination;
13: return  $C_q$ ;
```

Figure 2: DFS Search and Candidate Set Computation

the other hand, if the sequence from the root to node p is not a subsequence of q , node p and its subtree can be pruned according to Apriori pruning. Once reaching a leaf node, or before skipping a node, we need to find out the deepest labelled node, which is the maximum discriminative sequential pattern along this path. Only its id list should be intersected with C_q while other “smaller” subsequences on this path can be skipped. The algorithm using the DFS search is shown in Figure 2.

Further optimization can be explored by the following intuition. When more and more intersections are executed, the candidate answer set becomes smaller and smaller. At some point of the traversal, we may find that C_q is already small enough compared with a user-specified threshold $minCanSize$. We can decide at this point to stop traversing the index tree. The candidate set C'_q so obtained is a superset of the optimal candidate set C_q which can be obtained by searching the whole tree. If the tree is large, C'_q may be acceptable since the savings in traversing the remaining part of the tree would outweigh the reduction of C_q by further intersections. We call this technique *early termination*.

Besides the DFS search method introduced above, we developed an alternative search algorithm that performs better for dense datasets. Instead of traversing the index tree in a depth-first search manner, at each level, we visit the node that has the smallest id list.

Since we choose a highly selective node to follow, the size of C_q will be reduced very quickly. After traversing a path, we can start over from the root, pick the second smallest child node and perform the same operation until we get an answer set with acceptable size. We call this search method *MaxSel* since we always greedily select the path with the highest selectivity.

The reason that two search algorithms are proposed is that we expect that their performance is correlated with data distribution. If the dataset has a small number of distinct items, the tree becomes very “thin” with a small branching factor and each node is associated with a long id list. In this case, MaxSel algorithm can easily find the most selective path and reduce the size of C_q quickly by skipping those nodes with a long list. On the other hand, if the dataset has a large number of distinct items, the tree becomes very “wide” with a large branching factor and each node is associated with only a small id list. In this case, MaxSel will spend a lot of time at each level searching for the node with the smallest id list. Thus, DFS search with early termination will perform better. We will compare these two algorithms to verify our reasoning in the experiments.

5 Experimental Results

In this section, we report our experimental results that validate the effectiveness and efficiency of SeqIndex. We compare SeqIndex with ItemIndex.

5.1 ItemIndex Algorithm ItemIndex builds index for each single item in the sequence database. For an item i , keep every occurrence of i in the sequence database with a tuple $\langle seq_id, position \rangle$. Therefore, the index of an item i contains a list of such tuples.

Given a query $q = i_0 i_1 \dots i_n$, ItemIndex will compute the intersection of all pairs of adjacent items in the query q . When intersecting the index list of i_j and i_{j+1} , if a pair of tuples from the two lists match in the *seq_id* part, we will further check if *position* in i_j list is smaller than *position* in i_{j+1} list. If so, we know that i_j and i_{j+1} occur in the same sequence and i_j precedes i_{j+1} in that sequence.

We compare the performance of SeqIndex and ItemIndex in terms of CPU time and I/O costs. The data is generated by a synthetic data generator provided by IBM¹. More details are referred to [3].

5.2 SeqIndex vs. ItemIndex We compare SeqIndex and ItemIndex in terms of CPU time and I/O costs. Since we proposed two search algorithms – DFS Search and Maximum Selectivity Search, we will test SeqIndex

with these two alternatives. In the following figures, we use **DFS** to denote SeqIndex with DFS search and **MaxSel** to denote SeqIndex with maximum selectivity search.

We first test how the execution time of SeqIndex and ItemIndex changes when we vary the number of distinct items in the sequence database. The sequence database has 10,000 sequences with an average of 30 items in each sequence. The number of distinct items is varied from 10 to 1000. The result is shown in Figure 3(a).

When the number of distinct items is very small, e.g. 10, MaxSel is the fastest, DFS stands in the middle while ItemIndex is the slowest. This is because the data distribution is very dense with a small number of items. MaxSel is effective by picking the most selective path, as we analyzed in the previous section.

As the number of items increases, CPU time of ItemIndex decreases dramatically and that of DFS also decreases but more slowly. When the number of items is 500, both DFS and ItemIndex run faster while MaxSel starts to slow down. This is due to sparser data and shorter id lists. On the other hand, the index tree of MaxSel turns to be bulky with lots of nodes at each level. Searching the most selective path involves visiting many nodes, thus becomes very inefficient. In this case, DFS turns out to be very efficient. This also gives us a hint – when the data is very dense, we can employ SeqIndex with maximum selectivity search; when the data is very sparse, we can switch to SeqIndex with DFS search. The “hybrid” SeqIndex will perform uniformly well while ItemIndex is quite sensitive to the data distribution.

The second experiment is to test the performance of SeqIndex and ItemIndex with varied size of queries. We test a database with 10,000 sequences. Each sequence has 30 items on average. The number of distinct items is 10. We vary the query length from 5 to 50. The result is shown in Figure 3(b).

Figure 3(b) shows that MaxSel performs very well as the query length increases. MaxSel first searches the index tree and produces a candidate set. Then it verifies those candidate sequences. When the query length is small, e.g. 5, there are more candidate sequences since it is usually easier to satisfy a short query. As the query length increases, the size of candidate set decreases since it is harder to satisfy a long query. That is why the performance of MaxSel improves as the query length increases. On the other hand, the performance of ItemIndex degrades as the query length increases. Since the number of intersections executed in ItemIndex is proportional to the query length, the execution time increases roughly linearly as the query length increases.

¹<http://www.almaden.ibm.com/cs/quest>

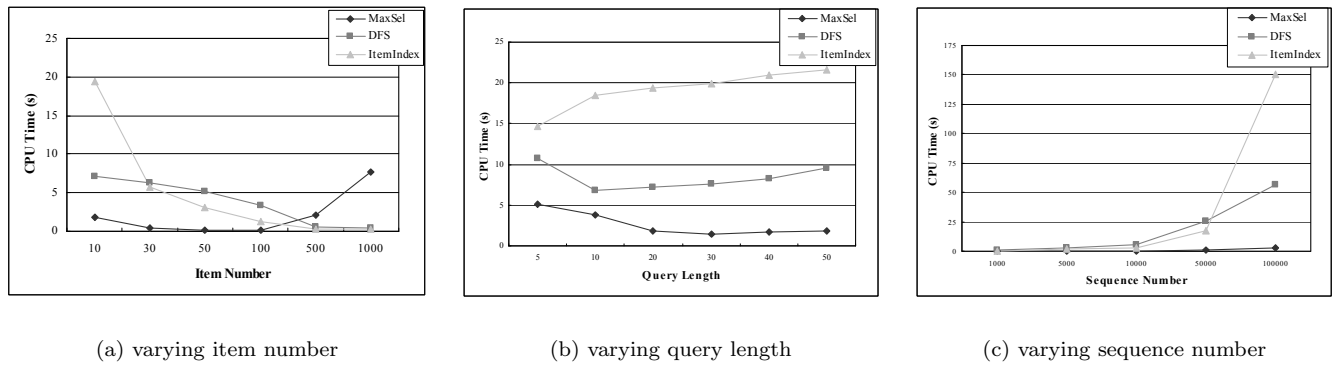


Figure 3: Performance study

We also test the performance of SeqIndex and ItemIndex with varied database size. We vary the number of sequences in the sequence database from 1,000 to 100,000 (number of distinct items is 50 and the query length is 20). The result is shown in Figure 3(c). As we increase the number of sequences in the database, MaxSel has the best scalability, DFS stands in between and ItemIndex shows poor scalability.

We finally test how effective our algorithm AutoSupport is. Experimental results show that the index is more compact using AutoSupport in comparison with the uniform-support method. The mining time of AutoSupport also outperforms the uniform-support method significantly. The figure is omitted due to space limit.

6 Conclusions

In this paper, we broaden the scope of sequential pattern mining beyond the “narrowly defined” spectrum of knowledge discovery. Our study is focused on the design and construction of high-performance nonconsecutive sequence index structures in large sequence databases. A novel method, SeqIndex, is proposed, in which the selection of indices is based on the analysis of discriminative, frequent sequential patterns mined from large sequence databases. Such an analysis leads to the construction of compact and effective indexing structures. The effectiveness of the approach has been verified by our performance study.

References

- [1] R. Agrawal, C. Faloutsos, and A.N. Swami. Efficient Similarity Search In Sequence Databases. In *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993.
- [2] R. Agrawal, K.I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proc. 1995 Int. Conf. on Very Large Databases*, pages 490–501, 1995.
- [3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering (ICDE’95)*, pages 3–14, March 1995.
- [4] J. Ayres, J. E. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using bitmaps. In *Proc. 2002 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD’02)*, July 2002.
- [5] K. Chan and A.W. Fu. Efficient time-series matching by wavelets. In *Proc. 15th IEEE Int. Conf. on Data Engineering*, pages 126–133, 1999.
- [6] T. Kahveci and A.K. Singh. Efficient index structures for string databases. In *The VLDB Journal*, pages 351–360, 2001.
- [7] E. Keogh. Exact indexing of dynamic time warping. In *Proc. 2002 Int. Conf. Very Large Data Bases (VLDB’02)*, Aug 2002.
- [8] C. Meek, J.M. Patel, and S. Kasetty. Oasis: An online and accurate technique for local-alignment searches on biological sequences. In *Proc. 2003 Int. Conf. Very Large Data Bases (VLDB’03)*, Sept 2003.
- [9] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. 2001 Int. Conf. Data Engineering (ICDE’01)*, pages 215–224, April 2001.
- [10] X. Yan, P.S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proc. 2004 Int. Conf. on Management of Data (SIGMOD’04)*, pages 253–264, 2004.
- [11] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 40:31–60, 2001.
- [12] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *Proc. 2003 Int. Conf. on Management of Data (SIGMOD’03)*, pages 181–192, 2003.

On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering

Chris Ding*

Xiaofeng He*

Horst D. Simon*

Abstract

Current nonnegative matrix factorization (NMF) deals with $X = FG^T$ type. We provide a systematic analysis and extensions of NMF to the symmetric $W = HH^T$, and the weighted $W = HSH^T$. We show that (1) $W = HH^T$ is equivalent to Kernel K -means clustering and the Laplacian-based spectral clustering. (2) $X = FG^T$ is equivalent to simultaneous clustering of rows and columns of a bipartite graph. Algorithms are given for computing these symmetric NMFs.

1 Introduction

Standard factorization of a data matrix uses singular value decomposition (SVD) as widely used in principal component analysis (PCA). However, for many dataset such as images and text, the original data matrices are nonnegative. A factorization such as SVD contain negative entries and thus has difficulty for interpretation. Nonnegative matrix factorization (NMF) [7, 8] has many advantages over standard PCA/SVD based factorizations. In contrast to cancellations due to negative entries in matrix factors in SVD based factorizations, the nonnegativity in NMF ensures factors contain coherent parts of the original data (images).

Let $X = (\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathbb{R}_+^{p \times n}$ be the data matrix of nonnegative elements. In image processing, each column is a 2D gray level of the pixels. In text mining, each column is a document.

The NMF factorizes X into two nonnegative matrices,

$$X \approx FG^T, \quad (1)$$

where $F = (\mathbf{f}_1, \dots, \mathbf{f}_k) \in \mathbb{R}_+^{p \times k}$ and $G = (\mathbf{g}_1, \dots, \mathbf{g}_k) \in \mathbb{R}_+^{n \times k}$. k is a pre-specified parameter. The factorizations are obtained by the least square minimization. A number of researches on further developing NMF computational methodologies [12, 11, 10], and applications on text mining [9, 14, 11].

Here we study NMF in the direction of data clustering. The relationship between NMF and vector quantization, especially the difference, are discussed by Lee

and Seung [7] as a motivation for NMF. The clustering aspect of NMF is also studied in [14, 10].

In this paper, we provide a systematic analysis and extensions of NMF and show that NMF is equivalent to Kernel K -means clustering and Laplacian-based spectral clustering.

(1) We study the symmetric NMF of

$$W \approx HH^T \quad (2)$$

where W contains the pairwise similarities or the Kernels. We show that this is equivalent to K -means type clustering and the Laplacian based spectral clustering.

(2) We generalize this to bipartite graph clustering i.e., simultaneously clustering rows and columns of the rectangular data matrix. The result is the standard NMF.

(3) We extend NMFs to weighted NMF:

$$W \approx HSH^T. \quad (3)$$

(4) We derive the algorithms for computing these factorizations.

Overall, our study provides a comprehensive look at the nonnegative matrix factorization and spectral clustering.

2 Kernel K-means clustering and Symmetric NMF

K -means clustering is one of most widely used clustering method. Here we first briefly introduce the K -means using spectral relaxation [15, 3]. This provides the necessary background information, notations and paves the way to the nonnegative matrix factorization approach in §2.1.

K -means uses K prototypes, the centroids of clusters, to characterize the data. The objective function is to minimize the sum of squared errors,

$$J_K = \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \mathbf{m}_k\|^2 = c_2 - \sum_k \frac{1}{n_k} \sum_{i,j \in C_k} \mathbf{x}_i^T \mathbf{x}_j, \quad (4)$$

where $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ is the data matrix, $\mathbf{m}_k = \sum_{i \in C_k} \mathbf{x}_i / n_k$ is the centroid of cluster C_k of n_k points,

*Lawrence Berkeley National Laboratory, University of California, Berkeley, CA 94720. {chqding, xhe, hdsimon}@lbl.gov

and $c_2 = \sum_i \|\mathbf{x}_i\|^2$. The solution of the clustering is represented by K non-negative indicator vectors:

$$H = (\mathbf{h}_1, \dots, \mathbf{h}_K), \quad \mathbf{h}_k^T \mathbf{h}_\ell = \delta_{k\ell}. \quad (5)$$

where

$$\mathbf{h}_k = (0, \dots, 0, \overbrace{1, \dots, 1}^{n_k}, 0, \dots, 0)^T / n_k^{1/2} \quad (6)$$

Now Eq.(4) becomes $J_K = \text{Tr}(X^T X) - \text{Tr}(H^T X^T X H)$. The first term is a constant. Let $W = X^T X$. Thus $\min J_K$ becomes

$$\max_{H^T H = I, H \geq 0} J_W(H) = \text{Tr}(H^T W H). \quad (7)$$

The pairwise similarity matrix $W = X^T X$ is the standard inner-product linear Kernel matrix. It can be extended to any other kernels. This is done using a non-linear transformation (a mapping) to the higher dimensional space

$$\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$$

The clustering objective function under this mapping, with the help of Eq.(4), can be written as

$$\min J_K(\phi) = \sum_i \|\phi(\mathbf{x}_i)\|^2 - \sum_k \frac{1}{n_k} \sum_{i,j \in C_k} \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j). \quad (8)$$

The first term is a constant for a given mapping function $\phi(\cdot)$ and can be ignored. Let the kernel matrix $W_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. Using the cluster indicators H , the kernel K -means clustering is reduced to Eq.(7).

The objective function in Eq.(7). can be symbolically written as

$$J_W = \sum_k \frac{1}{n_k} \sum_{i,j \in C_k} w_{ij} = \text{Tr}(H^T W H). \quad (9)$$

Kernel K -means aims at maximizing within-cluster similarities. The advantage of Kernel K -means is that it can describe data distributions more complicated than Gaussian distributions.

2.1 Nonnegative relaxation of Kernel K -means

We show the spectral relaxation of Eq.(7) can be solved by the matrix factorization

$$W \approx H H^T, \quad H \geq 0. \quad (10)$$

Casting this in an optimization framework, an appropriate objective function is

$$\min_{H \geq 0} J_1 = \|W - H H^T\|^2, \quad (11)$$

where the matrix norm $\|A\|^2 = \sum_{ij} a_{ij}^2$, the Frobenius norm.

Theorem 1. NMF of $W = H H^T$ is equivalent to Kernel K -means clustering with the the strict orthogonality relation Eq.(5) relaxed.

Proof. The maximization of Eq.(7) can be written as

$$\begin{aligned} H &= \arg \min_{H^T H = I, H \geq 0} -2\text{Tr}(H^T W H) \\ &= \arg \min_{H^T H = I, H \geq 0} -2\text{Tr}(H^T W H) + \|H^T H\|^2 \\ &= \arg \min_{H^T H = I, H \geq 0} (\|W\|^2 - 2\text{Tr}(H^T W H) + \|H^T H\|^2) \\ &= \arg \min_{H^T H = I, H \geq 0} \|W - H H^T\|^2. \end{aligned} \quad (12)$$

Relaxing the orthogonality $H^T H = I$ completes the proof. \square

A question immediately arises. Will the orthogonality $H^T H = I$ get lost?

Theorem 2. NMF of $W = H H^T$ retains near-orthogonality of $H^T H = I$.

proof. One can see $\min J_1$ is equivalent to

$$\max_{H \geq 0} \text{Tr}(H^T W H), \quad (13)$$

$$\min_{H \geq 0} \|H^T H\|^2. \quad (14)$$

The first objective recovers the original optimization objective Eq.(7). We concentrate on 2nd term. Note

$$\|H^T H\|^2 = \sum_{\ell k} (H^T H)_{\ell k}^2 = \sum_{\ell \neq k} (\mathbf{h}_\ell^T \mathbf{h}_k)^2 + \sum_k (\mathbf{h}_k^T \mathbf{h}_k)^2.$$

Minimizing the first term is equivalent to enforcing the orthogonality among \mathbf{h}_ℓ : $\mathbf{h}_\ell^T \mathbf{h}_k \approx 0$. Minimizing the second term is equivalent to

$$\min \|\mathbf{h}_1\|^4 + \dots + \|\mathbf{h}_K\|^4. \quad (15)$$

However, H cannot be all zero, otherwise we would have $\text{Tr}(H^T W H) = 0$. More precisely, since $W \approx H H^T$, we have

$$\sum_{ij} w_{ij} \approx \sum_{ij} (H H^T)_{ij} = \sum_{kij} h_{ik} h_{jk} = \sum_k \|\mathbf{h}_k\|^2, \quad (16)$$

where $\|\mathbf{h}\| = \sum_i |h_i| = \sum_i h_i$ is the L_1 of vector \mathbf{h} . This means $\|\mathbf{h}_\ell\| > 0$. Therefore, optimization of Eq.(14) with consideration of Eq.(13) implies H has near orthogonal columns, i.e.,

$$\mathbf{h}_\ell^T \mathbf{h}_k \approx \begin{cases} 0 & \text{if } \ell \neq k, \\ \|\mathbf{h}_k\|^2 > 0 & \text{if } \ell = k. \end{cases} \quad (17)$$

Furthermore, given that $\|\mathbf{h}_\ell\| > 0$, minimization of Eq.(15) will naturally lead to the column equalization condition

$$\|\mathbf{h}_1\| = \dots = \|\mathbf{h}_K\|. \quad (18)$$

□ the orthogonality constraints

The near-orthogonality of columns of H is important for data clustering. An exact orthogonality implies that each row of H can have only one nonzero element, which implies that each data object belongs only to 1 cluster. This is hard clustering, such as in K -means. The near-orthogonality condition relaxes this a bit, i.e., each data object could belong fractionally to more than 1 cluster. This is soft clustering. A completely non-orthogonality among columns of H does not have a clear clustering interpretation.

3 Bipartite graph K -means clustering and NMF

A large number of datasets in today's applications are in the form of rectangular nonnegative matrix, such as the word-document association matrix in text mining or the DNA gene expression profiles. These types of datasets can be conveniently represented by a bipartite graph where the graph adjacency matrix B contains the association among row and column objects, which is the input data matrix $B = X$.

The above kernel K -means approach can be easily extended to bipartite graph. Let \mathbf{f}_k be the indicator for the k -th row cluster. \mathbf{f}_k has the same form of \mathbf{h}_k as in Eq.(6). Put them together we have the indicator matrix $F = (\mathbf{f}_1, \dots, \mathbf{f}_k)$. Analogously, we define the indicator matrix $G = (\mathbf{g}_1, \dots, \mathbf{g}_k)$ for column-clusters.

We combine the row and column nodes together as

$$W = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}, \mathbf{h}_k = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{f}_k \\ \mathbf{g}_k \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} F \\ G \end{pmatrix} \quad (19)$$

where the factor $1/\sqrt{2}$ allows the simultaneous normalizations $\mathbf{h}_k^T \mathbf{h}_k = 1$, $\mathbf{f}_k^T \mathbf{f}_k = 1$, and $\mathbf{g}_k^T \mathbf{g}_k = 1$. The Kernel K -means type clustering objective function becomes

$$\max_{F,G} J_K^B = \frac{1}{2} \text{Tr} \begin{pmatrix} F \\ G \end{pmatrix}^T \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} F \\ G \end{pmatrix} \quad (20)$$

Following the Kernel K -means clustering optimization of Eq.(11), F, G are obtained by minimizing

$$J_1^B = \left\| \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} - \begin{pmatrix} F \\ G \end{pmatrix} \begin{pmatrix} F \\ G \end{pmatrix}^T \right\|^2 \quad (21)$$

Note that

$$J_1^B = 2\|B - FG^T\|^2 + \|F^T F\|^2 + \|G^T G\|^2.$$

Minimization of the second and third terms, following the analysis of Eqs.(14, 17), is equivalent to enforcing

$$\mathbf{f}_\ell^T \mathbf{f}_k \approx 0, \mathbf{g}_\ell^T \mathbf{g}_k \approx 0, \ell \neq k. \quad (22)$$

Thus $\min J_1^B$ becomes

$$\min_{F,G} J_2 = \|B - FG^T\|^2 \quad (23)$$

subject to the orthogonality constraints in Eq.(22). This is the form of NMF proposed by Lee and Seung. Therefore, we have shown that the bipartite graph kernel K -means clustering leads to NMF for nonnegative rectangular matrix.

The orthogonality constraints play an important role. If the orthogonality holds vigorously, we can show directly that NMF of $\|B - FG^T\|^2$ is equivalent to simultaneously K -means clustering of rows and columns of B .

To show this, we first prove it for clustering of columns of $B = (\mathbf{b}_1, \dots, \mathbf{b}_n) = (b_{ij})$, under the normalization

$$\sum_{i=1}^p b_{ij} = 1, \sum_{r=1}^k g_{ir} = 1, \sum_{j=1}^p f_{jk} = 1. \quad (24)$$

For any given data, normalization of X is first applied. The second normalization indicates that the i -th row of G are the posterior probabilities for \mathbf{b}_i belonging to k clusters; they should add up to 1. The 3rd normalization $\sum_j (\mathbf{f}_k)_j$ is a standard length normalize of the vector \mathbf{f}_k . Since we approximate $B \approx FG^T$, the normalization of FG^T should be consistent with the normalization of B . Indeed, $\sum_{i=1}^p (FG^T)_{ij} = \sum_{i=1}^p \sum_{r=1}^k F_{ir} G_{jr} = 1$, consistent with $\sum_i B_{ij} = 1$.

With this self-consistent normalization and imposing strict orthogonality on G : $\mathbf{g}_\ell^T \mathbf{g}_k = 0$, $\ell \neq k$, we call the resulting factorization as Orthogonal NMF.

Theorem 3. Orthogonal NMF is identical to K -means clustering.

Proof. We have

$$J_2 = \|B - FG^T\|^2 = \sum_{i=1}^n \left\| \mathbf{b}_i - \sum_{k=1}^{\kappa} g_{ik} \mathbf{f}_k \right\|^2, \quad (25)$$

because the Frobenious norm of a matrix is equivalent to sum of column norms. Now due to the normalization of G , we have

$$\begin{aligned} J_2 &= \left\| \sum_{k=1}^{\kappa} g_{ik} (\mathbf{b}_i - \mathbf{f}_k) \right\|^2 = \sum_{k=1}^{\kappa} g_{ik}^2 \|\mathbf{b}_i - \mathbf{f}_k\|^2 \\ &= \sum_{k=1}^{\kappa} g_{ik} \|\mathbf{b}_i - \mathbf{f}_k\|^2 = \sum_{k=1}^{\kappa} \sum_{i \in C_k} \|\mathbf{b}_i - \mathbf{f}_k\|^2 \end{aligned}$$

The 2nd equality is due to the orthogonality condition of G which implies that on each row of G , only one

element is nonzero. This also implies $g_{ik} = 0, 1$. Thus $g_{ik}^2 = g_{ik}$, giving the 3rd equality. Thus the final result is the standard K -means clustering of $\{\mathbf{b}_i\}_{i=1}^n$. \mathbf{f}_k is the cluster centroid. ■

In NMF of optimizing $\min \|B - FG^T\|$, rows and columns are treated in equal footing, since we could equally write $J_2 = \|B^T - GF^T\|$. Thus clustering of columns of B is happening *simultaneously* as the clustering the rows of B .

4 Spectral clustering and NMF

In recent years spectral clustering using the Laplacian of the graph emerges as solid approach for data clustering. Here we focus on the clustering objective functions. There are three clustering objective functions. the Ratio Cut [6], the Normalized Cut [13], and the MinMax Cut [4]. We are interested in the multi-way clustering objective functions,

$$J = \sum_{1 \leq p < q \leq \kappa} \frac{s(C_p, C_q)}{\rho(C_p)} + \frac{s(C_p, C_q)}{\rho(C_q)} = \sum_{k=1}^{\kappa} \frac{s(C_k, \bar{C}_k)}{\rho(C_k)} \quad (26)$$

$$\rho(C_k) = \begin{cases} |C_k| & \text{for Ratio Cut} \\ \sum_{i \in C_k} d_i & \text{for Normalized Cut} \\ s(C_k, \bar{C}_k) & \text{for MinMax Cut} \end{cases} \quad (27)$$

where \bar{C}_k is the complement of subset C_k in graph G .

Here we show that the minimization of these objective functions can be equivalently carried out via the non-negative matrix factorizations. The proof follows the multi-way spectral relaxation of Ratio Cut clustering objective function[1], and Normalized Cut and MinMax Cut clustering objective functions[5].

For concreteness and simplicity, here we outline the proof for the case of Normalized Cut. Let \mathbf{h}_k be the cluster indicators as in Eq.(6). One can easily see that

$$s(C_k, \bar{C}_k) = \sum_{i \in C_k} \sum_{j \in \bar{C}_k} w_{ij} = \mathbf{h}_k^T (D - W) \mathbf{h}_k \quad (28)$$

and $\sum_{i \in C_k} d_i = \mathbf{h}_k^T D \mathbf{h}_k$. Define the scaled cluster indicator vector $\mathbf{z}_\ell = D^{1/2} \mathbf{h}_\ell / \|D^{1/2} \mathbf{h}_\ell\|$, which obey the orthonormal condition $\mathbf{z}_\ell^T \mathbf{z}_k = \delta_{\ell k}$, or $Z^T Z = I$, where $Z = (\mathbf{z}_1, \dots, \mathbf{z}_\kappa)$. Substituting into the Normalized Cut objective function, we have

$$J_{\text{NC}} = \sum_{\ell=1}^{\kappa} \frac{\mathbf{h}_\ell^T (D - W) \mathbf{h}_\ell}{\mathbf{h}_\ell^T D \mathbf{h}_\ell} = \sum_{\ell=1}^{\kappa} \mathbf{z}_\ell^T (I - \widetilde{W}) \mathbf{z}_\ell$$

where

$$\widetilde{W} = D^{-1/2} W D^{-1/2}. \quad (29)$$

The first term is a constant. Thus the minimization problem becomes

$$\max_{Z^T Z = I, Z \geq 0} \text{Tr}(Z^T \widetilde{W} Z) \quad (30)$$

Now if we remove the restriction that Z takes the discrete values, and allow Z to be any continuous value, this is the multi-way spectral relaxation of the Normalized Cut[5]. The solution of the maximization problem is given by the k principal eigenvectors of the matrix \widetilde{W} .

The point of departure from spectral relaxation of Normalized Cut is to recognize that the maximization problem of Eq.(30) is identical to the maximization problem of Eq.(7) with the same orthogonality constraints. Following the same discussions there, the maximization problem of Eq.(30) is equivalent to

$$\min_{Z \geq 0} J_3 = \|\widetilde{W} - Z Z^T\|^2, \quad (31)$$

Once the solution \widehat{Z} for $\min J_3(Z)$ is obtained, we can recover H by optimizing

$$\min_{H \geq 0} \sum_{\ell} \left\| \widehat{\mathbf{z}}_\ell - \frac{D^{1/2} \mathbf{h}_\ell}{\|D^{1/2} \mathbf{h}_\ell\|} \right\|^2, \quad (32)$$

The solution can be easily shown to be $\mathbf{h}_k = D^{-1/2} \widehat{\mathbf{z}}_k$. This gives the solution to Normalized Cut via the NMF approach. This can be extended to RatioCut and MinMaxCut. Summarizing, we have proved that

Theorem 4. NMF is equivalent to spectral clustering.

5 Weighted Nonnegative $W = HSH^T$

In both Kernel K -means and spectral clustering, we assume the pairwise similarity matrix W are semi positive definite. For kernel matrices, this is true. But a large number of similarity matrices is nonnegative, but not s.p.d. This motivates us to propose the following more general NMF:

$$\min_H J_5 = \|W - HSH^T\|^2, \quad (33)$$

When the similarity matrix W is indefinite, W has negative eigenvalues. HH^T will not provide a good approximation, because HH^T can not absorb the subspace associated with negative eigenvalues. However, HSH^T can absorb subspaces associated with both positive and negative eigenvalues, i.e., the indefiniteness of W is passed on to S . This distinction is well-known in linear algebra where matrix factorizations have Cholesky factorization $A = LL^T$ if matrix A is s.p.d. Otherwise, one does $A = LDL^T$ factorization, where the diagonal matrix D takes care of the negative eigenvalues.

The second reason for nonnegative $W = HSH^T$ is that the extra degrees of freedom provided by S allow H to be more closer to the form of cluster indicators. This benefits occur for both s.p.d. W and indefinite W .

The third reason for nonnegative $W = HSH^T$ is that S provides a good characterization of the quality of the clustering. Generally speaking, given a fixed W and number of clusters κ , the residue of the matrix approximation $J_5^{\text{opt}} = \min \|W - HSH^T\|^2$ will be smaller than $J_1^{\text{opt}} = \min \|W - HH^T\|^2$. Furthermore, the K-by-K matrix S has a special meaning. To see this, let us assume H are vigorous cluster indicators, i.e., $H^T H = I$. Setting the derivative $\partial J_5 / \partial S = 0$, we obtain

$$S = H^T W H, \text{ or } S_{\ell k} = \mathbf{h}_\ell^T W \mathbf{h}_k = \frac{\sum_{i \in C_\ell} \sum_{j \in C_k} w_{ij}}{\sqrt{n_\ell n_k}} \quad (34)$$

S represents properly normalized within-cluster sum of weights ($\ell = k$) and between-cluster sum of weights ($\ell \neq k$). For this reason, we call this type of NMF as weighted NMF. The usefulness of weighted NMF is that if the clusters are well-separated, we would see the off-diagonal elements of S are much smaller than the diagonal elements of S .

The fourth reason is the consistency between standard $W = HH^T$ and $B = FG^T$. Since we can define a kernel as $W = B^T B$. Thus the factorization $W \approx B^T B \approx (FG^T)^T (FG^T) = G(F^T F)G^T$. Let $S = F^T F$, we obtain the weighted NMF.

6 Algorithms for computing symmetric NMF

We briefly outline the algorithms for computing symmetric factorizations $W = HH^T$ and $W = HSH^T$. For $W = HH^T$, the updating rule is

$$H_{ik} \leftarrow H_{ik} \left(1 - \beta + \beta \frac{(WH)_{ik}}{(HH^T H)_{ik}} \right). \quad (35)$$

where $0 < \beta \leq 1$. In practice, we find $\beta = 1/2$ is a good choice. A faster algorithm¹

$$H \leftarrow \max(WH(H^T H)^{-1}, 0). \quad (36)$$

can be used in the first stage of the iteration. Algorithmic issues of symmetric NMF is also studied in [2].

For weighted NMF $W = HSH^T$, the update rules are

$$S_{ik} \leftarrow S_{ik} \frac{(H^T W H)_{ik}}{(H^T H S H^T H)_{ik}}. \quad (37)$$

¹For the nonsymmetric NMF of Eq.(1), the algorithm is $F \leftarrow \max(BG(G^T G)^{-1}, 0)$, $G \leftarrow \max(B^T F(F^T F)^{-1}, 0)$. Without nonnegative constraints, these algorithms converge respectively to global optimal solutions of J_1 in Eq.(11) and J_2 in Eq.(23).

$$H_{ik} \leftarrow H_{ik} \left(1 - \beta + \beta \frac{(WHS)_{ik}}{(HSH^T HS)_{ik}} \right). \quad (38)$$

Acknowledgement. This work is supported partially by U.S. Department of Energy, Office of Science, under contract DE-AC03-76SF00098 through a LBNL LDRD grant.

References

- [1] P.K. Chan, M.Schlag, and J.Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Trans. CAD-Integrated Circuits and Systems*, 13:1088–1096, 1994.
- [2] M. Catral, L. Han, M. Neumann, and R.J. Plemmons. On reduced rank nonnegative matrix factorizations for symmetric matrices. *Linear Algebra and Its Applications*, to appear.
- [3] C. Ding and X. He. K-means clustering and principal component analysis. *LBNL-52983. Int'l Conf. Machine Learning (ICML2004)*, 2004.
- [4] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. *Proc. IEEE Int'l Conf. Data Mining*, 2001.
- [5] M. Gu, H. Zha, C. Ding, X. He, and H. Simon. Spectral relaxation models and structure analysis for k-way graph clustering and bi-clustering. *Penn State Univ Tech Report CSE-01-007*, 2001.
- [6] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Design*, 11:1074–1085, 1992.
- [7] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.
- [8] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13. 2001.
- [9] S.Z. Li, X. Hou, H. Zhang, Q. Cheng. Learning spatially localized, parts-based representation. In *Proc. IEEE Computer Vision and Pattern Recognition*, pp:207–212, Hawaii, 2001.
- [10] T. Li and S. Ma. IFD: Iterative feature and data clustering. In *Proc SIAM Int'l conf. on Data Mining (SDM 2004)*, pages 472–476, 2004.
- [11] V. P. Pauca, F. Shahnaz, M.W. Berry, and R. J. Plemmons. Text mining using non-negative matrix factorization. In *Proc SIAM Int'l conf. Data Mining (SDM 2004)*, pages 452–456, 2004.
- [12] F. Sha, L.K. Saul, and D.D. Lee. Multiplicative updates for nonnegative quadratic programming in support vector machines. *Advances in Neural Information Processing Systems 15*, pp:1041–1048. MIT Press, Cambridge, MA, 2003.
- [13] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.
- [14] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proc. SIGIR pp.267–273*, 2003.
- [15] H. Zha, C. Ding, M. Gu, X. He, and H.D. Simon. Spectral relaxation for K-means clustering. *Advances in Neural Information Processing Systems 14 (NIPS'01)*, pages 1057–1064, 2002.

Kronecker Factorization for Speeding up Kernel Machines

Gang Wu, Zhihua Zhang, and Edward Chang

Electrical and Computer Engineering, UCSB

{gwu@engineering, zzhang@mmdb.ece, echang@ece}.ucsb.edu

Abstract

In kernel machines, such as kernel principal component analysis (KPCA), Gaussian Processes (GPs), and Support Vector Machines (SVMs), the computational complexity of finding a solution is $O(n^3)$, where n is the number of training instances. To reduce this expensive computational complexity, we propose using *Kronecker factorization*, which approximates a positive definite kernel matrix by the *Kronecker product* of two smaller positive definite matrices. This approximation can speed up the calculation of the kernel-matrix inverse or eigendecomposition involved in kernel machines. When the two factorized matrices have about the same dimensions, the computational complexity is improved from $O(n^3)$ to $O(n^2)$. We propose two methods to carry out Kronecker factorization and apply them to speed up KPCA. In Experiments show that our methods can drastically reduce the computation time of kernel machines without any significant degradation in their effectiveness.

1 Introduction

The kernel matrix, such as the *Gram matrix* in Kernel PCA (KPCA) [5] and SVMs, and the covariance matrix in Gaussian Processes (GPs), plays an important role in kernel machines. In general, these machines are required to calculate the inverse of the kernel matrix or to make the eigendecomposition on it. Both operations take $O(n^3)$ where n denotes the number of training instances. Specifically, KPCA requires an eigendecomposition on an $n \times n$ Gram matrix, SVMs need to resolve a quadratic programming problem that involves an $n \times n$ Gram matrix, and GPs need to invert an $n \times n$ covariance matrix. Several methods have been proposed to address this problem of high computational cost, such as randomized techniques [1], sparse greedy approximation [6], and the Nyström method [8], etc. All these methods are based on sampling techniques. In this paper, we tackle the same computational challenge by a different route, using the *Kronecker product* of the matrices.

Suppose that two matrices $\mathbf{B} = [b_{ij}]$ and $\mathbf{C} = [c_{ij}]$ are $m_1 \times n_1$ and $m_2 \times n_2$, respectively. The Kronecker product of \mathbf{B} and \mathbf{C} (denoted by $\mathbf{B} \otimes \mathbf{C}$) is an $m_1 m_2 \times n_1 n_2$

matrix, defined as the following block matrix

$$\mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} b_{11}\mathbf{C} & \cdots & b_{1n_1}\mathbf{C} \\ \vdots & \ddots & \vdots \\ b_{m_1 1}\mathbf{C} & \cdots & b_{m_1 n_1}\mathbf{C} \end{bmatrix}.$$

Conversely, let \mathbf{A} be an $m \times n$ matrix with $m = m_1 m_2$ and $n = n_1 n_2$. Our current problem is to find two matrices \mathbf{B} ($m_1 \times n_1$) and \mathbf{C} ($m_2 \times n_2$) so that $\mathbf{B} \otimes \mathbf{C}$ approximates \mathbf{A} . We denote this problem as the *Kronecker factorization* of the matrix \mathbf{A} . A generic structure of the Kronecker factorization and its applications are detailed in [7]. In this paper, we focus on a special case in that the matrices \mathbf{A} , \mathbf{B} and, \mathbf{C} are all symmetric positive definite.

1.1 Related Work

The topic of speeding up kernel machines has been an important and actively research one. However, the scalability of kernel machines is still a serious problem if they are to be used for large-scale problems. Many sampling based methods have been proposed to reduce the $O(n^3)$ computational time of kernel machines. Take SVMs as an example. (Since SVMs involve solving a quadratic programming optimization problem, its computational complexity is $O(n^3)$.) Sampling based algorithms such as Osuna's Decomposition algorithm [3] and SMO [4] are able to reduce the training time to $O(n^2)$. Other sampling based algorithms for KPCA and GPs such as [1, 4, 6, 8] (details are presented shortly) can also achieve speedup of an order of magnitude. However, large-scale applications demand even faster algorithms. The Kronecker factorization enjoys two advantages over the sampling based methods. First, the factorization can be recursively applied to a large matrix. Second, this divide-and-conquer approach can be parallelized and take advantage of the new generation *multi-core processor architecture* (multiple processors included on one chip). On the contrary, traditional iterative methods (e.g., SMO) can hardly be parallelized because of strong data dependencies between iterations.

Due to the space limitations, we will leave GPs and SVMs to a future, comprehensive treatment. In this paper, we focus on speeding up Kernel methods that require an eigendecomposition on kernel matrix such as

KPCA and KICA. For this purpose, Williams et al [8] propose to use the Nyström method to approximate the eigendecomposition of the Gram matrix \mathbf{K} . Specifically, the authors use a reduced-rank kernel matrix generated from a set of randomly sampled training data of size $m < n$. Then, the Nyström method is applied on the eigenspectrum (eigenvalues and eigenvectors) of the reduced-rank kernel matrix to recover the corresponding eigenspectrum of the large-size \mathbf{K} .

Smola et al [6] propose a sparse greedy approximation technique to construct a lower-rank representation of the Gram matrix \mathbf{K} . It turns out that the form of the Nyström approximation is almost identical to the sparse greedy matrix approximation [8]. The only difference comes from how to sample a set of training data so as to form a reduced-rank approximation of the Gram matrix. The Nyström method randomly samples training data, but the sparse greedy matrix method searches over the column or basis function space incrementally until a selection rule is satisfied. In [6], the authors defined three selection rules. For example, one rule could be the Frobenius norm of the difference between the approximated $\tilde{\mathbf{K}}$ and the original \mathbf{K} , $\|\mathbf{K} - \tilde{\mathbf{K}}\|_F^2$. However, as argued by the authors, there exists a tradeoff between the quality of the selection and the amount of computational resources needed to compute the best set of columns or basis functions. For a given m (the number of selected training data), the sparse greedy method has a better approximation to \mathbf{K} , but with more computational cost than the Nyström approximation. Considering both issues, in this paper, we use the Nyström method as a baseline to compare with our proposed Kronecker factorization method on speeding up KPCA.

One more method for speeding up KPCA is proposed by Achlioptas et al [1]. Similar with the Nyström and sparse greedy matrix method, this one is also based on sampling techniques. The basic idea is to randomly sample the kernel matrix \mathbf{K} according to some pre-defined probability so as to form a sparse kernel matrix. Such a sparse \mathbf{K} is then employed to accelerate the eigendecomposition in KPCA.

1.2 Contribution Summary

In summary, almost all traditional methods attempt to find a lower-rank matrix to approximate the kernel matrix by sampling the training dataset. The difference of these methods mainly lies in the way sampling is done. In this paper, we propose speeding up kernel machines by factorizing a large-size kernel matrix \mathbf{K} into multiple smaller ones and approximate the original \mathbf{K} using their Kronecker product. Since both eigendecomposition and inverse of \mathbf{K} can be recovered from the corresponding operations on multiple factorized smaller matrices, the computation of kernel machines can be parallelized.

Considering that the *multi-core architecture* is the trend of future processors, we believe that the Kronecker factorization approach is a viable one to allow kernel machines to scale. Our experiments show that our methods can reduce the computation time of kernel machines without significant degradation in their effectiveness.

2 Mathematical Background

Given a positive definite (p.d.) matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $n = n_1 n_2$, we have the following factorization problem

$$\mathbf{A} \triangleq \mathbf{B} \otimes \mathbf{C},$$

where $\mathbf{B} \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{C} \in \mathbb{R}^{n_2 \times n_2}$ are both positive definite. In other words, we factorize a large-size p.d. matrix into two small-size p.d. matrices with the Kronecker product. We call this formulation the *Kronecker factorization* of the positive definite matrix. Some theoretical properties of the Kronecker products [2] are listed below, which provide the foundation to apply the Kronecker factorization in our paper. For more details, such as the Kronecker products on multiple matrices, please refer to [2].

THEOREM 2.1. *If \mathbf{B} and \mathbf{C} are $n_1 \times n_1$ and $n_2 \times n_2$ respectively, then*

- (a) $(\mathbf{B} \otimes \mathbf{C})' = \mathbf{B}' \otimes \mathbf{C}'$;
- (b) $\text{tr}(\mathbf{B} \otimes \mathbf{C}) = \text{tr}(\mathbf{B})\text{tr}(\mathbf{C})$;
- (c) $|\mathbf{B} \otimes \mathbf{C}| = |\mathbf{B}|^{n_2} \cdot |\mathbf{C}|^{n_1}$; and
- (d) *If \mathbf{B} and \mathbf{C} are nonsingular, then $(\mathbf{B} \otimes \mathbf{C})^{-1} = \mathbf{B}^{-1} \otimes \mathbf{C}^{-1}$.*

THEOREM 2.2. *Let $\sigma(\mathbf{B})$, $\sigma(\mathbf{C})$ and $\sigma(\mathbf{B} \otimes \mathbf{C})$ be the spectrums (or set of eigenvalues) of \mathbf{B} , \mathbf{C} , and $\mathbf{B} \otimes \mathbf{C}$, respectively. If $\lambda \in \sigma(\mathbf{B})$ and \mathbf{b} is the corresponding eigenvector of \mathbf{B} ; and if $\mu \in \sigma(\mathbf{C})$ and \mathbf{c} is the corresponding eigenvector of \mathbf{C} , then $\lambda\mu \in \sigma(\mathbf{B} \otimes \mathbf{C})$ and $\mathbf{b} \otimes \mathbf{c}$ is the corresponding eigenvector of $\mathbf{B} \otimes \mathbf{C}$. Furthermore, if $\sigma(\mathbf{B}) = \{\lambda_1, \dots, \lambda_{n_1}\}$ and $\sigma(\mathbf{C}) = \{\mu_1, \dots, \mu_{n_2}\}$, then $\sigma(\mathbf{B} \otimes \mathbf{C}) = \{\lambda_i \mu_j : i = 1, \dots, n_1, j = 1, \dots, n_2\}$.*

For an $n \times n$ matrix \mathbf{A} , the computational complexity is $O(n^3)$ to invert it or to make its eigendecomposition. The memory usage is $O(n^2)$. By means of the Kronecker factorization, the complexity of the same operations is reduced to $O(n_1^3) + O(n_2^3)$, following the above theorems. Moreover, the memory usage is reduced to $O(n_1^2) + O(n_2^2)$. Thus, the Kronecker factorization can effectively reduce the computational complexity. Especially when $n_1 \approx n_2$, the computation complexity of kernel machines is greatly reduced to $O(n^{\frac{3}{2}})$. Please note that this $O(n^{\frac{3}{2}})$ does not consider the factorization cost. When that cost is added, as we will show in Section 3, the computation complexity is $O(n^2)$.

3 Kronecker Factorization and KPCA

Given \mathbf{A} , our problem is to estimate $\mathbf{B} = [b_{ij}]$ and $\mathbf{C} = [c_{ij}]$. To achieve this goal, we propose two

iterative methods based on two different criteria, i.e., the least-squares (LS) error and the Kullback-Leibler (KL) divergence. At the end of this section, we show how these methods are applied to speed up KPCA. Because of the space limitation, we refer the reader to [9] for detailed derivations.

Partition \mathbf{A} into the block matrices, where \mathbf{A}_{ij} 's ($i, j = 1, \dots, n_1$) are all $n_2 \times n_2$ and $\hat{\mathbf{A}}_{ij}$'s ($i, j = 1, \dots, n_2$) are all $n_1 \times n_1$. Since \mathbf{A} is symmetric positive definite, it is clear that $\mathbf{A}_{ji}^T = \mathbf{A}_{ij}$ and $\hat{\mathbf{A}}_{ji}^T = \hat{\mathbf{A}}_{ij}$. Moreover, \mathbf{A}_{ii} 's and $\hat{\mathbf{A}}_{ii}$'s are positive definite.

3.1 Least-Squares Method

The first method is derived from minimizing the least-squares error defined in [7], which is expressed as follows:

$$\begin{aligned} e_{\mathbf{A}}(\mathbf{B}, \mathbf{C}) &= \|\mathbf{A} - \mathbf{B} \otimes \mathbf{C}\|_F^2 \\ (3.1) \quad &= \text{tr}((\mathbf{A} - \mathbf{B} \otimes \mathbf{C})'(\mathbf{A} - \mathbf{B} \otimes \mathbf{C})), \end{aligned}$$

We attempt to find a p.d. \mathbf{B} ($n_1 \times n_1$) and a p.d. \mathbf{C} ($n_2 \times n_2$), which minimize $e_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$. Practically, we use a *separable least squares framework*, which consists of two successive parts, to solve this minimization problem. The computational cost of this LS-based factorization algorithm is $O(n_1^2 n_2^2) + O(n_2^2 n_1^2) \approx O(n^2)$ and the memory usage $O(n_1^2 + n_2^2)$.

3.2 Kullback-Leibler Divergence Method

The second method is derived from minimizing the KL divergence between \mathbf{A} and $\mathbf{B} \otimes \mathbf{C}$, which is defined as follows:

$$\begin{aligned} d_{\mathbf{A}}(\mathbf{B}, \mathbf{C}) &= \text{tr}(\mathbf{A}(\mathbf{B} \otimes \mathbf{C})^{-1}) - \log |\mathbf{A}(\mathbf{B} \otimes \mathbf{C})^{-1}| - n \\ &= \text{tr}(\mathbf{A}(\mathbf{B}^{-1} \otimes \mathbf{C}^{-1})) - \log |\mathbf{A}| \\ (3.2) \quad &- n_2 \log |\mathbf{B}^{-1}| - n_1 \log |\mathbf{C}^{-1}| - n. \end{aligned}$$

Similar to $e_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ in Equation 3.1, $d_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ arrives at its minimum 0 iff $\mathbf{A} = \mathbf{B} \otimes \mathbf{C}$. We attempt to find positive definite \mathbf{B} and \mathbf{C} that minimize $d_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$.

To solve the optimization problem of minimizing $d_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ with respect to \mathbf{B} and \mathbf{C} , we also use a separable framework and devise an iterative algorithm. Both \mathbf{B} and \mathbf{C} are positive definitive. For the proof and the detail of the algorithm, please consult [9]. The computational cost of this method is $O(n_2^3 + n_1^2 n_2^2) + O(n_1^3 + n_2^2 n_1^2) \approx O(n^2)$ and the memory usage is $O(n_1^2 + n_2^2)$. Compared with that of the LS-based method presented in Section 3.1, the computational cost of our proposed method is slightly higher.

3.3 Speeding up KPCA

According to Theorem 2.2, the eigen-spectrum of a large-size matrix can be calculated from the eigen-spectrums of its factorized small-size matrices. This enables us to apply the Kronecker factorization in KPCA for reducing its computational complexity. Figure 1 illustrates the

Input: original kernel matrix \mathbf{K} ($n \times n$), the number of principle components to be extracted m .

Output: dataset $\tilde{\mathcal{X}}$ with only extracted principle components after KPCA.

1. Run Kronecker factorization algorithm on \mathbf{K} to factorize two small-size matrices \mathbf{B} and \mathbf{C} .
2. Run KPCA on \mathbf{B} and \mathbf{C} to generate the corresponding eigen-spectrum $\sigma(\mathbf{B})$ and $\sigma(\mathbf{C})$, respectively.
3. Choose m largest $\lambda\mu$, where $\lambda \in \sigma(\mathbf{B})$ and $\mu \in \sigma(\mathbf{C})$, and calculate a set of eigenvectors \mathcal{U}_{eig} via $\mathbf{x} \otimes \mathbf{y}$, where \mathbf{x} and \mathbf{y} are eigenvectors whose corresponding eigenvalues are λ and μ , respectively.
4. Calculate the new dataset $\tilde{\mathcal{X}}$ after KPCA by $K * [u_1, u_2, \dots, u_m]$, where u_i is one eigenvector from the set \mathcal{U}_{eig} with the i -th largest eigenvalue $\lambda\mu$.

Figure 1: Kronecker Factorization for KPCA

detailed algorithm. Note that in general, KPCA works with a centered kernel matrix. In this case, the centered kernel matrix becomes singular, so theoretically we are not allowed to implement the Kronecker factorization on the matrix. However, they can still work in practice, since the two factorization algorithms involve the trace operations of only sub-block matrices of the original matrix. Another alternative way is to first implement the Kronecker factorization on the original kernel matrix and then to centerize the two factorized matrices. In the experiments of this paper, we choose the latter method.

4 Experiments

We used both artificial and real-world datasets to examine our proposed Kronecker factorization algorithms and compare them with one representative sampling-based method, the Nyström method, for speeding up KPCA.

The datasets used in our experiments are eight toy datasets (TOY), the USPS handwritten digit dataset, and the ABALONE dataset from the UCI Repository. Each of eight toy datasets has binary classes and ten features. The first feature was generated according to the Gaussian distribution $N(1, 1)$ for one class and $N(-1, 1)$ for the other class. The other nine features were generated according to $N(0, 3)$ so as to make the datasets noisy. Eight toy datasets are of size 100, 400, 900, 1600, 2500, 3600, 4900, and 6400, respectively. For each toy dataset, we randomly extracted about 67% as training data and the rest 33% as test data. The USPS dataset has 7291 training instances and 2007 test instances, each of which has 256 features. This dataset has 10 classes, corresponding to the digits 0, ..., 9. The ABALONE dataset has 3000 training instances and 1177 test instances, each of which has 8 dimensions. ABALONE has 29 classes. In the experiments, we set its first 10 classes to be positive and the remaining 19 classes to be negative so as to form a binary dataset.

To achieve the maximum performance of our algo-

gorithms, we chose the sizes of two factorized matrices close to \sqrt{n} . When n couldn't be factorized, we simply removed some training instances. Our experiments showed that for a large-size dataset, doing so had almost no influence on the performance of KPCA and GPs. Following the experimental setup in [6, 8], we used the Gaussian kernel, $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$. For TOY datasets, we chose $\sigma = 10.0$. For USPS and ABALONE, we followed the parameter settings in [6, 8] and then chose $\sigma^2 = 0.5d$, where d is the dimensionality of the data. In our experiments, we initialize $\mathbf{C}(0) = \mathbf{I}$ and chose 5 as the running iterations in our algorithms. This heuristic number of iterations came from our observations in the experiments that the Kronecker factorization algorithms became convergent after about 5 iterations.

For a better illustration of comparison, we used a modified measurement for each algorithm to report the experimental results. Instead of just using $e_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ in Eqn. 3.1 and $d_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ in Eqn. 3.2, we used the average least-square error defined as $\frac{e_{\mathbf{A}}(\mathbf{B}, \mathbf{C})}{n^2}$ for the factorization method based on least-square error, and used the decimal logarithm $\log_{10} d_{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ for the method based on Kullback-Leibler divergence,

4.1 Experiments on KPCA

In this experiment, we examined the effectiveness and efficiency of our proposed Kronecker factorization on KPCA. We compared with the regular KPCA without any speedup and the Nystrom method, which is one representative sampling-based method for speeding up KPCA and GPs. Moreover, besides examining Kronecker factorization applied at one level, we also examined the factorization applied at two levels, which was to run factorization one more time on the factorized \mathbf{B} and \mathbf{C} so that $\mathbf{B} = \mathbf{B}_1 \otimes \mathbf{B}_2$ and $\mathbf{C} = \mathbf{C}_1 \otimes \mathbf{C}_2$. From Theorem 2.2, the eigen-spectrum $\sigma(\mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \mathbf{C}_1 \otimes \mathbf{C}_2)$ can be exactly recovered from the eigen-spectrum of the smaller-size matrices, which can be conducted in parallel¹.

Since running KPCA on the whole USPS can be very time-consuming on our hardware platform², we followed the setting reported in [5] by randomly sampling 5,000 out of 7291 instances as training data, and 1400 out of 2,007 as test data. We set the size of matrices \mathbf{B} and \mathbf{C} to be 80, and the sizes of \mathbf{B}_1 , \mathbf{B}_2 , \mathbf{C}_1 , and \mathbf{C}_2 to be 8, 10, 8, and 10, respectively. For ABALONE, to make the Kronecker factorization feasible, we removed

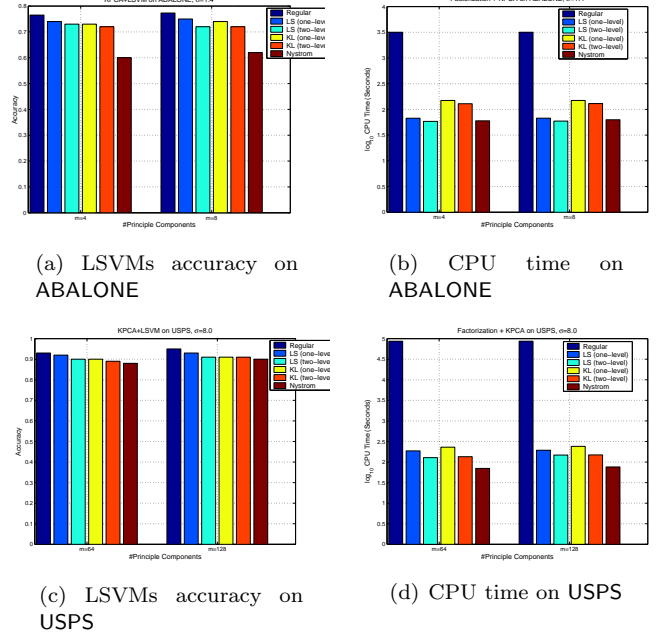


Figure 2: Comparison of Speeding Up KPCA: For each figure, x -axis denotes the number of principle components which is pre-defined before running KPCA, y -axis denotes the accuracy of running LSVMs after KPCA in (a) and (c), or the CPU time (in the decimal logarithm) of running factorization (if possible) and KPCA together. In each figure, from left to right, the six bars represent the results by running regular KPCA (denoted as Regular in the legend), KPCA after a one-level factorization based on least square error (LS (one-level)), KPCA after a two-level factorization based on least square error (LS (two-level)), KPCA after a one-level factorization based on KL divergence (KL (one-level)), KPCA after a two-level factorization based on KL divergence (KL (two-level)), and KPCA approximated by the Nystrom method (Nystrom).

one instance³ from the training dataset so as to form two feasible small matrices, a 58×58 matrix \mathbf{B} and a 72×72 matrix \mathbf{C} ($58 \times 72 = 4176$). Then, we set the sizes of \mathbf{B}_1 , \mathbf{B}_2 , \mathbf{C}_1 , and \mathbf{C}_2 to be 2, 29, 8, and 9, respectively. As for the Nystrom method, we selected the number of sampled instances to be 80 for USPS and 60 for ABALONE so that its computational cost could be almost the same or even less than that of our algorithms.

In order to quantitatively evaluate the performance of each approximate method for KPCA, we ran linear SVMs (LSVMs) on the dimensionality-reduced dataset

¹Since we do not have a hardware platform to conduct the parallelized experiments, we just sequentially conducted the KPCA on each small matrix, which could not show the benefits of two-level factorization.

²All experiments were done on a PIII 900MHZ Workstation with 1GB memory.

³To achieve the maximum performance of our algorithms, we choose the sizes of two factorized matrices to \sqrt{n} . When n cannot be factorized, we simply remove some training instances. Our experiments show that doing so does not impact the performance of KPCA and GPs.

via KPCA following the setting in [5]. Figures 2(a) and (c) give the classification accuracy on ABALONE and USPS, respectively. Figures 2(b) and (d) show the corresponding running time of KPCA (in log-scale), which includes the time of running the factorization for our method and of running the approximation for the Nystrom method. From the figures, we make three observations. First, compared to the regular KPCA using the full kernel matrix, all our methods, one-level and two-level factorization based on LS or KL, can reduce the wall-clock time of running KPCA by hundreds of times while maintaining the same level of accuracy. (Again, Figures 2(b) and (d) are plotted in the log scale.) Second, the Nyström method, which we intentionally forced to have almost the same computational time as our methods, did not approximate well to the regular KPCA, especially for ABALONE. This means that more instances need to be sampled for Nyström, which will increase its computational time. Third, compared to one-level factorization, two-level factorization indeed helps speeding up the KPCA, and still enjoys a good approximation to the regular KPCA. For our experiment, we also found that the factorization part took up about 70% of the total running time, including the time for conducting factorization and running kernel machines. We thus believe that running KPCA on multiple smaller matrices in parallel can further reduce the 30% part of computational time (which is not reflected in the figures).

5 Concluding Remarks

In this paper, we have presented the idea of *Kronecker factorization* of the positive definite matrix to speed up kernel machines. Specifically, we factorize a large-size matrix \mathbf{A} into two considerably smaller matrices \mathbf{B} and \mathbf{C} , and we approximate \mathbf{A} with the Kronecker product of \mathbf{B} and \mathbf{C} . Our empirical studies showed that the proposed method can substantially speed up KPCA while maintaining high class-prediction accuracy. Please refer to [9] for detailed mathematical derivations and also the success of applying *Kronecker factorization* to GPs. Our future research plans to parallelize our proposed algorithms and apply the factorization recursively when the matrix dimension is very large.

References

- [1] D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. *Advances in Neural Information Processing Systems 14*, 2002.
- [2] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, UK, 1991.
- [3] E. Osuna and F. Girosi. Reducing runtime complexity of svms. Technical Report Proceedings of the 14th International Conference on Pattern Recognition, 1998.

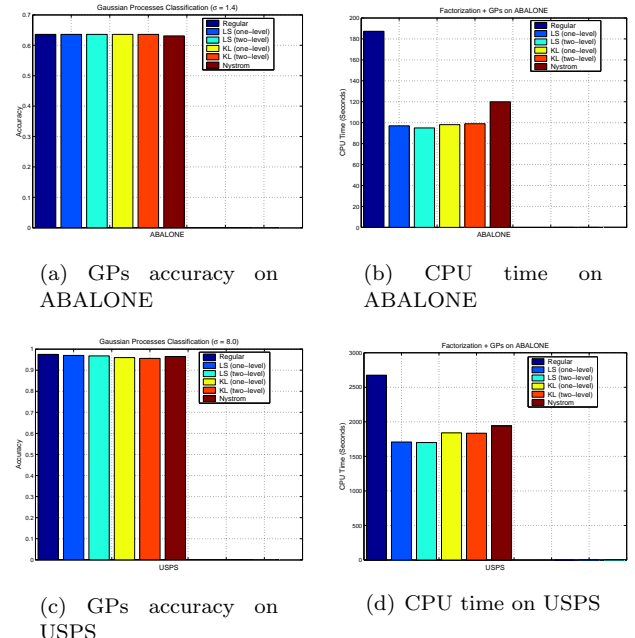


Figure 3: Comparison of Speeding Up GPs. (a) and (c) illustrate the test accuracy on Gaussian processes with different approximate methods. (b) and (d) then illustrate the corresponding cpu time (in seconds) of running factorization and GPs together. In each figure, from the left to the right, the six bars represent the results by running regular GPs (denoted as Regular in the legend), GPs after a one-level factorization based on least square error (LS (one-level)), GPs after a two-level factorization based on least square error (LS (two-level)), GPs after a one-level factorization based on KL divergence (KL (one-level)), GPs after a two-level factorization based on KL divergence (KL (two-level)), and GPs approximated by the Nyström method (Nystrom).

- [4] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.
- [5] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
- [6] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *The 17th International Conference on Machine Learning*, 2000.
- [7] C. F. Van Loan and N. Pitslanis. Approximation with kronecker products. In M. Moonen, G. Golub, and B. de Moor, editors, *Linear Algebra for Large Scale and Real-Time Applications*, pages 293–314. Kluwer Academic Publisher, Dordrecht, 1993.
- [8] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 2001.
- [9] G. Wu, Z. Zhang, and E. Chang. Kronecker factorization for speeding up kernel machines (extended version). *UCSB Technical Report*, June 2004.

Symmetric Statistical Translation Models for Automatic Image Annotation

Feng Kang and Rong Jin*

Abstract

Automatic image annotation provides means for users to search image collections on the semantic level using natural language queries. In the past, statistical machine translation models have been successfully applied to automatic image annotation. A problem with this approach is that, due to the skewed distribution of term frequency for annotation words, common words have been overly favored, which leaves little room for uncommon words to be used in auto-annotations. In contrast, studies on information retrieval have revealed that uncommon words are at least as important as common words since they are also often used in users' queries. Unlike the previous studies where a single type of statistical translation model is considered for automatic image annotation, in this paper, we studied two types of statistical translation models: a forward translation model, which *translates* visual information into textual words, and a backward model, which *translates* textual words into visual images. In particular, we propose a new statistical translation model, named regularization-based symmetric statistical translation model, which combines strength of forward and backward models to alleviate the problem of overly favoring common words. Our empirical studies with the Corel dataset have shown that the proposed model performs considerably better than the existing translation model and a state-of-the-art approach for automatic image annotation.

1 Introduction.

Efficient access to image databases requires the ability to search and organize images effectively. While images could be retrieved based on their features such as color, texture, it is usually more natural and desirable for users to search image databases using textual queries. One important reason is that textual queries allow users to express their information needs on the semantic level instead of the level of preliminary image features.

A key to image retrieval using textual queries is image annotation. Given annotated words for images, the problem of image retrieval becomes a problem of textual retrieval. Many well-developed textual retrieval algorithms, such as language modeling approaches [11,

12, 18, 19], can be applied to find images that are relevant to textual queries. Since manual annotation is usually expensive and subjective, many methods have been developed to annotate images automatically [1-3, 5, 6, 8, 10, 13-17].

A statistical machine translation model for automatic image annotation [8] views the process of annotating images as a process of translating information from a 'visual language' to textual words. Images are first segmented into different regions, which are further grouped into a number of clusters, or image blobs as in [8]. Then, correspondence between image blobs and annotated words is learned through a statistical machine translation model [4].

One difficulty with translation models for automatic image annotation arises from the skewed distribution of word frequency. According to [4], a key for translation models to disambiguate the alignment between image regions and annotated words is the co-occurrence statistics. If an image blob co-occurs more frequently with a word 'A' than with any other words, it will be more likely for the image blob to be associated with 'A'. According to [16], the term frequency of annotation words follows the Zipf's law, namely a small number of words appear very often and most words are used only by a few images. As a result, a common word can *accidentally* co-occur with a blob that in fact is more associated with an uncommon word. The problem with the co-occurrence statistics is further complicated by the fact that massive number of image regions are first clustered into a small number of blobs. Very often, image regions for different annotation words have similar distributions over the space of image features and thus are clustered into the same group (i.e., image blob). As a result, an image region related to a rare word could be grouped with other image regions related with common words, which leads to more errors in co-occurrence statistics.

To correct the potential errors in the co-occurrence statistics, we examine two types of translation models. Most previous studies on translation models for automatic image annotation focus on the model that *translates* image regions/blobs into textual words, which is called *forward translation model* in this paper. Apparently, we can apply the translation model in a reverse way, namely translating textual words into im-

*Department of Computer Science and Engineering, Michigan State University, East Lansing MI 48824. {kangfeng, rongjin}@cse.msu.edu

	w1	w2
b1	100	50
b2	200	35

Forward translation model $p(w|b)$

	w1	w2
b1	0.67	0.33
b2	0.85	0.15

Backward translation model $p(b|w)$

	w1	w2
b1	0.33	0.58
b2	0.67	0.42

Table 1: An example of forward and backward translation models for two image blobs(b1 and b2), and two words (w1 and w2).

age blobs, which we call a *backward translation model*. These two kinds of translation models make different assumptions between blobs and words. The forward translation model assumes that each image blob is translated into a single word, while each word can be translated into multiple blobs. The backward translation model is based on the assumption that each word is translated into a single image blob.

In order to better illustrate the difference between these two types of translation models, consider a simple example of co-occurrence statistics shown in Table1. On one hand, for the forward translation model, the translation probabilities for word ‘w1’ are dominative for both image blobs. It is unlikely for word ‘w2’ to be used in any auto-annotations. On the other hand, for the backward translation model, we do find that image blob ‘b1’ is strongly associated with word ‘w2’ and the chance for image blob ‘b2’ to be associated with word ‘w2’ is also high. The difference motivates us to propose a symmetric model, which combines the two models together.

The rest of this paper is arranged as follows: Section 2 describes the background knowledge; Section 3 describes the proposed symmetric translation model that combines the forward and backward translation models; The empirical studies are described in Section 4; Section 5 draws the conclusions.

2 Related Work.

In this section, we describe the overview of statistical methods for automatic image annotation with the focus on translation model approaches.

2.1 Automatic Image Annotation. The key to automatic image annotation is to learn the annotation

models that automatically predict annotation words given extracted image features. A variety of machine learning methods have been applied to automatic image annotation, including machine translation model [8], co-occurrence model [17], latent space approaches [1, 16], graphic models [3], classification approaches [5, 6, 14], and relevance language models [10, 13]. The co-occurrence model [17] collects the co-occurrence counts between words and image features and uses them to predict annotated words for images. Duygulu et al. [8] improved the co-occurrence model by utilizing machine translation models. Another way of capturing co-occurrence information is to introduce latent variables that link image features with words. Methods in this category include latent semantic analysis (LSA), probabilistic latent semantic analysis (PLSA) [16], hierarchical aspect model [1], Gaussian Mixture Model (GMM), Latent Dirichlet Allocator (LDA), and correspondence LDA [3]. The classification approaches for automatic image annotation treat each annotated word as an independent class and create a different image classification model for every word. Work such as linguistic indexing of pictures [14], image annotation using SVM [6] and Bayes point machine [5] fall into this category. More recently, relevance language models have been applied to automatic image annotation [10, 13]. Empirical studies [10, 13] have shown that relevance language models for image annotation are better than translation models.

2.2 Machine Translation Models for Automatic Image Annotation. Using the IBM translation model I [4, 8], the probability of annotating image blobs $\vec{b}_i = \{b_{i,1}, b_{i,2}, \dots, b_{i,m}\}$ with words $\vec{w}_i = \{w_{i,1}, w_{i,2}, \dots, w_{i,n}\}$, i.e., $p(\vec{w}_i | \vec{b}_i)$, is expressed as follows:

$$\begin{aligned}
 p(\vec{w}_i | \vec{b}_i) &= \prod_{j=1}^n p(w_{i,j} | \vec{b}_i) \\
 &\propto \prod_{\{j | w_{i,j}=1\}} \sum_{k=1}^m t_{j,k} b_{i,k}
 \end{aligned}
 \tag{2.1}$$

where $t_{i,j}$ stands for the probability of translating the k -th blob into the j -th word. In order to annotate an image $I = \{b_1, b_2, \dots, b_m\}$, (2.1) is applied to find the set of words \vec{w} that maximizes $p(\vec{w} | \vec{b})$. The translation probabilities $\{t_{j,k}\}$ can be obtained by maximizing the log-likelihood of training images, i.e.,

$$\begin{aligned}
 \log l(T) &= \sum_{i=1}^{|T|} p(\vec{w}_i | \vec{b}_i) \\
 &= \sum_{i=1}^{|T|} \sum_{j=1}^n w_{i,j} \log \left(\sum_{k=1}^m t_{j,k} b_{i,k} \right)
 \end{aligned}
 \tag{2.2}$$

Expectation-Maximization (EM) algorithm [7] is applied to find the optimal solution for (2.2) and updating equation is:

$$(2.3) \quad t_{j,k}^{new} = \frac{1}{Z_k} \sum_i \frac{w_{i,j} b_{j,k} t_{j,k}^{old}}{\sum_{k'} w_{j,k'} t_{j,k'}^{old}}$$

Z_k is a normalization factor that ensures $\sum_j t_{j,k}^{new} = 1$, namely each blob has to be translated into a single annotation word. Detailed EM algorithm is described in [4]. The above translation model takes the direction of translating image blobs into words, which we call *forward translation model*. We can take another direction of translation, i.e., translating words into blobs, which we call *backward translation model*. In backward translation model, the translation probability from annotation words to image blobs is written as:

$$(2.4) \quad p(\vec{b}_i | \vec{w}_i) = \prod_{j=1}^m p(b_{i,j} | \vec{w}_i) \propto \prod_{j=1}^m \left\{ \sum_{k=1}^m u_{j,k} w_{i,k} \right\}^{b_{i,j}}$$

Similarly, EM is used to find the set of translation probabilities and the updating equation is written as:

$$(2.5) \quad u_{j,k}^{new} = \frac{1}{Z_k} \sum_i \frac{w_{i,j} b_{i,k} u_{j,k}^{old}}{\sum_{k'} w_{i,k'} u_{j,k'}^{old}}$$

where $u_{j,k}$ stands for the probability of translating the k -th word into the j -th blob. Z_k is a normalization factor that ensures $\sum_j u_{j,k}^{new} = 1$, namely each word has to be translated into a single image blob.

In next section, we propose a new translation model, which combines the forward and backward translation models to enhance the quality of automatic image annotations.

3 Regularization-based Symmetric Translation Model: Combining the Forward and Backward Translation Models.

The main idea of the proposed model, a **regularization-based symmetric translation model (RSTM)**, is first to examine the discrepancy between the forward and backward models, and then to correct them by utilizing the information across the two models. We first introduce a symmetric KL divergence term that measures the discrepancy between the forward and backward models:

$$(3.6) \quad KL = \sum_j \sum_k p(w_j, b_k; f) \log\left(\frac{p(w_j, b_k; f)}{p(w_j, b_k; b)}\right) + \sum_j \sum_k p(w_j, b_k; b) \log\left(\frac{p(w_j, b_k; b)}{p(w_j, b_k; f)}\right)$$

According to the property of KL divergence, the above expression becomes zero iff $p(w_j, b_k; f) = p(w_j, b_k; b)$ for any $j \in [1...n]$ and $k \in [1...m]$. Then, we add the KL divergence term into the objective function as the regularization term to ensure the consistency between the forward and backward translation models:

$$(3.7) \quad \Omega_{RSTM} = \underbrace{\left\{ \sum_{i=1}^{|T|} \sum_{\{w_{i,j}=1\}} \log\left(\sum_{k=1}^m t_{j,k} b_{i,k}\right) + \sum_{i=1}^{|T|} \sum_{\{b_{i,k}=1\}} \log\left(\sum_{j=1}^n u_{k,j} w_{i,j}\right) \right\}}_{\text{translation}} - \lambda \underbrace{\left\{ \sum_j \sum_k p(w_j, b_k; f) \log\left(\frac{p(w_j, b_k; f)}{p(w_j, b_k; b)}\right) + \sum_j \sum_k p(w_j, b_k; b) \log\left(\frac{p(w_j, b_k; b)}{p(w_j, b_k; f)}\right) \right\}}_{\text{regularization}}$$

where λ is to determine the degree of consistency between the two models. Efficiently finding the optimal solution to (3.7) is more complicated than the EM algorithm for standard translation model. Here, we list the updating equations for the forward and backward translation models, and leave out the detailed derivation for brevity.

$$(3.8) \quad t_{j,k}^{new} = \frac{2C_{j,k}}{B_{j,k} + \sqrt{B_{j,k}^2 + 4A_{j,k}C_{j,k}}}$$

$$\begin{aligned} A_{j,k} &= 2\lambda \frac{p(b_k)}{t_{j,k}^{old}} \\ B_{j,k} &= \lambda p(b_k) \log(t_{j,k}^{old}) - \lambda p(b_k) \log(u_{k,j} p(w_j)) + \alpha_k \\ C_{j,k} &= \sum_{i=1}^{|T|} \frac{t_{j,k}^{old} b_{i,k}}{\sum_{k'=1}^m t_{j,k'}^{old} b_{i,k'}} + \lambda p(w_j) u_{k,j} \end{aligned}$$

where α_k is the normalization factor that ensures $\sum_j t_{j,k}^{new} = 1$.

$$(3.9) \quad u_{k,j}^{new} = \frac{2F_{k,j}}{E_{k,j} + \sqrt{E_{k,j}^2 + 4D_{k,j}F_{k,j}}}$$

$$\begin{aligned} D_{k,j} &= 2\lambda \frac{p(w_j)}{u_{k,j}^{old}} \\ E_{k,j} &= \lambda p(w_j) \log(u_{k,j}^{old}) - \lambda p(w_j) \log(t_{j,k} p(b_k)) + \beta_j \\ F_{k,j} &= \sum_{i=1}^{|T|} \frac{u_{k,j}^{old} w_{i,j}}{\sum_{j'=1}^n u_{k,j'}^{old} w_{i,j'}} + \lambda p(b_k) t_{j,k} \end{aligned}$$

where β_j is the normalization factor that ensures $\sum_k u_{k,j}^{new} = 1$.

Probabilities $p(w)$ and $p(b)$ are used for joint probability $p(w, b)$. One natural choice for $p(w)$ and $p(b)$ is to use the empirical values that are estimated from the training corpus. However, one problem is that, the empirical distribution for term frequency follows a skewed distribution (Zipf's law). As a result, if the empirical $p(w)$ is used directly, the regularization term will mainly focus on the consistency checking for common words. For rare words, since its empirical $p(w)$ is very small, its impact in the regularization term is almost ignorable. To put equal emphasis on both common words and uncommon words, we decide to use a uniform distribution for both $p(w)$ and $p(b)$, which turns out to have better performance in our empirical studies.

4 Experiments.

In the following experiments, we compare the effectiveness of the proposed model to the existing translation model and a state-of-art statistical model for automatic image annotation.

4.1 Experiment Data. The same subset of Corel data used in [8] is used in this experiment. It consists of 5000 annotated images, among which 4500 of them are used for training and selection of parameters and the rest 500 images used for testing. 371 different words are used for annotating both training and testing images. Similar to the previous studies on automatic image annotation, the quality of automatic image annotation is measured by the performance of retrieving auto-annotated images regarding to single-word queries. For each single-word query, **precision** and **recall** are computed using the retrieved lists that are based on the true annotations and the auto-annotations. Let I_j be a test image, t_j be its true annotation, and g_j be its auto-annotation. For a given query word w , precision and recall are defined respectively as:

$$\text{precision}(w) = \frac{|\{I_j | w \in t_j \wedge w \in g_j\}|}{|\{I_j | w \in g_j\}|}$$

$$\text{recall}(w) = \frac{|\{I_j | w \in t_j \wedge w \in g_j\}|}{|\{I_j | w \in t_j\}|}$$

The $\text{precision}(w)$ measures the accuracy in annotating images with word w and the $\text{recall}(w)$ measures the completeness in annotating images with word w . The average precision and recall over different single-word queries are used to measure the overall quality of automatically generated annotations. The third metric, **#Ret_Query**, is the number of single-word queries for

	TM	RM	RSTM
#Ret_Query	63	76	86
Average recall	0.2106	0.2656	0.3373
Average precision	0.1836	0.2299	0.2141

Table 2: Performance comparison of different models: the Translation Model(TM), the Relevance Model(RM), and the Regularization-based Symmetric Translation Model(RSTM).

which at least one relevant image can be retrieved:

$$\#Ret_Query = |\{w | \text{precision}(w) > 0 \wedge \text{recall}(w) > 0\}|$$

This metric compensates the metrics of average precision and average recall by providing information about how wide is the range of words that contribute to the average precision and recall.

4.2 RSTM VS. Original Translation Model.

In this section, we compare the performance of the proposed translation model to the original one. First, we employ the cross-validation approach to decide the best value for λ . In particular, the training data is divided into two parts: 70% of data is used for training the model, and 30% of data is used for cross validating the value of λ . We find $\lambda = 50,000$ is a good choice for the RSTM.

The comparison result is listed in Table 2. The RSTM performs substantially better than the original translation model in all three metrics. The most noticeable difference between them is in the metric of #Ret_Query and average recall, which is 86 and 33.73% for the RSTM, and is only 63 and 21.06% for the original translation model. A few examples of the annotation words are listed in Table 3. We can see that, the RSTM is able to come up with more specific words for annotation than the original translation model. For instance, consider the first example in Table 3, the original translation model is only able to come up with a general/common term 'buildings' to describe the object in the image, while the RSTM is able to identify the building as 'lighthouse'.

4.3 Comparison to Other Annotation Models.

We also compare RSTM to the relevance language model for automatic image annotation, which has shown good performance in recent studies [10, 13]. The result is listed in Table 2. Compared to the RSTM, the relevance language model achieves slightly better performance in terms of average precision. However, its #Ret_Query and average recall are substantially worse than the RSTM, with 76 and 26.56% versus 86 and 33.73% for the RSTM.

Image	TM	RSTM	Manual
	sky water tree buildings rocks	sky water rocks booby lighthouse	water hills coast lighthouse
	sky water tree buildings snow	water tree snow forest zebra	tree snow forest coyote
	sky water tree grass rocks	sky rocks fox giraffe tusks	rocks fox kit baby
	water tree people grass buildings	tree field horses albatross foals	field horses mare foals

Table 3: Examples of annotations generated by the Translation Model (TM), the Regularization-based Symmetric Translation Model (RSTM). The manual annotations are included in the last column.

5 Conclusion.

In this paper, we propose a regularization-based symmetric translation model(RSTM) to explore the correlation between the forward and the backward translation models. In particular, it introduces a soft regularization term, based on the KL divergence, to enforce the consistency between two translation models. Empirical studies have shown that the model is able to effectively enhance the quality of automatic image annotations.

References

- [1] K. Barnard, P. Duygulu and D. Forsyth, *Clustering Art*. in Proceedings of the IEEE Computer Society Conference on Pattern Recognition. 2001.
- [2] K. Barnard, P. Duygulu, N. d. Freitas, D. Forsyth, D. Blei and M. I. Jordan, *Matching Words and Pictures*. Journal of Machine Learning Research, 2003. 3: p. 1107-1135.
- [3] D. Blei and M. Jordan, *Modeling annotated data*. in Proceedings of 26th International Conference on Research and Development in Information Retrieval (SIGIR). 2003.
- [4] P. Brown, S. D. Pietra, V. D. Pietra and R. Mercer, *The Mathematics of Statistical Machine Translation*. Computational Linguistics, 1993. 19(2): p. 263-311.
- [5] E. Chang, K. Goh, G. Sychay and G. Wu, *CBSA: content-based soft annotation for multimodal image retrieval using bayes point machines*. CirSysVideo, 2003. 13(1): p. 26-38.
- [6] C. Cusano, G. Ciocca and R. Schettini, *Image annotation using SVM*. in Proceedings of Internet imaging IV, Vol. SPIE 5304. 2004.
- [7] A. P. Dempster, N. M. Laird and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*. Journal of Royal Statistical Society, 1977. 39(1): p. 1-38.
- [8] P. Duygulu, K. Barnard, N. d. Freitas and D. A. Forsyth, *Object recognition as machine translation: learning a lexicon for a fixed image vocabulary*. in Proceedings of 7th European Conference on Computer Vision. 2002.
- [9] S. F. Chen, and J. Goodman, *An empirical study of smoothing techniques for language modeling*. in Annual Meeting of the ACL Proceedings of the 34th conference on Association for Computational Linguistics. 1996. Santa Cruz, California.
- [10] J. Jeon, V. Lavrenko and R. Manmatha, *Automatic Image Annotation and Retrieval using Cross-Media Relevance Models*. in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval. 2003.
- [11] R. Jin, C. X. Zhai and A. G. Hauptmann, *Title Language Model for Information Retrieval*. in Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2002.
- [12] V. Lavrenko and B. Croft, *Relevance-based language models*. in The 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2001.
- [13] V. Lavrenko, R. Manmatha and J. Jeon, *A Model for Learning the Semantics of Pictures*. in Proceedings of Advance in Neural Information Processing. 2003.
- [14] J. Li and J. Z. Wang, *Automatic linguistic indexing of pictures by a statistical modeling approach*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2003. 25(19): p. 1075-1088.
- [15] O. Maron, *Learning from Ambiguity*. 1998, MIT.
- [16] F. Monay and D. Gatica-Perez, *On Image Auto-Annotation with Latent Space Models*. in Proc. ACM International Conference on Multimedia. 2003.
- [17] Y. Mori, H. TAKAHASHI and R. Oka, *Image-to-Word Transformation Based on Dividing and Vector Quantizing Images With Words*. in MISRM'99 First International Workshop on Multimedia Intelligent Storage and Retrieval Management. 1999.
- [18] J. Ponte, *A Language Modeling Approach to Information Retrieval*. in Department of Computer Science. 1998, Univ. of Massachusetts at Amherst.
- [19] Zhai, C. X. and J. Lafferty. *Model-based feedback in the KL-divergence retrieval model*. in Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM). 2001.

Correcting Sampling Bias in Structural Genomics through Iterative Selection of Underrepresented Targets

Kang Peng*

Slobodan Vucetic*

Zoran Obradovic*

Abstract

In this study we proposed an iterative procedure for correcting sampling bias in labeled datasets for supervised learning applications. Given a much larger and unbiased unlabeled dataset, our approach relies on training contrast classifiers to iteratively select unlabeled examples most highly underrepresented in the labeled dataset. Once labeled, these examples could greatly reduce the sampling bias present in the labeled dataset. Unlike active learning methods, the actual labeling is not necessary in order to determine the most appropriate sampling schedule. The proposed procedure was applied on an important bioinformatics problem of prioritizing protein targets for structural genomics projects. We show that the procedure is capable of identifying protein targets that are underrepresented in current protein structure database, the Protein Data Bank (PDB). We argue that these proteins should be given higher priorities for experimental structural characterization to achieve faster sampling bias reduction in current PDB and make it more representative of the protein space.

1 Introduction.

In data mining and machine learning it is commonly assumed that the training, or labeled, dataset is unbiased, i.e., a random sample from the same underlying distribution as the data on which the learned model will be applied. In real-life applications, however, this assumption is often violated due to the *sampling bias* or *sample selection bias* problem [4], and the labeled dataset is no longer a representative of the whole population. Consequently, a predictor model learned on such data may be suboptimal and will not generalize well on out-of-sample examples. Thus, it is crucial to be able to detect and correct such bias, and bias detection and correction should be considered an integral part of the learning process.

The sampling bias can be quantified by the *sample selection probability* $p(s=1|\mathbf{x}, y)$ [11], where \mathbf{x} is feature vector, y is target variable (label) and s is a binary random variable. If $s = 1$, an example (\mathbf{x}, y) is selected

into the labeled dataset, while if $s = 0$, it is not selected. Thus, if $p(s=1|\mathbf{x}, y)$ is constant for all (\mathbf{x}, y) , the labeled dataset is unbiased; otherwise, it is biased. In numerous real-life scenarios, a reasonable and realistic assumption is that s is dependent on \mathbf{x} but conditionally independent of y given \mathbf{x} , i.e. $p(s=1|\mathbf{x}, y) = p(s=1|\mathbf{x})$.

A suitable tool for detecting and characterizing the sampling bias are contrast classifiers [9] that measure the distributional difference between the labeled dataset and an unlabeled dataset, which is a large, unbiased sample from the underlying distribution. The contrast classifier is a 2-class classifier trained to discriminate between examples of labeled and unlabeled datasets. Its output is directly related to the sample selection probability $p(s=1|\mathbf{x})$ and thus could be a useful measure of sampling bias, as illustrated on a synthetic dataset as well as in real-life bioinformatics applications [8, 9].

This study was motivated by an important bioinformatics problem that is directly related to sampling bias. It is well-known that three dimensional (3-D) structure of a protein is a crucial determinant of its biological function and biochemical properties. However, only a small fraction of known proteins has experimentally determined 3-D structure; currently, there are about 25,000 protein structures deposited in the main protein structure database, Protein Data Bank (PDB) [1], as compared to more than 1.5 million known proteins. Furthermore, current content of PDB is highly biased in the sense that it does not adequately sample the protein sequence space, due to various issues related to experimental structure determination [8].

The ongoing structural genomics projects [2] try to address this problem by experimentally determining structures of a carefully selected set of representative protein targets which, along with those already in PDB, could achieve maximal coverage of the protein sequence space. However, most of the existing target selection strategies [5] rely on sequence comparison methods and, as a result, produce long lists of representative proteins without structurally characterized homologues (i.e. proteins evolved from the same ancestor that typically have similar sequence and structure). It is evident that, using the available technology, the

*Information Science and Technology Center, Temple University, 303 Wachman Hall, 1805 N. Broad St., Philadelphia, PA 19122, USA.

progress of these projects is likely to be slow. Thus, in order to rapidly achieve a good coverage of the protein space, novel computational methods for prioritization of protein targets should be developed.

In such application, labeled examples are proteins with experimentally determined 3-D structures. A protein is also considered as labeled if at least one of its homologues has known structure. Unlabeled examples are defined as all known proteins. Thus stated, our specific goal is developing methodologies to select the most informative unlabeled proteins for labeling, i.e. experimental structure characterization, as to rapidly reduce the sampling bias existing in the labeled proteins in PDB.

In this study we propose an iterative procedure for the prioritization based on the contrast classifier framework. Starting from a biased labeled dataset and an unbiased unlabeled dataset, the procedure iteratively builds contrast classifiers to (a) determine if sampling bias in current labeled dataset is significant, and (b) select a certain number of underrepresented unlabeled examples based on the contrast classifier output. The selected examples are then assumed as labeled and added into the current labeled data, thus reducing the sampling bias.

The proposed procedure was applied on a complete genome whose protein sequences are currently used as structural genomics targets. We argue that the proposed approach is highly suitable for prioritizing structural genomics protein targets since it emphasizes importance of the most underrepresented protein sequences. Revealing structural properties of such proteins is likely to produce highly significant biological results.

2 Methodology.

2.1 Problem formulation. Given two datasets sampled from the same unknown distribution, where D_L is *labeled* and *biased*, while D_U is *unlabeled* and *unbiased*, our objective is to identify a set of G most informative examples from D_U which, once labeled, would maximally reduce the labeled data bias. Here G is a user-specified parameter that depends on total labeling cost. We assume that labeling cost is uniform for all examples.

2.2 Contrast classifier for detecting sampling bias. As shown in [9] the contrast classifier is a 2-class classifier trained to learn the distributional difference between labeled and unlabeled datasets. When using classification algorithms that estimate posterior conditional class probability and balanced training data, the contrast classifier output $cc(\mathbf{x})$ approxi-

mates $u(\mathbf{x})/(u(\mathbf{x})+l(\mathbf{x}))$, or, equivalently, $l(\mathbf{x})/u(\mathbf{x}) = (1-cc(\mathbf{x}))/cc(\mathbf{x})$, where $l(\mathbf{x})$ and $u(\mathbf{x})$ are probability density functions (pdfs) for D_L and D_U respectively.

It is straightforward to show the connection between $cc(\mathbf{x})$ and the sample selection probability $p(s=1|\mathbf{x})$ [11], where s is a binary random variable indicating whether \mathbf{x} is sampled into the labeled dataset D_L . If $p(s=1|\mathbf{x})$ is constant for all \mathbf{x} , there is no sampling bias; otherwise, the labeled dataset is biased. Following Bayes theorem, $p(s=1|\mathbf{x}) = p(s=1)p(\mathbf{x}|s=1)/p(\mathbf{x})$, where $p(\mathbf{x}|s=1)$ and $p(\mathbf{x})$ can be approximated by $l(\mathbf{x})$ and $u(\mathbf{x})$, respectively. Thus, $p(s=1|\mathbf{x}) = p(s=1)l(\mathbf{x})/u(\mathbf{x}) = p(s=1)(1-cc(\mathbf{x}))/cc(\mathbf{x})$.

The contrast classifier output could therefore be a useful measure of sampling bias. If $cc(\mathbf{x}) < 0.5$, then $l(\mathbf{x}) > u(\mathbf{x})$ and \mathbf{x} is *overrepresented* in D_L . If $cc(\mathbf{x}) > 0.5$, then $l(\mathbf{x}) < u(\mathbf{x})$ and \mathbf{x} is *underrepresented* in D_L . Otherwise, $l(\mathbf{x}) = u(\mathbf{x})$ and \mathbf{x} is equally represented in D_L and D_U . Thus, the $cc(\mathbf{x})$ distribution for D_U could reveal the overall level of sampling bias: the bias is negligible if it is concentrated around 0.5, otherwise the bias is significant. Alternatively, the difference between the two $cc(\mathbf{x})$ distributions for D_L and D_U could also be used to measure the overall bias.

2.3 An iterative procedure for correcting sampling bias. Based on the discussion above, we propose to use contrast classifier output as criterion to select underrepresented examples for labeling to correct sampling bias. A one-step approach for this purpose would be building a single contrast classifier from the initial D_L and D_U and selecting G underrepresented examples from D_U . An open question is what G underrepresented examples would be the most suitable for selection. A one-step approach for selection may be too aggressive and fail to properly correct the bias. Therefore, a challenging problem is how to determine an appropriate selection schedule that would minimize bias of the resulting labeled dataset.

We propose a procedure (Figure 1) that iteratively builds contrast classifiers and incrementally selects underrepresented examples. At each iteration, a contrast classifier is built from current D_L and D_U and then applied to D_U . If the $cc(\mathbf{x})$ distribution for D_U indicates the overall bias is significant, a set of B underrepresented examples will be selected and added into D_L for building the contrast classifier at the next iteration. Details about how to select the B examples will be discussed in the next section. The whole procedure iterates until the required G examples have been selected or the sampling bias becomes negligible.

It should be noted that the actual labeling is not really necessary during the procedure since the label

Inputs:	a) Two datasets sampled from the same distribution, D_L is biased while D_U is unbiased, and $ D_L \ll D_U $ b) p – a small constant within (0, 1) c) G – total number of examples to select d) B – number of examples to select at each iteration
Outputs:	D_S – selected examples
	$D_S = \emptyset$;
	for $i = 1$ to G/B
	Train a contrast classifier from D_L and D_U
	Apply the contrast classifier on D_U to calculate the Δ measure
	Stop the procedure if the bias is negligible
	Select B underrepresented examples from D_U
	Add selected examples to D_L and D_S
	end

Figure 1: The proposed iterative procedure.

information is not used. It can be done after the procedure stops for all selected examples (D_S), which is the differential set between the final and initial D_L . This is one of the major differences between the proposed procedure and active learning methods [3].

2.4 Selection of underrepresented examples.

After a contrast classifier is built and significant bias is detected, potentially underrepresented examples can be selected from D_U . A straightforward method (named *Top-B*) is to select B examples with the highest $cc(\mathbf{x})$ from D_U . However, this approach might not be efficient in reducing sampling bias, since these examples may be redundant. They may come from a same underrepresented region that produces similarly high $cc(\mathbf{x})$ values.

An alternative method (named *Random-B*) first determines a threshold θ^p such that only 100p% of labeled examples have $cc(\mathbf{x})$ values higher than it, where p is a small constant in (0, 1). Then, all unlabeled examples that satisfy $cc(\mathbf{x}) > \theta^p$ are considered as underrepresented, i.e. $U^p = \{\mathbf{x} \mid cc(\mathbf{x}) > \theta^p \wedge \mathbf{x} \in D_U\}$. Finally, B examples are randomly drawn from U^p according to the uniform distribution. In this way the selected examples could cover the underrepresented regions more evenly.

2.5 Quantitative measure of overall bias. As discussed in § 2.2, the sampling bias can be assessed qualitatively by visual inspection of the $cc(\mathbf{x})$ distributions for D_L and D_U , i.e. whether the distribution for D_U is concentrated around 0.5, or whether the two distributions are largely overlapped. We also defined a quantitative measure of overall bias $\Delta = \sqrt{\sum_{i=1}^{|D_U|} (cc(\mathbf{x}_i) - 0.5)^2 / |D_U|}$, where $cc(\mathbf{x}_i)$ is contrast classifier output for the i -th example in D_U . The value of Δ will approach 0 when bias is negligible since all $cc(\mathbf{x}_i)$ should be close to 0.5. It will be large if the bias is significant

and many $cc(\mathbf{x}_i)$ are far away from 0.5. In the extreme case when all $cc(\mathbf{x}) = 1$, $\Delta = 0.5$.

3 Bioinformatics Application in Structural Genomics.

In this section we applied the proposed procedure to prioritizing structural genomics targets from a model organism. We show that the proposed iterative procedure is more effective than simple random sampling and a one-step approach discussed in § 2.3.

3.1 Datasets. We limited our study to the 28,334 protein sequences (40 amino acids or longer) from a model organism called *Arabidopsis thaliana* extensively studied in plant biology. It is also a major source of structural genomics targets for the Center for Eukaryotic Structural Genomics (CESG, <http://www.uwstructuralgenomics.org/>). By now CESG has selected about 4,000 target proteins from this genome and finished structure determination for 19 of them.

The amino acid sequences were obtained from website of the Arabidopsis Information Resource (TAIR, <http://www.arabidopsis.org/>). As in a previous study [8], a non-redundant representative subset of 14,988 sequences was selected, with no two sequences having pairwise identity higher than 40%. Out of these sequences 838 were identified to have known structures and thus formed D_L , while all of the 14,988 proteins were assigned to D_U . We assume that at most 500 proteins, i.e. $G = 500$, can be selected from D_U for labeling due to available resources for experimental structure determination.

3.2 Knowledge representation and contrast classifier training. A similar knowledge representation was adopted as in the previous study [8], i.e. constructing one example per sequence position instead of one example per sequence. Each example consisted of 30 attributes and a class label of 0 or 1 indicating if it was from D_L or D_U . In addition to the 25 attributes used in the previous study [8], 3 transmembrane helix predictions by PHDHtm predictor [10], 1 disorder prediction by VL3 predictor [7] and 1 coiled-coils prediction by COILS predictor [6] were also included.

The contrast classifiers were built as an ensemble of 20 neural networks each having 10 hidden neurons and 1 output neuron with sigmoid activation function. A two-stage sampling procedure [8] was employed to construct balanced training sets (12,000 examples) for training component networks. The contrast classifiers were applied to each sequence \mathbf{s} in D_U to obtain $cc(\mathbf{x})$ for each sequence position. These *per-position*

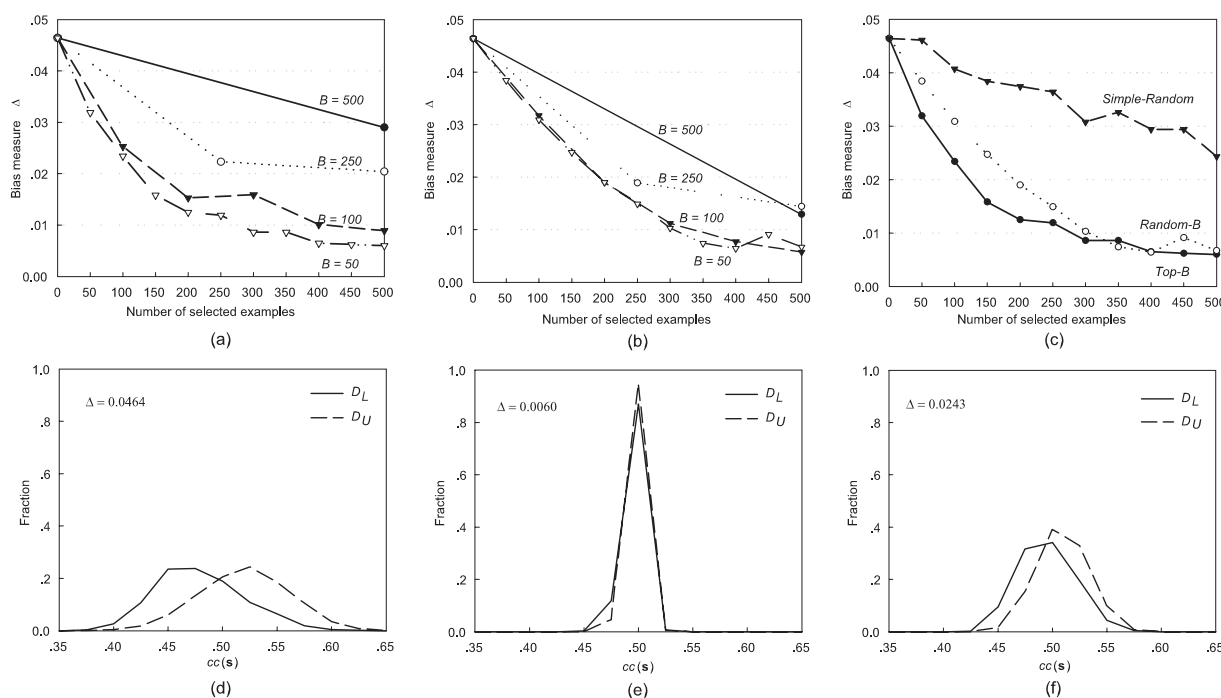


Figure 2: Application of the proposed procedure on structural genomics data. Upper row - plots of Δ vs. *Number of examples selected* for (a) *Top-B*, (b) *Random-B* and (c) Comparison to *Simple-Random*. Lower row - $cc(s)$ distributions for D_L and D_U (d) at the beginning, after 500 proteins selected using (e) *Top-B* ($B = 50$), and (f) *Simple-Random*. The distributions for *Random-B* are similar to those of *Top-B* and thus not shown.

predictions were then averaged over the sequence as the *per-sequence* prediction $cc(s)$, which was used to select underrepresented sequences.

3.3 Application of the proposed procedure. We examined different values of $B = \{50, 100, 250, 500\}$ for the proposed procedure. Note that $B = 500$ corresponds to the one-step approach of selecting all 500 proteins at one time (§ 2.3). The two methods *Top-B* and *Random-B* ($p = 0.05$) for selecting underrepresented examples were compared. The performance was assessed using the Δ measure of overall bias (§ 2.5) calculated in each iteration. A successful bias correction procedure should result in a rapid decrease in Δ measure as more proteins are selected. In Figure 2a and 2b we show the plots of Δ measure vs. number of selected proteins for the two methods.

As evident from the plots, the proposed iterative procedure ($B = 50$) performed better than the one-step approach ($B = 500$) in the sense that it achieved much lower Δ value, or lower level of overall bias, after selected 500 proteins. However, the difference is smaller

for *Random-B* than for *Top-B*. It is worth noting that smaller B typically leads to better bias reduction but with larger computational efforts. However, the small difference between the plots for $B = 50$ and $B = 100$ indicates that the gain of using even smaller B may be marginal.

In addition to the proposed procedure, we also examined the simple approach (*Simple-Random*) of randomly selecting M unlabeled proteins in a single step. For each $M = \{50, 100, 150, \dots, 500\}$, a contrast classifier was built to calculate the Δ measure of overall level of bias after the M proteins were added into D_L . The plot of Δ measure vs. number (M) of selected proteins is shown in Figure 2c, along with those for the proposed procedure ($B = 50$). As expected, the *Simple-Random* method was less effective in bias reduction, with the final $\Delta = 0.0243$, as compared to 0.0060 for *Top-B* and 0.0067 for *Random-B*. This is further confirmed by the $cc(s)$ distributions in Figure 2. The two resulting distributions are almost completely overlapped for *Top-B* with $B = 50$ (Figure 2e), but still clearly separated for *Simple-Random* (Figure 2f), after

500 proteins were selected. The distributions for initial D_L and D_U are shown in Figure 2d.

Out of the 500 proteins selected using *Top-B* (*Random-B*) method, only 68 (79) are currently selected as structural genomics targets by the Center for Eukaryotic Structural Genomics and none of them have been solved. We argue that the remaining 432 (421) proteins should also be selected as structural genomics targets and should be given higher priorities. Along with proteins with known structures, these proteins should be very helpful in achieving maximal coverage of the protein space for *Arabidopsis thaliana* genome.

4 Conclusions.

In this study we proposed an iterative procedure for correcting sampling bias in labeled data. This approach is applicable if an unbiased unlabeled dataset is available. It iteratively builds contrast classifiers to detect sampling bias and selects underrepresented examples which, once labeled, can be very effective in correcting the sampling bias in labeled data. As illustrated on an important bioinformatics problem of prioritizing protein targets for structural genomics projects, the proposed procedure is more effective than randomly choosing unlabeled proteins for labeling in reducing sampling bias, and can rapidly achieve a good coverage of the protein space.

The ultimate solution for correcting sampling bias is to add new labeled examples. A simple approach is to randomly select examples for labeling. As more examples are labeled, the bias in the resulting labeled dataset would be gradually reduced. This is especially good if the total number of new labeled examples could be relatively large. However, in real-life problems like protein structure determination it is often the case that the costs for labeling even a single example could be very high. Consequently, only a small number of examples can be allowed and thus the selected ones should be the most informative. In such scenarios, our procedure could be very appropriate since it emphasizes the underrepresented examples based on contrast classifier output directly related to the sample selection probability $p(s=1|\mathbf{x})$.

More work is needed to fully characterize the proposed procedure. An improved performance measure is needed since the Δ measure may depend on the learning algorithms used in learning contrast classifiers. As shown in § 3.3, smaller B might be desirable for better bias reduction but would require more computational efforts. Thus, additional analytical and experimental work needs to be done to determine the optimal trade-off between the computational effort and the level of bias reduction, given a fixed total number G of allowed

examples for labeling. Finally, if G is very small, it is likely that more elaborate procedure would be needed than the proposed *Top-B* and *Random-B* procedures. These issues are the subjects of our ongoing research.

Acknowledgements. We thank A. K. Dunker, director of Center for Computational Biology and Bioinformatics at Indiana University School of Medicine, for pointing us to structural targets prioritization in *Arabidopsis thaliana* genome as a potentially high impact application domain for our sampling bias correction method. We are also grateful to J. L. Markley, director of the Center for Eukaryotic Structural Genomics at University of Wisconsin-Madison, for providing us access to CESG structural genomics target database for a more detailed biological relevance validation of our method (results not included in this report).

References

- [1] H. M. Berman, T. N. Bhat, P. Bourne, Z. Feng, G. Gilliland and H. Weissig, *The Protein Data Bank and the challenge of structural genomics*, Nat. Struct. Biol., 7 (2000), pp. 957–959.
- [2] S. E. Brenner, *A tour of structural genomics*, Nat. Rev. Genet., 2 (2001), pp. 801–809.
- [3] D. Cohn, L. Atlas and R. Ladner, *Improved generalization with active learning*, Mach. Learning, 15 (1994), pp. 201–221.
- [4] J. Heckman, *Sample selection bias as a specification error*, Econometrica, 47 (1979), pp. 153–161.
- [5] M. Linial and G. Yona, *Methodologies for target selection in structural genomics*, Prog. Biophys. Mol. Biol., 73 (2000), pp. 297–320.
- [6] A. Lupas, M. Van Dyke and J. Stock, *Predicting coiled coils from protein sequences*, Science, 252 (1991), pp. 1162–1164.
- [7] Z. Obradovic, K. Peng, S. Vucetic, P. Radivojac, C. J. Brown and A. K. Dunker, *Predicting intrinsic disorder from amino acid sequence*, Proteins, 53 Suppl 6 (2003), pp. 566–572.
- [8] K. Peng, Z. Obradovic and S. Vucetic, *Exploring bias in the Protein Data Bank using contrast classifiers*, Pacific Symposium on Biocomputing, Hawaii, (2004), pp. 435–446.
- [9] K. Peng, S. Vucetic, B. Han, H. Xie and Z. Obradovic, *Exploiting unlabeled data for improving accuracy of predictive data mining*, 3rd IEEE Int’l Conf. on Data Mining, Melbourne, FL, (2003), pp. 267–274.
- [10] B. Rost, R. Casadio, P. Fariselli and C. Sander, *Prediction of helical transmembrane segments at 95% accuracy*, Protein Sci., 4 (1995), pp. 521–533.
- [11] B. Zadrozny, *Learning and evaluating classifiers under sample selection bias*, in C. E. Brodley, ed., 21st Int’l Conf. Machine Learning, ACM Press, Banff, Alberta, Canada, (2004).

Statistical Models for Unequally Spaced Time Series

Emre Erdogan ^{*} Sheng Ma [†] Alina Beygelzimer [†] Irina Rish [†]

January 16, 2005

Abstract

Irregularly observed time series and their analysis are fundamental for any application in which data are collected in a distributed or asynchronous manner. We propose a theoretical framework for analyzing both stationary and non-stationary irregularly spaced time series. Our models can be viewed as extensions of the well known autoregression (AR) model. We provide experiments suggesting that, in practice, the proposed approach performs well in computing the basic statistics and doing prediction. We also develop a resampling strategy that uses the proposed models to reduce irregular time series to regular time series. This enables us to take advantage of the vast number of approaches developed for analyzing regular time series.

1 Introduction

Unevenly sampled time series are common in many real-life applications when measurements are constrained by practical conditions. The irregularity of observations can have several fundamental reasons. First, any event-driven collection process (in which observations are recorded only when some event occurs) is inherently irregular. Second, in such applications as sensor networks, or any distributed monitoring infrastructure, data collection is distributed, and collection agents cannot easily synchronize with one another. In addition, their sampling intervals and policies may be different. Finally, in many applications, measurements cannot be made regularly or have to be interrupted due to some events (either foreseen or not).

Time series analysis has a long history. The vast majority of methods, however, can only handle regular time series and do not easily extend to unevenly sampled data. Continuous time series models can be directly applied for the problem (e.g., [5]), but they tend to be complicated (mostly due to the difficulty of estimating and evaluating them

from discretely sampled data) and do not provide a satisfying solution in practice.

In data analysis practice, irregularity is a recognized data characteristic, and practitioners deal with it heuristically. It is a common practice to ignore the times and treat the data as if it were regular. This can clearly introduce a significant bias leading to incorrect predictions. Consider, for example, the return of a slowly, but consistently changing stock, recorded first very frequently and then significantly less frequently. If we ignore the times, it would appear as if the stock became more rapidly changing, thus riskier, while in fact the statistical properties of the stock did not change.

Many basic questions that are well understood for regular time series, are not dealt with for unequally spaced time series. The goal of this paper is to provide such a theoretical foundation. At the very least, we would like to be able to compute the basic statistics of a given time series (e.g., its mean, variance, autocorrelation), and predict its future values.

Our contributions can be summarized as follows:

- We propose two statistical models for handling irregularly sampled time series. The first model assumes stationarity, and can be viewed as a natural extension of the classical AR(1) model for regular time series. The second model relaxes the stationarity assumption by allowing a more general dependence on the current time, time difference, and the state of the process at a given time.
- We show how to efficiently estimate the parameters of both models using the maximum likelihood method.
- We propose and give solutions for two strategies based on the proposed models. The first strategy is to compute the basic statistics (e.g., autocorrelation function) and do prediction directly from the models. This approach does not easily extend to non-linear time series and multiple

^{*}Columbia University, New York, NY, ee168@columbia.edu

[†]IBM T. J. Watson Research Center, Hawthorne, NY, {shengma,beygel,rish}@us.ibm.com

irregular time series. The second strategy avoids these problems by using the model to convert irregular time series to regular time series by resampling. The reduction reduces the problem to a problem that has already been thoroughly analyzed and for which many approaches are available. The resampling approach can be found in the full version of the paper [4].

Related work A vast amount of techniques were developed for analyzing regularly sampled time series. Unfortunately, most of these techniques do not take into account sampling times, and cannot be easily generalized to irregularly sampled time series.

As a simple heuristic, we can ignore the times and treat the values as regularly sampled. Obviously, if there is enough structure and irregularity in sampling times, we lose a lot of information about the dynamics of the system.

Many techniques have been proposed to handle time series with missing data, which in the limit can be viewed as irregularly sampled [8]. One approach is to interpolate the data to equally spaced sampling times. A survey of such interpolation techniques can be found in [1]. While this is a reasonable heuristic for dealing with missing values, the interpolation process typically results in a significant bias (e.g., smooths the data) changing the dynamics of the process, thus these models can not be applied if the data is truly unequally spaced. Another problem is that there is little understanding of which interpolation method does best on a given dataset.

A number of authors suggested to use continuous time diffusion processes for the problem. Jones [6] proposed a state-space representation, and showed that for Gaussian inputs and errors, the likelihood of data can be calculated exactly using Kalman filters. A nonlinear (non-convex) optimization can then be used to obtain maximum likelihood estimates of the parameters. Brockwell [2] improved on this model and suggested a continuous time ARMA process driven by the Lévy process. His models, however, assume stationarity, and parameter estimation is done via non-convex optimization using Kalman filtering limiting the practical use of these models.

2 Background: Basic Definitions

A *time series* $X(t)$ is an ordered sequence of observations of a variable X sampled at different points t over time. Let the sampling times be t_0, t_1, \dots, t_n satisfying $0 \leq t_0 < t_1 < \dots < t_n$. If the time points are equally spaced (i.e., $t_{i+1} - t_i = \Delta$ for all $i = 0, \dots, n-1$, where $\Delta > 0$ is some constant), we call the time series *regularly sampled*. Otherwise, the

sequence of pairs $\{X(t_i), t_i\}$ is called an *irregularly sampled time series*.

Definition 1. (AUTOCOVARANCE [3]) For a process $\{X(t), t \in T\}$ with $\text{var}(X(t)) < \infty$ for each $t \in T$, the auto-covariance function $\text{cov}_X(X(t), X(s))$, for $t, s \in T$, is given by

$$\mathbf{E}[(X(t) - \mathbf{E}[X(t)])(X(s) - \mathbf{E}[X(s)])].$$

Definition 2. (STATIONARITY [3]) A time series $\{X(t), t \in T\}$ is said to be stationary if

- $\mathbf{E}[|X(t)|^2] < \infty$, $\mathbf{E}[X(t)] = c$, for all $t \in T$,
- $\text{cov}_X(X(t), X(s)) = \text{cov}_X(X(t+h), X(s+h))$ for all $t, s, h \in T$.

In other words, a stationary process is a process whose statistical properties do not vary with time.

Definition 3. (AR(1) PROCESS [3]) A regularly sampled process $\{X(t), t = 0, 1, 2, \dots\}$ is said to be an AR(1) process if $\{X(t)\}$ is stationary and if for every t ,

$$X(t) = \theta X(t-1) + \sigma \epsilon_t,$$

where $\{\epsilon_t\}$ is a series of random variables with $\mathbf{E}(\epsilon_t) = 0$, $\text{var}(\epsilon_t) = 1$, and $\text{cov}(\epsilon_t, \epsilon_s) = 0$ for every $t \neq s$. Notice that by recursive substitution, we can write $X(t+h)$ for any positive integer h in terms of $X(t)$ as

$$X(t+h) = \theta^h X(t) + \sigma \sum_{j=0}^{h-1} \theta^j \epsilon_{t+1+j}.$$

The process $\{\epsilon_t\}$ is also called “white noise”. We will assume that $\epsilon_t \sim N(0, 1)$ for all t .

3 Overview of our Approach

Suppose that our irregularly sampled time series $Y(t)$ can be decomposed as

$$(3.1) \quad Y(t) = a(t) + X(t),$$

where $a(t)$ is a slowly changing deterministic function called the “trend component” and $X(t)$ is the “random noise component”.

In general, one can observe only the values $Y(t)$. Therefore, our first goal is to estimate the deterministic part $a(t)$, and extract the random noise component $X(t) = Y(t) - a(t)$. Our second goal is to find a satisfactory probabilistic model for the process $X(t)$, analyze its properties, and use it together with $a(t)$ to predict $Y(t)$.

Let $\{y(t_i), t_i\}$, $i = 0, 1, \dots, n$ be a sample of $Y(t)$. We assume that $a(t)$ is a polynomial of degree p in t ,

$$(3.2) \quad a_p(t) = \rho_0 + \rho_1 t + \rho_2 t^2 + \dots + \rho_p t^p,$$

where p is a nonnegative integer and $\boldsymbol{\rho} = [\rho_0; \rho_1; \dots; \rho_p]$ is a vector of coefficients. A more general structure can also be used. The vector $\boldsymbol{\rho}$ can be estimated using the least squares method, by choosing the vector minimizing $\sum_{i=0}^n (y(t_i) - a(t_i))^2$. This is straightforward, and we will turn to developing a statistical model for $X(t)$.

We propose two parametric statistical models for analyzing $X(t)$. The first model, described in the next section, is a direct extension of the classical AR(1) model given in Definition 3 and assumes that $X(t)$ is stationary. The second model, presented in Section 5, relaxes the stationarity assumption and allows a more general dependence of $X(t + \Delta)$ on t , Δ , and $X(t)$.

4 A Statistical Model for Stationary $X(t)$

Suppose that $X(t)$ obtained from $Y(t)$ after removing the trend component $a_p(t)$ is a stationary process. We define an irregularly sampled stationary AR(1) process as follows.

Definition 4. (IS-AR(1) PROCESS) *A time series $X(t)$ is an irregularly sampled stationary AR(1) process if it is stationary and if for every t and $\Delta \geq 0$,*

$$(4.3) \quad X(t + \Delta) = \theta^\Delta X(t) + \sigma_\Delta \epsilon_{t+\Delta},$$

where $\epsilon_t \sim N(0, 1)$ and $\mathbf{cov}(\epsilon_t, \epsilon_s) = 0$ for every $t \neq s$ and $\sigma_\Delta^2 = \sigma^2 \left(\frac{1 - \theta^{2\Delta}}{1 - \theta^2} \right)$ for some $\sigma > 0$.

If the times are regularly spaced, IS-AR(1) can be reduced to the original AR(1) process by observing that $\sigma \sum_{j=0}^{h-1} \theta^j \epsilon_{t+1+j} \sim N(0, \sigma^2 \left(\frac{1 - \theta^{2h}}{1 - \theta^2} \right))$ and comparing with Equation 4.3.

4.1 Parameter estimation In this section, we show how to estimate parameters θ and σ , given a set of observations $\{x(t_0), x(t_1), \dots, x(t_n)\}$ of $X(t)$.

Define $\Delta_i = t_{i+1} - t_i$ for all $i = 0, \dots, n-1$. We can assume that all $\Delta_i \geq 1$; otherwise, we can rescale each Δ_i by $\min_i \{\Delta_i\}$.

Since $\mathbf{E}[\epsilon_t] = 0$ and $\mathbf{cov}(\epsilon_t, \epsilon_s) = 0$ for all $t \neq s$, we can estimate θ by the least squares method. We need to find $\hat{\theta} \in (-1, 1)$ minimizing $\sum_{i=0}^{n-1} (X(t_{i+1}) - \theta^{\Delta_i} X(t_i))^2$. Since $\Delta_i \geq 1$ for all i , this sum is a convex function of θ that can be efficiently minimized using convex optimization.

To estimate σ , we set $z_i = x(t_{i+1}) - (\hat{\theta})^{\Delta_i} x(t_i)$. By Definition 4, we have $z_i \sim N(0, \sigma_{\Delta_i}^2)$ where $\sigma_{\Delta_i}^2 = \sigma^2 \left(\frac{1 - (\hat{\theta})^{2\Delta_i}}{1 - (\hat{\theta})^2} \right)$. We can thus estimate σ by maximizing the Gaussian likelihood of the residuals $z(t_0), \dots, z(t_{n-1})$ at times t_0, t_1, \dots, t_{n-1} . The maximum

likelihood estimator of σ is given by

$$(4.4) \quad \hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} \frac{(x(t_{i+1}) - (\hat{\theta})^{\Delta_i} x(t_i))^2}{\rho_i}},$$

where $\rho_i = \left(\frac{1 - \hat{\theta}^{2\Delta_i}}{1 - \hat{\theta}^2} \right)$ for all i . The derivation is omitted due to page limit (see [4]).

4.2 Prediction using the IS-AR(1) model We first establish conditions for $X(t_0)$ under which $X(t)$ is a stationary process. Then assuming the stationarity of $X(t)$, we derive equations for one-step prediction and the auto-covariance function. We can assume without loss of generality that $t_0 = 0$.

Using Equation 4.3, independence of ϵ_t and $X(0)$, and the fact that $\mathbf{E}[\epsilon_t] = 0$ for all t , we can express $\mathbf{E}[X(t)]$, $\mathbf{var}[X(t)]$, and $\mathbf{cov}[X(t), X(t + \Delta)]$ in terms of $X(0)$ as follows.

$$(4.5) \quad \mathbf{E}[X(t)] = \theta^t \mathbf{E}[X(0)]$$

$$(4.6) \quad \mathbf{var}[X(t)] = \theta^{2t} \mathbf{var}[X(0)] + \sigma^2 \frac{1 - \theta^{2t}}{1 - \theta^2}$$

$$(4.7) \quad \mathbf{cov}[X(t), X(t + \Delta)] = \theta^\Delta \mathbf{var}[X(0)]$$

PROPOSITION 4.1. *Assume that $\mathbf{E}[X(t)^2] < \infty$ and $\mathbf{E}[X(0)] = 0$. Then $X(t)$ in Definition 4 is a stationary process if $\mathbf{var}[X(0)] = \frac{\sigma^2}{1 - \theta^2}$ and $\mathbf{cov}_X(\Delta) = \mathbf{cov}[X(t), X(t + \Delta)] = \theta^\Delta \frac{\sigma^2}{1 - \theta^2}$.*

Proof. For $\mathbf{var}[X(0)] = \frac{\sigma^2}{1 - \theta^2}$, Equation 4.6 gives

$$\mathbf{var}[X(t)] = \theta^{2t} \frac{\sigma^2}{1 - \theta^2} + \sigma^2 \frac{1 - \theta^{2t}}{1 - \theta^2} = \frac{\sigma^2}{1 - \theta^2},$$

yielding

$$\mathbf{cov}[X(t), X(t + \Delta)] = \theta^\Delta \mathbf{var}[X(t)] = \theta^\Delta \frac{\sigma^2}{1 - \theta^2}.$$

Since $\mathbf{cov}[X(t), X(t + \Delta)]$ does not depend on t , $X(t)$ is stationary. ■

A one-step predictor of $X(t + \Delta)$ given $X(t)$ for any $\Delta > 0$ is given by the conditional expectation of $X(t + \Delta)$ (using Equation 4.5):

$$\hat{X}(t + \Delta) = \mathbf{E}[X(t + \Delta) | X(t)] = \theta^\Delta X(t).$$

4.3 Analyzing $Y(t)$ with a Stationary Component $X(t)$: The following procedure can be used for estimating the auto-covariance function $\mathbf{cov}_Y(\Delta)$ of irregularly sampled time series $Y(t)$ and for predicting $Y(t + \Delta)$ given $Y(t)$. First, we fit a polynomial $a_p(t)$ to $Y(t)$ as described before, and set $X(t) = Y(t) - a_p(t)$. We then estimate θ as $\hat{\theta} =$

$\arg \min_{\theta} \sum_{i=0}^{n-1} (X(t_{i+1}) - \theta^{\Delta_i} X(t_i))^2$, and σ using $\hat{\sigma}$ in Equation (4.4). Since $a_p(t)$ is deterministic, set $\mathbf{cov}_Y(\Delta) = \mathbf{cov}_X(\Delta) = \theta^{\Delta} \frac{\sigma^2}{1 - \theta^{\Delta}}$. Finally, prediction is given by

$$\begin{aligned}\hat{Y}(t + \Delta) &= a_p(t + \Delta) + \hat{X}(t + \Delta) \\ &= a_p(t + \Delta) + \mathbf{E}[X(t + \Delta)|X(t)] \\ &= a_p(t + \Delta) + \theta^{\Delta} X(t).\end{aligned}$$

5 A model for non-stationary $X(t)$

The model introduced in the previous section assumes that $X(t)$ is stationary. Mean and variance of a stationary process are time independent, and the covariance between any two observations depends only on their time difference. This allowed us to derive a simple expression for the auto-covariance function. In practice, however, one may not have stationarity – statistical properties of $X(t)$ may vary with time. Thus it is natural to focus on estimating these properties in the near future instead of trying to obtain some global, time-independent values. To achieve this goal, we model $X(t + \Delta)$ as a function of t , Δ , and $X(t)$, plus a random noise whose variance also depends on t , Δ , and $X(t)$. As before, after fitting a polynomial of degree p to $Y(t)$ we get $X(t) = Y(t) - a_p(t)$.

5.1 General IN-AR(1) Process Let $\theta \in \mathbf{R}^m$ be an m -dimensional drift parameter vector and $\sigma \in \mathbf{R}$ be a scalar variance parameter. Let $\alpha(\Delta, t, X(t)) : [0, \infty) \times [0, \infty) \times \mathbf{R} \rightarrow \mathbf{R}^m$ and $\beta(\Delta, t, X(t)) : [0, \infty) \times [0, \infty) \times \mathbf{R} \rightarrow \mathbf{R}$ be functions of Δ , t , and $X(t)$.

Definition 5. (IN-AR(1) PROCESS) *An irregularly sampled non-stationary IN-AR(1) process is defined as*

$$\begin{aligned}X(t + \Delta) &= X(t) + \theta^T \alpha(\Delta, t, X(t)) \\ &\quad + \sigma \beta(\Delta, t, X(t)) \epsilon_{t+\Delta},\end{aligned}$$

where $\epsilon_{t+\Delta} \sim N(0, 1)$, $\mathbf{cov}(\epsilon_t, \epsilon_s) = 0$ for all $t \neq s$, θ is the vector of drift parameters, $\alpha(\Delta, t, X(t))$ is the drift function, σ is the variance parameter, and $\beta(\Delta, t, X(t))$ is the variance function. In addition, if $\Delta = 0$ then the functions α and β satisfy

$$\alpha(0, t, X(t)) = \mathbf{0} \text{ and } \beta(0, t, X(t)) = 0.$$

Since the above condition is the only assumption on the structure of α and β , the model covers a wide range of irregularly sampled time series.

5.2 Parameter Estimation Since ϵ_t and ϵ_s are independent for all $t \neq s$, the distribution of $X(t_{i+1})$ given $X(t_i)$ is normal with mean $X(t_i) +$

$\theta^T \alpha(\Delta_i, t_i, X(t_i))$ and variance $\beta^2(\Delta_i, t_i, X(t_i)) \sigma^2$. Therefore, θ and σ can be estimated by maximizing the Gaussian likelihood of observations $x(t_1), \dots, x(t_n)$ at times t_1, \dots, t_n .

PROPOSITION 5.1. *The maximum likelihood estimators of θ and σ are given by*

$$\begin{aligned}\hat{\theta} &= \left(\sum_{i=1}^n \frac{\alpha(\Delta_i, t_i, x(t_i)) \alpha^T(\Delta_i, t_i, x(t_i))}{\beta^2(\Delta_i, t_i, x(t_i))} \right)^{-1} \\ &\quad \cdot \sum_{i=1}^n \frac{(x(t_{i+1}) - x(t_i)) \alpha(\Delta_i, t_i, x(t_i))}{\beta^2(\Delta_i, t_i, x(t_i))}, \\ \hat{\sigma} &= \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{(x(t_{i+1}) - x(t_i) - \hat{\theta}^T \alpha(\Delta_i, t_i, x(t_i)))^2}{\beta^2(\Delta_i, t_i, x(t_i))}}.\end{aligned}$$

Proof. See the full version of the paper [4].

The functions $\alpha(\Delta, t, X(t))$ and $\beta(\Delta, t, X(t))$ can be chosen by first selecting a set of candidate functions (which can be based on data generation process, some preliminary analysis, or interaction with domain experts). Proposition 5.1 can then be used to estimate the parameters for each pair of candidate functions, choosing the pair that gives the best fit to the data. One can also use various greedy search-based methods in more general families of candidate functions.

5.3 Prediction using the general IN-AR(1) model Since $a_p(t)$ is deterministic, we only need to predict $X(t + \Delta)$. We have

$$\begin{aligned}\hat{Y}(t + \Delta) &= \mathbf{E}[Y(t + \Delta)|Y(t)] \\ &= a_p(t + \Delta) + \mathbf{E}[X(t + \Delta)|X(t)], \\ \mathbf{var}[Y(t + \Delta)|Y(t)] &= \mathbf{var}[X(t + \Delta)|X(t)], \\ \mathbf{cov}[Y(t + \Delta_1 + \Delta_2), Y(t + \Delta_1)|Y(t)] \\ &= \mathbf{cov}[X(t + \Delta_1 + \Delta_2), X(t + \Delta_1)|X(t)].\end{aligned}$$

Since we did not assume that $X(t)$ is stationary, we might not have a time independent expression for the mean, variance, and the auto-covariance function.

Using Definition 5, the independence of $\epsilon_{t+\Delta}$ and $X(t)$, and the assumption that $\mathbf{E}[\epsilon_t] = 0$ for all t , we can write the conditional expectation of $X(t + \Delta)$ given $X(t)$ and conditional variance as

$$\begin{aligned}\mathbf{E}[X(t + \Delta)|X(t)] &= X(t) + \theta^T \alpha(\Delta, t, X(t)) + \\ &\quad \sigma \beta(\Delta, t, X(t)) \mathbf{E}[\epsilon_{t+\Delta}] = X(t) + \hat{\theta}^T \alpha(\Delta, t, X(t)) \\ \mathbf{var}[X(t + \Delta)|X(t)] &= \mathbf{E}[(X(t + \Delta) - \\ &\quad - \mathbf{E}[X(t + \Delta)|X(t)])^2 | X(t)] = \sigma^2 \beta^2(\Delta, t, X(t))\end{aligned}$$

Let $\Delta_1, \Delta_2 > 0$. The conditional covariance between $X(t + \Delta_1 + \Delta_2)$ and $X(t + \Delta_1)$ given $X(t)$ is

$$\begin{aligned} \text{cov}[X(t + \Delta_1 + \Delta_2), X(t + \Delta_1)|X(t)] &= \sigma^2 \beta^2(\Delta, t, X(t)) \\ &+ \mathbf{E}[\theta^T \alpha(\Delta_2, t + \Delta_1, X(t + \Delta_1))X(t + \Delta_1)|X(t)] \\ &- (X(t) + \hat{\theta}^T \alpha(\Delta, t, X(t))) \cdot \\ &\cdot \mathbf{E}[\theta^T \alpha(\Delta_2, t + \Delta_1, X(t + \Delta_1))|X(t)] \end{aligned}$$

It can be further simplified using the structure of α . The expressions inside the expectation operators are functions of $\epsilon_{t+\Delta_1}$, and thus are independent of $X(t)$, so the operators can be removed. Finally, $\text{cov}[X(t + \Delta_1 + \Delta_2), X(t + \Delta_1)|X(t)]$ is a function of $\alpha, \beta, \theta, \sigma$ and $X(t)$. Since $X(t)$ is given, if we replace θ and σ by their estimators $\hat{\theta}$ and $\hat{\sigma}$, we can estimate $\text{cov}[X(t + \Delta_1 + \Delta_2), X(t + \Delta_1)|X(t)]$.

A one-step predictor of $X(t + \Delta)$ given $X(t)$ for any $\Delta > 0$ is given by:

$$\hat{X}(t + \Delta) = \mathbf{E}[X(t + \Delta)|X(t)] = X(t) + \hat{\theta}^T \alpha(\Delta, t, X(t)).$$

5.4 Analyzing $Y(t)$ with a non-stationary $X(t)$: To predict $Y(t + \delta)$ given $Y(t)$, we first fit a polynomial $a_p(t)$ to $Y(t)$ as in Section 4.3, and estimate $\hat{\theta}$ and $\hat{\sigma}$ using Proposition 5.1. Then a predictor $\hat{Y}(t + \delta)$ is given by

$$\begin{aligned} \hat{Y}(t + \delta) &= a_p(t + \delta) + \hat{X}(t + \delta) \\ &= a_p(t + \delta) + X(t) + \hat{\theta}^T \alpha(\delta, t, X(t)). \end{aligned}$$

Similarly, we can estimate the variance and auto-covariance functions of $Y(t)$ for given δ_1 and δ_2 :

$$\begin{aligned} \text{var}[Y(t + \delta)|Y(t)] &= \hat{\sigma}^2 \beta^2(\delta, t, X(t)), \\ \text{cov}[Y(t + \delta_1 + \delta_2), Y(t + \delta_1)|Y(t)] &= \\ \text{cov}[X(t + \delta_1 + \delta_2), X(t + \delta_1)|X(t)]. \end{aligned}$$

6 Computational Experiments

We tested prediction abilities of our IN-AR(1) model on several real datasets. Figure 1 shows a dataset containing a historical isotopic temperature record from the Vostok ice core (about 1K irregularly sampled points), due to Petit et al. [7]. Figure 1(a) overlays the dataset with a 10-point prediction given by the model trained on 100 points. For comparison, we did a similar prediction using a vanilla algorithm that always predicts the last value it sees (Figure 1(b)). The vanilla algorithm produces a step function with a good overall fit to the data, but with no attempt to give accurate short-term predictions. The curve produced by the IN-AR(1) model provides a smoother, much more accurate fit to the data. See the full version of the paper for more experiments [4].

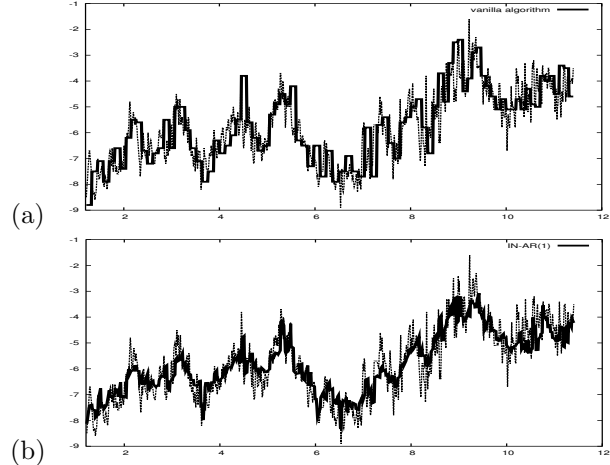


Figure 1: 10-step prediction (a) vanilla, (b) IS-AR(1)

7 Conclusion

We proposed two AR(1)-type models for analyzing irregularly sampled time series. The first model is based on the assumption of stationarity, the second model relaxes this assumption. Both models are extremely simple and can be efficiently fit to the data. The presented approach can be extended to higher order auto-regression processes $\text{AR}(p)$, moving average $\text{MA}(q)$ and autoregressive moving average processes $\text{ARMA}(p, q)$. An interesting research question is to develop a model for analyzing irregularly sampled time series with a non-Gaussian noise.

References

- [1] H. M. Adorf. Interpolation of irregularly sampled data series. In R. Shaw, H. Payne, and J. Hayes, editors, *Astronomical Data Analysis Software and Systems IV, APS Series 77*, 460–463, 1995.
- [2] P. J. Brockwell. Lévy driven carma processes. *Ann.Inst.Statist.Math.*, 53(1):113–124, 2001.
- [3] P. J. Brockwell and R. A. Davis. *Time series: Theory and Methods*. Springer Verlag, 1991.
- [4] E. Erdogan, S. Ma, A. Beygelzimer, and I. Rish. Technical Report, IBM T. J. Watson Center, 2005.
- [5] R. H. Jones and P. V. Tryon. Continuous time series models for unequally spaced data applied to modeling atomic clocks. *SIAM J. Sci. Stat. Comput.*, 4(1):71–81, January 1987.
- [6] R. H. Jones. *Time series analysis with unequally spaced data*. In E. Hannan, P. Krishnaiah, and M. Rao, editors, *Handbook of statistics*, 5, 157–178, North Holland, Amsterdam, 1985.
- [7] J. R. Petit, et al. Climate and atmospheric history of the past 420,000 years from the vostok ice core, antarctica. *Nature*, 399:429–436, 1999.
- [8] D. B. Rubin. *Statistical Analysis With Missing Data*. Wiley, New York, 2002.

Term-Document Matrices [†]

Efstathios Gallopoulos[‡]

Keywords: Low rank approximations, Clustering, LSI.

631

the data matrix A by $\mathcal{U}X$ where

$$(2.1) \quad X = \arg \min_{Y \in \mathbb{R}^{m \times k}} \|A - \mathcal{U}Y\|_F.$$

As in the Centroids Method ([16]), we can find the solution as $\tilde{A} = \mathcal{U}(\mathcal{U}^\top \mathcal{U})^{-1} \mathcal{U}^\top A$. The columns of \mathcal{U} are not orthogonal, in general, though $G = \mathcal{U}^\top \mathcal{U}$ is oblique. We generalize the above idea and consider, for better approximation, using more leading left singular vectors from each cluster block A_j . Let the number of cluster blocks be $l < k$. We solve (2.1), where matrix \mathcal{U} becomes

$$(2.2) \quad \begin{aligned} \mathcal{U} &= [u_1^{(1)}, \dots, u_{k_1}^{(1)}, u_1^{(2)}, \dots, u_{k_2}^{(2)}, \dots, u_1^{(l)}, \dots, u_{k_l}^{(l)}] \\ &= [U_{k_1}^{(1)}, \dots, U_{k_l}^{(l)}], \quad U_{k_j}^{(j)} \in \mathbb{R}^{m \times k_j}, \end{aligned}$$

and $\{u_1^{(j)}, \dots, u_{k_j}^{(j)}\}$ are the k_j leading left singular vectors of A_j . We reserved k for the total number of columns of \mathcal{U} rather than for the number of clusters. To fully specify the algorithm we need a strategy for selecting the k_j 's. We would like to select these values so that A is not too far from the subspace spanned by the columns of \mathcal{U} . More precisely, we want to estimate the number of singular triplets that are necessary for each submatrix so as to have similar (and small) approximation error for all submatrices. Because of the relation between the Frobenius norm and singular values, $\|A\|_F^2 = \sum_{i=1}^l \|A_i\|_F^2 = \sum_{i=1}^l \sum_{j=1}^{r_i} (\sigma_j^{(i)})^2$ we use the ratio

$$(2.3) \quad f(\|A_i\|_F) = \frac{\|A_i\|_F}{\|A\|_F}.$$

as gauge for the number of singular vectors to use from each submatrix. It is worth noting that a strategy based on the ratio of the Frobenius norms of the columns of A vs. A_i was key to the implementation of a recent randomized SVD algorithm [7]. Note that the described heuristic is one of many possibilities. Related ideas are used in Image Processing and Compression. In particular, k_i is selected to be the minimum of the integer nearest to $k f(\|A_i\|_F)$ and the number of columns of A_i . We would be calling the special case of the above algorithm, when $k_j = 1$ for $j = 1, \dots, l$ and $l = k$, as Algorithm 1 and the more general case we just described Algorithm 2; see Table 1.

Rank reduction framework The general result we use is the following:

THEOREM 2.1. (GUTTMAN) Let $A \in \mathbb{R}^{m \times n}$, $X \in \mathbb{R}^{n \times k}$, $Y \in \mathbb{R}^{m \times k}$. Then:

$$\text{rank}(A - AX\Omega^{-1}Y^\top A) = \text{rank}(A) - k$$

if and only if $\Omega = Y^\top AX$ is nonsingular.

If we set $\mathcal{P} = I - AX\Omega^{-1}Y^\top$, it is straightforward to show that $\mathcal{P}^2 = \mathcal{P}$ though, in general $\mathcal{P} \neq \mathcal{P}^\top$. Therefore, \mathcal{P} is an oblique projector and since

$$A - AX\Omega^{-1}Y^\top A = \mathcal{P}A,$$

Input: $m \times n$ tdm A , number of clusters l , integer k

Output: Matrices $X \in \mathbb{R}^{m \times k}$, $Y \in \mathbb{R}^{k \times n}$

I. Run selected clustering algorithm for l clusters;

Reorder the tdm $A = \begin{bmatrix} A_1 & A_2 & \dots & A_l \end{bmatrix}$;

II. Let $k_i = \min(\text{round}(kf(\|A_i\|_F)), n_i)$;

Compute leading left singular vectors for each A_i :

$\mathcal{U} = [u_1^{(1)}, \dots, u_{k_1}^{(1)}, u_1^{(2)}, \dots, u_{k_2}^{(2)}, \dots, u_1^{(l)}, \dots, u_{k_l}^{(l)}]$

III: Solve $\min_Y \|A - \mathcal{U}Y\|_F$: $Y = (\mathcal{U}^\top \mathcal{U})^{-1} (\mathcal{U}^\top A)$;

Compute $\tilde{A} = \mathcal{U}(\mathcal{U}^\top \mathcal{U})^{-1} \mathcal{U}^\top A$;

Table 1: Algorithm 2.

rank reduction follows by the oblique projection of the columns of A . Algorithms 1-2 can be expressed in this framework. We do this here for the former. Let

$$(2.4) \quad S = \text{diag}([\begin{smallmatrix} (1) \\ 1 \end{smallmatrix}, \dots, \begin{smallmatrix} (k) \\ 1 \end{smallmatrix}]) \in \mathbb{R}^{k \times k},$$

and $\mathcal{V} \in \mathbb{R}^{n \times k}$ the block diagonal matrix where each diagonal “block” is the vector $v_1^{(j)}$, $j = 1, \dots, k$. \mathcal{U} is not orthogonal in general, hence the decomposition is not an SVD. Therefore

$$E = A - \tilde{A} = A - A\mathcal{V}S(\mathcal{U}^\top A\mathcal{V}S)^{-1} \mathcal{U}^\top A.$$

This is the same as the rank reduction formula shown in Theorem 2.1 above with $X = \mathcal{V}S$ and $Y = \mathcal{U}$. The above can be extended to Algorithm 2.

Clustered LSI (CLSI). CLSI can be viewed as a block version of LSI. Whereas LSI achieves dimensional reduction by projecting on a number of the leading left singular vectors of A , say k , in CLSI we use k orthogonal vectors for the subspace spanned by a total of k vectors selected from the leading singular vectors of the l cluster blocks; in particular, the algorithm uses k_j leading vectors from each block $j = 1, \dots, l$. Irrespective of the strategy employed to compute the k_j 's, we enforce $k = \sum_{j=1}^l k_j$. Actually, CLSI is a family of methods; e.g. when $k_j = 1$, CLSI corresponds to Alg. 1, and when $l = 1$, $k_1 = k$ (in which case clustering phase I is redundant) it corresponds to LSI. CLSI represents documents by means of the columns of $(\mathcal{U}^\top \mathcal{U})^{-1} \mathcal{U}^\top A \in \mathbb{R}^{k \times n}$. In practice, it is sufficient to construct an orthogonal basis $Q_{\mathcal{U}}$ for \mathcal{U} . Assuming that $Q_{\mathcal{U}}R_{\mathcal{U}} = \mathcal{U}$, where $R_{\mathcal{U}} \in \mathbb{R}^{k \times k}$, it immediately follows that documents are equivalently represented by $R_{\mathcal{U}}(R_{\mathcal{U}}^\top R_{\mathcal{U}})^{-1} R_{\mathcal{U}}^\top Q_{\mathcal{U}}^\top A = Q_{\mathcal{U}}^\top A$. Similarly, any query q is projected by means of its coefficients with respect to the basis $Q_{\mathcal{U}}$, that is by $Q_{\mathcal{U}}^\top q$. To save space we do not list CLSI but note that it is similar to Algorithm 2. The differences are in phase II, where the scheme for selecting the parameters k_i is not specified but left to the user and in phase III, where we explicitly construct an orthogonal basis $Q_{\mathcal{U}}$ for $\text{range}(\mathcal{U})$ and set $X = Q_{\mathcal{U}}$, $Y = Q_{\mathcal{U}}^\top A$.

CLSI adds the extra clustering overhead of phase I, which, according to our experience with the method, is small. On the other hand, the necessary SVD's are applied on l matrix blocks of size $m \times n_j$, $j = 1, \dots, l$ each, whereas LSI would require the leading k singular vectors of the full, $m \times n$, matrix. This can lead to significant reductions in runtime, especially when l is close to k . We assume, of course, that the SVD is computed by means of some iterative algorithm; cf. [2]. The gain in runtime is expected to be larger for Algorithm 1 and the approximation better for Algorithm 2. Further time gains are possible in Algorithm 2 and CLSI, with block methods to compute consecutive singular triplets. We defer discussion of updating strategies for a full version of this paper.

Reference [21] addresses the following important issue. If we possess truncated SVD approximations for blocks of A , how does this information help us to approximate A ? This is directly related to our methodology, since it uses the leading left singular vectors of each block. Let each cluster block A_j be approximated using the optimal rank-1 matrix. Note now that the matrix $[\begin{smallmatrix} (1) \\ 1 \end{smallmatrix} u_1^{(1)} (v_1^{(1)})^\top, \dots, \begin{smallmatrix} (k) \\ 1 \end{smallmatrix} u_1^{(k)} (v_1^{(k)})^\top]$, whose columns are the best rank-1 approximations of each block A_j , can be rewritten as $\mathcal{B} := [\text{best}_1(A_1), \dots, \text{best}_1(A_k)] = \mathcal{U}\mathcal{S}\mathcal{V}^\top$, where $\text{best}_1(A_j)$ is the optimal rank 1 approximation of A_j and $\mathcal{U}, \mathcal{S}, \mathcal{V}$ are as discussed in formula 2.4 above. Consider next using \mathcal{B} or a lower rank approximation of it to approximate A . The error is $\|A - \mathcal{B}\|_F = \|A - \mathcal{U}\mathcal{S}\mathcal{V}^\top\|_F$ in the former case. How does this compare with the approximation offered by Algorithm 1? There we use $\min_{Y \in \mathbb{R}^{k \times m}} \|A - \mathcal{U}Y\|_F$, and thus search the space of $k \times n$ matrices to minimize the Frobenius norm; instead, with the former approach, \mathcal{B} is used directly and no minimization is performed. It immediately follows, therefore, that

$$(2.5) \quad \min_{Y \in \mathbb{R}^{k \times m}} \|A - \mathcal{U}Y\|_F \leq \|A - \mathcal{B}\|_F$$

The above argument can easily be extended to the case where each cluster block is optimally approximated with respect to the Frobenius norm by matrices of rank-1 or higher, except that this time $\mathcal{B} := [\text{best}_{k_1}(A_1), \dots, \text{best}_{k_l}(A_l)]$ so that $\mathcal{B} = \mathcal{U}\mathcal{S}\mathcal{V}^\top$, where \mathcal{U} is as in (2.3) and \mathcal{S}, \mathcal{V} modified accordingly (cf. formula 2.4). Proceeding as above we can show that for the above \mathcal{U} and \mathcal{B} , an inequality as (2.5) holds, except that $Y \in \mathbb{R}^{K \times m}$, where $K = \sum_{j=1}^l k_j$ and $K \geq k$. We will call an algorithm based on approximation \mathcal{B} and will denote the method based on this approach by BSVD and will use it in our experiments. We will assume that the clustering phase results in the same groups as Algorithm 2 and use as rank of the partial SVD approximation of each cluster A_i the value k_i , obtained in Algorithm 2.

A related approach to CLSI is CSVD ([4]) that, like ours, has strong connections with methods that attempt to find local structure in data space. CSVD uses clustering

and then obtains the principal components of each block $\hat{A}_i \hat{A}_i^\top$, where $\hat{A}_i = A_i - A_i e e^\top / n_i$ and e is the vector of all 1's. All mk eigenvalues of these matrices are sorted and components are selected for the reconstruction. For the size of problems we are considering here, a disadvantage of this well-designed method is cost. For related work see also [10], where clustering is first used and LSI is applied independently on each cluster though there is no discussion of low rank approximations of tdm. Of great interest are also the probabilistic techniques in [9]; we plan to study them in light of CLSI. Here we evaluate our methods vs. the powerful algorithms of [19] and further expanded upon together with MATLAB codes in [3]: these are SPQR (sparse pivoted QR approximation) and SCRA (sparse column-row approximation), that construct approximations of the form $A \cong XTY^\top$, where X, Y are sparse and T is small.

3 Numerical experiments

We next explore the performance of our approach using MATLAB 6.5 running on a 2.4 GHz Pentium IV PC with 512 MB of RAM. We examine runtimes, approximation error and IR precision. The SVD and QR decompositions are computed via MATLAB's functions `svds` and ("economy size") `qr`. Our data sets were built using TMG², a MATLAB tool for the construction of tdm's from text collections [20]. Here we only present experiments with tdm `cran lec`. This was produced from the CRANFIELD data set using the best term weighting and normalization combinations according to results in [20]. Runtimes are in seconds and include clustering, though we refer to the extended paper for further analysis. The methods considered are: Centroids ([16]); Algorithms 1 and 2 (Table 1); CLSI (Section 2); SPQR and SCRA (Section 2); BSVD (Section 2). The baseline method is the truncated SVD method (when it is clear from the context we simply call it SVD). We first evaluate the matrix approximation algorithms³. The preprocessing algorithm used was Spherical k-means (S-kmeans) [6]. We record the approximation error $\|A - \hat{A}\|_F$ and compare it with $\|A - U_k S_k V_k^\top\|_F$ from partial SVD using ranks $k = 10 : 10 : 150$. Algorithm 2 employed $l = 2, 3, k$. Because of the nondeterminism in the clustering, we ran 5 experiments and recorded the mean approximation error. Fig. 1 depicts errors (top) and runtimes (bottom) for truncated SVD, SPQR, SCRA, BSVD, Centroids and Algorithms 1 and 2. For Algorithm 2 we depict the errors obtained when $l = 2, 3, k$. Centroids appears to provide reasonable approximations at low cost compared to truncated SVD. Algorithms 1 and 2 appear to return better approximations with the latter being more

²See <http://scgroup.hpclab.ceid.upatras.gr/scgroup/Projects/TMG/>.

³Due to space limitations we only present results with `cran lec` and defer analysis of the effect of clustering.

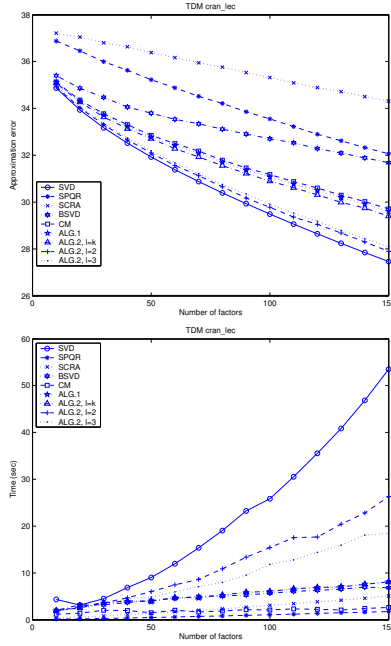


Figure 1: Approximation errors (top) and runtimes (bottom) for methods under review ($l = 2, 3, k$)

accurate. Algorithm 2 provides the best approximations than all other methods, with the approximations becoming better as l drops from k to 2. This is not surprising, as the $l = 1$ corresponds to the optimal (Eckard-Young) rank- k approximation. Fig. 1 (bottom) shows that runtimes increase as l decreases; in particular they are halved between $l = 2$ and $l = 1$, underlying the linearity of the expected runtime of (iterative) SVD algorithms with respect to the number of columns. The experiments indicate that Algorithms 1 and 2 can be economical effective alternatives to the classical rank- k approximation. Centroids also returns good overall performance; faster than Algorithms 1 and 2 with accuracy close to that returned by the former.

We next comment on SPQR, SCRA and BSVD. The compressed representation of A in Algorithms 1 and 2 requires \mathcal{U} (needing mk words) and Y (another nk words). SPQR and SCRA, on the other hand, need much less storage, namely $(n + 1)k$ and $(k + 2)k$ words, respectively, as well as some rows and columns of the tdm. From Fig. 1(top), for our datasets, Algorithms 1 and 2 return better approximations than SPQR, SCRA and BSVD. On the other hand, Centroids and SPQR are faster. Table 2 shows the runtimes that SPQR, SCRA, BSVD and Algorithms 1, 2, need to reach a specific error in the approximation achieved by the truncated SVD. Each table entry contains the number of factors required for such an approximation error as well as the precision in the querying process. Here we only present

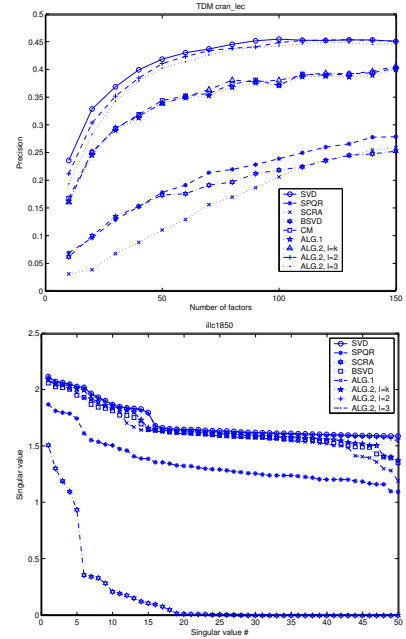


Figure 2: Precision (top); approximate singular values (bottom) for methods under review ($l = 2, 3, k$)

results for the `cran_lec` tdm, with $k = 70, 80$. It appears that SPQR is the fastest, BSVD and SCRA the slowest, while Algorithms 1 and 2 take time comparable with SPQR (for $l = k$). SPQR, SCRA and BSVD lead to larger values of k before the sought error threshold is reached. We also considered matrices unrelated to IR: Fig. 2(bottom) illustrates the 50 largest singular values of matrix `illc1850` (from www.nist.gov/MatrixMarket) approximated using SPQR, SCRA, BSVD and Algorithm 1 and 2. The latter two appear to return the best approximations.

Alg.- k	70	80
SVD	15.3, 70, 0.44	17.7, 80, 0.45
SPQR	2.7, 200, 0.30	3.3, 220, 0.31
SCRA	16.4, 310, 0.34	17.7, 330, 0.35
BSVD	8.9, 200, 0.28	9.4, 230, 0.27
Alg.1	6.1, 110, 0.38	6.8, 130, 0.39
Alg.2, $l = k$	6.1, 110, 0.39	6.4, 120, 0.40
Alg.2, $l = 2$	11.4, 80, 0.44	12.3, 90, 0.44
Alg.2, $l = 3$	8.0, 80, 0.44	9.8, 90, 0.44

Table 2: Results for SVD, SPQR, SCRA, BSVD, Algorithm 1 and 2 (runtime, k , precision) for the same approximation error in `cran_lec` tdm.

We next examine the effectiveness of the schemes for IR. Fig. 2(top) illustrates the $N = 11$ -point interpolated av-

erage precision for the algorithms. Centroids and Algorithm 1 appear to be competitive with LSI. Interestingly, for small values of l , Algorithm 2 can return better precision than LSI. Table 3 tabulates the maximum precision and corresponding k for each method and confirms that Centroids and Algorithms 1 and 2 can be quite effective for IR. Furthermore, for small l , Algorithm 2 may achieve higher precision than LSI.

Alg.-Tdm	med_lfx	cran_lec	cisi_lfc
LSI	0.70, 100	0.46, 100	0.25, 90
SPQR	0.56, 150	0.28, 150	0.17, 150
SCRA	0.51, 150	0.26, 150	0.18, 120
BSVD	0.46, 150	0.25, 150	0.14, 150
CM	0.65, 120	0.40, 150	0.20, 110
Alg.1	0.65, 110	0.40, 150	0.20, 140
Alg.2, $l = k$	0.65, 110	0.41, 150	0.20, 140
Alg.2, $l = 2$	0.70, 70	0.46, 130	0.23, 110
Alg.2, $l = 3$	0.69, 80	0.45, 120	0.23, 110

Table 3: Best precision and corresponding value of k pairs for LSI, SPQR, SCRA, BSVD, Centroids Method and Algorithms 1-2. Boldface indicates best precision/dataset.

Overall, it appears that our methodology can produce approximations with good performance. All proposed methods require smaller runtimes than partial SVD. Our experiments also indicate that CLSI is suitable for the querying process as in some cases it gives better results, faster than LSI. Therefore, CLSI could be used as basis of new algorithmic suites for IR from large and dynamic collections and merits additional study.

Acknowledgements We thank M. Berry, P. Drineas, E. Kokiopoulou and C. Bekas for helpful discussions and the reviewers for their comments regarding this work; due to lack of space, however, several issues and further experiments must be addressed in the full version of the paper; see also our group's website <http://scgroup.hpclab.ceid.upatras.gr>.

References

- [1] D. Achlioptas and F. McSherry, *Fast computation of low rank matrix approximations*, Proc. 33rd Annual ACM STOC, 2001, pp. 611–618.
- [2] M.W. Berry, *Large scale singular value decomposition*, Int. J. Supercomp. Appl. **6** (1992), 13–49.
- [3] M.W. Berry, S. A. Pulatova, and G. W. Stewart, *Computing sparse reduced-rank approximations to sparse matrices*, Tech. Rept. No. CS-04-525, Dept. Comput. Sci., Univ. Tennessee, Knoxville, May 2004.
- [4] V. Castelli, A. Thomasian, and C.-S. Li, *CSVD: Clustering and singular value decomposition for approximate similarity search in high-dimensional spaces*, IEEE Trans. Knowl. Data Eng. **15** (2003), no. 3, 671–685.
- [5] M. Chu and R. Funderlic, *The centroid decomposition: Relationships between discrete variational decompositions and SVDs*, SIAM J. Matrix Anal. Appl. **23** (2002), no. 4, 1025–1044.
- [6] I. S. Dhillon and D. S. Modha, *Concept decompositions for large sparse text data using clustering*, Machine Learning **42** (2001), no. 1, 143–175.
- [7] P. Drineas, R. Kannan, A. Frieze, S. Vempala, and V. Vinay, *Clustering of large graphs via the singular value decomposition*, Machine Learning **56** (2004), 9–33.
- [8] P. Drineas, R. Kannan, and M. Mahoney, *Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix*, Tech. Report TR-1270, Yale University, Department of Computer Science, February 2004.
- [9] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, Proc. 39th Annual Symp. FOCS, ACM, 1998, pp. 370–378.
- [10] J. Gao and J. Zhang, *Clustered SVD strategies in latent semantic indexing*, Tech. rept. no. 382-03, Dept. Comput. Science, Univ. Kentucky, Lexington, KY, 2003.
- [11] L. Guttman, *General Theory and Methods for Matrix Factorizing*, Psychometrika **9** (1944), 1–16.
- [12] L. Hubert, J. Meulman, and W. Heiser, *Two purposes for matrix factorization: A historical appraisal*, SIAM Rev. **42** (2000), no. 1, 68–82.
- [13] E. Kokiopoulou and Y. Saad, *Polynomial filtering in latent semantic indexing for information retrieval*, Proc. 27th ACM SIGIR (New York), ACM, 2004, pp. 104–111.
- [14] T. Kolda and D. O'Leary, *A semidiscrete matrix decomposition for latent semantic indexing information retrieval*, ACM Trans. Inform. Sys. **16** (1998), 322–346.
- [15] D. D. Lee and H. S. Seung, *Algorithms for non-negative matrix factorizations*, Advances in Neural Information Processing Systems **13** (2001), 556–562.
- [16] H. Park and L. Elden, *Matrix rank reduction for data analysis and feature extraction*, Tech. report, University of Minnesota CSE Tech. Rept., 2003.
- [17] H. Park, M. Jeon, and J.B. Rosen, *Lower dimensional representation of text data based on centroids and least squares*, BIT **43-2** (2003), 1–22.
- [18] G. Salton, C. Yang, and A. Wong, *A Vector-Space Model for Automatic Indexing*, CACM **18** (1975), no. 11, 613–620.
- [19] G. W. Stewart, *Four algorithms for the efficient computation of truncated pivoted QR approximations to a sparse matrix*, Numer. Math. **83** (1999), 313–323.
- [20] D. Zeimpekis and E. Gallopoulos, *TMG: A MATLAB toolbox for generating term-document matrices from text collections*, Technical Report, HPCLAB-SCG 1/6-04, Comput. Eng. & Inform. Dept., U. Patras, Greece, June 2004.
- [21] Z. Zhang and H. Zha, *Structure and perturbation analysis of truncated SVD for column-partitioned matrices*, SIAM J. Matrix Anal. Appl. **22** (2001), 1245–1262.
- [22] Z. Zhang, H. Zha, and H. Simon, *Low-rank approximations with sparse factors I: Basic algorithms and error analysis*, SIAM J. Matrix Anal. Appl. **23** (2002), no. 3, 706–727.

WFIM: Weighted Frequent Itemset Mining with a weight range and a minimum weight

Unil Yun and John J. Leggett
Department of Computer Science
Texas A&M University
College Station, TX 77843, U.S.A.
{yunei, leggett@cs.tamu.edu}

ABSTRACT

Researchers have proposed weighted frequent itemset mining algorithms that reflect the importance of items. The main focus of weighted frequent itemset mining concerns satisfying the downward closure property. All weighted association rule mining algorithms suggested so far have been based on the Apriori algorithm. However, pattern growth algorithms are more efficient than Apriori based algorithms. Our main approach is to push the weight constraints into the pattern growth algorithm while maintaining the downward closure property. In this paper, a weight range and a minimum weight constraint are defined and items are given different weights within the weight range. The weight and support of each item are considered separately for pruning the search space. The number of weighted frequent itemsets can be reduced by setting a weight range and a minimum weight, allowing the user to balance support and weight of itemsets. WFIM generates more concise and important weighted frequent itemsets in large databases, particularly dense databases with low minimum support, by adjusting a minimum weight and a weight range.

1. Introduction

Before the FP-tree based mining method [7] was developed, Apriori approaches [13, 14] were usually used based on the downward closure property. That is, if any length k pattern is not frequent in a transaction database, superset patterns can not be frequent. Using this characteristic, Apriori based algorithms prune candidate itemsets. However, Apriori based algorithms need to generate and test all candidates. Moreover, they must repeatedly scan a large amount of the original database in order to check if a candidate is frequent or not. This is inefficient and ineffective. To overcome this problem, pattern growth based approaches [7, 8, 9, 10] were developed. FP-tree based methods mine the complete set of frequent patterns using a divide and conquer method to reduce the search space without generating all the candidates. An association mining algorithm generates frequent patterns and then makes association rules satisfying a minimum support. One of the main limitations of the traditional model for mining frequent itemsets is that all the items are treated uniformly, but real items have different importance. For this reason, weighted frequent itemset mining algorithms [1, 2, 5] have been suggested. The items are given different weights in the transaction database. The main focus in weighted frequent

itemset mining concerns satisfying the downward closure property. The downward closure property is usually broken when different weights are applied to the items according to their significance. Most of the weighted association rule mining algorithms such as [1, 2, 5] have adopted an Apriori algorithm based on the downward closure property. They have suggested the sorted closure property [4], the weighted closure property [1] or other techniques [2, 3, 5] in order to satisfy the downward closure property. However, Apriori based algorithms use candidate set generation and test approaches. It can be very expensive to generate and test all the candidates. Performance analyses [11, 12] have shown that frequent pattern growth algorithms are efficient at mining large databases and more scalable than Apriori based approaches. However, there has been no weighted association rule mining using the pattern growth algorithm because the downward closure property is broken by simply applying the FP growth methodology.

We propose an extended model to tackle these problems of previous frequent itemset mining. Our main goal in this framework is to push weight constraints into the pattern growth algorithm while keeping the downward closure property. The remainder of the paper is organized as follows. In section 2, we describe the problem definition of weighted association rules. In Section 3, we develop a WFIM (Weighted Frequent Itemset Mining). Section 4 shows the extensive experimental results. Finally, conclusions are presented in section 5.

2. Problem definition

Let $I = \{i_1, i_2, \dots, i_n\}$ be a unique set of items. A transaction database, TDB, is a set of transactions in which each transaction, denoted as a tuple $\langle \text{tid}, X \rangle$, contains a unique tid and a set of items. An itemset is called a k -itemset if it contains k items. An itemset $\{x_1, x_2, \dots, x_n\}$ is also represented as x_1, x_2, \dots, x_n . The support of itemset A is the number of transactions containing itemset A in the database. A weight of an item is a non-negative real number that shows the importance of each item. We can use the term, *weighted itemset* to represent a set of weighted items. A simple way to obtain a weighted itemset is to calculate the average value of the weights of the items in the itemset. As defined by previous studies [1, 2, 5], the problem of weighted association rule mining is to find the complete set of association rules satisfying a support constraint and a weight constraint in the database.

TID	Set of items	Frequent Item list
100	a, c, d, f, i, m	c, d, f, m
200	a, c, d, f, m, r	c, d, f, m, r
300	b, d, f, m, p, r	d, f, m, p, r
400	b, c, f, m, p	c, f, m, p
500	c, d, f, m, p, r	c, d, f, m, p, r
600	d, m, r	d, m, r

Table 1: transaction database TDB.

3. WFIM (Weighted Frequent Itemsets Mining)

We suggest an efficient weighted frequent itemset mining algorithm in which the main approach is to push weight constraints into the pattern growth algorithm and provide ways to keep the downward closure property. WFIM adopts an ascending weight ordered prefix tree. The tree is traversed bottom-up because the previous matching can not maintain the downward closure property. A support of each itemset is usually decreased as the length of an itemset is increased, but the weight has a different characteristic. An itemset which has a low weight sometimes can get a higher weight after adding another item with a higher weight, so it is not guaranteed to keep the downward closure property. We present our algorithm in detail and show actual examples in order to illustrate the steps in the FP-tree construction for weighted frequent itemset mining and the mining of a weighted frequent itemset from the FP-tree.

3.1 Preliminaries

Definition 3.1 Weight Range (WR)

The weight of each item is assigned to reflect the importance of each item in the transaction database. A weight is given to an item with a weight range, $W_{\min} \leq W \leq W_{\max}$.

Definition 3.2 minimum weight constraint (min_weight)

In the WFIM, we want to give a balance between the two measures of weight and support. Therefore, we define a minimum weight constraint like a minimum support (min_sup) in order to prune items which have lower weights. Itemset X is defined as a useless itemset if the support of itemset X is less than a minimum support and its weight is also less than a minimum support.

Definition 3.3 Maximum Weight and Minimum Weight

The maximum weight (MaxW) is defined as the value of the maximum weight of items in a transaction database or a conditional database. The minimum weight (MinW) is defined as the value of the minimum weight of items in a transaction database or a conditional database. In WFIM, a MaxW is used in a transaction database and a MinW is used in a conditional database.

Definition 3.4 Weighted Frequent Itemset (WFI)

An itemset X is a weighted infrequent itemset if following pruning, condition 1 or condition 2 below is satisfied. If the itemset X does not satisfy both of these, the itemset X is called a weighted frequent itemset.

Item (min_sup = 3)	a	b	c	d	f	i	m	p	r
Support	2	2	4	5	5	1	6	3	4
Weight ($1.0 \leq WR_1 \leq 1.5$)	1.3	1.1	1.4	1.2	1.5	1.1	1.3	1.0	1.5
Weight ($0.7 \leq WR_2 \leq 1.3$)	1.1	1.0	0.9	1.0	0.7	0.9	1.2	0.8	1.3
Weight ($0.7 \leq WR_3 \leq 0.9$)	0.85	0.75	0.8	0.9	0.75	0.7	0.85	0.7	0.9
Weight ($0.2 \leq WR_4 \leq 0.7$)	0.5	0.3	0.6	0.4	0.7	0.3	0.5	0.2	0.7

Table 2: example of sets of items with different WRs.

Pruning condition 1 (support < min_sup && weight < min_weight) The support of an itemset is less than a minimum support and the weight of an itemset is less than a minimum weight constraint.

Pruning condition 2 (support * MaxW (MinW) < min_sup)

In a transaction database, the value of multiplying itemset's support with a maximum weight (MaxW) among items in the Transaction database is less than a minimum support. In conditional databases, the value of multiplying the support of an itemset with a minimum weight (MinW) of a conditional pattern in the FP-trees is less than a minimum support.

TID	WFI list ($1.0 \leq WR_1 \leq 1.5$) MinW = 1.0	WFI list ($0.7 \leq WR_2 \leq 1.3$) MinW = 0.7	WFI list ($0.7 \leq WR_3 \leq 0.9$) MinW = 0.7	WFI list ($0.2 \leq WR_4 \leq 0.7$) MinW = 0.2
100	a, c, d, f, m	c, d, f, m	c, d, f, m	d, f, m
200	a, c, d, f, m, r	c, d, f, m, r	c, d, f, m, r	d, f, m
300	b, d, f, m, p, r	d, f, m, p, r	d, f, m, r	d, f, m
400	b, c, f, p, m	c, f, p, m	c, f, m	f, m
500	c, d, f, m, p, r	c, d, f, m, p, r	c, d, f, m, r	d, f, m
600	d, m, r	d, m, r	d, m, r	d, m

Table 3: weighted frequent itemsets with different WRs.

Example 1: Table 1 shows transaction database TDB. Table 2 shows example sets of items with different weights. Assume that the minimum support threshold is 3. The frequent list is: Frequent_list = <a:2, b:2, c:4, d:5, f:5, i:1, m:6, p:3, r:4>. The columns in Table 3 show the set of weighted frequent itemsets after pruning weighted infrequent itemsets using pruning condition1 and pruning condition 2 in definition 3.4 by applying different WRs. For example, when WR_3 is applied, item p's support is 3, MaxW is 0.9 and the value (2.7) of multiplying item's support (3) with a MaxW (0.9) of an itemset in the TDB is less than minimum support (3) so item "p" can be removed. Meanwhile, the number of WFI can be increased when WR_1 is used as the weight range. The support of Item "a" in the transaction database is 2. However, a maximum weight is 1.5 so the value (3) of multiplying item's support (2) with a MaxW (1.5) of an itemset is greater than or equal to a minimum support (3) so this item is added in the WFI list.

Example 2: Let us show another example by changing a WR and a min_weight. In this example, Table1 and Table 2 are used and pruning condition 1 in definition 3.4 is applied using WR_2 as a weight range. If the min_weight is 1.2, items "a", "b" and "i" are pruned because the support of these items is less than a minimum support and the weight of these items is also less than a minimum weight.

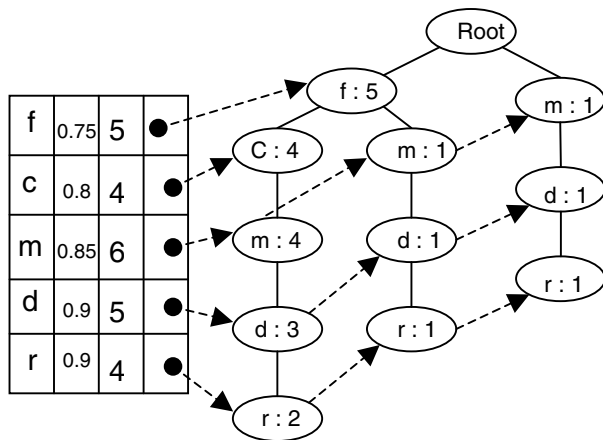


Fig. 1. The global FP tree

In a similar way, we can get the following results by changing min_weight . If min_weight is 1.1, items “i” and “b” are pruned and if min_weight is 1.0, item “i” is pruned. As a result, the number of weighted frequent items can be changed according to different min_weights . When pruning condition 2 of definition 3.4 is considered, if a weight range is $1.0 \leq \text{WR}_1 \leq 1.5$, only item i is pruned because the value of multiplying item i’s support (1) with a maximum weight (1.5) is less than a minimum support (3). However, the items “a” and “b” with 2 as a support are not pruned because the value of multiplying the support (2) of item “a” and “b” with a maximum weight (1.5) is equal to a minimum support (3). In a similar fashion, we can get the following results by changing WRs. If the weight range is $0.7 \leq \text{WR}_3 \leq 0.9$, item “a”, “b”, “i” and “p” are pruned and if the weight range is $0.2 \leq \text{WR}_4 \leq 0.7$, item “a”, “b”, “c”, “i”, “p” and “r” are pruned. Thus, the number of weighted frequent items can be adjusted by using the different WRs.

3.2 FP (Frequent Pattern) tree structure

WFIM uses FP-trees as a compression technique. FP-trees are mainly used in pattern growth algorithms. WFIM computes local frequent items of a prefix by scanning its projected database. The FP-trees in our algorithm are made as follows. Scan the transaction database one time and count the support of each item and check the weight of each item. After this, sort the items in weight ascending order. Although supports of items may be lower than the minimum support and infrequent, the items can not be deleted since infrequent items may become weighted itemsets in the next step. The weighted infrequent items are removed according to pruning conditions 1 and 2 in definition 3.4. For instance, assume that WR_3 is used as a WR, min_sup is 3 and min_weight is 0.7. Then, items “a”, “b”, “p”, and “i” are removed. When an item is inserted in the FP-tree, as already discussed, a weighted infrequent item is removed and the rest, weighted frequent items and itemsets, are sorted by weight ascending order. As shown in [7], each node in the FP-tree has item-id, a weight, count and node link. Separate header tables exist for each FP tree and there is an entry for each item in the header table. Fig.1 presents the global FP-tree and corresponding header table for this example in WFIM.

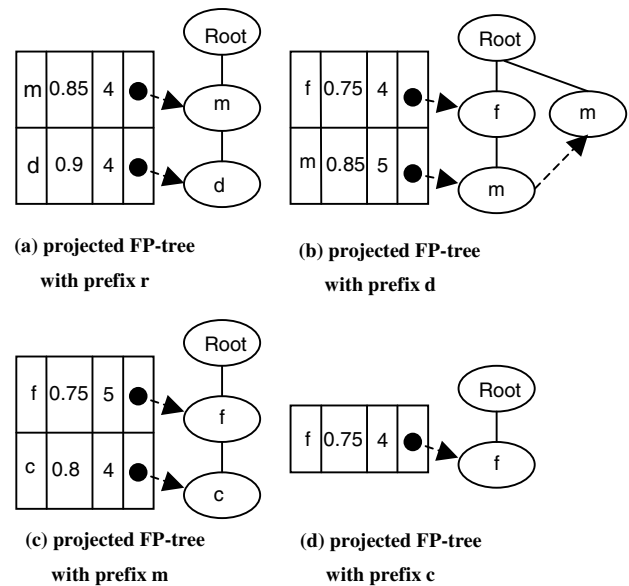


Fig. 2. Conditional FP trees.

3.3 Bottom up Divide and Conquer

After a global FP-tree is generated from the transaction database, WFIM mines frequent itemsets from the FP-tree. The weighted frequent itemsets are generated by adding items one by one. WFIM adapts the divide and conquer approach for mining weighted frequent itemsets. It divides mining the FP-tree into mining smaller FP trees. In the above example of Fig 1, WFIM mines (1) the patterns containing item “r” which has the highest weight, (2) the patterns including “d” but not “r”, (3) the patterns containing “m” but no “r” nor “d”, (4) the patterns containing “c” but no “r”, “d” nor “m”, and finally the patterns containing item “f”. WFIM can find all the subsets of weighted frequent itemsets.

To begin with, for node r, we generate a conditional database by starting from r’s head and following r’s node link. The conditional database for prefix “r:4” contains three transactions: <dmcf:2>, <dmf:1> and <dm:1>. Previous FP-growth algorithms only consider a support in each conditional database so an item “c” is only pruned because the support (2) of item “c” is less than a minimum support (3). However, in WFIM, before constructing the FP-tree, conditions in definition 3.4 are applied in order to check whether it is a weighed frequent itemset. From Table 2, we know that the weights of item “r”, “d”, “m”, “c”, and “f” are 0.9, 0.9, 0.85, 0.8 and 0.75 respectively. There is only one item “r” in the conditional pattern, so the MinW of the conditional pattern is 0.9. By applying pruning conditions, we can see that not only an item “c” but also an item “f” are pruned because the value (2.7) of multiplying item f’s support (3) with a minimum weight (0.9) is less than a minimum support (3). Although the support of item “f” is equal to the minimum support, the item “f” with a lower weight can be deleted. After that, Fig. 2 (a) shows a FP- tree with prefix “r”. As you can see, the conditional FP-tree for prefix “r:4” is a single path tree <md:4>. We can generate all kinds of combinations of items including a conditional pattern “r”. That is, “r”, “rm”, “rd” and “rmd”. The important point is that we can use the minimum weight (MinW) of conditional patterns instead of the maximum

weight (MaxW) of the conditional database or the MaxW of a conditional pattern to prune weighted infrequent itemsets because WFIM uses an ascending weight ordered prefix tree structure to construct conditional databases and the MinW of a prefix is always greater than or equal to the weights of all the items in a conditional database. Hence, the minimum weight of a conditional pattern in the conditional database can be used for applying pruning conditions in definition 3.4.

For node “d”, WFIM derives a frequent pattern (d:5) and three paths in the FP-tree : <mc:3>, <mf:1> and <m:1>. In the FP-growth algorithm, no item is pruned. However, in WFIM, item “c” is pruned in this conditional database with prefix “d”, since the value (2.7) of multiplying item c’s support (3) with the minimum weight (0.9) is less than minimum support (3) although the supports of these items are greater than a minimum support. However, item “m” and item “f” are not pruned. After removing weighted infrequent itemsets in the conditional database, the projected FP-tree for the prefix “d:5” is constructed. Fig 2 (b) shows a conditional FP-tree for prefix d:5. It’s not a single FP-tree, so we divide the conditional FP-tree to mine even smaller conditional FP-trees recursively. As a result, after a recursive process, we obtain weighted frequent itemsets based on conditional pattern “d”: “d:5”, “dm:5”, “df:4” and “dmf:4”. Similarly, we can build projected FP-trees from the global FP-tree and mine weighted frequent itemsets from them recursively. These FP-trees are shown in Fig. 2(c)-(d). (The FP-tree for prefix f:5 is empty and not shown here).

3.4 WFIM algorithm

WFIM can push weight constraints into the pattern growth algorithm and show how to keep the downward closure property. A weight range and a minimum weight are defined and items are given different weights within the weight range. Now, we summarize the weighted frequent itemset mining process and present the mining algorithm.

ALGORITHM. (WFIM): Weighted frequent itemset mining with a weight range and a minimum weight constraint in a large transaction database.

Input: (1) A transaction database : TDB,

(2) minimum support threshold : min_sup,

(3) weights of the items within weight range : w_i ,

(4) minimum weight threshold : min_weight

Output: The complete set of weighted frequent itemsets.

Method:

1. Scan TDB once to find the global weighted frequent items satisfying the following definition: An itemset X is a weighted frequent itemset if the following pruning conditions 1 and 2 are not satisfied.

Condition 1.1: (support < min_sup && weight < min_weight)

Condition 1.2: (support * MaxW < min_sup)

2. Sort items in weight ascending order. The sorted weighted frequent item list forms the weight_order and header of FP tree.

3. Scan the TDB again and build a global FP-tree using weight_order.

4. Mine a global FP-tree for weighted frequent itemset mining in a bottom up manner. Form conditional databases for all remaining

Data sets	Size	#Trans	#Items	A.(M.) t. l.
Connect	12.14M	67557	150	43 (43)
BMS-webView-1	1.28M	59601	497	2.51 (267)
T10I4Dx	5.06-50.6M	200k-1000k	1000	10 (31)

Table 4: Data Characteristics

items in weight_list and complete local weighted frequent itemsets for the conditional databases. (An itemset X is a weighted frequent itemset if the following pruning conditions 1 and 2 are not satisfied).

Condition 4.1: (support < min_sup && weight < min_weight)

Condition 4.2: (support * MinW < min_sup)

5. When all the items in the global header table have been mined, WFIM is finished.

4. Performance Evaluation

In this section, we present our performance study over various datasets. In our experiments, we compared WFIM with FP-growth. We used two real datasets and one synthetic dataset which have been used in pervious experiments [7, 8, 9, 10, 11, 12]. Table 4 shows the characteristic of these datasets. The two real datasets used are connect, and BMS-webView-1. The connect dataset is very dense and includes game state information. The BMS-webView-1 dataset is a very sparse dataset with a web log. These real datasets can be obtained from the UCI machine learning repository [15]. The synthetic datasets were generated from the IBM dataset generator. We use T10I4D100k which is very sparse and contains 100,000 transactions. However, the synthetic datasets T10I4Dx contain 100k to 1000k transactions. These datasets have been used to test scalability in the previous performance evaluations [9, 10, 11, 12]. WFIM is written in C++. In our experiment, a random generation function generates weights for each item. All experiments were performed on a Unix machine. First, we show how the number of weighted frequent itemsets in dense databases and sparse databases with very low minimum support can be adjusted by user’s feedback. Second, we show how WFIM has good scalability against the number of transactions in the datasets. In Fig. 3, we can see that the number of WFI is decreased as the weight range is decreased. In addition, WFIM can adjust the number of weighted frequent itemsets in the dense database with very low minimum support. For instance, in the connect dataset with 10% minimum support, WFIM generates 11003 with a WR: 0.05 – 0.15 and 1989 with a WR: 0.05 – 0.14. Therefore, the number of weighted frequent itemsets can be adjusted by user’s feedback. From Fig. 4, the runtime of WFIM is shown under different WRs.

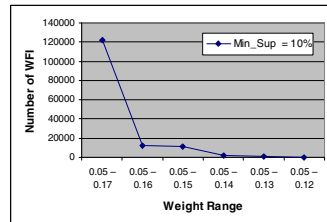


Fig. 3. Num of WFI (Connect dataset)

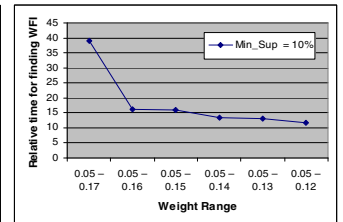


Fig. 4. Runtime (Connect dataset)

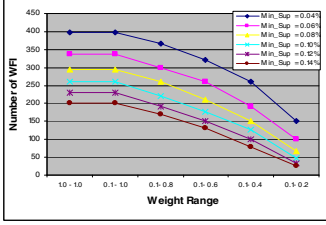


Fig. 5. Num of WFI (BMS-WebView-1)

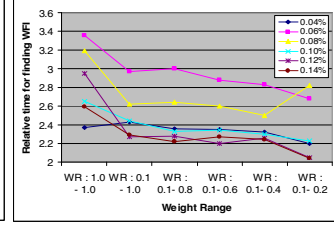


Fig. 6. Run time (BMS-WebView-1)

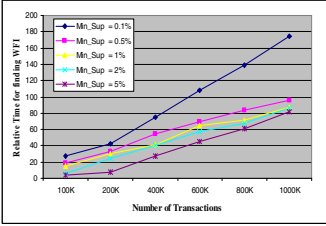


Fig. 7 Runtime (T10I4DxK) (WR: 0.2–0.8)

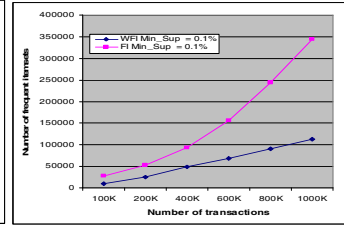


Fig. 8. Num of WFI (T10I4DxK) (WR: 0.2–0.8)

The runtime is sharply reduced even in the connect dataset with a low minimum support as the weight range becomes lower. Fig. 5 and Fig. 6 demonstrate the results of using sparse dataset BMS-WebView-1. In Fig. 5, we can see that the number of WFI is reduced as the minimum support becomes lower in the sparse dataset with the same WR. Moreover, the number of weighted frequent itemsets becomes smaller by diminishing WRs although the minimum support is the same. In Fig. 6, the runtime for finding weighted frequent itemsets is shown according to different WRs as the minimum support is lower. We can see that the run time also can be reduced by diminishing WRs. In the several datasets, WFIM can generate smaller weighted frequent itemsets. It is difficult to reduce frequent itemsets without changing a minimum support. However, WFIM can reduce only the number of frequent itemsets by adjusting WRs when giving weights to each item. In addition, WFIM can generate orders of magnitude fewer patterns than the traditional weighted frequent itemset mining algorithms. To test the scalability with the number of transactions, the T10I4DxK dataset was used. WFIM scales much better than previous weighted frequent itemset mining algorithms which are based on Apriori like approaches [13, 14] since WFIM is based on FP-growth algorithms. In this scalability test, WFIM is compared with FP-growth. Both WFIM and FP-growth show linear scalability with the number of transactions from 100k to 1000k. However, WFIM is much more scalable than FP-growth. In Fig. 7 and Fig. 8, the difference between WFIM and FP-growth becomes clear. We first tested the scalability in terms of base size from 100K tuples to 1000K tuples and different minimum support of 0.1% to 5%. From Fig. 7, we can see that WFIM has much better scalability in terms of base size. The slope ratio for each different minimum support is almost similar. We can also compare WFIM with the FP-growth algorithm in terms of run time. In Fig. 7, you can see that WFIM is faster than FP-growth. More importantly, WFIM can reduce the runtime by adjusting WRs. That is, users can diminish the runtime and the number of WFI by reducing a WR. From Fig. 8, we compared WFIM with FP-growth in order to show that WFIM can generate a more concise result set considering not only frequency of itemsets but also their importance. Our above experiments showed that WFIM can generate smaller but important weighted frequent itemset with various WRs.

Support of connect dataset	Number of W.F.I WR: 0.5–1.5 MW: 1.5	Number of W.F.I WR: 0.5–1.5 MW: 1.0	Number of W.F.I WR: 0.5–1.5 MW: 0.5	Num of F.C.I	Num of F.I
64179 (95%)	125	784	1471	812	2205
60801 (90%)	690	2346	5312	3486	27127
54046 (80%)	2769	2989	3044	15107	533975
47290 (70%)	3997	4089	4093	35875	4129839

Table 5. Comparison of frequent itemsets.

The Table 5 lists the number of weighted frequent itemsets (WFI) with various WRs and min_weights, frequent itemsets (FI) and frequent closed itemsets (FCI). From Table 5, WFIM can generate smaller WFI by using different WRs and min_weights. For example, in Table 4, the number of WFI at a minimum support: 90%, a WR: 0.5 – 1.5 and a min_weight: 0.5, is 5312. However, the number of WFI can be reduced to 2346 with a min_weight: 1.0 and can be further reduced to 690 with a MW: 0.5. The numbers of frequent closed itemsets and frequent itemsets are 3486 and 27127 respectively. In this way, the proper number of weighted frequent itemsets can be found by adjusting a minimum weight.

5. Conclusion

In this paper, we developed WFIM which focuses on weighted frequent itemset mining based on a pattern growth algorithm. The extensive performance analysis shows that WFIM is efficient and scalable in weighted frequent itemset mining. Many improved algorithms using divide and conquer methods have been suggested. In future work, the WFIM can be extended by using a combination of FP-growth based algorithms with better performance.

6. REFERENCES

- [1] Feng Tao, *Weighted Association Rule Mining using Weighted Support and Significant framework*, ACM SIGKDD, Aug 2003.
- [2] Wei Wang, Jiong Yang, Philip S. Yu, *Efficient mining of weighted association rules (WAR)*, ACM SIGKDD, Aug 2000.
- [3] K. Wang, Y. He and J. Han, *Mining Frequent Itemsets Using Support Constraints*, VLDB, Sep 2000.
- [4] Bing Liu, Wynne Hsu, Yiming Ma, *Mining Association Rules with Multiple Minimum Supports*, ACM SIGKDD, June 1999.
- [5] C. H. Cai, Ada Wai-Chee Fu, C. H. Cheng, and W. W. Kwong, *Mining association rules with weighted items*, IDEAS'98, July 1998.
- [6] J. Han and Y. Fu, *Mining Multiple-Level Association Rules in Large Databases*, IEEE TKDE, September/October 1999, pp. 798-805.
- [7] Jiawei Han, Jian Pei, Yiwen Yin, *Mining frequent patterns without candidate generation*, ACM SIGMOD, May 2000.
- [8] Guimei Liu, Hongjun Lu, Yabo Xu, Jeffrey Xu Yu: *Ascending Frequency Ordered Prefix-tree: Efficient Mining of Frequent Patterns*. DASFAA 2003: 65-72.
- [9] Jian Pei, Jiawei Han, *CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets*, DMKD, May 2000.
- [10] Jianyong Wang, Jiawei Han, Jian Pei, *CLOSET+: searching for the best strategies for mining frequent closed itemsets*, ACM SIGKDD, Aug 2003.
- [11] Zijian Zheng, *Real World Performance of Association Rule Algorithms*, ACM SIGKDD, 2001.
- [12] Bart Goethals, Mohammed J. Zaki, *FIMI '03: Workshop on Frequent Itemset Mining Implementations*, FIMI'03, 2003.
- [13] Rakesh Agrawal, Tomasz Imieliński, Arun Swami, *Mining association rules between sets of items in large databases*, ACM SIGMOD, May 1993.
- [14] Rakesh Agrawal, Ramakrishnan Srikant, *Fast Algorithms for Mining Association Rules in Large Databases*, VLDB, September, 1994.
- [15] <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

Model-based Clustering With Probabilistic Constraints

Martin H. C. Law*

Alexander Topchy*

Anil K. Jain*

Abstract

The problem of clustering with constraints is receiving increasing attention. Many existing algorithms assume the specified constraints are correct and consistent. We take a new approach and model the uncertainty of constraints in a principled manner by treating the constraints as random variables. The effect of specified constraints on a subset of points is propagated to other data points by biasing the search for cluster boundaries. By combining the a posteriori enforcement of constraints with the log-likelihood, we obtain a new objective function. An EM-type algorithm derived by variational method is used for efficient parameter estimation. Experimental results demonstrate the usefulness of the proposed algorithm. In particular, our approach can identify the desired clusters even when only a small portion of data participates in constraints.

1 Introduction

The goal of (partitional) clustering [8] is to discover the “intrinsic” grouping of a data set without any class labels. Clustering is an ill-posed problem because the absence of class labels obfuscates the goal of analysis: what is the proper definition of “intrinsic”? In some applications, however, there is a preference for certain clustering solutions. This preference or extrinsic information is often referred to as *side-information*. Examples include alternative metrics between objects, orthogonality to a known partition, additional labels or attributes, relevance of different features and ranks of the objects.

Perhaps the most natural type of side-information in clustering is a set of *constraints*, which specify the relationship between cluster labels of different objects. Constraints are naturally available in many clustering applications. For instance, in image segmentation one can have partial grouping cues for some regions of the image to assist in the overall clustering [20]. Clustering of customers in market-basket database can have multiple records pertaining to the same person. In video retrieval tasks different users may provide alternative annotations of images in small subsets of

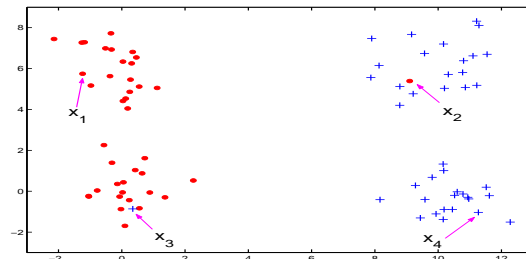


Figure 1: A counter-intuitive clustering solution with pairs (x_1, x_2) and (x_3, x_4) in must-link constraints. The cluster labels of the neighbors of x_2 and x_3 are different from those of x_2 and x_3 . This is the consequence of computing cluster labels instead of cluster boundaries.

a large database [2]; such groupings may be used for semi-supervised clustering of the entire database.

A pairwise *must-link/positive* constraint corresponds to the requirement that two objects should be placed in the same cluster. A pairwise *must-not-link/negative* constraint, on the contrary, means that two objects should be placed in different clusters. Positive constraints tend to be more informative, and the experimental results in [17] suggest that negative constraints only help the clustering results marginally, at the expense of increased computation. Therefore, in this paper we shall focus on positive constraints, though negative constraints can also be incorporated in our model [14]. Note that clustering with constraints is different from learning with unlabelled data, because constraints only specify the relative relationship between labels.

It is important that the effect of constraints be propagated: not only the labels of points involved with constraints should be affected, but also their neighbors [12]. Without this, one can obtain a weird clustering solution, as shown in Figure 1. This intuitive requirement of constraint propagation, unfortunately, is not satisfied by many existing approaches, which estimate the *cluster labels* directly. Our algorithm instead searches for *cluster boundaries* that are most consistent with the constraints and the data.

Different algorithms have been proposed for clustering with constraints. COBWEB and k -means with constraints were proposed in [18] and [19], respectively. Spectral clustering has also been modified to work with

*Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48823, USA. This research was supported by ONR contract # N00014-01-1-0266.

constraints [11, 20]. Metric-learning and clustering with constraints in k -means were considered simultaneously in [4], and was extended to a Hidden Markov random field formulation in [3]. Correlation clustering [1] uses only constraints for clustering. Coordinated conditional information bottleneck [6] discovers novel cluster structure in a data set.

We earlier had proposed a graphical model to represent constraints in model-based clustering [13]. In this paper, we extend that model by (i) incorporating a posterior term in the objective function that corresponds to the enforcement of constraints, (ii) introducing tradeoff parameters of such terms as the strengths of constraints, and (iii) deriving an EM-like algorithm for parameter estimation based on variational method.

2 Method

Let $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$ be the set of d -dimensional data to be clustered by mixture model-based clustering [5] with K clusters. Let $z_i \in \{1, 2, \dots, K\}$ be the iid (hidden) cluster label of \mathbf{y}_i , and let $q_j(\cdot|\theta_j)$ be the probability distribution of the j -th component with parameter θ_j , which is assumed to be Gaussian. Extensions to other type of component distributions are straightforward. Let α_j be the prior probability of the j -th cluster. Consider *group constraints*, a generalization of pairwise constraints, where multiple data points (possibly more than two) are constrained to be in the same cluster. Let w_l be the cluster label of the l -th constraint group, with L as the total number of groups. The random variable z_i takes the value of w_l when the constraint on \mathbf{y}_i is enforced. Introduce the random variable v_i , which corresponds to the constraint on \mathbf{y}_i . When it is “on” (non-zero), the constraint is enforced, i.e., $z_i = w_l$. When it is “off” (zero), the constraint is disabled, and z_i is distributed independently according to its prior probabilities. The probability that the constraint is “on” corresponds to the certainty of the constraint. For example, to represent the constraints for the data in Figure 3, we should assign $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$ to the first group and $\mathbf{y}_5, \mathbf{y}_6$ to the second group. Since there are two groups, $L = 2$. If we assume the confidence of all the constraints to be 0.5, the first group constraint is represented by setting the parameters $\gamma_{i2} = 0$ and $\gamma_{i1} = 0.5$ for $i = 1, 2, 3, 4$, whereas the second group constraint is represented by $\gamma_{i1} = 0$ and $\gamma_{i2} = 0.5$ for $i = 5, 6$. The meaning of γ_{il} will be defined shortly after.

The presence of constraints introduces dependence only among z_i . Different \mathbf{y}_i are still independent given z_i . Therefore, our model can be factorized as

$$P(\mathcal{Y}) = \sum_{\mathbf{z}, \mathbf{v}, \mathbf{w}} \left(\prod_i P(\mathbf{y}_i | z_i) P(z_i | v_i, \mathbf{w}) P(v_i) \right) \prod_{l=1}^L P(w_l).$$

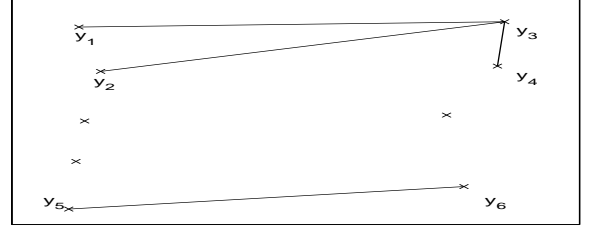


Figure 3: An example set of constraints. Points that should be put in the same cluster are joined by lines.

The rest of the model is specified as follows:

$$(2.1) \quad \begin{aligned} P(w_l = j) &= \alpha_j, \quad 1 \leq l \leq L, 1 \leq j \leq K, \\ P(v_i = l) &= \gamma_{il}, \quad 1 \leq i \leq N, 1 \leq l \leq L, \\ P(z_i = j | v_i, \mathbf{w}) &= \begin{cases} \alpha_j & \text{if } v_i = 0 \\ \delta_{w_l, j} & \text{if } v_i = l \end{cases}, \\ P(\mathbf{y}_i | z_i = j) &= q_j(\mathbf{y}_i), \end{aligned}$$

where $\mathbf{z} = (z_1, \dots, z_N)$, $\mathbf{v} = (v_1, \dots, v_N)$ and $\mathbf{w} = (w_1, \dots, w_L)$ are the hidden variables. Here, γ_{il} denotes the probability that the constraint of tying \mathbf{y}_i to the l -th group is “on”. The values of γ_{il} are either specified by the user to represent the confidence of different constraints, or they can be set to 0.5 when the certainties of the constraints are unknown. An example of such a model with seven data points and three group labels is shown in Figure 2. The model in [17] is a special case of this model when all γ_{il} are binary.

An EM algorithm can be derived to learn the parameters of this model by maximizing the data log-likelihood [13]. The M-step is described by

$$(2.2) \quad a_j = \sum_{l=1}^L P(w_l = j | \mathcal{Y}) + \sum_{i=1}^N P(v_i = 0, z_i = j | \mathcal{Y}),$$

$$(2.3) \quad \hat{\alpha}_j = \frac{a_j}{\sum_{j'=1}^K a_{j'}},$$

$$(2.4) \quad \hat{\boldsymbol{\mu}}_j = \frac{\sum_{i=1}^N P(z_i = j | \mathcal{Y}) \mathbf{y}_i}{\sum_{i=1}^N P(z_i = j | \mathcal{Y})},$$

$$(2.5) \quad \hat{\mathbf{C}}_j = \frac{\sum_{i=1}^N P(z_i = j | \{\mathbf{y}_i\}) (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_j) (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_j)^T}{\sum_{i=1}^N P(z_i = j | \mathcal{Y})}.$$

Here, the j -th component is assumed to be a Gaussian with mean $\boldsymbol{\mu}_j$ and covariance \mathbf{C}_j , $\theta_j = (\boldsymbol{\mu}_j, \mathbf{C}_j)$. The E-step consists of computing the probabilities $P(w_l = j | \mathcal{Y})$, $P(z_i = j | \mathcal{Y})$ and $P(v_i = 0, z_i = j | \mathcal{Y})$, which can be done by standard Bayesian network inference algorithms like belief propagation or junction tree [10]. Because of the simplicity of the structure of the graphical model, inference can be carried out efficiently. In particular, the complexity is virtually the same as the standard

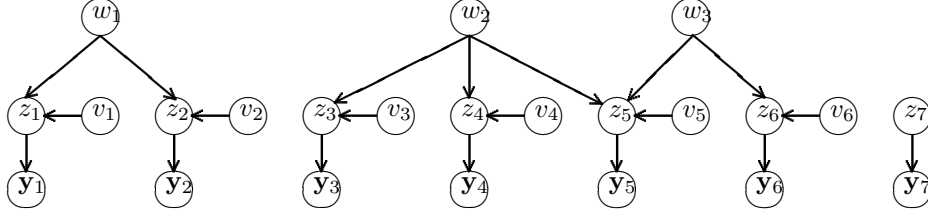


Figure 2: An example of the graphical model with constraint uncertainties for 9 points in 3 groups. Note that each connected component in the graph is a polytree and hence the belief propagation algorithm can be used to calculate the probabilities exactly.

EM algorithm when there are only positive constraints that tie each of the z_i to one group label.

2.1 Modification of the Objective Function The proposed graphical model can handle the uncertainties of constraints elegantly. However, the tradeoff between the constraint information and the data information cannot be controlled explicitly. To cope with this, an additional term that represents the *a posteriori* enforcement of constraints is included in the objective function. This is a distinct characteristic of the proposed model: since each constraint is represented as a random variable, we can consider its posterior probability. The posterior probability that a constraint is “on” reflects how strongly a constraint is enforced by the current parameter estimate. Instead of the binary statement that a constraint is satisfied or violated, we can now consider the *partial* degree of satisfaction of a constraint. This way, the violation of constraints is measured more accurately. Formally, the new objective function is

$$(2.6) \quad E = \log P(\mathcal{Y}|\Theta) + \sum_{i,l} \beta_{il} \log P(v_i = l | \mathcal{Y}, \Theta),$$

with the convention that β_{il} is zero when $P(v_i = l) = 0$. The posterior enforcement of the constraint on \mathbf{y}_i is represented by $\log P(v_i = l | \mathcal{Y}, \Theta)$, because the event $v_i = l$ corresponds to the constraint that z_i is tied to w_l . The strengths of the constraints are represented by β_{il} , which are the user-specified tradeoff parameters between the influence of the posterior probability and the data log-likelihood. In this paper, we set $\beta_{il} = \alpha \tau_{il}$, where α is a global constraint strength parameter and τ_{il} represents the goodness of the constraint tying \mathbf{y}_i to the l -th group. If τ_{il} is unknown, we can assume that all constraints are equally important and set τ_{il} to one. The only parameter that needs to be specified is α .

Direct optimization of E is difficult and we resort to variational method. Due to limitation of space, we defer the derivation and the update formulae in the long version of the paper [14]. In brief, there is no change in the E-step, whereas the M-step (Equations

(2.3) to (2.5)) is modified by replacing the cluster label probabilities with a weighted sum of constraint satisfaction and cluster label probabilities.

3 Experiments

3.1 Synthetic Data Set Four 2D Gaussian distributions with mean vectors $\begin{bmatrix} 1.5 \\ 6 \end{bmatrix}$, $\begin{bmatrix} -1.5 \\ 6 \end{bmatrix}$, $\begin{bmatrix} -1.5 \\ -6 \end{bmatrix}$, $\begin{bmatrix} 1.5 \\ -6 \end{bmatrix}$, and identity covariance matrix are considered. 150 data points are generated from each of the four Gaussians. The number of target clusters (K) is two. In the absence of any constraints, two horizontal clusters are successfully discovered by the EM algorithm (Figure 4(d)). Ten multiple random restarts were used to avoid poor local minima. Now suppose that prior information favors two vertical clusters instead of the more natural horizontal clusters. This prior information can be incorporated by constraining a data point in the leftmost (rightmost) top cluster to belong to the same cluster as a data point in the leftmost (rightmost) bottom cluster. To determine the strength of a constraint, τ_{il} is randomly drawn from the interval $[0.6, 1]$, and we set $\beta_{il} = \alpha \tau_{il}$, where α is the global constraint strength specified by the user. To demonstrate the importance of constraint uncertainty, the constraints are corrupted with noise: a data point is connected to a randomly chosen point with probability $1 - \tau_{il}$. An example set of constraints with 15% of data points involved in the constraints is shown in Figure 4(a). Different portions of points participating in constraints are studied in the experiment. In all cases, the proposed algorithm can recover the desired two “vertical” clusters, whereas other algorithms (such as [17]) fail. It is worthy of note that our algorithm can recover the target structure with as few as 2.5% of the data points participating in constraints. If a clustering with constraint algorithm that deduces cluster labels directly is used, the anomaly illustrated in Figure 1 can happen, because of the small number of constraints.

3.2 Real World Data Sets Experiments are also performed on three data sets in the UCI machine learning repository (Table 1). For each data set, K is set

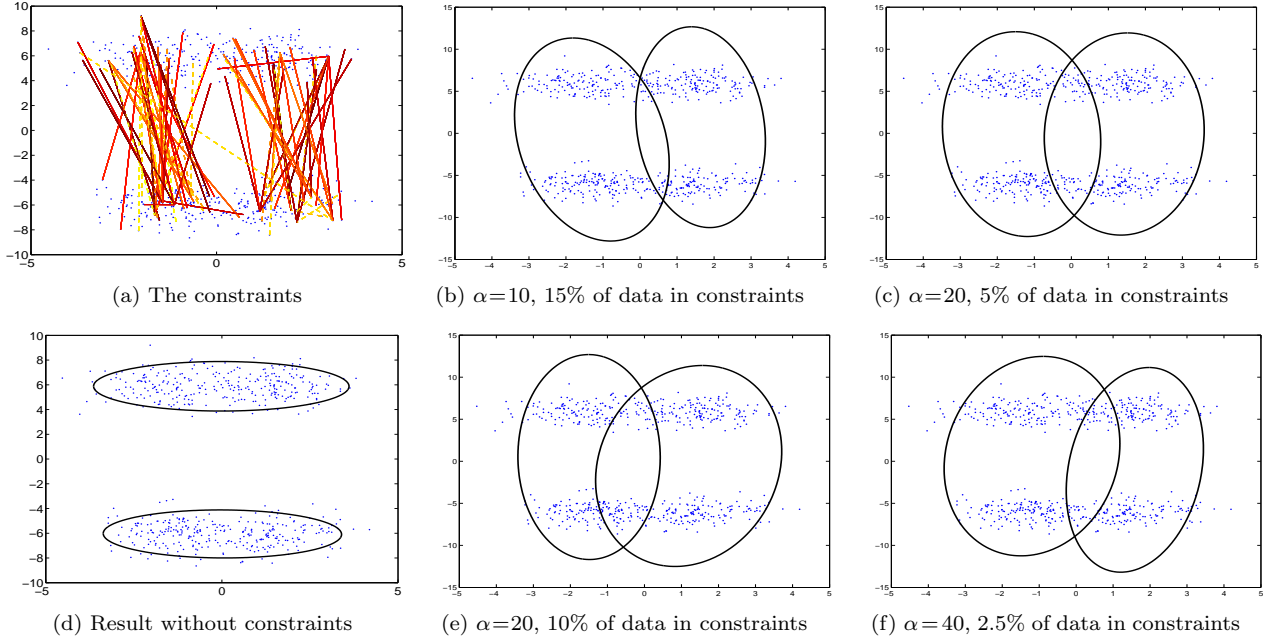


Figure 4: Results of the proposed algorithm when different number of data points participate in constraints.

	Full name	N	D	K	d	m
wdbc	Wisconsin breast cancer	569	30	2	10	6
derm	Dermatology	366	33	6	5	12
image	Image Segmentation	2410	18	7	10	14

Table 1: Data sets used in the experiments. N : size of data. D : no. of features. K : no. of classes. d : PCA dimension. m : no. of points labelled by a teacher.

to the true number of classes. The distributed learning scenario described in [17], where different teachers label a small subset of the data, is used to generate the constraints. Each teacher labels $2K$ or $3K$ data points, depending on the size of the data. The labels assigned by the teachers are corrupted with noise with probability based on the confidence of those labels. The confidence is used as constraint strengths as in the case for synthetic data. The number of teachers is determined by the percentage of points in constraints. PCA is used to reduce the dimensionality of the data sets to d to ensure there are a sufficient number of data points to estimate the covariance matrix, with d determined by the size of the data. For each data set, half of the data is used for clustering, while the other half is used to evaluate the clusters based on the ground truth labels. We compare the performance of soft constraints [13], hard constraints (equivalent to [17]) and the proposed method in Table 2. The proposed algorithm leads to superior clusters when compared with the results of

hard and soft constraints. The improvement due to constraints is not very significant for the Dermatology data set, because the standard EM algorithm is able to find a good quadratic cluster boundary. The degradation of performance in **image** for hard constraints is due to the existence of erroneous constraints.

4 Discussion

The proposed algorithm can be viewed from alternative perspectives. It can be regarded as training a mixture of Gaussians in a discriminative manner [9], with the constraints serving as relaxed label information. If different Gaussians share the same covariance matrix, EM algorithm is related to performing discriminant analysis with posterior probabilities as weights [7]. This provides an alternative justification of our approach even when the Gaussian assumption is not satisfied, because the EM algorithm finds the clusters that are best separated.

The global constraint strength α is the only parameter that requires tuning. In practice, α is chosen automatically by setting apart a set of “validation constraints” or “validation teachers”, which are not used to estimate the clusters. The smallest value of α that leads to clusters that violate the validation information the least is chosen. Note that we do not observe significant overfitting in our experiments. So, one may as well use the value of α that leads to the smallest violation of the training constraints.

	20% of data in constraints					10% of data in constraints					5% of data in constraints				
	H	S	P	$P \geq H$	$P \geq S$	H	S	P	$P \geq H$	$P \geq S$	H	S	P	$P \geq H$	$P \geq S$
wdbc	6.5	1.9	16.7	9	10	2.8	1.5	13.3	9	9	3.4	-1.1	9.4	9	10
derm	0.5	1.0	3.5	5	6	2.9	2.5	5.2	5	6	1.4	2.3	4.5	6	9
image	-2.6	2.6	6.1	8	10	-3.1	0.5	9.0	9	8	-5.8	2.2	5.0	9	6

Table 2: Results on real world data sets. Average improvements in accuracy (in %) with respect to no constraints for soft constraints (S), hard constraints (H), posterior constraints, i.e., the proposed algorithm, (P), are shown. The number of times that the proposed algorithm outperforms the other two constraint algorithms in 10 runs is also shown.

5 Conclusion

We have proposed a graphical model with the constraints as random variables. This principled approach enables us to state the prior certainty and posterior enforcement of a constraint. The model is more robust towards noisy constraints, and it provides a more general approach to estimate constraint violation. Metric learning is automatic because covariance matrices are estimated. The use of variational method provides an efficient approach for parameter estimation. Experimental results show the utility of the proposed method. For future work, we plan to estimate the number of clusters automatically. Can traditional criteria like AIC, BIC or MDL be modified to work in the presence of constraints? A kernel version of this algorithm can be developed for clusters with general shapes.

References

- [1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proc. Annual IEEE Symp. on Foundations of Computer Science*, 2002.
- [2] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning via equivalence constraints, with applications to the enhancement of image and video retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.
- [3] S. Basu, M. Bilenko, and R.J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. ACM SIGKDD, Intl. Conf. on Knowledge Discovery and Data Mining*, 2004.
- [4] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. Intl. Conf. on Machine Learning*, 2004.
- [5] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.
- [6] D. Gondek and T. Hofmann. Non-redundant data clustering. In *Proc. Intl. Conf. on Data Mining*, 2004.
- [7] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society series B*, 58:158–176, 1996.
- [8] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.
- [9] T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Advances in Neural Information Processing Systems 11*, pages 494–500. MIT Press, 1998.
- [10] M. I. Jordan. *Learning in Graphical Models*. Institute of Mathematical Statistics, 1999.
- [11] S. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *Proc. Intl. Joint Conf. on Artificial Intelligence*, pages 561–566, 2003.
- [12] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. Intl. Conf. on Machine Learning*, pages 307–314, 2002.
- [13] M. H. Law, A. Topchy, and A. K. Jain. Clustering with soft and group constraints. In *Proc. Joint IAPR International Workshops on Structural, Syntactic, And Statistical Pattern Recognition*, 2004.
- [14] M. H. C. Law, A. P. Topchy, and A. K. Jain. Model-based clustering with soft and probabilistic constraints. Technical report, Michigan State University, 2004.
- [15] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, San Francisco, 1999.
- [16] S. T. Roweis, L. K. Saul, and G. E. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14*, pages 889–896. MIT Press, 2002.
- [17] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.
- [18] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. Intl. Conf. on Machine Learning*, pages 1103–1110, 2000.
- [19] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. Intl. Conf. on Machine Learning*, pages 577–584, 2001.
- [20] S. X. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):173–183, 2004.

- Aggarwal, C. C., 56, 80, 115
 Alhajj, R., 496
 Alhammady, H., 481
 Arunasalam, B., 173
 Atallah, M., 404
- Bailey-Kellogg, C., 427
 Barker, K., 496
 Bar-Or, A., 466
 Barr, S., 161
 Berrospi, C., 581
 Beygelzimer, A., 626
 Bhatnagar, R., 591
 Bi, J., 476
 Bian, H., 591
 Biswas, H., 546
 Brand, M., 12
 Brodley, C. E., 439
- Calders, T., 250
 Casas-Garriga, G., 380
 Castelli, V., 449
 Chang, C.-H., 501
 Chang, E., 611
 Chang, E. Y., 322
 Chawla, S., 173
 Chen, A. L. P., 68
 Chen, Z., 262
 Cheng, H., 601
 Cheung, D. W., 461
 Chi, Y., 346
 Chiu, D.-Y., 68
 Cho, M., 461
 Choy, T. -S., 161
 Chung, S. M., 415
 Cormode, G., 44
- Davidson, I., 138
 Ding, C., 32, 606
 Domeniconi, C., 217
 Dundar, M. M., 476
- Elisseeff, A., 581
 Erdogan, E., 626
- Fan, W., 486
 Fern, X. Z., 439
 Friedl, M. A., 439
 Fu, A. W.-C., 516
 Fung, G., 476
- Gallopoulos, E., 631
 Garg, A., 126
- Goethals, B., 239, 250
 Gondek, D., 126
 Greco, G., 561
 Guzzo, A., 561
 Gwadera, R., 404
- Han, J., 286, 601
 Han, Y., 227
 Hauskrecht, M., 370
 He, X., 606
 Hu, Q., 195
 Huang, K.-Y., 501
 Huang, S., 541
- Idé, T., 571
 Inoue, K., 571
- Jain, A. K., 641
 Jin, R., 616
- Kabán, A., 183
 Kalashnikov, D. V., 262
 Kang, F., 616
 Karypis, G., 205, 358, 556
 Keogh, E., 506, 531
 Keren, D., 466
 Kokiopoulou, E., 511
 Kumar, M. B., 566
 Kumar, N., 531
- Labbi, A., 581
 Lam, W., 227, 298
 Lane, T., 334
 Law, M. H. C., 641
 Lee, C.-H., 596
 Leggett, J. J., 636
 Lemire, D., 471
 Li, T., 521, 526
 Li, X.-Y., 551
 Li, Y., 103
 Lin, C.-H., 68
 Lin, K.-Z., 501
 Liu, C., 286
 Liu, S., 1
 Liu, Y., 496
 Lolla, V. N., 531
 Lonardi, S., 531
 Lu, C.-t., 486
 Luo, C., 415
- Ma, S., 626
 Machiraju, R., 161
 Maclachlan, A., 471
- Manco, G., 561
 Mathuria, J., 486
 Mehrotra, S., 262
 Mehta, S., 161
 Mitchell, T. M., 310
 Muhonen, J., 239
 Muntz, R. M., 346
 Muthukrishnan, S., 44
- Niculescu, R. S., 310
 Nolan, L. A., 183
- Obradovic, Z., 621
 Özyer, T., 496
- Pal, S., 546
 Panda, N., 322
 Pandey, V. N., 427
 Pang, K.-P., 392
 Parthasarathy, S., 161
 Pearson, R. K., 20
 Pei, J., 461
 Peng, J., 150
 Peng, K., 621
 Perng, C.-S., 586
- Qiu, S., 334
- Ramakrishnan, N., 427
 Ramamohanarao, K., 481
 Rao, B., 476
 Rao, R. B., 310
 Rashid, A. M., 556
 Ratanamahatana, C. A., 506, 531
 Ravi, S. S., 138
 Raychaudhury, S., 183
 Riedl, J., 556
 Rish, I., 626
- Saad, Y., 511
 Saccà, D., 561
 Sathyakama, S., 476
 Schuster, A., 466
 She, R., 536, 576
 Shen, X., 1
 Simon, H. D., 606
 Singliar, T., 370
 Smyth, P., 274
 Srikant, Y. N., 566
 Sun, P., 173
 Sun, Z., 551
 Szpankowski, W., 404

- Tadepalli, S., 427
Ting, K.-M., 392
Tirenni, G., 581
Toivonen, H., 239
Topchy, A., 641
Tseng, V. S.-M., 596

Vaithyanathan, S., 126
Vlachos, M., 449
Vucetic, S., 621

Wang, H., 346, 586
Wang, J., 205
Wang, K., 536, 576
Wang, Y., 103
Webb, G. I., 541
Wei, L., 531
White, S., 274
Wilkins, J., 161
Wolff, R., 466
Wong, R. C.-W., 516
Wong, T.-L., 298
Wong, W. H., 1
Wright, R. N., 92
Wu, G., 611
Wu, X., 103
Wu, Y., 103
Wu, Y.-H., 68

Xia, Y., 150
Xie, Z., 195
Xu, S., 491
Xu, Y., 536, 576

Yan, B., 217
Yan, X., 286, 601
Yang, H., 161
Yang, Z., 92
Ye, J., 32
Yu, D., 195
Yu, H., 286
Yu, P., 449
Yu, P. S., 56, 115, 286, 346, 536,
551, 576, 586
Yun, U., 636

Zeimpekis, D., 631
Zhang, J., 491
Zhang, Z., 611
Zhao, Y., 358
Zhong, S., 92
Zhu, S., 521, 526