Workshop on Clustering High Dimensional Data and its Applications

April 23, 2005
Sutton Place Hotel
Newport Beach, California

To be Held in Conjunction with the
Fifth SIAM International Conference on Data Mining
(SDM 2005)

Organizing Committee

Inderjit Dhillon
Department of Comp. Sciences
University of Texas
Austin, TX 78712-1188
Phone: (512) 471-9725
Fax: (512) 471-8885
inderjit@cs.utexas.edu

Jacob Kogan
Dept. of Mathematics and Statistics
University of Maryland
Baltimore, MD 21250
Phone: (410) 455-3297
Fax: (410) 455-1066
kogan@math.umbc.edu

Joydeep Ghosh
Dept. of Electrical and Computer Engineering
University of Texas
Austin, TX 78712
Phone:(512)471-8980
Fax: (512) 471-2893
ghosh@ece.utexas.edu

Preview

Clustering, or partitioning of datasets into subsets (also called clusters) so that the members of a cluster are more similar to each other than to members of other clusters is a long standing problem with rich history of documented research. Explosive use of the Internet and recent advances in information technology bring new challenges to this exciting research area. In many important applications the data resides in high dimensional vector spaces. While processing of high dimensional data represents computational challenges, in many cases the high dimensional vectors are sparse. The sparsity allows an efficient data clustering. In this one day workshop papers will be presented by experts from academia and industry. Clustering issues that will be covered include: high dimensional clustering using both spectral and iterative relocation techniques, and applications to intrusion detection and text analysis. The workshop held in conjunction with the Fifth SIAM Conference on Data Mining brings together applied mathematicians, computers scientists, and computational statisticians working toward design of next generation clustering algorithms and software.

Inderjit S. Dhillon	Jacob Kogan	Joydeep Ghosh
Department of Comp. Sc.	Department of Math Stats.	Department of Elect. Comp. Engr.
Univ. Texas, Austin	Univ.of Maryland Baltimore County	Univ. of Texas, Austin

Acknowledgments

Special thanks to the members of the Program Committee for their diligent efforts in reviewing all the manuscripts submitted.

Program Committee

Workshop on Clustering High Dimensional Data
and its Applications

April 23, 2005

Table of Contents

Clustering by Maximizing Sum-of-Squared Separation Distance

Yixin Chen*† Jinbo Bi‡

Abstract

Maximizing the separating margin is crucial for the good generalization performance of Support Vector Machines (SVMs). Analogous to the definition of separation distance or separating margin in SVMs, we propose a definition on separation distance in clustering tasks when a hyperplane is used to separate clusters. For given training data and a given metric distance, by maximizing the proposed separation distance, our clustering algorithm constructs an "optimal" hyperplane that can be applied to unseen data in the future. The resulting hyperplane corresponds to a nonlinear decision boundary in the input feature space through an appropriate distance feature mapping. A graph-theoretic perspective of the proposed method is discussed. In particular, we show that, under certain conditions, the proposed clustering algorithm is equivalent to a spectral relaxed graph cut. Extensive experimental results are provided to validate the method.

Keywords: biclustering, optimization, graph partitioning, spectral relaxation, spectral clustering.

1 Introduction and Overview.

As an important branch in unsupervised learning, cluster analysis aims at partitioning a collection of objects into groups or "clusters" so that members within each cluster are more closely related to one another than objects assigned to different clusters [12]. In a variety of areas including bio-informatics, computer vision, information retrieval, data mining, and VLSI design, clustering algorithms provide automated tools to help identify a structure from an unlabeled data set. There is a rich resource of prior work on this subject. The works reviewed below are most related to ours, which by no means represent a comprehensive list.

1.1 Related Work. Depending on the underlying model assumption, clustering algorithms roughly fall into two categories: *generative approach* and *discriminative approach*.

A generative clustering algorithm supposes that the data are independent and identically distributed samples generated from an unknown probability density function. This density function is usually parameterized by a mixture model: weighted sum of a collection of component density functions, each of which characterizes one of the clusters. Consequently, clustering turns into a density estimation problem, which is commonly tackled by Expectation Maximization (EM) algorithm [6]. However, in practice there is usually no *a priori* knowledge about the parametric forms of component density functions. In many applications the Gaussian assumption does not lead to satisfactory performance. Moreover, estimation techniques, such as EM algorithm, only guarantee local optimality. Nonetheless, generative clustering techniques have several advantages, such as the scalability to large data sets [4] and the ability to handle examples outside the training set.

A discriminative clustering algorithm works directly on the training data without explicitly assuming an underlying probability model. Each training sample is assigned to one and only one of the clusters. A "loss" function is defined over all possible assignments. It measures the degree to which the clustering goal is met. Optimal cluster assignments for all training samples are achieved by optimizing the loss function. Since this optimization problem is essentially combinatorial, discriminative clustering algorithms are also called *combinatorial clustering algorithms* [14]. Combinatorial optimization is straightforward in principle: searching all possible assignments. Unfortunately, this is feasible only for very small data sets since the number of distinct assignments is $O(k^n)$, where k is the number of clusters and n is the sample size. Therefore, practical discriminative clustering algorithms typically seek for a trade-off between optimality and computational complexity. For example, the k-means [13] and k-medoids algorithms [16] use an iterative greedy descent strategy to search for a suboptimal partition. Agglomerative (divisive) clustering methods [16] generate a hierarchy of clusters via recursively merging (splitting) clusters according to certain greedy heuristics. Spectral clustering [20, 8, 17] formulates clustering as a graph partitioning problem. The

*Department of Computer Science, University of New Orleans, New Orleans, LA 70148. Email: yixin@cs.uno.edu

†The Research Institute for Children, 200 Henry Clay Avenue, Research & Education, New Orleans, LA 70118

‡Computer-Aided Diagnosis & Therapy Group, Siemens Medical Solutions, Inc., 51 Valley Stream Parkway, Malvern, PA 19355. Email: jinbo.bi@siemens.com

optimal partition is approximated by eigenvectors of a properly normalized affinity matrix of the graph. The relationships between spectral partitioning methods and kernel k-means are discussed in [7].

Interesting connections between generative and discriminative approaches have been discussed in [2]. The equivalence between a class of generative and discriminative clustering algorithms were established based on Bregman Divergence loss function [2]. Unlike generative approaches, which can predict on examples outside the training set, many discriminative clustering algorithms, including those mentioned above, cannot do so without rerunning the algorithm. In a recent work by Bengio *et al.*, a family of discriminative clustering methods were extended to deal with "out-of-sample" examples [3].

The work proposed in this paper is a new discriminative clustering algorithm based on a loss function motivated by the separating margin of Support Vector Machines (SVM) [21]. An overview is provided below.

1.2 Overview of Our Approach. In the past decade, SVM has become an effective and robust tool for solving supervised classification problems. Loosely speaking, SVM finds an "optimal" hyperplane, in a kernel-induced feature space, which separates two classes of samples with the maximal margin. The characteristics of SVM can be summarized as follows:
1. maximizing a separation distance, i.e., the so-called separating margin, and
2. applying appropriate feature mappings, i.e., the kernel mapping.
By a kernel mapping, SVM is able to construct nonlinear models using a linear learning mechanism. The margin concept provides a theoretical basis for SVM since maximizing margin is related to minimizing upper bounds on the generalization error [21]. This paper proposes a novel clustering algorithm aiming to make analogous uses of the above two well-established characteristics of SVM in unsupervised learning, in particular, cluster analysis.

From a basic rationale of clustering, members of different clusters should be as dissimilar as possible. In terms of a linear separating boundary, intuitively, a good bipartition should divide the samples into two groups, and put them away from the separating hyperplane as far as possible. In SVM, the separating margin of two classes is defined as the shortest distance from the vectors in either of the two classes to the separating hyperplane [1]. Figure 1 illustrates the definition. Let L

[1] The margin defined here refers to the "hard margin" when separation is perfect. Margin can also be defined when perfect separation is impossible or undesirable [21]. It is then called the "soft margin."

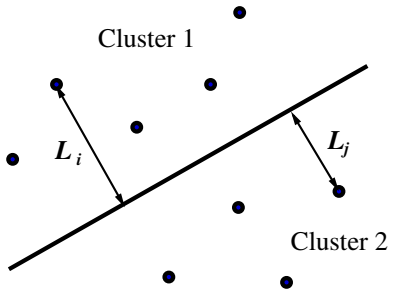


Figure 1: The separation distance definition.

denote the geometric distance from an example to the hyperplane. Assume indices i and j run through examples from cluster 1 and cluster 2, respectively. The margin can be written as:

$$\text{Margin} = \min_i L_i + \min_j L_j \ .$$

The use of margin was derived from VC theory for supervised learning tasks. Unfortunately, maximizing the margin on all examples with unknown labels in unsupervised learning is an NP-complete problem. The evaluation of margin varies for different possible label assignments. Hence we propose a different separation distance, which we call the sum-of-squared (SS) separation distance and is defined:

$$\text{SS} = \sum_i L_i^2 + \sum_j L_j^2 = \sum_t L_t^2$$

where t runs through all training examples. We will show that the sum-of-squared separation distance (SS) is much easier to evaluate and maximize, and maximizing SS produces neat properties similar to those obtained for spectral clustering.

For a nontrivial training set, there usually exist cluster assignments under which the samples are not linearly separable in the input space. Therefore it is possible that some bipartitions, which may correspond to good clustering, could not be realized by hyperplanes in the input space. This is certainly undesirable since the model itself may have already eliminated potentially good bipartitions. To avoid such an intrinsic deficiency, nonlinear feature mapping is adopted so that a hyperplane constructed in the feature space corresponds to a nonlinear model in the original input space, such as the kernel mapping used in SVMs. Notice that when perfect separation is impossible in supervised learning, a "soft margin" can be defined in terms of the given class labels. With unknown labels in unsupervised learning, no way exists to define a soft margin. Bearing this in mind, we propose a simple feature mapping, based on the given data set and the distance (or dissimilarity)

measure, which maps input vectors to a new feature space where samples are always linearly separable.

The proposed algorithm, Maximal Separation Clustering (MSC), possesses several properties:

- MSC only requires the knowledge of a *metric distance* function measuring the dissimilarity between samples with an assumption that prior knowledge can be properly incorporated in the chosen metric distance. For applications where a similarity function is specified, a transformation is needed to map the similarity into a metric distance. The transformation will be discussed in Section 2.1. Note that a metric distance function is a more general concept than a positive-definite (PD) kernel function since as long as an inner product (the PD kernel) is given, the corresponding metric distance can be induced, but not vice versa.

- Although MSC is a discriminative approach, the algorithm can predict labels for out-of-sample examples because it learns an optimal separating hyperplane that transforms to a nonlinear decision boundary in the input space.

- MSC has interesting connections with a class of graph partitioning methods. Specifically, the optimal bipartition generated by the algorithm can be viewed as an approximation of an "optimal" graph partition under spectral relaxation.

- MSC can be formulated as an eigenvalue decomposition problem similar to spectral clustering under certain circumstances. Hence it does not require any integer programming solvers.

1.3 Outline of the Paper. The remainder of the paper is organized as follows: In Section 2, we first introduce a feature mapping such that any bipartition of a given training set can be achieved by a hyperplane in the new feature space. Clustering is then formulated as maximizing the sum-of-squared separation distance in the feature space. In Section 3, we propose a class of graph partitioning problems and prove that under certain conditions the optimal bipartition given by a hyperplane is the solution of a graph partitioning problem with spectral relaxation. Section 4 describes the experiments we have performed and provides the results. We conclude in Section 5, together with a discussion of future work.

2 Maximal Separation Clustering.

We focus on problems of clustering examples into two clusters. For applications requiring more than two clusters, the proposed method can be recursively applied, but only local optimality is ensured.

2.1 Distance Feature Mapping. Given a set of n distinct training samples $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{X} : i = 1, ..., n\}$ and a metric distance function [2] $m : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$, we consider linear separation functions:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{d}(\mathbf{x}) \tag{2.1}$$

where $\mathbf{d} : \mathbb{X} \to \mathbb{R}^m$ realizes a set of features induced by the metric distance m and $\mathbf{w}$ are the model parameters. An example $\mathbf{x}_i$ is assigned into one cluster if $f(\mathbf{x}_i) \geq 0$; the other cluster if $f(\mathbf{x}_i) < 0$.

Note that the metric distance m can be any suitable metric and does not have to be the Euclidean distance in $\mathbb{X}$. Since the distance function m provides the only prior knowledge about clustering, $m(\mathbf{x}_i, \mathbf{x}_j)$ should be fully explored in the feature mapping $\mathbf{d}$. We propose to map any input vector $\mathbf{x}$ to an n dimensional space where the j^{th} dimension represents $d_j(\mathbf{x}) = m(\mathbf{x}, \mathbf{x}_j)$.

The mapping $\mathbf{d}_{\mathcal{X}} : \mathbb{X} \to \mathbb{R}^n$ can be written as:

$$\mathbf{d}_{\mathcal{X}}(\mathbf{x}) = \begin{bmatrix} m(\mathbf{x}, \mathbf{x}_1) \\ m(\mathbf{x}, \mathbf{x}_2) \\ \vdots \\ m(\mathbf{x}, \mathbf{x}_n) \end{bmatrix}.$$

Clearly, $\mathbf{d}_{\mathcal{X}}$ is data dependent. Now we validate if the training samples are linearly separable for all possible label assignments in the new feature space generated by $d_{\mathcal{X}}$. Let us first stack n cluster assignments made by (2.1) into a matrix equation:

$$\begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix} = \mathbf{D}\mathbf{w} \tag{2.2}$$

where

$$\mathbf{D} = \begin{bmatrix} m(\mathbf{x}_1, \mathbf{x}_1) & m(\mathbf{x}_1, \mathbf{x}_2) & \cdots & m(\mathbf{x}_1, \mathbf{x}_n) \\ m(\mathbf{x}_2, \mathbf{x}_1) & m(\mathbf{x}_2, \mathbf{x}_2) & \cdots & m(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ m(\mathbf{x}_n, \mathbf{x}_1) & m(\mathbf{x}_n, \mathbf{x}_2) & \cdots & m(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

is called the *distance matrix* or the *dissimilarity matrix*. $\mathbf{D}$ is symmetric and nonnegative (all elements are greater than or equal to zero.) For Euclidean distance $m(\cdot, \cdot)$, $\mathbf{D}$ is always invertible with one positive eigenvalue and $n - 1$ negative eigenvalues [18]. It has been

[2] A metric distance function, $m(\cdot, \cdot)$, is a nonnegative function satisfying: 1) $m(\mathbf{x}, \mathbf{y}) \geq 0$, and $m(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$; 2) $m(\mathbf{x}, \mathbf{y}) = m(\mathbf{y}, \mathbf{x})$; and 3) $m(\mathbf{x}, \mathbf{y}) + m(\mathbf{y}, \mathbf{z}) \geq m(\mathbf{x}, \mathbf{z})$.

proved that this property holds for an arbitrary metric distance $m(\cdot,\cdot)$ as well [1]. Consequently, for an arbitrary label assignment $[f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^T$, equation (2.2) has a unique solution of $\mathbf{w}$. This implies that $\mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)$'s are always linearly separable.

In some clustering tasks, a distance metric is not given directly. Instead, a similarity measure is specified. A commonly used class of similarity measures is defined by PD kernels [20, 17, 19]. A PD kernel, $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$, computes the inner product in a kernel-induced feature space $\mathcal{H}$ via a mapping $\Phi : \mathbb{X} \to \mathcal{H}$, i.e., $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$. Correspondingly,

$$\begin{aligned} m(\mathbf{x}, \mathbf{x}') &= \sqrt{[\Phi(\mathbf{x}) - \Phi(\mathbf{x}')]^T [\Phi(\mathbf{x}) - \Phi(\mathbf{x}')]} \\ &= \sqrt{K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2K(\mathbf{x}, \mathbf{x}')} \end{aligned} \tag{2.3}$$

defines a metric distance in $\mathcal{H}$. This suggests that $\mathbf{d}_{\mathcal{X}}$ and $\mathbf{D}$ can also be constructed from a PD kernel. Therefore, for the rest of the paper, we assume that either a metric distance or a similarity measure (described by a PD kernel) is given.

2.2 Computing the Optimal Partition. In the space where $\mathbf{d}_{\mathcal{X}}(\mathbf{x})$ resides, the decision boundary of the separation function $f(\mathbf{x}) = \mathbf{w}^T \mathbf{d}_{\mathcal{X}}(\mathbf{x})$ is a hyperplane defined by $\mathbf{w}^T \mathbf{d}_{\mathcal{X}} = 0$. The geometric distance from a point $\mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)$ to the hyperplane is $L_i = \frac{|\mathbf{w}^T \mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)|}{\|\mathbf{w}\|}$ where $\|\cdot\|$ denotes the 2-norm of a vector unless otherwise stated. If we assume the separation function is normalized such that $\|\mathbf{w}\| = 1$, then the above geometric distance can be simply calculated as $L_i = |\mathbf{w}^T \mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)|$. The term inside $|\cdot|$ gives the signed distance. The hence proposed *sum-of-squared separation distance* is calculated as follows by taking all samples in $\mathcal{X}$ into consideration:

$$\sum_{i=1}^{n} \left[\mathbf{w}^T \mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)\right]^2 = \mathbf{w}^T \mathbf{D}^2 \mathbf{w}. \tag{2.4}$$

The optimal clustering corresponds to a unit length $\mathbf{w}$ that separates two clusters with maximal value of (2.4).

It is not difficult to see that the unit length eigenvector $\mathbf{p}$ corresponding to the largest eigenvalue λ_1 of $\mathbf{D}^2$, maximizes (2.4). Unfortunately, the resulting separation function, $f(\mathbf{x}) = \mathbf{d}_{\mathcal{X}}(\mathbf{x})^T \mathbf{p}$, assigns all training samples in $\mathcal{X}$ to one cluster because $\mathbf{Dp} = \lambda_1 \mathbf{p}$ has components either all positive or all negative. This is due to the positivity of $\mathbf{D}^2$. The distance matrix $\mathbf{D}$ is a symmetric and non-negative matrix, so the eigenvalue decomposition exists. The corresponding $\mathbf{D}^2$ is thus a positive matrix [3] which has the same eigenvectors as those of $\mathbf{D}$ and eigenvalues equal to the square of the eigenvalues of $\mathbf{D}$. A positive matrix also has the following properties(Ch.8.2, [15]):

- The largest eigenvalue λ_1 is positive with algebraic multiplicity 1;
- The corresponding eigenvector satisfies either $\mathbf{p} > 0$ (called the Perron vector if $\mathbf{p}^T \mathbf{e} = 1$ where $\mathbf{e}$ is a vector of 1's) or $\mathbf{p} < 0$ [4]. All other eigenvectors have elements with mixed signs.

To avoid trivial clustering like $\mathbf{p}$, the following constraint is imposed:

$$\boldsymbol{\alpha}^T \mathbf{Dw} = 0 .$$

Here $\boldsymbol{\alpha} > 0$ is a user-specified normalized weight vector satisfying $\boldsymbol{\alpha}^T \mathbf{e} = 1$. Since $\mathbf{Dw}$ contains the signed distances (cluster membership is indicated by the sign), the above constraint enforces that the weighted summation of signed distances is zero. In other words, neither $\mathbf{Dw} > 0$ nor $\mathbf{Dw} < 0$ is allowed. The clustering problem is then formulated as below.

DEFINITION 2.1. (Maximal Separation Clustering) *Given a distance matrix $\mathbf{D}$ and a normalized weight vector $\boldsymbol{\alpha} > 0$ ($\boldsymbol{\alpha}^T \mathbf{e} = 1$), an optimal clustering is given by a separation function $f(\mathbf{x}) = \mathbf{w}^{*T} \mathbf{d}_{\mathcal{X}}(\mathbf{x})$ where*

$$\mathbf{w}^* = \underset{\substack{\mathbf{w},\, \|\mathbf{w}\|=1 \\ \boldsymbol{\alpha}^T \mathbf{Dw}=0}}{\operatorname{argmax}} \quad \mathbf{w}^T \mathbf{D}^2 \mathbf{w} .$$

Since the weight vector should be chosen by a user beforehand, a few insights about different choices of $\boldsymbol{\alpha}$ will be helpful. Three interesting choices are discussed:

- $\boldsymbol{\alpha}_1 \propto \mathbf{e}$
  It assumes that all samples are equally important. Therefore, the signed distances should be weighted uniformly, i.e., $\boldsymbol{\alpha}_1 = \frac{\mathbf{e}}{\mathbf{e}^T \mathbf{e}}$.
- $\boldsymbol{\alpha}_2 \propto \mathbf{De}$
  A sample is weighted according to its overall dissimilarity to the rest of the samples. The weight for the signed distance $\mathbf{w}^T \mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)$ is proportional to $\mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)^T \mathbf{e} = \sum_{k=1}^{n} m(\mathbf{x}_i, \mathbf{x}_k)$, which is the summation of the distances between $\mathbf{x}_i$ and all samples in $\mathcal{X}$. After normalization, we get $\boldsymbol{\alpha}_2 = \frac{\mathbf{De}}{\mathbf{e}^T \mathbf{De}}$.
- $\boldsymbol{\alpha}_3 \propto \mathbf{D}\boldsymbol{\alpha}_3$
  This scheme carries a similar flavor as that of $\boldsymbol{\alpha}_2$. Here the weight for the signed distance $\mathbf{w}^T \mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)$ is proportional to $\mathbf{d}_{\mathcal{X}}(\mathbf{x}_i)^T \boldsymbol{\alpha}_3$, which is a *weighted*

[3] A matrix is positive if all it elements are positive [15].

[4] For a matrix or a vector $\mathbf{A}$, we write $\mathbf{A} > 0$ if all it elements are positive. This notation can be generalized to $\geq$, $<$, and $\leq$.

summation of the distances between $\mathbf{x}_i$ and all samples in $\mathcal{X}$. In this case, $\boldsymbol{\alpha}_3$ has to be an eigenvector of $\mathbf{D}$, i.e., $\lambda\boldsymbol{\alpha}_3 = \mathbf{D}\boldsymbol{\alpha}_3$. Hence the Perron vector $\mathbf{p} > 0$ (satisfying $\mathbf{p}^T\mathbf{e} = 1$) of $\mathbf{D}$ is the only choice.

Empirical comparisons of the above weighting strategies are provided in Section 4.

THEOREM 2.1. *The optimal solution* $\mathbf{w}^*$ *of the MSC problem described in Definition 2.1 is*

$$\mathbf{w}^* = \mathbf{U}_{\boldsymbol{\alpha}}\mathbf{v}, \tag{2.5}$$

where $\mathbf{U}_{\boldsymbol{\alpha}} \in \mathbb{R}^{n\times(n-1)}$ *is a matrix whose columns form an orthonormal basis of the null space of* $\boldsymbol{\alpha}^T\mathbf{D}$*, and* $\mathbf{v}$ *is a unit length eigenvector corresponding to the largest eigenvalue of* $\mathbf{U}_{\boldsymbol{\alpha}}^T\mathbf{D}^2\mathbf{U}_{\boldsymbol{\alpha}}$.

Proof. The feasible region of the optimization problem in Definition 2.5 is

$$\mathcal{F} = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\| = 1, \boldsymbol{\alpha}^T\mathbf{D}\mathbf{w} = 0\} .$$

It is not difficult to check that $\mathcal{F}$ can be equivalently written as

$$\mathcal{F} = \{\mathbf{w} = \mathbf{U}_{\boldsymbol{\alpha}}\mathbf{z} : \mathbf{z} \in \mathbb{R}^{n-1}, \|\mathbf{z}\| = 1\} .$$

So the original optimization problem becomes

$$\begin{aligned} \mathbf{z}^* &= \arg\max_{\|\mathbf{z}\|=1} \mathbf{z}^T\mathbf{U}_{\boldsymbol{\alpha}}^T\mathbf{D}^2\mathbf{U}_{\boldsymbol{\alpha}}\mathbf{z} \\ \mathbf{w}^* &= \mathbf{U}_{\boldsymbol{\alpha}}\mathbf{z}^* . \end{aligned}$$

Then (2.5) follows from the fact that $\mathbf{z}^* = \mathbf{v}$. □

2.3 An Algorithmic View.

ALGORITHM 2.1. (Maximal Separation Clustering)

Input: Set $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, ..., n\}$, a metric distance function $m(\cdot,\cdot)$, and weight $\boldsymbol{\alpha}$.
Output: Separation function $f(\mathbf{x}) = \mathbf{w}^{*T}\mathbf{d}_{\mathcal{X}}(\mathbf{x})$ where the sign of $f(\mathbf{x})$ defines the cluster assignment of $\mathbf{x}$.
Method:
1 Compute the $n \times n$ distance matrix $\mathbf{D}$
2 Find an orthonormal basis of the null space of $\boldsymbol{\alpha}^T\mathbf{D}$ and stack them as columns of $\mathbf{U}_{\boldsymbol{\alpha}}$
3 Compute $\mathbf{z}^*$, an eigenvector corresponding to the largest eigenvalue of $\mathbf{U}_{\boldsymbol{\alpha}}^T\mathbf{D}^2\mathbf{U}_{\boldsymbol{\alpha}}$
4 $\mathbf{w}^* \leftarrow \mathbf{U}_{\boldsymbol{\alpha}}\frac{\mathbf{z}^*}{\|\mathbf{z}^*\|}$
5 OUTPUT $f(\mathbf{x}) = \mathbf{w}^{*T}\mathbf{d}_{\mathcal{X}}(\mathbf{x})$

Note that the input metric distance can be constructed from a PD kernel based on (2.3). The orthonormal basis of the null space of $\boldsymbol{\alpha}^T\mathbf{D}$ is computed using the Symmetric QR Algorithm (Ch.8.3, [10]). Therefore the computational cost of Step 2 is $O(n^3)$. The eigenvalue problem in Step 3 is solved by a Lanczos method (Ch.9, [10]). The running time of a Lanczos method is $O(kn^2)$ where k is the maximum number of matrix-vector computations required. Usually, k is much smaller than n. So the overall running time is $O(n^3 + kn^2)$. It is worth mentioning a special case where $\boldsymbol{\alpha}$ is given as the Perron vector of $\mathbf{D}$, i.e., $\boldsymbol{\alpha} = \boldsymbol{\alpha}_3$. Then one can show that $\mathbf{w}^*$ is a unit length eigenvector corresponding to the second largest eigenvalue of $\mathbf{D}^2$ (a proof will be given in Section 3.3). In this case, the computational cost becomes $O(kn^2)$ because there is no need to compute the null space of $\boldsymbol{\alpha}^T\mathbf{D}$.

3 Connections with Graph Partitioning.

This section provides a graph-theoretic view of the MSC method. We first propose a new graph-theoretic criterion for measuring the goodness of graph bipartition. A graph partitioning problem divides vertices into groups so that the between-group dissimilarity is high, and/or within-group dissimilarity is low. The novel criterion measures the disparity between the between-group dissimilarity and the within-group dissimilarity. Thus we name the resulting bipartition – the *disparity cut*. The maximization of the criterion can be formulated as an eigenvalue problem. Then connections between Algorithm 2.1 and the disparity cut are established.

3.1 Disparity Cut Criterion.

Given a set of samples and a dissimilarity measure, one can construct a weighted undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V} = \{1, 2, \ldots, n\}$ is the vertex set, $\mathbf{E} = \{(i,j) : i,j \in \mathbf{V}\}$ is the set of edges. The vertices represent the samples. An edge is formed between every pair of vertices. The weight d_{ij} of an edge (i,j) indicates the similarity or dissimilarity between vertices i and j. The weights can be organized into an *affinity matrix*, $\mathbf{D}$. To be consistent with notations in Section 2, we assume that d_{ij} describes the dissimilarity. Note that a similarity measure can be converted to a dissimilarity measure via (2.3).

Let sets $\mathbf{A}, \mathbf{B} \subset \mathbf{V}$. In graph theory, a *cut* is defined as:

$$cut(\mathbf{A}, \mathbf{B}) = \sum_{i\in\mathbf{A}, j\in\mathbf{B}} d_{ij} .$$

Finding a bipartition of the graph that minimizes this cut value is known as the minimum cut problem [5]. Efficient algorithms exist for solving this problem. How-

[5] In the minimum cut problem, the affinity matrix is defined by vertex similarities. If affinity matrix captures vertex dissimilarities, as in this paper, the problem should be named the maximum cut problem instead.

ever the minimum cut criterion tends to group small sets of isolated nodes in the graph because the cut defined above does not contain any within-group information [20]. Many modified graph partition criteria have been proposed to produce more balanced partitions [11, 20, 8, 17]. Next, we introduce a new criterion, *disparity cut*.

Let each vertex i be associated with a positive weight, β_i. Without loss of generality we assume $||\boldsymbol{\beta}|| = 1$. For $\mathbf{A} \subset \mathbf{V}$, we define a *weighted cardinality* of $\mathbf{A}$, $|\mathbf{A}|_{\boldsymbol{\beta}}$, to be

$$|\mathbf{A}|_{\boldsymbol{\beta}} = \sum_{i \in \mathbf{A}} \beta_i^2 \ . \tag{3.6}$$

If weights are uniform (i.e., $\boldsymbol{\beta} = \mathbf{e}$), then (3.6) is identical to the standard definition of set cardinality. By taking vertex weights, $\boldsymbol{\beta}$, into consideration, we define a *weighted cut* as

$$cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{B}) = \sum_{i \in \mathbf{A}, j \in \mathbf{B}} \beta_i \beta_j d_{ij} \ .$$

It is not difficult to see that $cut_{\mathbf{e}}(\mathbf{A}, \mathbf{B}) = cut(\mathbf{A}, \mathbf{B})$. Let $\mathbf{A}$ and $\mathbf{B}$ form a bipartition of $\mathbf{V}$ (i.e., $\mathbf{A} \cap \mathbf{B} = \emptyset$, $\mathbf{A} \cup \mathbf{B} = \mathbf{V}$), the disparity cut, $Dcut(\mathbf{A}, \mathbf{B})$, is then defined as

$$\begin{aligned} Dcut(\mathbf{A}, \mathbf{B}) &= 2\, cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{B}) - \\ &\frac{|\mathbf{B}|_{\boldsymbol{\beta}}}{|\mathbf{A}|_{\boldsymbol{\beta}}} cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{A}) - \frac{|\mathbf{A}|_{\boldsymbol{\beta}}}{|\mathbf{B}|_{\boldsymbol{\beta}}} cut_{\boldsymbol{\beta}}(\mathbf{B}, \mathbf{B}) \ . \end{aligned} \tag{3.7}$$

- $cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{B})$ measures the dissimilarity between vertex sets $\mathbf{A}$ and $\mathbf{B}$;
- $cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{A})$ and $cut_{\boldsymbol{\beta}}(\mathbf{B}, \mathbf{B})$ capture the vertex dissimilarities within $\mathbf{A}$ and within $\mathbf{B}$, respectively;
- $\frac{|\mathbf{B}|_{\boldsymbol{\beta}}}{|\mathbf{A}|_{\boldsymbol{\beta}}}$ and $\frac{|\mathbf{A}|_{\boldsymbol{\beta}}}{|\mathbf{B}|_{\boldsymbol{\beta}}}$ indicate the relative size of the two groups. An "unbalanced" bipartition will make one of the ratios a large number.

A "good" bipartition should generate two "balanced" groups that have high between-group dissimilarity and low within-group dissimilarity. This goal is achieved in this article by maximizing the disparity cut criterion. Finding the maximum disparity cut is NP-complete. Nevertheless, it is possible to find an approximation via spectral relaxation. This is described as below.

3.2 Spectral Relaxation. Given a weighted undirected graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ with affinity matrix $\mathbf{D}$ and vertex weights $\boldsymbol{\beta}$, a bipartition of $\mathbf{V}$ into $\mathbf{A}$ and $\mathbf{B}$ can be defined by a *partition vector* $\mathbf{q} \in \{1, -1\}^n$ with elements

$$q_i = \begin{cases} 1, & i \in \mathbf{A} \ , \\ -1, & i \in \mathbf{B} \ . \end{cases}$$

Let Γ denote the diagonal matrix formed by the vertex weights $\boldsymbol{\beta}$, i.e., $\Gamma_{ii} = \beta_i$. Then we have the following identities:

$$\begin{aligned} cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{A}) &= \frac{1}{4} (\mathbf{e} + \mathbf{q})^T \Gamma \mathbf{D} \Gamma (\mathbf{e} + \mathbf{q}) \ , \\ cut_{\boldsymbol{\beta}}(\mathbf{B}, \mathbf{B}) &= \frac{1}{4} (\mathbf{e} - \mathbf{q})^T \Gamma \mathbf{D} \Gamma (\mathbf{e} - \mathbf{q}) \ , \\ cut_{\boldsymbol{\beta}}(\mathbf{A}, \mathbf{B}) &= \frac{1}{4} (\mathbf{e} + \mathbf{q})^T \Gamma \mathbf{D} \Gamma (\mathbf{e} - \mathbf{q}) \ . \end{aligned}$$

If we define the ratio of weighted cardinality, r, as

$$r = \frac{|\mathbf{A}|_{\boldsymbol{\beta}}}{|\mathbf{B}|_{\boldsymbol{\beta}}} \ ,$$

then (3.7) is equivalent to

$$\begin{aligned} Dcut(\mathbf{A}, \mathbf{B}) &= \frac{(\mathbf{e} + \mathbf{q})^T \Gamma \mathbf{D} \Gamma (\mathbf{e} - \mathbf{q})}{2} - \\ &\frac{(\mathbf{e} + \mathbf{q})^T \Gamma \mathbf{D} \Gamma (\mathbf{e} + \mathbf{q})}{4r} - \frac{r (\mathbf{e} - \mathbf{q})^T \Gamma \mathbf{D} \Gamma (\mathbf{e} - \mathbf{q})}{4} \\ &= \frac{[(\mathbf{e} + \mathbf{q}) - r (\mathbf{e} - \mathbf{q})]^T \Gamma \mathbf{D} \Gamma [r (\mathbf{e} - \mathbf{q}) - (\mathbf{e} + \mathbf{q})]}{4r} \ . \end{aligned}$$

Let

$$\mathbf{y} = \frac{(\mathbf{e} + \mathbf{q}) - r (\mathbf{e} - \mathbf{q})}{2},$$

and $\mathbf{y}$ can be viewed as a generalized partition vector with elements

$$y_i = \begin{cases} 1, & i \in \mathbf{A} \ , \\ -r, & i \in \mathbf{B} \ . \end{cases}$$

Then we have

$$Dcut(\mathbf{A}, \mathbf{B}) = -\frac{\mathbf{y}^T \Gamma \mathbf{D} \Gamma \mathbf{y}}{r} \ .$$

In addition, it is straightforward to derive that

$$\begin{aligned} \boldsymbol{\beta}^T \Gamma \mathbf{y} &= 0 \ , \\ \mathbf{y}^T \Gamma^2 \mathbf{y} &= r \|\boldsymbol{\beta}\|^2 = r \ . \end{aligned}$$

Therefore, we can write $Dcut$ in terms of the generalized partition vector $\mathbf{y}$ as

$$Dcut(\mathbf{A}, \mathbf{B}) = -\frac{\mathbf{y}^T \Gamma \mathbf{D} \Gamma \mathbf{y}}{\mathbf{y}^T \Gamma^2 \mathbf{y}} \ .$$

Finding the maximum disparity cut can then be stated as the following discrete optimization problem.

DEFINITION 3.1. (Maximal Disparity Cut) *Given a weighted undirected graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ with affinity matrix $\mathbf{D}$ and unit length vertex weights $\boldsymbol{\beta}$, the maximum disparity cut is defined by the generalized partition vector*

$$\mathbf{y}^* = \underset{\substack{\mathbf{y}, \boldsymbol{\beta}^T \Gamma \mathbf{y} = 0, \\ \mathbf{y} \in \{1, -r\}^n}}{\operatorname{argmax}} \ -\frac{\mathbf{y}^T \Gamma \mathbf{D} \Gamma \mathbf{y}}{\mathbf{y}^T \Gamma^2 \mathbf{y}} \ .$$

Even though the above discrete optimization problem is still NP-complete, it is possible to find an approximation by relaxing the condition that y is either 1 or $-r$. When we only require y to be continuous and set $\mathbf{z} = \Gamma\mathbf{y}$, the following optimization problem is obtained.

DEFINITION 3.2. (Spectral Relaxation for Maximal Disparity Cut) *Given a weighted undirected graph* $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ *with affinity matrix* $\mathbf{D}$ *and unit length vertex weights* $\boldsymbol{\beta}$*, a continuous approximation of the optimal generalized partition vector is*

$$\mathbf{z}^* = \underset{\substack{\mathbf{z}, \boldsymbol{\beta}^T\mathbf{z}=0, \\ \|\mathbf{z}\|=1}}{\text{argmax}} \quad -\mathbf{z}^T\mathbf{D}\mathbf{z} \ . \tag{3.8}$$

The corresponding bipartition is $\mathbf{A} = \{i \in \mathbf{V} : z_i^* \geq 0\}$ *and* $\mathbf{B} = \{i \in \mathbf{V} : z_i^* < 0\}$.

Since Γ is a diagonal matrix with positive diagonal entries, $\mathbf{z}^*$ and $\Gamma^{-1}\mathbf{z}^*$ generate identical bipartitions because their corresponding elements have identical signs. This optimization problem is solved as below.

THEOREM 3.1. *The optimal solution* $\mathbf{z}^*$ *of the problem described in Definition 3.2 is*

$$\mathbf{z}^* = \mathbf{U}_{\boldsymbol{\beta}}\mathbf{v}, \tag{3.9}$$

where $\mathbf{U}_{\boldsymbol{\beta}} \in \mathbb{R}^{n\times(n-1)}$ *is a matrix whose columns form an orthonormal basis of the null space of* $\boldsymbol{\beta}^T$*, and* $\mathbf{v}$ *is a unit length eigenvector corresponding to the smallest eigenvalue of* $\mathbf{U}_{\boldsymbol{\beta}}^T\mathbf{D}\mathbf{U}_{\boldsymbol{\beta}}$.

The proof is similar to that of Theorem 2.1, therefore, is omitted.

3.3 Connections between MSC and Dcut. As shown in Theorem 2.1, the maximal separation bipartition relies on the weight vector $\boldsymbol{\alpha}$. Similarly, the spectral relaxed maximal disparity cut depends on the choice of the vertex weights $\boldsymbol{\beta}$ as shown in Theorem 3.1. We prove that under certain conditions they generate identical bipartitions.

LEMMA 3.1. *Let the affinity matrix* $\mathbf{D}$ *in Definition 3.2 be the distance matrix in Definition 2.1 and* $\mathbf{p}$ *be the Perron vector of* $\mathbf{D}$*. If the smallest eigenvalue of* $\mathbf{D}$ *is algebraically simple and we choose the weight vectors* $\boldsymbol{\alpha} = \mathbf{p}$ *and* $\boldsymbol{\beta} = \frac{\mathbf{p}}{\|\mathbf{p}\|}$*, then the maximal separation bipartition in Definition 2.1 is identical to that generated by the spectral relaxed maximal disparity cut in Definition 3.2.*

Proof. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be eigenvalues of $\mathbf{D}$. Since $\mathbf{D}$ is a nonnegative distance matrix, from the properties of (metric) distance matrices [18, 1] and nonnegative matrices (Ch.8.2,[15]), we have that

$$\lambda_1 > 0 > \lambda_2 \geq \ldots \geq \lambda_n \ ,$$

and $\mathbf{p}$ is an eigenvector corresponding to λ_1. Moreover, since $\sum_{i=1}^n \lambda_i = \text{Trace}(\mathbf{D}) = 0$, we have

$$\lambda_1 > |\lambda_i|, \ i = 2, \ldots, n.$$

Since λ_n is algebraically simple, $-\lambda_n$ is the largest eigenvalue of $-\mathbf{D}$ and λ_n^2 is the second largest eigenvalue of $\mathbf{D}^2$.

Let $\mathbf{u}_2, \ldots, \mathbf{u}_n$ be unit length eigenvectors associated with $\lambda_2, \ldots, \lambda_n$, respectively. Clearly, $\mathbf{u}_2, \ldots, \mathbf{u}_n$ form an orthonormal basis of the null space of $\boldsymbol{\beta}^T$. Therefore

$$\mathbf{z}^* = \mathbf{u}_n \ .$$

Note that $\mathbf{u}_2, \ldots, \mathbf{u}_n$ also form an orthonormal basis of the null space of $\boldsymbol{\alpha}^T\mathbf{D}$. Since

$$\lambda_1^2 > \lambda_n^2 > \lambda_{n-1}^2 \geq \ldots \geq \lambda_2^2$$

are eigenvalues of $\mathbf{D}^2$ with corresponding eigenvectors $\mathbf{p}, \mathbf{u}_n, \mathbf{u}_{n-1}, \ldots, \mathbf{u}_2$, respectively, we get

$$\mathbf{w}^* = \mathbf{u}_n \ .$$

The proof then follows from $\mathbf{D}\mathbf{w}^* = \lambda_1\mathbf{z}^*$. □

Therefore, the maximal separation bipartition in Definition 2.1 is equivalent to the spectral relaxed maximal disparity cut when the affinity matrix is defined by the distance matrix and the Perron vector of the distance matrix is used as the weight vectors. Given an arbitrary positive weight vector $\boldsymbol{\alpha}$ for maximal separation bipartition, can we find positive vertex weights $\boldsymbol{\beta}$ such that $\mathbf{z}^* = \frac{\mathbf{D}\mathbf{w}^*}{\|\mathbf{D}\mathbf{w}^*\|}$, i.e., two bipartitions are identical? A necessary condition is presented as below.

LEMMA 3.2. *Let the affinity matrix* $\mathbf{D}$ *in Definition 3.2 be the distance matrix in Definition 2.1 and* $\mathbf{w}^*$ *be calculated by (2.5). Let* $\mathbf{z}^*$ *be computed by (3.9) and* $\mathbf{z}^*$ *is not an eigenvector of* $\mathbf{D}$*. If* $\mathbf{z}^* = \frac{\mathbf{D}\mathbf{w}^*}{\|\mathbf{D}\mathbf{w}^*\|}$*, then*

$$\boldsymbol{\beta} = \frac{\left(\mathbf{I} - \frac{\mathbf{D}\mathbf{w}^*\mathbf{w}^{*T}\mathbf{D}}{\|\mathbf{D}\mathbf{w}^*\|^2}\right)\mathbf{D}^2\mathbf{w}^*}{\left\|\left(\mathbf{I} - \frac{\mathbf{D}\mathbf{w}^*\mathbf{w}^{*T}\mathbf{D}}{\|\mathbf{D}\mathbf{w}^*\|^2}\right)\mathbf{D}^2\mathbf{w}^*\right\|} \tag{3.10}$$

where $\mathbf{I}$ *is an identity matrix.*

Proof. For the constrained optimization problem (3.8), the Lagrange function is

$$\mathcal{L}(\mathbf{z}, \sigma_1, \sigma_2) = -\mathbf{z}^T\mathbf{D}\mathbf{z} - \sigma_1\left(\mathbf{z}^T\mathbf{z} - 1\right) - \sigma_2\boldsymbol{\beta}^T\mathbf{z}$$

where σ_1 and σ_2 are Lagrange multipliers. Setting the respective derivatives to zero yields

$$2\mathbf{D}\mathbf{z} + 2\sigma_1\mathbf{z} + \sigma_2\boldsymbol{\beta} = \mathbf{0}\,, \tag{3.11}$$

$$\mathbf{z}^T\mathbf{z} = 1\,, \tag{3.12}$$

$$\boldsymbol{\beta}^T\mathbf{z} = 0\,. \tag{3.13}$$

Left multiplying both sides of (3.11) with $\mathbf{z}^T$ and applying identities (3.12) and (3.13), we obtain

$$\sigma_1 = -\mathbf{z}^T\mathbf{D}\mathbf{z}\,. \tag{3.14}$$

Similarly, we get

$$\sigma_2 = -\frac{2\boldsymbol{\beta}^T\mathbf{D}\mathbf{z}}{\|\boldsymbol{\beta}\|^2} \tag{3.15}$$

by multiplying both side of (3.11) with $\boldsymbol{\beta}^T$ and applying identities (3.13) and (3.14). Substituting (3.14) and (3.15) into (3.12) gives

$$\left(\mathbf{I} - \mathbf{z}\mathbf{z}^T\right)\mathbf{D}\mathbf{z} = \frac{\boldsymbol{\beta}^T\mathbf{D}\mathbf{z}}{\|\boldsymbol{\beta}\|^2}\boldsymbol{\beta}\,.$$

Since $\mathbf{z}^*$ is not an eigenvector of $\mathbf{D}$, we have $\boldsymbol{\beta}^T\mathbf{D}\mathbf{z}^* \neq 0$. Therefore

$$\boldsymbol{\beta} = \frac{\|\boldsymbol{\beta}\|^2}{\boldsymbol{\beta}^T\mathbf{D}\mathbf{z}^*}\left(\mathbf{I} - \mathbf{z}^*\mathbf{z}^{*T}\right)\mathbf{D}\mathbf{z}\,. \tag{3.16}$$

By substituting $\mathbf{z}^* = \frac{\mathbf{D}\mathbf{w}^*}{\|\mathbf{D}\mathbf{w}^*\|}$ into (3.16) and enforcing the unit length constraint on $\boldsymbol{\beta}$, we get (3.10). □

4 Experiments.

Based on an artificial data set, the USPS data set, and a COREL image data set, we evaluate the performance of Algorithm 2.1. Comparisons with the normalized cut (Ncut) method in [20] are provided in some of our experimental results.

4.1 Artificial Data Set. The artificial data set consists of 200 samples belonging to two classes. Each class contains 100 samples. The samples in the first class are distributed as a two dimensional Gaussian with zero means and identity covariance matrix. The samples in the second class are generated by the following stochastic process:

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

where $[Y_1, Y_2]^T$ is distributed as a two dimensional Gaussian with means $[3, 3]^T$ and identity covariance matrix, and θ is a random variable with uniform distribution over the interval $[0, 2\pi]$. Figure 2 shows 200 randomly generated samples from the two classes.

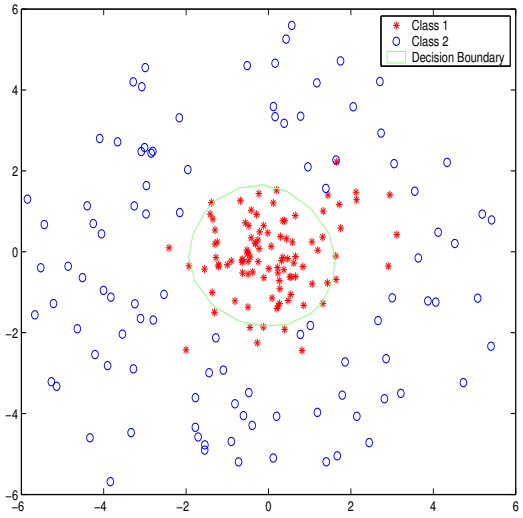


Figure 2: An artificial data set. Samples in Class 1 and Class 2 are denoted by stars and circles, respectively. The curve in the middle is the decision boundary of maximal separation clustering.

We compare the performance of Algorithm 2.1 with that of Ncut. The affinity matrix in Ncut is defined by Gaussian kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}}$. Since Gaussian kernel computes the inner produce in a kernel-induced feature space $\mathcal{H}$, a metric distance in $\mathcal{H}$ can be defined according to (2.3) and is used in Algorithm 2.1 to calculate the distance matrix $\mathbf{D}$. Because we know the "true" class label for each sample, the classification error is one way to capture the goodness of clustering. Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be a bipartition of the data set $\mathbf{C}$. The classification error is defined as

$$\mathrm{Err}(\mathbf{C}_1, \mathbf{C}_2) = \frac{1}{|\mathbf{C}|}\min\left(|\mathbf{C}_{1,1}| + |\mathbf{C}_{2,2}|, |\mathbf{C}_{1,2}| + |\mathbf{C}_{2,1}|\right)$$

where $|\mathbf{C}|$ is the size of the data set $\mathbf{C}$, $\mathbf{C}_{i,j}$ consists of samples in $\mathbf{C}_i$ that belong to class j $(i, j = 1, 2)$.

The kernel parameter, σ^2, is allowed to take values of 1, 3, 5, ..., 29. Under each value, experiments are repeated over 10 randomly generated data sets for each algorithm. The minimum of the average classification errors is listed in Table 1 along with the corresponding σ^2 values and 95% confidence intervals. Clearly, for this artificial data set, the MSC algorithm performs significantly better than Ncut. For the MSC algorithm, the Perron weight vector works better than the other two weighting schemes. But the difference is not statistically significant.

Since the MSC algorithm learns a decision boundary (the curve in the center of Figure 2), the clustering generalizes to unseen inputs. Each decision boundary is

Table 1: Comparing the MSC algorithm and Ncut on artificial data sets. The numbers listed are the average classification errors over 10 randomly generated data sets and the corresponding 95% confidence intervals. MSC1: MSC with $\boldsymbol{\alpha}_1$ weight vector; MSC2: MSC with $\boldsymbol{\alpha}_2$ weight vector; MSC3: MSC with $\boldsymbol{\alpha}_3$ weight vector. α_1, α_2, and α_3 are defined in Section 2.2

	σ^2	Average Classification Error	95% Confidence Interval
MSC1	7	3.9%	[2.81%, 4.99%]
MSC2	15	3.6%	[2.53%, 4.67%]
MSC3	7	3.1%	[2.47%, 3.73%]
Ncut	23	29.95%	[27.79%, 32.11%]

Table 2: The generalization performance of the MSC algorithm on artificial data sets. The numbers listed are the average generalization errors over 10 randomly generated testing sets and the corresponding 95% confidence intervals.

	σ^2	Average Generalization Error	95% Confidence Interval
MSC1	7	4.15%	[3.14%, 5.16%]
MSC2	15	4.85%	[3.72%, 5.98%]
MSC3	7	3.90%	[2.52%, 5.28%]

tested over a new set of 200 samples generated by the above distributions (each class contains 100 samples). The average generalization errors over 10 testing sets are reported in Table 2 along with the corresponding 95% confidence intervals.

4.2 USPS Data Set. The USPS data set contains 9298 grayscale images of handwritten digits. The images are size normalized to fit in a 16×16 pixel box while preserving their aspect ratio. The data set is divided into a training set of 7291 samples and a testing set of 2007 samples. For each sample, the input feature vector consists of 256 grayscale values. Since MSC deals with binary clustering, the training set is divided into 45 subsets, $\mathbf{S}_{i,j}$, $i = 0, \ldots, 9$, $j = 1, \ldots, 9$, $i \neq j$. The subset $\mathbf{S}_{ij}$ consists of digits i and j. In the same way, the testing set is divided into 45 subsets.

We compare the performance of the MSC algorithm with that of Ncut using the training set. The affinity matrix in Ncut is defined by Gaussian kernel. The distance matrix in Algorithm 2.1 is computed using (2.3). The kernel parameter, σ^2, is allowed to take values of 50, 100, ..., 600. For each value of σ^2, the average classification error (defined in Section 4.1) is computed over the 45 subsets. The numbers reported in Table 3 are the minimum average classification errors along with the corresponding σ^2 values and standard deviations.

Table 3: Comparing the MSC algorithm and Ncut on the USPS data set. The numbers listed are the average classification errors over 45 subsets and the corresponding standard deviations.

	σ^2	Average Classification Error	Standard Deviation
MSC1	350	7.86%	8.05%
MSC2	300	7.68%	8.17%
MSC3	250	7.70%	8.12%
Ncut	100	7.05%	8.39%

Table 4: The generalization performance of the MSC algorithm on the USPS data set. The numbers listed are the average generalization errors over 45 subsets and the corresponding standard deviations.

	σ^2	Average Generalization Error	Standard Deviation
MSC1	350	9.26%	7.54%
MSC2	300	8.91%	7.78%
MSC3	250	9.06%	7.67%

As we can see Ncut performs slightly better than the proposed method for the USPS data set. However, the difference is not statistically significant. The separation functions learned from the training sets are also applied to the testing sets. The average generalization errors over 45 subsets are reported in Table 4 along with the corresponding standard deviations.

4.3 COREL Data Set. The image data set employed in our empirical study consists of 2000 images taken from 20 CD-ROMs published by COREL Corporation. Each COREL CD-ROM of 100 images represents one distinct topic of interest. Therefore, the data set has 20 thematically diverse image categories. All the images are in JPEG format with size 384×256 or 256×384. The image category names and some randomly selected sample images from each category are shown in Figure 3. Each image is represented as a collection of regions obtained from image segmentation. Nine features are extracted from each region. They capture the color, texture, and shape properties of the region. For a detailed discussion of the image segmentation algorithm and imagery features, please refer to [5]. The image data set and region features are available at `http://www.cs.uno.edu/~yixin/ddsvm.html`.

Let $\mathbf{B}_i = \{\mathbf{x}_{ij} \in \mathbb{R}^9 : j = 1, \ldots, N_i\}$ be the collection of region features for image i. The distance between two images, with respective collection of regions features $\mathbf{B}_k$ and $\mathbf{B}_l$, is defined by the Hausdorff distance [9]

$$H(\mathbf{B}_k, \mathbf{B}_l) = \max\left(h(\mathbf{B}_k, \mathbf{B}_l), h(\mathbf{B}_l, \mathbf{B}_k)\right)$$

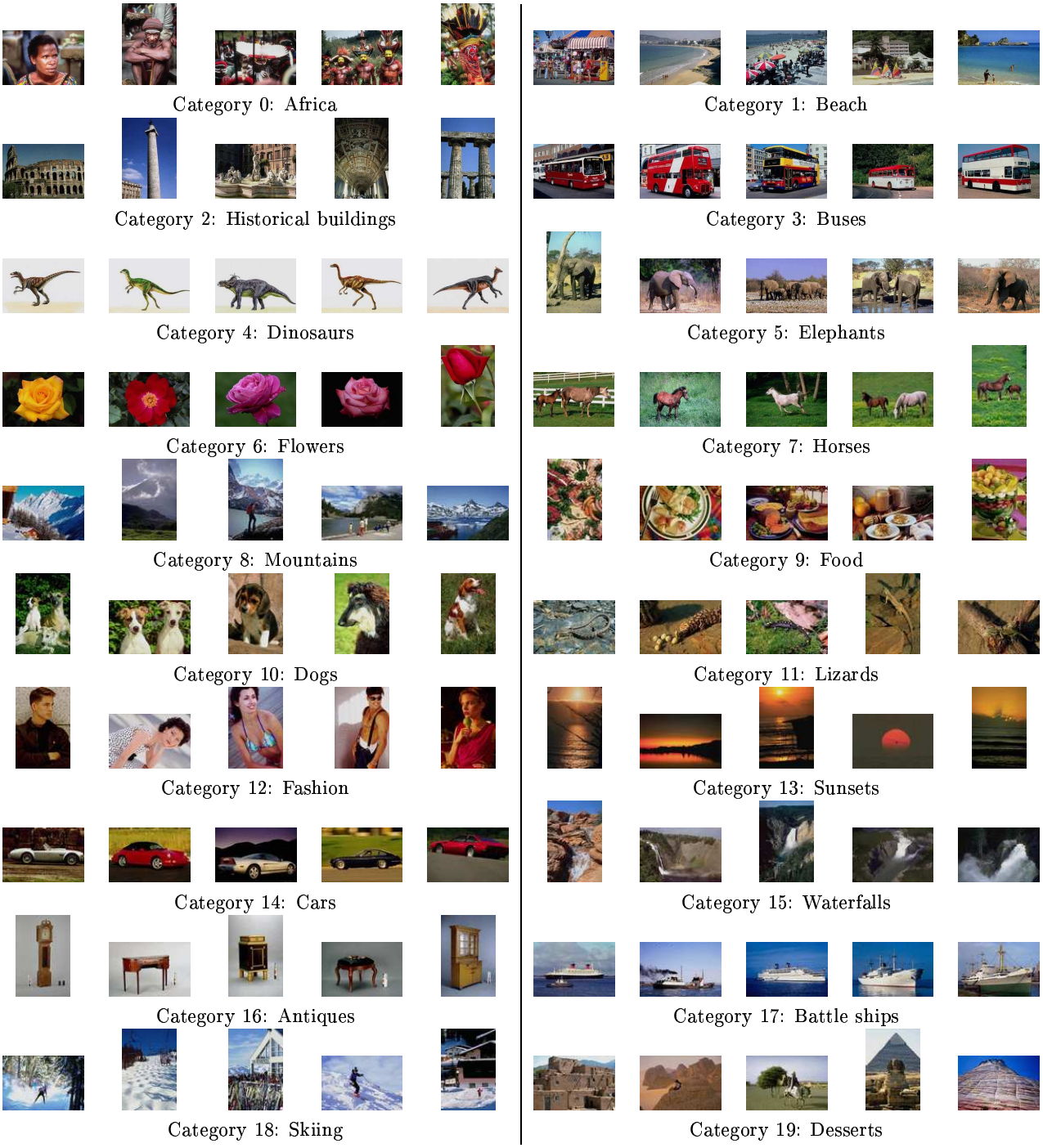


Figure 3: Sample images taken from 20 image categories.

where

$$h(\mathbf{B}_k, \mathbf{B}_l) = \max_{\mathbf{x} \in \mathbf{B}_k} \min_{\mathbf{y} \in \mathbf{B}_l} \|\mathbf{x} - \mathbf{y}\| \ .$$

Since the Hausdorff distance is a metric, it is applied to construct the distance matrix in Algorithm 2.1.

The MSC algorithm is recursively applied to the data set. Each time, the largest cluster is bipartitioned. Clearly, t iterations produce $t+1$ clusters. We use *purity* and *entropy* to measure the goodness of image clustering. Assume we are given a set of n images ($n = 2000$ in this experiment) belonging to c distinctive classes denoted by $0, \ldots, c-1$ ($c = 20$ in this experiment) and the

Table 5: Maximal separation clustering of images with $\boldsymbol{\alpha}_1$ weight vector.

Cluster	Size	Purity	Entropy	Dominant Class
1	65	0.2769	0.7397	Antiques
2	138	0.1594	0.8709	Horses
3	90	0.2667	0.7445	Cars
4	111	0.2162	0.8037	Lizards
5	120	0.1500	0.8160	Beach
6	83	0.2651	0.6563	Battle ships
7	111	0.2613	0.7869	Fashion
8	129	0.3333	0.7690	Food
9	106	0.3868	0.6793	Horses
10	83	0.5783	0.4564	Dinosaurs
11	81	0.1852	0.7625	Waterfalls
12	123	0.3821	0.6062	Buses
13	97	0.2371	0.7923	Lizards
14	101	0.5149	0.5440	Sunsets
15	124	0.2258	0.6615	Mountains
16	91	0.1868	0.8178	Africa
17	86	0.5930	0.4603	Flowers
18	93	0.2258	0.7799	Waterfalls
19	98	0.2653	0.7582	Elephants
20	70	0.3571	0.5619	Dinosaurs

Table 6: Maximal separation clustering of images with $\boldsymbol{\alpha}_2$ weight vector.

Cluster	Size	Purity	Entropy	Dominant Class
1	109	0.2844	0.7384	Elephants
2	111	0.4865	0.5437	Buses
3	89	0.2921	0.7782	Fashion
4	135	0.2148	0.7078	Mountains
5	106	0.3019	0.7481	Waterfalls
6	100	0.2200	0.8197	Lizards
7	108	0.4444	0.6142	Horses
8	70	0.3143	0.5938	Dinosaurs
9	84	0.5595	0.5252	Flowers
10	134	0.1493	0.8197	Beach
11	41	0.2195	0.7470	Lizards
12	151	0.2119	0.8411	Lizards
13	125	0.3840	0.7480	Food
14	81	0.2716	0.6567	Battle ships
15	104	0.1731	0.8434	Africa
16	91	0.1868	0.7654	Sunsets
17	100	0.5300	0.5323	Sunsets
18	78	0.2564	0.7555	Antiques
19	90	0.3111	0.7150	Cars
20	93	0.5484	0.5092	Dinosaurs

images are grouped into $t+1$ clusters $\mathbf{C}_j$, $j = 1, \ldots, m$. Cluster $\mathbf{C}_j$'s purity can be defined as

$$p(\mathbf{C}_j) = \frac{1}{|\mathbf{C}_j|} \max_{k=1,\ldots,c} |C_{j,k}|$$

where $\mathbf{C}_{j,k}$ consists of images in $\mathbf{C}_j$ that belong to class k. Each cluster may contain images of different classes. Purity gives the ratio of the dominant class size in the cluster to the cluster size itself. The value of purity is always in the interval $[\frac{1}{c}, 1]$ with a larger value means that the cluster is a "purer" subset of the dominant class. Entropy is another cluster quality measure, which is defined as follows:

$$h(\mathbf{C}_j) = -\frac{1}{\log c} \sum_{k=1}^{c} \frac{|\mathbf{C}_{j,k}|}{|\mathbf{C}_j|} \log \frac{|\mathbf{C}_{j,k}|}{|\mathbf{C}_j|} .$$

Since entropy considers the distribution of semantic classes in a cluster, it is a more comprehensive measure than purity. Note that we have normalized entropy so that the value is between 0 and 1. Contrary to the purity measure, an entropy value near 0 means the cluster is comprised mainly of 1 category, while an entropy value close to 1 implies that the cluster contains a uniform mixture of all categories. For example, if half of the images of a cluster belong to one class and the rest of the images are evenly divided into 19 different classes, then the entropy is 0.7228 and the purity is 0.5.

Table 5, Table 6, and Table 7 show the purity and entropy of clusters generated by Algorithm 2.1 (with weight vector $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$, and $\boldsymbol{\alpha}_3$, respectively) after 19 bipartitions (i.e., 20 clusters). Size of each cluster and the name of the dominant class in each cluster are also listed. Since the images belong to 20 classes, ideally, each of the 20 clusters should contains 100 images from a unique classes. In our experiments, the average purities under $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$, and $\boldsymbol{\alpha}_3$ are 0.3034, 0.3180, and 0.3155, respectively. And the average entropies are 0.7034, 0.6999, 0.6992. Although the results we obtained is far from perfect, they are significantly better than a random guess where the average purity would be 0.05 and the average entropy would be 1.0. It is observed that in each of the three experiments, there are 4 classes which do not appear as the dominant class in any of the clusters: *Historical buildings*, *Dogs*, *Skiing*, and *Desserts*. It is interesting to observe that in all three experiments, *Historical buildings* is the second largest class in the cluster where the dominant class is *Battle ships*; *Dogs* is the second largest class in the cluster where *Horses* is the dominant class; and *Skiing* is the second largest class in the cluster where *Beach* is the dominant class. *Desserts* does not even show up as the second largest class in any clusters. But it turns out to be the third largest class in the cluster where the top two largest classes are *Battle ships* and *Historical buildings*.

5 Conclusions and Future Work.

In this paper, we propose a new clustering algorithm which computes an "optimal" hyperplane maximizing the sum of squared distance in a feature space induced

Table 7: Maximal separation clustering of images with $\boldsymbol{\alpha}_3$ weight vector.

Cluster	Size	Purity	Entropy	Dominant Class
1	100	0.5300	0.5323	Sunsets
2	50	0.2200	0.7720	Lizards
3	101	0.1980	0.8214	Africa
4	81	0.2840	0.6540	Battle ships
5	100	0.2900	0.7474	Waterfalls
6	151	0.2053	0.8331	Lizards
7	113	0.2566	0.7616	Elephants
8	93	0.5376	0.5128	Dinosaurs
9	86	0.5698	0.5057	Flowers
10	129	0.1550	0.8076	Beach
11	113	0.4779	0.5472	Buses
12	101	0.2178	0.8167	Lizards
13	125	0.3760	0.7439	Food
14	135	0.2148	0.7029	Mountains
15	108	0.4259	0.6345	Horses
16	93	0.2903	0.7757	Fashion
17	89	0.1798	0.7641	Sunsets
18	74	0.2568	0.7462	Antiques
19	88	0.2955	0.7222	Cars
20	70	0.3286	0.5833	Dinosaurs

by the training data and the given metric distance. The separating hyperplane transforms to a nonlinear decision boundary in the input space. Hence the clustering generalizes to unseen samples. The connection between the proposed clustering algorithm and spectral graph partition methods is discussed. Specifically, we prove that, under proper weight vectors, the proposed clustering algorithm is equivalent to a spectral relaxed graph cut – disparity cut. The disparity cut criterion takes into account the between-cluster dissimilarity, the within-cluster dissimilarity, and the size of the clusters. We provide extensive experimental results to verify the method.

Acknowledgements.

This work was supported in part by the University of New Orleans, The Research Institute for Children, and NASA/EPSCoR DART Grant NCC5-573. The authors would like to thank Bin Fu for valuable discussions.

References

[1] J. W. Auer, An Elementary Proof of the Invertibility of Distance Matrices, *Linear and Multilinear Algebra*, 40:119–124, 1995.

[2] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, Clustering with Bregman Divergences, *Proc. 4th SIAM Int'l Conf. on Data Mining*, pages 234–245, 2004.

[3] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering, *Advances in Neural Information Processing Systems 16*, 2003.

[4] P. S. Bradley, U. M. Fayyad, and C. Reina, Scaling Clustering Algorithms to Large Databases, *Proc. 4th Int'l Conf. on Knowledge Discovery and Data Mining*, pages 9–15, 1998.

[5] Y. Chen and J. Z. Wang, Image Categorization by Learning and Reasoning with Regions, *Journal of Machine Learning Research*, 5:913–939, 2004.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, Series B, 39:1–38, 1977.

[7] I. S. Dhillon, Y. Guan, and B. Kulis Kernel k-means, Spectral Clustering and Normalized Cuts, *Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2004.

[8] C. Ding, X. He, H. Zha, M. Gu and H. Simon, Spectral Min-Max Cut for Graph Partitioning and Data Clustering, *Proc. 1st IEEE Int'l Conf. on Data Mining*, pages 107–114, 2001.

[9] G. B. Folland *Real Analysis: Modern Techniques and Their Applications*, 2nd edition, John Wiley & Sons, Inc., 1999.

[10] G. H. Golub and C. F. Van Loan, *Matrix Analysis*, 3rd ed., Johns Hopkins University Press, 1996.

[11] L. Hagen and A. B. Kahng, New Spectral Methods for Ratio Cut Partitioning and Clustering, *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, 1992.

[12] J. A. Hartigan, *Clustering Algorithms*, John Wiley & Sons, 1975.

[13] J. A. Hartigan and M. A. Wong, *Algorithm AS136: A k-means Clustering Algorithm*, Applied Statistics, 28:100-108, 1979.

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, 2001.

[15] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

[16] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

[17] A. Y. Ng, M. I. Jordan, and Y. Weiss, On Spectral Clustering: Analysis and an Algorithm, *Advances in Neural Information Processing Systems 14*, 2001.

[18] I. J. Schoenberg, On Certain Metric Spaces Arising from Euclidean Spaces by a Change of Metric and Their Imbedding in Hilbert Space, *The Annals of Mathematics*, 38(4):787–793, 1937.

[19] N. Shental, A. Zomet, T. Hertz, and Y. Weiss, Pairwise Clustering and Graphical Models, *Advances in Neural Information Processing Systems 16*, 2003.

[20] J. Shi and J. Malik, Normalized Cuts and Image Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[21] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

Deterministic Annealing and a k-Means Type Smoothing Optimization Algorithm for Data Clustering *

Marc Teboulle† Jacob Kogan‡

Abstract

The classical k-means clustering algorithm is probably one of the most popular techniques for data clustering. While partitioning a dataset into a prescribed number of clusters, the algorithm alternates between the computation of clusters and centroids of a dataset partition. The clustering problem can in fact be formulated as a non-smooth and non-convex optimization problem. Within such a formulation, the k-means algorithm can be viewed as an application of a gradient based method to an exact smooth reformulation of the clustering optimization problem that leads to a local minimum of the clustering objective function. In this paper we introduce an alternative approach, whereby we approximate the original non-smooth clustering optimization problem by a family of parametrized smooth optimization problems. By so doing we obtain a new derivation and interpretation of the Deterministic Annealing (DA) for clustering which was introduced by Rose, Gurewitz and Fox [20]. The DA was inspired by principles of statistical mechanics and intended to deal with non-convex optimization problems. Within the proposed optimization framework, unlike DA, we focus on non-smoothness, rather than on non-convexity. This allows us to obtain the algorithm convergence and other important properties and provide new insights into the DA method and its performance. Numerical experiments are given to indicate the superiority of the algorithm when compared to known versions of k–means clustering algorithms.

Key words: clustering, k-means algorithms, deterministic annealing, mathematical optimization, smoothing methods, generalized means.

1 Introduction

The clustering problem is fundamental in data analysis, and arises in a great variety of fields, such as pattern recognition, image processing, text mining, and many others. Given a finite dataset one has to identify its "best possible" partition into a finite set of disjoint groups or clusters by optimizing a certain objective function. More specifically, given a dataset $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\} \subset \mathbf{R}^n$, a k cluster partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of $\mathcal{A}$ is a collection of k disjoint subsets (clusters) of $\mathcal{A}$ so that $\bigcup_{i=1}^{k} \pi_i = \mathcal{A}$. Alternatively, by identifying a cluster π_i with its centroid $\mathbf{x}_i$ one has to select k vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \mathbf{R}^n$ to generate a k cluster partition Π. For each $i = 1, \ldots, m$, let

$$D_i(\mathbf{x}_1, \ldots, \mathbf{x}_k) := \min_{1 \le l \le k} d(\mathbf{a}_i, \mathbf{x}_l)$$

be the distance from $\mathbf{a}_i \in \mathcal{A}$ to the nearest centroid from the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, where $d(\cdot, \cdot)$ is a distance–like function. Our objective is to minimize the sum of the distances between the points of the dataset $\mathcal{A}$ and the nearest centroids. In other words we are interested in solving the continuous optimization problem:

$$(1.1) \quad \min_{\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbf{R}^n} f(\mathbf{x}_1, \ldots, \mathbf{x}_k) := \sum_{i=1}^{m} D_i(\mathbf{x}_1, \ldots, \mathbf{x}_k).$$

In this paper, we focus on the classical choice of the quadratic Euclidean distance $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$. It will be convenient to introduce the following notations. We denote the $k \times n$ dimensional vector $(\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T)^T$ by $\mathbf{x}$, the function $f(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ by $F(\mathbf{x})$, $F : \mathbf{R}^N \to \mathbf{R}$, with $N = kn$. Then, for all $k > 1$, problem (1.1) can be equivalently written as the following *non-convex* and *non-smooth* optimization problem in $\mathbf{R}^N$:

$$(1.2) \quad \min_{\mathbf{x} \in \mathbf{R}^N} F(\mathbf{x}) = \sum_{i=1}^{m} \min_{1 \le l \le k} \|\mathbf{x}_l - \mathbf{a}_i\|^2.$$

Therefore, the clustering problem (1.1) belongs to the class of continuous optimization problems which combines two of the most difficult characteristics one encounters in optimization: non-convexity and non-smoothness. In this paper we focus on an approach handling the second difficulty, namely non-smoothness.

The classical k–means algorithm is probably the most celebrated and widely used general clustering tech-

*Supported by the United States–Israel Binational Science Foundation (BSF).

†teboulle@post.tau.ac.il, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv, Israel.

‡kogan@umbc.edu, Department of Mathematics and Statistics, UMBC, Baltimore, MD 21250. Research of this author was also supported by the Fulbright Program.

nique which attempts to solve this problem by an iterative procedure. It starts with an initial set of k centroids $\{\mathbf{x}_l(0) : l = 1,\ldots,k\}$, and assigns each point of the dataset to the the nearest centroid. Then the centroids are updated. The two step procedure continues until some stopping criterion is satisfied [11]. The k-means algorithm is in fact a local search optimization type algorithm, which handles the non-smoothness by using an *exact* smooth reformulation of the problem on which a Gauss-Seidel gradient type method [5] is applied (see e.g., [22] for more details and references therein).

In this paper, we propose to use a smooth approximation approach which replaces the non-smooth objective $F(\mathbf{x})$ by a *family of smooth functions* $F_s(\mathbf{x})$ that depends on a smoothing parameter s. The special structure of the smoothed objective lends itself to construct a simple algorithm described via a fixed point explicit iteration formula. This algorithm is a special case of a new and general class of smoothing iterative methods for solving various types of clustering problems we recently introduced in our companion paper [22] where more details and theoretical analysis can be found. In this paper we briefly describe the approach that lays the ground to the proposed clustering algorithm. This is outlined in Section 3 where we suggest a way to approximate $F(\mathbf{x})$ by a family of smooth functions. We introduce a particular algorithm, called `smoka` (`SMOothing K-means Algorithm`), to solve the clustering continuous optimization problem, and we state its main properties. It turns out that the approach we propose leads to a new derivation and interpretation of the DA algorithm introduced by Rose, Gurewitz and Fox [20] and motivated by statistical physics arguments. The DA was claimed to be very fast and effective to handle the first difficulty alluded above, namely non-convexity. Our optimization approach provides new insights into the DA method and its performance, and further clarifies the convergence and other properties of the DA algorithm. This is discussed in Section 2. To illustrate the validity of our approach, we first recall in Section 4 the classical $k-$means clustering algorithm, as well as its recently introduced enhancement (see e.g., [25], [12], [15]) and describe the behavior of `smoka` and the $k-$means like algorithms when applied to a particular scalar dataset. In section 5 we report on some encouraging numerical experiments with `smoka` when applied to several multidimensional datasets and compare its performance with the k-means type algorithms. The current numerical experiments indicate the superiority of `smoka` over the known versions of k-means clustering algorithms. Section 6 concludes the paper.

2 Deterministic Annealing and Optimization

In 1990 Rose, Gurewitz and Fox introduced a statistical mechanics formulation for clustering and derived a remarkable algorithm called the Deterministic Annealing (DA). The DA is used in order to avoid local minima of the given non-convex objective function (see [20]). A well known method for solving non-convex optimization problems is the simulated annealing (SA, see e.g. [14]), which is a stochastic method motivated by its analogy to statistical physics and is based on the Metropolis algorithm [16]. However, unlike the SA method, the DA algorithm replaces the stochastic search by a deterministic search, whereby the annealing is performed by minimizing a cost function (the free energy) directly, rather than via stochastic simulations of the system dynamics. Building on this procedure, global minima is obtained by minimizing the cost function (the free energy) while gradually reducing a parameter that plays the role of "temperature." Among other things the method offers "the ability to avoid many poor local optima" (see [19], p. 2210). More recently, (see [19] and references therein) the DA has been re-interpreted within a purely probabilistic framework within basic information theory principles. Many reports in the literature have indicated that the DA algorithm outperforms the standard $k-$means clustering algorithm, and it can be also used in the context of other important applications (see [19] and references therein).

We present a new derivation and interpretation of the DA algorithm that do not rely on any statistical physics concepts. Our approach is based solely on mathematical optimization techniques. Within this interpretation we derive its convergence properties, provide new insights into the clustering problem that also re–enforce the past mentioned studies. Thus not only our approach can be viewed as complementary, but more importantly it paves the way to further use of mathematical and optimization tools for future works.

We complete this background section by mentioning two additional contributions that treat the $k-$mean like clustering as a finite dimensional optimization problem. Zhang, Hsu and Dayal [24] proposed an algorithm called the $k-$harmonic means clustering algorithm. The algorithm is based on minimizing a cost whereby the min operator in (1.2) is replaced by the harmonic mean. In a similar vein, Nasraoui and Krishnapuram [17] proposed to use other performance measures based on l_p norms. In fact, both approaches have suggested cost functions for the clustering problem that can be seen as special cases of the so-called generalized means introduces by Hardy, Littlewood and Polya in their 1934 monograph [13] (further details will be given in our forthcoming paper [22]).

3 Smoothing $k-$Means Algorithm

Consider the clustering optimization problem defined in $\mathbf{R}^N$ via:

$$\min_{\mathbf{x}\in\mathbf{R}^N} F(\mathbf{x}) = \sum_{i=1}^{m} \min_{1\le l\le k} \|\mathbf{x}_l - \mathbf{a}_i\|^2. \tag{3.3}$$

The underlying idea of this paper is based on combining a smoothing and successive approximation technique we now briefly describe. This approach is based on a general mathematical technique that opens the door to reformulation of k–means clustering with a wide variety of distance like functions as finite dimensional optimization problems.

For any $\mathbf{z} \in \mathbf{R}^N$, one has

$$\max_{1\le j\le N} \mathbf{z}[j] = \lim_{s\to 0^+} s \log\left(\sum_{j=1}^{N} e^{\mathbf{z}[j]/s}\right) := f_s(\mathbf{z}).$$

This can be verified by a direct computation, or by observing that it is just a reformulation of the definition of the l_∞ norm of a vector in $\mathbf{R}^N$.

Thus for small $s > 0$ we can *approximate* the non-smooth max function $f(\mathbf{z}) = \max_{1\le j\le N} \mathbf{z}[j]$ by the smooth function $f_s(\mathbf{z})$. This fact is very well known and widely used in the field of optimization. In fact, this is a special case of a general smoothing approach introduced by Ben–Tal and Teboulle [2]. The approach relies on the use of asymptotic functions which play a central role in optimization (see the recent book by Auslender and Teboulle [1] for more details and results, and references therein). Returning to our problem (3.3), since

$$\max_{1\le l\le k} \mathbf{y}[l] = -\min_{1\le l\le k} \{-\mathbf{y}[l]\},$$

we substitute $F(\mathbf{x})$ by

$$F_s(\mathbf{x}) = \sum_{i=1}^{m} -s \log\left(\sum_{l=1}^{k} e^{-\frac{\|\mathbf{x}_l - \mathbf{a}_i\|^2}{s}}\right)$$

so that $\lim_{s\to 0^+} F_s(\mathbf{x}) = F(\mathbf{x})$. Thus, it is natural to replace the objective $F(\mathbf{x})$ by a family of smooth functions defined by:

$$F_s(\mathbf{x}) = \sum_{i=1}^{m} -s \log\left(\sum_{l=1}^{k} e^{-\frac{\|\mathbf{x}_l - \mathbf{a}_i\|^2}{s}}\right), \tag{3.4}$$

and thus replace problem (3.3) by the approximate family of smoothed problems parametrized by $s > 0$:

$$\min_{\mathbf{x}\in\mathbf{R}^N} F_s(\mathbf{x}). \tag{3.5}$$

We remark that for each positive and small enough s, the approximation function F_s is a nonnegative infinitely differentiable function of $\mathbf{x}$. The function F_s is not necessarily convex, so finding a global minimum is not guaranteed. Thus the new problem is still not easy, since there are many local minima. Note that this difficulty is also not avoided within the standard k-means algorithm. Furthermore, the proposed smoothed reformulation of (1.1) can now be solved by standard smooth optimization techniques, a possibility that will not be pursued in this paper. Instead, here, we will fix the smoothing parameter $s > 0$, and use the very special structure of F_s to derive a simple algorithm based on an explicit fixed point iteration formula. We note that the idea of combining smoothing with successive approximations is a widely known technique in optimization that can be traced back to the so-called Weiszfeld algorithm derived in 1937 for location theory problems [23]. The algorithm has provided a fertile ground for many problems in a variety of research areas. For more recent studies see, for example, Ben–Tal and Teboulle [3] and Brimberg and Love [7] and references therein.

Smoothing k–means algorithm–`smoka`

We fix $s > 0$ and denote a local minimum of F_s by $\overline{\mathbf{x}}(s)$. To simplify the notations, when it does not lead to ambiguity, we shall drop the parameter s and denote $\overline{\mathbf{x}}(s)$ just by $\overline{\mathbf{x}} = \left(\overline{\mathbf{x}}_1^T, \ldots, \overline{\mathbf{x}}_k^T\right)^T$. The necessary local optimality condition for problem (3.5) is $\nabla F_s(\overline{\mathbf{x}}) = 0$, and thus for each $l = 1, \ldots, k$ we obtain

$$\sum_{i=1}^{m} \frac{(\overline{\mathbf{x}}_l - \mathbf{a}_i) e^{-\frac{\|\overline{\mathbf{x}}_l - \mathbf{a}_i\|^2}{s}}}{\sum_{j=1}^{k} e^{-\frac{\|\overline{\mathbf{x}}_j - \mathbf{a}_i\|^2}{s}}} = 0. \tag{3.6}$$

For each $l = 1, \ldots, k$ and $i = 1, \ldots, m$ and $\mathbf{x} = \left(\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T\right)^T$ we define the positive numbers

$$\rho^{il}(\mathbf{x}, s) = \frac{e^{-\frac{\|\mathbf{x}_l - \mathbf{a}_i\|^2}{s}}}{\sum_{j=1}^{k} e^{-\frac{\|\mathbf{x}_j - \mathbf{a}_i\|^2}{s}}}. \tag{3.7}$$

Equation (3.6) now becomes

$$\sum_{i=1}^{m} (\overline{\mathbf{x}}_l - \mathbf{a}_i) \rho^{il}(\overline{\mathbf{x}}, s) = 0, \tag{3.8}$$

and yields

$$\overline{\mathbf{x}}_l = \frac{\sum_{i=1}^{m} \rho^{il}(\overline{\mathbf{x}}, s) \mathbf{a}_i}{\sum_{j=1}^{m} \rho^{jl}(\overline{\mathbf{x}}, s)}. \tag{3.9}$$

To simplify the expression for $\overline{\mathbf{x}}_l$ we define

$$\lambda^{il}(\mathbf{x}, s) = \frac{\rho^{il}(\mathbf{x}, s)}{\sum_{j=1}^{m} \rho^{jl}(\mathbf{x}, s)}, \tag{3.10}$$

so that

$$\lambda^{il}(\mathbf{x}, s) > 0, \text{ and } \sum_{i=1}^{m} \lambda^{il}(\mathbf{x}, s) = 1.$$

Expression for $\overline{\mathbf{x}}_l$ can now be written in a compact form as follows:

$$\overline{\mathbf{x}}_l = \sum_{i=1}^{m} \lambda^{il}(\overline{\mathbf{x}}, s)\mathbf{a}_i, \text{ for each } l = 1, \ldots, k. \tag{3.11}$$

The fixed point characterization of $\overline{\mathbf{x}}$ suggests to use successive approximations to compute $\overline{\mathbf{x}}$. Thus, the basic algorithm is as follows:

ALGORITHM 3.1. (`smoka`)
For a user defined non negative tolerance `tol`, smoothing parameter $s > 0$, and initial choice of k centroids $\mathbf{x}(0) = (\mathbf{x}_1^T(0), \ldots, \mathbf{x}_k^T(0))^T \in \mathbf{R}^N$ do the following:

1. Set $t = 0$.
2. For each $l = 1, \ldots, k$ compute
$$\mathbf{x}_l(t+1) = \sum_{i=1}^{m} \lambda^{il}(\mathbf{x}(t), s)\mathbf{a}_i.$$
3. If $F_s(\mathbf{x}(t)) - F_s(\mathbf{x}(t+1)) >$ `tol`
set $t = t + 1$
goto step 2
4. Stop.

The formula for $\overline{\mathbf{x}}_l$ is in fact nothing else but the one derived by Rose, Gurewitz and Fox [20], [21]. Here, the smoothing parameter s plays the role of the "temperature" there, and the sequence produced by the DA algorithm with constant "temperature" coincides with the sequence generated by `smoka` (see also Section 2).

Thus, using standard and elementary tools from optimization theory we have recovered the DA algorithm. It is further interesting to note that while the derivation of Rose–Gurewitz–Fox relied on Statistical Physics, which among other things, also requires assumptions concerning the probability distribution for vector-centroids associations, the proposed derivation is assumption free, and our approach is very much different. It paves the way of developing an analysis based solely on mathematical optimization techniques. At this juncture we also mention that the more recent probabilistic interpretation of the DA algorithm within information theory principles given in [19] can also be interpreted and derived via optimization principles. These interpretations and derivations together with a detailed analysis of a general class of algorithms that includes `smoka` is developed in our companion paper [22]. The remaining of this section briefly outlines some of properties that are obtained as by product of the optimization approach.

First, note that formula (3.11) indicates that $\overline{\mathbf{x}}_l$ belongs to the interior of the convex hull of the dataset $\mathcal{A}$. About the convergence of `smoka`, we have the following theorem (for a proof, see [22]):

THEOREM 3.1. *Let $\{\mathbf{x}(t)\}_{t=0}^{\infty}$ be a sequence generated by* `smoka`. *If $\mathbf{x}(t) \neq \mathbf{x}(t+1)$, then*

$$F_s(\mathbf{x}(t)) > F_s(\mathbf{x}(t+1)).$$

Furthermore, every limit point $\mathbf{x}$ of $\{\mathbf{x}(t)\}_{t=0}^{\infty}$ is a stationary point of F_s.

An important property of the function F_s is that it approximates the original objective F uniformly. Indeed, (see [2]), for each $\mathbf{x} \in \mathbf{R}^N$ and $s > 0$ one has

$$0 \leq F(\mathbf{x}) - F_s(\mathbf{x}) \leq sm \log k. \tag{3.12}$$

We focus on the upper bound $sm \log k$ for the nonnegative expression $F(\mathbf{x}) - F_s(\mathbf{x})$ and pick i, $1 \leq i \leq m$. To simplify the exposition we assume that

$$\|\mathbf{x}_1 - \mathbf{a}_i\|^2 \leq \|\mathbf{x}_2 - \mathbf{a}_i\|^2 \ldots \leq \|\mathbf{x}_k - \mathbf{a}_i\|^2 \tag{3.13}$$

and denote $\|\mathbf{x}_l - \mathbf{a}_i\|^2$ by Δ_l, $l = 1, \ldots, k$. Then

$$\begin{aligned}
& -s \log \left(\sum_{l=1}^{k} e^{-\frac{\|\mathbf{x}_l - \mathbf{a}_i\|^2}{s}} \right) \\
= \; & -s \log \left(\sum_{l=1}^{k} e^{\frac{-\Delta_l}{s}} \right) \\
= \; & -s \log \left[e^{\frac{-\Delta_1}{s}} \left(1 + e^{\frac{\Delta_1 - \Delta_2}{s}} + \ldots + e^{\frac{\Delta_1 - \Delta_k}{s}} \right) \right] \\
= \; & \Delta_1 - s \log \left(1 + e^{\frac{\Delta_1 - \Delta_2}{s}} + \ldots + e^{\frac{\Delta_1 - \Delta_k}{s}} \right) \\
\leq \; & \min_{1 \leq l \leq k} \|\mathbf{x}_l - \mathbf{a}_i\|^2 \\
& -s \log \left(1 + e^{\frac{\Delta_1 - \Delta_2}{s}} + \ldots + e^{\frac{\Delta_1 - \Delta_k}{s}} \right).
\end{aligned}$$

We remark that in many cases the first inequality in (3.13) is strict, and as $s \to 0$ the expression

$$\epsilon_i(s) = \log \left(1 + e^{\frac{\Delta_1 - \Delta_2}{s}} + \ldots + e^{\frac{\Delta_1 - \Delta_k}{s}} \right)$$

is much smaller than $\log k$. Since

$$(3.14)\quad -s\log\left(\sum_{l=1}^{k} e^{-\frac{\|\mathbf{x}_l-\mathbf{a}_i\|^2}{s}}\right)=\min_{1\le l\le k}\|\mathbf{x}_l-\mathbf{a}_i\|^2-s\epsilon_i(s),$$

one has

$$\begin{aligned} F_s(\mathbf{x}) &= -s\sum_{i=1}^{m}\log\left(\sum_{l=1}^{k} e^{-\frac{\|\mathbf{x}_l-\mathbf{a}_i\|^2}{s}}\right) \\ &= \sum_{i=1}^{m}\min_{1\le l\le k}\|\mathbf{x}_l-\mathbf{a}_i\|^2 - s\sum_{i=1}^{m}\epsilon_i(s) \\ &= F(\mathbf{x}) - s\sum_{i=1}^{m}\epsilon_i(s). \end{aligned}$$

Hence, as $s\to 0$ the nonnegative expression $F(\mathbf{x})-F_s(\mathbf{x})$ has the potential to be much smaller than $sm\log k$.

In the next section we review some classical clustering algorithms that will be tested against `smoka` on several datasets in Section 5.

4 Classical Clustering Algorithms

In this section we briefly review the classical batch $k-$means algorithm [11] and the classical batch $k-$means algorithm augmented by the incremental $k-$means algorithm [25], [12], [15], [9] (we shall refer to this algorithm simply as $k-$means). To illustrate the behavior of the different algorithms we apply the $k-$means algorithms and `smoka` to a simple scalar dataset.

We remind the reader how the classical batch $k-$means algorithm partitions a dataset $\mathcal{A}=\{\mathbf{a}_1,\ldots,\mathbf{a}_m\}\subset\mathbf{R}^n$ into k clusters. An iteration of the algorithm oscillates between computing clusters and centroids of a dataset partition. For a partition $\Pi=\{\pi_1,\ldots,\pi_k\}$ of $\mathcal{A}$ the **first** step of an iteration computes the corresponding centroids $\{\mathbf{x}_1,\ldots,\mathbf{x}_k\}$ via

$$(4.15)\qquad \mathbf{c}(\pi)=\operatorname{argmin}\sum_{\mathbf{a}\in\pi} d(\mathbf{a},\mathbf{x}),$$

and with $d(\cdot,\cdot)$ being the quadratic Euclidean distance, the centroid corresponding to cluster π is just the arithmetic mean of the cluster. The **second** step of the iteration uses centroids $\{\mathbf{x}_1,\ldots,\mathbf{x}_k\}$ to build a new partition $\Pi'=\{\pi'_1,\ldots,\pi'_k\}$ as follows: A vector $\mathbf{a}$ belongs to the new cluster π'_i if $\mathbf{x}_i$ is the centroid nearest $\mathbf{a}$, i.e.,

$$(4.16)\qquad \|\mathbf{a}-\mathbf{x}_i\|\le\|\mathbf{a}-\mathbf{x}_j\|,\ \text{ for each } j\ne i.$$

Using partition Π' one can compute the corresponding centroids $\{\mathbf{x}'_1,\ldots,\mathbf{x}'_k\}$ and it is known (see e.g. [11]) that

$$Q(\Pi)\ge Q(\Pi')$$

where $Q(\Pi)$ measures the quality of partition and, with a slight abuse of notations, is given by

$$(4.17)\qquad Q(\Pi)=Q(\pi_1)+\ldots+Q(\pi_k),$$

with

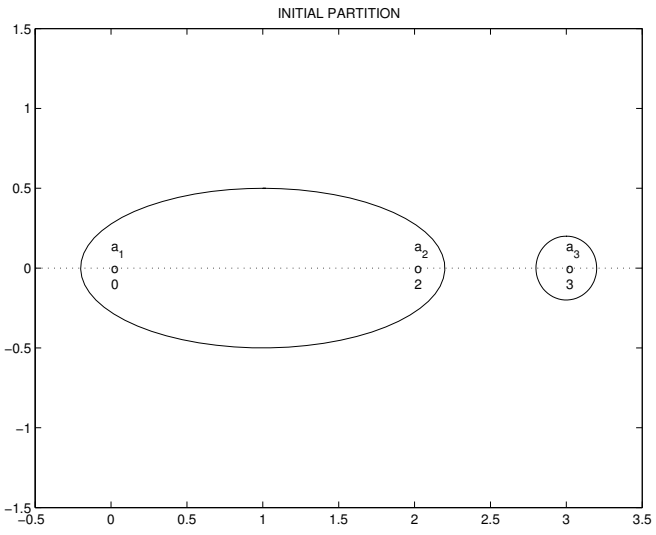
$$Q(\pi_i):=\sum_{\mathbf{a}\in\pi_i}\|\mathbf{a}-\mathbf{x}_i\|^2.$$

The algorithm starts with an initial partition $\Pi^{(0)}$ and generates a sequence of partitions $\Pi^{(1)},\Pi^{(2)},\ldots$. The algorithm runs as long as the decrease in the objective function

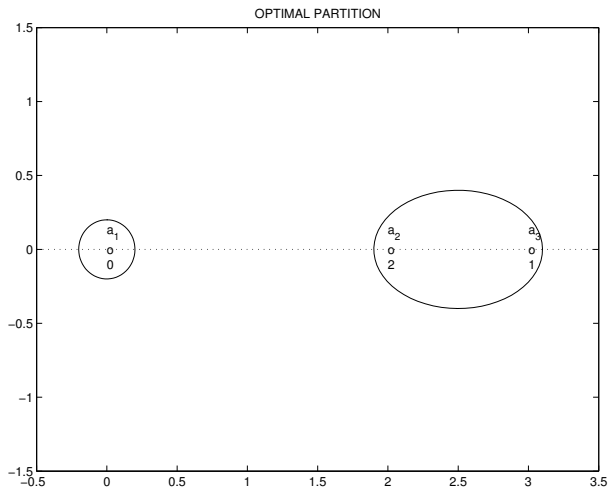
$$Q\left(\Pi^{(i)}\right)-Q\left(\Pi^{(i+1)}\right)$$

exceeds a user specified threshold `tol`. While fast and memory efficient the batch $k-$means often gets trapped at a local minimum (see Example 4.1 below).

EXAMPLE 4.1. *Let $\mathcal{A}=\{0,2,3\}$, and the initial partition $\Pi^{(0)}=\{\pi_1^{(0)},\pi_2^{(0)}\}$ where $\pi_1^{(0)}=\{0,2\}$, and $\pi_2^{(0)}=\{3\}$.*



An iteration of the batch $k-$means algorithm applied to $\Pi^{(0)}$ does not change the partition. On the other hand the partition $\Pi=\{\{0\},\{2,3\}\}$ is superior to $\Pi^{(0)}$.



The better partition Π *is undetected by the algorithm.*

Note that the decision whether a vector $\mathbf{a} \in \pi_i$ should be moved from cluster π_i with m_i vectors to cluster π_j with m_j vectors is made by the batch $k-$means algorithm based on examination of the expression $||\mathbf{a} - \mathbf{x}_i|| - ||\mathbf{a} - \mathbf{x}_j||$. The positive sign of

$$\Delta_k = ||\mathbf{a} - \mathbf{x}_i||^2 - ||\mathbf{a} - \mathbf{x}_j||^2, \tag{4.18}$$

may trigger the move. On the other hand the change in the value of the objective function caused by the move is

$$\Delta = \frac{m_i}{m_i - 1} ||\mathbf{a} - \mathbf{x}_i||^2 - \frac{m_j}{m_j + 1} ||\mathbf{a} - \mathbf{x}_j||^2 \tag{4.19}$$

(see e.g. [10]). The difference between the expressions

$$\Delta - \Delta_k = \frac{1}{m_i - 1} ||\mathbf{a} - \mathbf{x}_i||^2 + \frac{1}{m_j + 1} ||\mathbf{a} - \mathbf{x}_j||^2 \geq 0.$$

In particular, it is possible that Δ_k is non-positive, and the batch $k-$means iteration leaves $\mathbf{a}$ in cluster π_i. At the same time the value of Δ is positive, and reassigning $\mathbf{a}$ to π_j would decrease Q. Indeed, for the dataset of Example 4.1 and $\mathbf{a} = \mathbf{a}_2$ one has

$$\Delta_k = \|\mathbf{a}_2 - \mathbf{x}_1\|^2 - \|\mathbf{a}_2 - \mathbf{x}_2\|^2 = 1 - 1 = 0,$$

and

$$\Delta = \frac{2}{1}\|\mathbf{a}_2 - \mathbf{x}_1\|^2 - \frac{1}{2}\|\mathbf{a}_2 - \mathbf{x}_2\|^2 = 2 - \frac{1}{2} = \frac{3}{2}.$$

The remark and the example suggest to augment the batch $k-$means algorithm with iterations reassigning a single vector (so–called "incremental" iterations). The combination of batch and incremental iterations was introduced recently in [25], [12], [15]. The augmented algorithm (that we shall call simply $k-$means [15], [9]) always outperforms the batch $k-$means algorithm. Note that quantities $||\mathbf{a} - \mathbf{x}_i||$ needed for (4.19) are computed by an iteration of the $k-$means algorithm. An incremental iteration following a batch iteration comes, therefore, at virtually no additional computational expense. One application of the batch $k-$means algorithm iteration augmented by incremental iterations immediately leads from $\Pi^{(0)}$ to Π.

Finally we remark that an application of `smoka` with parameters $s = 0.1$ and `tol` $= 0.001$ to the partition $\Pi^{(0)}$ produces centroids $\mathbf{x}_1 = 0$ and $\mathbf{x}_2 = 2.5$, thus recovering Π.

In the next section we apply `smoka` and the aforementioned clustering algorithms to various datasets. The obtained results and performance of the algorithms are reported.

5 Numerical Experiments

In this section we report clustering results for the following document collections:

1. A merger of the three document collections available at `http://www.cs.utk.edu/~lsi/`:
   - DC0 (Medlars Collection, 1033 medical abstracts).
   - DC1 (CISI Collection, 1460 information science abstracts).
   - DC2 (Cranfield Collection, 1398 aerodynamics abstracts).
2. The 20 newsgroups dataset available at `http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html`.
   - The "mini" dataset is a subset of the "full" dataset with 100 documents from each of the 20 Usenet newsgroups.
   - The "full" dataset of 19997 messages taken from 20 Usenet newsgroups.

We remove stop-words (see `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`), and stem the texts (see [18]). We then select n "best" terms, apply IDFB normalization scheme [8], and build vectors of dimension n for each document collection (see [4] for the vector space model description and consult [9] for selection procedure details, unlike [9] in this experiment we rate all collection terms). The Principal Direction Divisive Partitioning (PDDP) introduced in [6] is applied to a vector set to generate the initial partition $\Pi^{(0)}$. We then apply the batch $k-$means, $k-$means, and `smoka` to the partition $\Pi^{(0)}$ and report qualities of the obtained final partitions along with the number of iterations performed by each algorithm.

Since the first three collections (classic3) are known to be well separated we also provide confusion matrices corresponding to partitions of these collections. In the experiments reported below batch $k-$means, and $k-$means are run with `tol` $= 0$, `smoka` is run with $s = 0.0001$ and `tol` $= 0.0001$.

We conduct the first experiment with classic3 collection. The confusion matrix for the partition $\Pi^{(0)}$ is given in Table 1. When the number of terms is relatively small some documents may contain no selected terms, and their corresponding vectors are zeros. We always remove these vectors ahead of clustering and assign the "empty" documents into a special cluster. This cluster concludes the "confusion" matrix (and is empty in this experiment). The zero vectors do appear when we decrease the dimension of the vector space model (see e.g. Table 6). Application of the batch k–means clus-

	DC0	DC1	DC2
cluster 0	907	91	13
cluster 1	120	7	1372
cluster 2	6	1362	13
"empty" documents			
cluster 3	0	0	0

Table 1: Collection: **classic3**. PDDP generated initial "confusion" matrix with **250** "misclassified" documents using 600 best terms, $Q = 3612.61$

tering algorithm to the partition $\Pi^{(0)}$ iterates 3 times only and generates a partition with confusion matrix given in Table 2. Application of the k–means clustering

	DC0	DC1	DC2
cluster 0	984	50	12
cluster 1	42	2	1368
cluster 2	7	1408	18
"empty" documents			
cluster 3	0	0	0

Table 2: Collection: **classic3**. Batch k–means generated "confusion" matrix with **131** "misclassified" documents using 600 best terms, $Q = 3608.06$

algorithm to the partition $\Pi^{(0)}$ produces 87 batch iterations and 72 incremental iterations. The confusions matrix corresponding to the generated final partition is given in Table 3. Finally, `smoka` applied to $\Pi^{(0)}$ after only 7 iterations builds a partition with the confusion matrix given in Table 4. While the quality of final partitions generated by k–means and `smoka` are almost

	DC0	DC1	DC2
cluster 0	1018	21	19
cluster 1	5	1	1356
cluster 2	10	1438	23
"empty" documents			
cluster 3	0	0	0

Table 3: Collection: **classic3**. $k-$means generated "confusion" matrix with **79** "misclassified" documents using 600 best terms, $Q = 3605.5$

	DC0	DC1	DC2
cluster 0	1019	22	15
cluster 1	5	1	1362
cluster 2	9	1437	21
"empty" documents			
cluster 3	0	0	0

Table 4: Collection: **classic3**. `smoka` generated "confusion" matrix with **73** "misclassified" documents using 600 best terms, $Q = 3605.5$

identical the number of iterations performed by `smoka` is significantly smaller. This information is collected in Table 5. We reduce the vector space dimension to 100

algorithm	iterations	Q
PDDP		3612.6
batch k–means	3	3608.1
k–means	87	3605.5
`smoka`	7	3605.5

Table 5: Collection: **classic3**. Number of iterations per clustering algorithm applied to the initial partition generated by PDDP, the vector space dimension is 600

and repeat the experiment with the same dataset. The confusion matrix for the 3 cluster partition $\Pi^{(0)}$ generated by PDDP is given in Table 6. Table 7, Table 8, and Table 9 present confusion matrices generated from $\Pi^{(0)}$ by batch $k-$means, $k-$means, and `smoka` respectively. Table 10 reports quality of partition generated and number of iterations performed by the algorithms applied to $\Pi^{(0)}$.

We now cluster the "mini" subset of the 20 newsgroups dataset (total of 2000 documents). We first apply PDDP to 2000 vectors of dimension 600 and generate the initial partition $\Pi^{(0)}$ with $Q = 1758.4$. Applications of batch $k-$means, $k-$means and `smoka` are

	DC0	DC1	DC2
cluster 0	384	35	1364
cluster 1	3	999	5
cluster 2	642	425	28
"empty" documents			
cluster 3	4	1	1

Table 6: Collection: **classic3**. PDDP generated initial "confusion" matrix with **880** "misclassified" documents using 100 best terms, $Q = 3379.17$

	DC0	DC1	DC2
cluster 0	193	34	1354
cluster 1	11	1063	7
cluster 2	825	362	36
"empty" documents			
cluster 3	4	1	1

Table 7: Collection: **classic3**. batch $k-$means generated initial "confusion" matrix with **643** "misclassified" documents using 100 best terms, $Q = 3365.3$

	DC0	DC1	DC2
cluster 0	154	19	1360
cluster 1	116	1420	29
cluster 2	759	20	8
"empty" documents			
cluster 3	4	1	1

Table 8: Collection: **classic3**. $k-$means generated "confusion" matrix with **346** "misclassified" documents using 100 best terms, $Q = 3341.06$

	DC0	DC1	DC2
cluster 0	150	19	1361
cluster 1	115	1419	28
cluster 2	764	21	8
"empty" documents			
cluster 3	4	1	1

Table 9: Collection: **classic3**. `smoka` generated "confusion" matrix with **341** "misclassified" documents using 100 best terms, $Q = 3341.07$

reported in Table 11. While `smoka` appears to generate partitions of quality comparable with k–means it again performs significantly less iterations. As Table 11

algorithm	iterations	Q
PDDP		3379.17
batch k–means	9	3365.30
k–means	550	3341.06
`smoka`	31	3341.07

Table 10: Collection: **classic3**. Number of iterations per clustering algorithm applied to the initial partition $\Pi^{(0)}$ generated by PDDP, the vector space dimension is 100

algorithm	iterations	Q
PDDP		1758.4
batch k–means	11	1737.7
k–means	473	1721.9
`smoka`	15	1726.3

Table 11: Collection: **the "mini" subset of the 20 newsgroups dataset**. Number of iterations per clustering algorithm applied to the initial partition generated by PDDP, 2000 vectors of dimension 600

shows an application of `smoka` to $\Pi^{(0)}$ leads to the partition $\Pi^{(1)}$ with $Q = 1726.3$ in 15 iterations only. A subsequent application of k–means to $\Pi^{(1)}$ stops after 226 batch and 194 incremental iterations that result in the final partition with $Q = 1721.9$.

The final experiment reported in the paper deals with the full set of 19997 documents. We build the vector space model of dimension 1000, generate the initial 20 cluster partition $\Pi^{(0)}$ with PDDP, and apply batch $k-$means, $k-$means and `smoka` to $\Pi^{(0)}$. The clustering results are reported in Table 12.

algorithm	iterations	Q
PDDP		18156
batch k–means	47	17956
k–means	5862	17808
`smoka`	51	17808

Table 12: Collection: **the "full" 20 newsgroups dataset**. Number of iterations per clustering algorithm applied to the initial partition generated by PDDP, 19997 vectors of dimension 1000 from the "full" 20 newsgroups dataset

6 Conclusion

The paper has shown that the DA algorithm can be derived by simple optimization techniques without recourse to statistical mechanics and probabilistic arguments. Our optimization approach provides further insights to the DA method, the way it works, and opens the door to further improve it within the use of more powerful optimization techniques.

The motivation for the DA algorithm was to built an algorithm that eliminates the non-convexity difficulty which arise in the clustering optimization problem, and to find global optimal solutions. However, thus far it is not clear that global optimality can be guaranteed or/and that the quality of a solution produced by DA can be estimated. On the other hand, in this paper, we have handled the second difficulty in the clustering optimization problem, namely the non-smoothness. By so doing, we have shown that the DA algorithm with fixed "temperature" in fact coincides with `smoka`. Thus from our analysis, it appears that in the context of clustering problems the DA algorithm, which was based on statistical physics analogies is also handling only the non-smoothness difficulty and does not mathematically resolve the non-convexity issue. Nevertheless, it turns out that the special "Gaussian–like" form of the function F_s tends to eliminate many potential local minima. Like the many simulations results presented in the past literature with the DA method, our preliminary numerical experiments indicate that `smoka` outperforms the classical batch $k-$means algorithm, as well as its recently introduced enhanced versions.

Acknowledgments The authors thank anonymous reviewers for bringing to their attention works of Zhang, Hsu and Dayal, Nasraoui and Krishnapuram, and Rose, Gurewitz, and Fox.

References

[1] A. Auslender and M. Teboulle. *Asymptotic Cones and Functions in Optimization and Variational Inequalities*. Springer–Verlag, New York, 2003.

[2] A. Ben-Tal and M. Teboulle. A smoothing technique for nondifferentiable optimization problems. In *Springer Verlag Lecture Notes in Mathematics*, volume 1405, pages 1–11, Berlin, 1989.

[3] A. Ben-Tal and M. Teboulle. A least-squares based method for a class of nonsmooth minimization problems with applications in plasticity. *Appl. Mathematics and Optimization*, 24:273–288, 1991.

[4] M. Berry and M. Browne. *Understanding Search Engines*. SIAM, 1999.

[5] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, second edition, 1999.

[6] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[7] J. Brimberg and Love R.F. Global convergence of a generalized iterative procedure for the minisum location problem with l_p distances. *Operations Research*, 41:1153–1163, 1993.

[8] E. Chisholm and T. Kolda. New term weighting formulas for the vector space method in information retrieval, 1999. Report ORNL/TM-13756, Computer Science and Mathematics Division, Oak Ridge National Laboratory.

[9] I. S. Dhillon, J. Kogan, and C. Nicholas. Feature selection and document clustering. In M.W. Berry, editor, *A Comprehensive Survey of Text Mining*, pages 73–100. Springer-Verlag, 2003.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, second edition, 2000.

[11] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(3):768, 1965.

[12] P. Hansen and Mladenovic N. J–Means: a new local search heuristic for minimum sum of squares clustering. *Patern Recognition*, 34:405–413, 2001.

[13] G. Hardy, Littlewood J.E., and G. Polya. *Inequalities*. Cambridge University Press, Cambridge, 1934.

[14] S. Kirkpatrick, C.D. Gelatt, and Vecchi M.P. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[15] J. Kogan. Means clustering for text data. In M.W.Berry, editor, *Proceedings of the Workshop on Text Mining at the First SIAM International Conference on Data Mining*, pages 47–54, 2001.

[16] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *TJ. Chem. Phys.*, 21(6):1087–1091, 1953.

[17] O. Nasraoui and R. Krishnapuram. Crisp interpretations of fuzzy and possibilistic clustering algorithms. In *Proceedings of 3rd European Congress on Intelligent Techniques and Soft Computing*, pages 1312–1318, Aachen, Germany, April 1995.

[18] M.F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[19] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[20] K. Rose, E. Gurewitz, and C.G. Fox. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 11(9):589–594, 1990.

[21] K. Rose, E. Gurewitz, and C.G. Fox. Vector quantization by deterministic annealing. *IEEE Trans. Info. Theory*, 38(4):1249–1257, 1992.

[22] M. Teboulle and J. Kogan. A smoothing optimization approach to clustering algorithms with general distance-like functions. *In preparation*.

[23] E. Weiszfeld. Sur le point lequel la some de n points dnnes est minimum. *Tohoku Math. J.*, 43:355–386, 1937.

[24] B. Zhang, M. Hsu, and U. Dayal. K-harmonic means - a data clustering algorithm. Technical Report HPL-1999-124 991029, HP Labs, Palo Alto, CA, 1999.

[25] G. Zhang, B. Kleyner and M. Hsu. A local search approach to k-clustering. *Tech Report HPL-1999-119*, 1999.

On the Benefit of Spectral Projection for Document Clustering

Santosh Vempala*
Mathematics Department and CSAIL, MIT
vempala@math.mit.edu

Grant Wang*
CSAIL, MIT
gjw@theory.csail.mit.edu

Abstract

Spectral algorithms have been applied with much experimental success to the problem of document clustering. However, their success is not fully explained – why do spectral algorithms perform better than traditional clustering algorithms such as k-means? In this paper, we propose that the benefit of spectral projection is based on two simple properties that aid clustering and give experimental evidence of these properties. The first property is that the spectral projection is a good approximation of the original matrix. The second is that the true clustering becomes more apparent in the spectral projection. We also discuss a connection between the second property and recent theoretical work on learning mixtures of distributions.

1 Introduction

Document clustering is a fundamental problem in information retrieval; it has been used to organize results of a query [CKPT92] and produce summaries of documents [HKH+01]. In this problem, an algorithm is given a set of documents. The algorithm's task is to partition the set of documents so that each subset comprises documents about the same topic. Spectral clustering algorithms have shown impressive experimental results in this domain [Bol98, ZDG+01, DHZ+01, Dhi01, ZK02, XLG03, LMO04, CKVW05][1]. A spectral clustering algorithm is one that partitions a data set using the eigenvalues and eigenvectors of a matrix representation of the data. Such a description is vague, but a more precise description cannot be given because spectral methods have been applied to clustering in many different ways. The varied uses of spectral methods and their common experimental success motivate the following question: do the proposed spectral algorithms share a common benefit for document clustering?

In this paper, we focus on spectral projection and give experimental evidence that the answer is yes; clustering algorithms that use the spectral projection of a document-term matrix do indeed share a common benefit in the form of two properties: *approximation* and *distinguishability*. The second property explains why traditional clustering algorithms such as k-means perform better on the spectral projection of a document-term matrix than on the document-term matrix itself. The purpose of this work is not to show that spectral methods *do* perform better than or can improve traditional clustering methods — this is well known. The goal of this paper is to explain the benefit of spectral projection by means of two specific properties.

The first property, approximation, means that the spectral projection of a document-term matrix remains close to the original document-term matrix. Thus, spectral projection reduces the dimension of the data, which speeds up clustering algorithms running on the data, while not incurring too much error. The experiments and results are described in Section 3.

The second property, distinguishability, is that clusters are more clearly demarcated after spectral projection. In particular, for natural (and commonly used) definitions of distance and similarity for documents, we give experimental evidence that inter-cluster distance/similarity is substantially more distinguishable from intra-cluster distance/similarity after spectral projection. Before spectral projection, these two quantities are indistinguishable. This explains why clustering algorithms that work solely on the basis of pairwise distances/similarities perform better on the projection compared to the original document-term matrix. This property of spectral projection coincides with recent theoretical work on spectral algorithms for learning mixtures of distributions [VW04, KSV04]. We explain this connection and the experimental results in Section 4.

1.1 Previous work

An early use of spectral techniques in information retrieval was the work of [DDL+90], which showed that a low rank approximation to a document-term matrix improved precision and recall for queries. Much work on spectral techniques

*Both authors are supported in part by NSF Award CCR-0312339.

[1] For instance, see http://eigencluster.csail.mit.edu

for clustering followed in the text domain, including [Bol98, SS97, ZDG+01, DHZ+01, Dhi01, ZK02, XLG03, LMO04, CKVW05]. Empirical success of spectral methods has also appeared in other domains such as image segmentation [SM00, NJW01]. Theoretical work explaining the success of spectral methods in clustering has focused on proposing generative models for data and showing that particular spectral algorithms perform well [PRTV00, AFK+01, KVV04, McS01, VW04, KSV04]; An exception is the work of [KVV04] that gives a worst-case approximation guarantee for recursive spectral partitioning. The work done in [DM01] and [SS97] is most similar to the work in this paper. In [DM01], Dhillon and Modha perform similar experiments investigating the approximation and distinguishability properties of *concept decomposition*, a technique distinct from SVD that represents a document-term matrix in a lower-dimensional space. In [SS97], Schutze and Silverstein empirically study the effect of projections on clustering speed and efficacy. They conclude that spectral projection significantly speeds up clustering but has no effect on efficacy. However, their measure of efficacy is respect to a fixed clustering algorithm. Although a fixed clustering algorithm may not see improved performance after spectral projection, other clustering algorithms may benefit from spectral projection. Our work suggests that spectral projection has a positive effect on efficacy; we measure this effect independent of any particular clustering algorithm.

2 Preliminaries and Notation

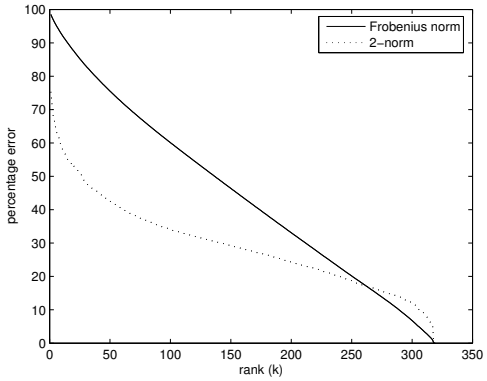
Let $A \in \mathbb{R}^{m \times n}$ be a document-term matrix; the rows of A are the documents and the columns of A are the terms. Let $A = U\Sigma V^T$ be the singular value decomposition of A, where Σ is a diagonal matrix and U and V are orthonormal matrices. The diagonal entries of Σ are the singular values of A, and the columns of U and V are the left and right singular vectors, respectively. Let Σ_k be a diagonal matrix with the first k singular values down the diagonal, and let $A_k = U\Sigma_k V^T$. We also refer to A_k as the spectral projection of A of rank k, since it is the projection of A onto the top k right singular vectors. A_k is the best rank k approximation to A in the following sense:

THEOREM 2.1. (ECKHART AND YOUNG [EY39])

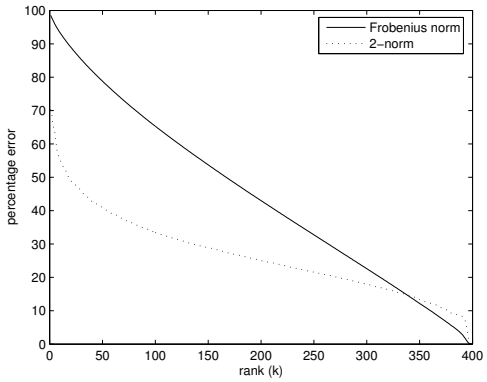
$$A_k = \mathit{argmin}_{B:rank(B)\leq k} ||A - B||_F$$

$$A_k = \mathit{argmin}_{B:rank(B)\leq k} ||A - B||_2$$

Recall that the 2-norm of a matrix is $\max_{v:||v||=1} ||Av||_2$ and the Frobenius norm is the sum of the squares of the matrix entries. More about the SVD and its properties can be found e.g., in [GL96].



(a) `alt.atheism`



(b) `rec.sport.hockey`

Figure 1: Percentage error vs. rank

3 Approximation

Reducing the dimensionality of data can speed up clustering algorithms computing over the data. However, a low-dimensional representation of high-dimensional data typically incurs some error. Thus, for dimension reduction to be effective, it is desirable that the error is small. We conducted an experiment to measure the error incurred by spectral projection of document-term matrices. From Theorem 2.1, we know that the spectral projection A_k is the *best* approximation to A for the Frobenius norm and 2-norm. The results show that this best approximation is indeed a good one — spectral projection to low dimensions introduces manageable error, even when the initial dimension is very high.

The data set we used was the 20 Newsgroups data set [Lan], which consists of 20,000 articles from 20

data set	rank (k) 1	2	4	8	16	32	64	128
alt.atheism	98.5%	97.7%	96.2%	93.7%	89.5%	82.3%	70.9%	52.4%
comp.graphics	98.7%	98.1%	97.1%	95.5%	92.5%	87.4%	78.7%	63.5%
comp.os.ms-windows.misc	98.7%	98.0%	96.8%	94.7%	91.1%	85.5%	76.2%	60.5%
comp.sys.ibm.pc.hardware	98.5%	97.6%	96.4%	94.4%	91.0%	85.5%	76.2%	60.4%
comp.sys.mac.hardware	98.8%	98.1%	96.8%	94.7%	91.2%	85.2%	75.1%	58.5%
comp.windows.x	98.9%	98.3%	97.2%	95.5%	92.5%	87.2%	78.2%	62.9%
misc.forsale	98.9%	98.2%	97.1%	95.3%	92.2%	87.0%	78.0%	62.3%
rec.autos	98.8%	98.0%	96.6%	94.5%	91.0%	85.3%	75.5%	59.6%
rec.motorcycles	98.9%	98.2%	97.0%	94.8%	91.2%	85.0%	74.6%	58.1%
rec.sport.baseball	98.7%	97.9%	96.5%	94.3%	90.8%	84.6%	74.4%	58.1%
rec.sport.hockey	98.6%	97.9%	96.5%	94.2%	90.6%	84.6%	74.7%	58.7%
sci.crypt	98.6%	97.9%	96.8%	94.6%	91.1%	85.2%	75.1%	58.8%
sci.electronics	98.9%	98.1%	96.7%	94.6%	91.1%	85.3%	75.2%	58.5%
sci.med	99.0%	98.2%	97.0%	94.8%	91.4%	85.7%	76.0%	59.9%
sci.space	98.7%	98.0%	96.7%	94.6%	91.0%	84.9%	75.2%	59.3%
soc.religion.christian	98.5%	97.8%	96.6%	94.6%	91.3%	85.9%	77.2%	62.8%
talk.politics.guns	98.5%	97.7%	96.4%	94.3%	90.7%	84.7%	74.5%	58.0%
talk.politics.mideast	98.4%	97.3%	95.7%	93.0%	88.9%	82.5%	72.4%	56.2%
talk.politics.misc	98.3%	97.1%	95.5%	93.0%	89.0%	82.2%	70.8%	51.9%
talk.religion.misc	98.5%	97.5%	95.8%	92.9%	88.3%	80.3%	66.6%	43.5%

Table 1: Percentage error (Frobenius norm) vs. rank for all 20 newsgroups

Usenet newsgroups. Each newsgroup contains roughly 1,000 articles divided into a training subset and a test subset. For each of the newsgroups, we constructed the document-term matrix for all the articles in the test subset of the newsgroup. We used the following common pre-processing steps.

- Stemming (using Porter's algorithm [Por80]).
- Removal of terms that appear too infrequently (less than two times).
- TF*IDF normalization.
- Normalization of term vectors to have Euclidean length one.

The document-term matrix was roughly 400 by $4,000$ for each newsgroup and contained roughly $24,000$ non-zero entries. The singular value decomposition was computed for each document-term matrix, and the percentage error

$$\frac{||A - A_k||}{||A||}$$

was measured for values of the rank (k) from 1 to 250 for both the Frobenius norm and the 2-norm.

The plots of percentage error vs. k for the newsgroups `alt.atheism` and `rec.sport.hockey` appear in Figures 1(a) and 1(b). The percentage error for the Frobenius norm for all 20 newsgroups is shown in Table 1 for k (rank) in powers of two. Table 2 contains the same information but for the 2-norm.

In Figures 1(a) and 1(b), the percentage error drops to 0% when $k = 319$ and $k = 399$, respectively. This is because `alt.atheism` consists of 319 documents and `rec.sport.hockey` consists of 399 documents. Thus, spectral projection to 319 and 399 dimensions results in perfect reconstructions of the document-term matrices for `alt.atheism` and `rec.sport.hockey`, respectively. Both plots show that the error for the Frobenius norm drops off roughly linearly, whereas the error for the 2-norm drops off more quickly initially. For high dimensions, though, the percentage error for the Frobenius norm is lower than that of the 2-norm.

The best low-rank approximations with respect to the Frobenius norm and 2-norm are indeed good approximations. When the dimensionality of the data was reduced from roughly 400 to 128, the percentage error for the Frobenius norm was less than 60% for all but two newsgroups (see Table 1). For the 2-norm (Table 2), we have even better results: when the dimensionality was reduced to just 32, the percentage error for all but three newsgroups was under 50%. It is interesting to note that all newsgroups behaved

data set	rank (k) 1	2	4	8	16	32	64	128
alt.atheism	75.4%	71.6%	66.7%	61.2%	54.9%	47.3%	39.5%	31.3%
comp.graphics	69.6%	63.7%	57.6%	54.0%	50.9%	45.4%	40.2%	33.5%
comp.os.ms-windows.misc	74.1%	71.2%	65.0%	59.1%	53.0%	45.7%	39.8%	33.0%
comp.sys.ibm.pc.hardware	72.3%	64.8%	58.2%	52.9%	47.8%	42.1%	36.6%	30.3%
comp.sys.mac.hardware	74.4%	70.6%	67.9%	60.9%	55.1%	49.0%	42.2%	33.8%
comp.windows.x	73.2%	70.9%	64.8%	59.8%	55.0%	49.5%	43.5%	36.2%
misc.forsale	78.7%	74.2%	64.7%	61.1%	55.1%	49.2%	43.7%	36.5%
rec.autos	82.4%	77.2%	69.7%	63.9%	55.4%	49.3%	42.8%	34.5%
rec.motorcycles	76.8%	72.2%	69.4%	64.9%	57.8%	52.2%	43.8%	34.5%
rec.sport.baseball	74.7%	71.7%	67.7%	58.2%	52.3%	47.4%	40.0%	31.6%
rec.sport.hockey	70.3%	69.2%	64.7%	57.1%	51.4%	45.1%	38.4%	30.8%
sci.crypt	66.1%	63.2%	61.5%	57.1%	50.6%	45.1%	39.1%	30.8%
sci.electronics	87.1%	81.2%	71.1%	63.7%	58.6%	51.7%	45.4%	36.1%
sci.med	83.7%	80.2%	71.7%	64.5%	58.1%	51.7%	45.4%	36.3%
sci.space	74.0%	73.5%	65.5%	60.3%	55.4%	47.8%	40.8%	33.0%
soc.religion.christian	66.6%	62.2%	56.6%	53.2%	47.4%	41.3%	35.4%	29.4%
talk.politics.guns	71.5%	65.4%	59.1%	54.1%	49.5%	43.9%	37.7%	30.3%
talk.politics.mideast	78.8%	74.9%	65.1%	56.2%	49.6%	42.5%	35.3%	28.5%
talk.politics.misc	85.3%	69.5%	60.9%	55.7%	50.2%	43.7%	37.3%	29.7%
talk.religion.misc	77.7%	76.8%	69.4%	62.5%	55.2%	49.6%	40.9%	31.6%

Table 2: Percentage error (2-norm) vs. rank for all 20 newsgroups

similarly with respect to percentage error versus rank, despite the difference in content in the newsgroups. For most values of k (rank), the maximum percentage error difference between any two newsgroups was roughly 10%.

4 Distinguishability

Many document clustering algorithms are based on a pairwise distance or similarity function. The distance function maps two documents to any non-negative number; the larger the distance, the more unlike the two documents are. On the other hand, a similarity function maps two documents to the interval $[0, 1]$; the closer to 1, the more *like* the two documents are. Many different distance and similarity functions have been used when documents are represented as term vectors. We consider two commonly used candidates for a distance function and similarity function. The distance between two term vectors u, v will be their Euclidean distance:

$$d(u, v) = ||u - v||.$$

For similarity, we use the Gaussian kernel:

$$s(u, v) = e^{-||u-v||^2}.$$

We give experimental evidence that under these measures, intra-cluster distance/similarity becomes substantially more distinguishable from inter-cluster distance/similarity after spectral projection.

4.1 Experiment and results The data set we used for this experiment was also the 20 Newsgroup data set [Lan]. The general experimental setup was as follows: we formed the document-term matrix A of a random sample of the articles from k random newsgroups, where k ranged from 2 to 10. We measured the inter/intra newsgroup distance/similarity of these documents. We then computed A_k, the spectral projection of A to k dimensions, and measured the inter/intra newsgroup distance/similarity in the new representation. The exact experimental details follow.

We chose 10 random sets of k different newsgroups. For a random set of k newsgroups, we chose 10 random samples $S_1 \ldots, S_{10}$. Each random sample consisted of 50 articles from each of the k newsgroups. Clearly, the articles in each random sample, $S_1 \ldots S_{10}$ can be partitioned into k clusters according to the newsgroup to which they belong (we will refer to the collection of documents from the same newsgroup as a cluster). We then computed the document-term matrix A for each random sample S_i, using the pre-processing steps described in Section 3. The singular value decomposition of A was computed and we formed the spectral projection A_k.

Each row vector in A_k was then normalized so that its Euclidean length was 1 (note that in A, each row vector was also normalized to 1).

For the distance function $d(u,v) = ||u-v||$, we measured the following two quantities both before and after projection:

- Average distance between cluster means.
- Average distance between cluster mean and a vector from the same cluster.

The cluster mean is simply the vector that is the average of all the term vectors from a newsgroup. Table 3 shows the results averaged together for each value of k. The key property is that the ratio of average distance between cluster means, $d(\mu_i, \mu_j)$, to average distance between a cluster mean and a term vector from the same cluster, $d(u,\mu)$, increases to the point that the two quantities become distinguishable. The factor by which this ratio increases after projection is the magnification factor (the last column in the table). Consider a situation in which the ratio $d(\mu_i,\mu_j)/d(u,\mu)$ is at least 2 for every pair of clusters; then we can put a ball around each cluster mean so that 1) the balls do not intersect each other, and 2) each ball contains only the term vectors from a single cluster. In our experiments, the ratio was at least 2 only for $k < 6$. Nevertheless, the magnification factor shows that it is easier to cluster the points in balls after spectral projection (most points can be clustered).

For the similarity function $s(u,v) = e^{-||u-v||^2}$, we measured the following quantities both before and after spectral projection:

- Average intra-cluster similarity, $s(C_i, C_i)$.
- Average inter-cluster similarity, $s(C_i, C_j)$.

The results appear in Table 4. The ratio between the average intra-cluster similarity and average inter-cluster similarity $(s(C_i,C_i)/s(C_i,C_j))$ is the key quantity. Before projection (4th column), the ratio is roughly 1. After projection (7th column), the ratio approaches 2 – meaning that, on average, documents from the same cluster are twice as similar as documents from different clusters. The magnification factor, the last column in the table, shows the increase in this ratio after projection. It is interesting that the magnification factor stays roughly constant as k increases. This is not true for distance; the magnification factor drops as k increases (see Table 3).

4.2 Connection to learning mixtures of distributions Measuring the effect of spectral projection on inter-cluster distance and intra-cluster distance was motivated by recent theoretical work on the problem of learning mixtures of distributions. In this problem, we are given a random sample from a mixture of k distributions $F_1, \ldots, F_k$ with mixing weights $w_1, \ldots, w_k$ that sum to 1, i.e. $\sum_i w_i = 1$. A random sample is generated from the mixture by first choosing a distribution F_i according to its mixing weight w_i, and then choosing a random sample according to F_i. We say that an algorithm *learns a mixture of distributions* if it can classify each random sample according to the distribution from which it was sampled.

Recent work has shown that spectral algorithms can provably learn mixtures of distributions for some important special cases. In [VW04], a spectral algorithm is given that correctly classifies a random sample from a mixture of spherical Gaussians, assuming a weak separation between the means of the Gaussians. This was generalized in [KSV04] using a stronger separation condition to mixtures of logconcave distributions, a class of distributions which includes general Gaussians. The key insight to both algorithms was relating the spectral projection subspace V, spanned by the top k right singular vectors of the sample matrix A (the rows of A are simply the sample points), to the subspace W spanned by the mean vectors of the distributions $F_1, \ldots, F_k$. In [VW04], it is shown that, in expectation, V is the same subspace as W, but only when the distributions are spherical. This is not true when the distributions are arbitrary, but an approximate theorem holds.

THEOREM 4.1. *([KSV04]) Let $S = S_1 \cup S_2 \ldots \cup S_k$ be a sample from a mixture with k distributions such that S_i is from the ith distribution F_i and let V be the k-dimensional SVD subspace of S. For each i, let μ_i be the mean of S_i and σ_i^2 be the maximum variance of S_i along any direction in V. Then,*

$$\sum_{i=1}^{k} |S_i| d(\mu_i, V)^2 \leq k \sum_{i=1}^{k} |S_i| \sigma_i^2$$

where $d(\mu_i, V)$ is the orthogonal distance between μ_i and V.

The theorem upper bounds the average distance between means and their projection to V. Thus a lower bound on the distance between any two means (i.e., $d(\mu_i,\mu_j)$), gives a lower bound on the distance between the means of any two projected distributions by the triangle inequality. Roughly speaking, the theorem says that, on average, the inter-mean distances do not decrease much.

On the other hand, it is reasonable to expect that the distance between a random point and the mean

	before projection			after projection			magnification
k	$d(\mu_i, \mu_j)$	$d(u, \mu)$	$\frac{d(\mu_i,\mu_j)}{d(u,\mu)}$	$d(\mu_i, \mu_j)$	$d(u, \mu)$	$\frac{d(\mu_i,\mu_j)}{d(u,\mu)}$	factor
2	0.2902	0.9690	0.2995	0.8985	0.2197	4.0889	13.6524
3	0.2858	0.9701	0.2945	0.8222	0.3269	2.5151	8.5402
4	0.2842	0.9714	0.2925	0.8259	0.3860	2.1396	7.3149
5	0.2891	0.9703	0.2980	0.9124	0.4122	2.2137	7.4285
6	0.2876	0.9707	0.2963	0.8325	0.4439	1.8755	6.3297
7	0.2876	0.9717	0.2960	0.7731	0.4627	1.6709	5.6449
8	0.2863	0.9719	0.2946	0.8376	0.4953	1.6909	5.7396
9	0.2863	0.9714	0.2947	0.8716	0.5202	1.6756	5.6858
10	0.2847	0.9725	0.2928	0.8348	0.5374	1.5534	5.3053

Table 3: Average distances before and after spectral projection to k dimensions

	before projection			after projection			magnification
k	$s(C_i, C_i)$	$s(C_i, C_j)$	$\frac{s(C_i,C_i)}{s(C_i,C_j)}$	$s(C_i, C_i)$	$s(C_i, C_j)$	$\frac{s(C_i,C_i)}{s(C_i,C_j)}$	factor
2	0.1485	0.1407	1.0556	0.8709	0.4460	1.9530	1.8501
3	0.1479	0.1404	1.0534	0.7836	0.4450	1.7609	1.6716
4	0.1471	0.1398	1.0524	0.7282	0.4025	1.8093	1.7192
5	0.1478	0.1400	1.0553	0.7007	0.3643	1.9235	1.8227
6	0.1475	0.1399	1.0543	0.6686	0.3444	1.9410	1.8410
7	0.1470	0.1393	1.0546	0.6513	0.3334	1.9533	1.8522
8	0.1468	0.1394	1.0532	0.6144	0.3099	1.9825	1.8824
9	0.1471	0.1396	1.0535	0.5889	0.3061	1.9271	1.8292
10	0.1464	0.1391	1.0525	0.5695	0.2948	1.9359	1.8393

Table 4: Average intra-cluster and inter-cluster similarity before and after spectral projection to k dimensions

of its distribution shrinks when projected to a lower dimension. As the ratio of the inter-mean distance to the distance between a random sample and the mean of its distribution increases, clustering based on pairwise distances becomes more effective. The assumption of Gaussian (or logconcave) distributions with separated means is sufficient to be able to classify the entire sample with high probability.

Table 3 suggests that this is indeed occurring for document clusters. Note that the average distance between means before and after spectral projection is roughly the same. This corresponds to the lower bound on the distance between means from Theorem 4.1. Meanwhile, the distance between a document and the mean of its cluster drops considerably. It is interesting that text corpora, which we have no reason to believe are mixtures of logconcave distributions benefit from spectral projection.

5 Conclusion

We have described two properties, approximation and distinguishability, that aid clustering and have given experimental evidence that the spectral projection of a document-term matrix has these properties. Besides more extensive experiments, several other questions also arise from this work: To what extent do the preprocessing steps aid the inter-cluster/intra-cluster distinguishability? What other definitions of distance and similarity can spectral projection magnify? The connection to the problem of learning mixtures of distribution also suggests a data-driven method of designing algorithms for document clustering. In this method, properties of real-world data are first experimentally verified and then an algorithm that provably works on data with such properties is designed. With this method, one might avoid having to use generative models (which may not be accurate) to show that an algorithm works on real-world data.

References

[AFK+01] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of*

the Thirty-Third Annual ACM Symposium on Theory of Computing, pages 619–626, 2001.

[Bol98] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[CKPT92] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[CKVW05] D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. In *Proceedings of the 24th Annual ACM Symposium on Principle of Database Systems*, page To Appear, 2005.

[DDL+90] S.C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[Dhi01] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[DHZ+01] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. Spectral min-max cut for graph partitioning and data clustering. In *Proceedings of the First IEEE International Conference on Data Mining*, pages 107–114, 2001.

[DM01] I.S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[EY39] C. Eckhart and G. Young. A principal axis transformation for non-hermitian matrices. *Bulletin of the American Mathematical Society*, 45:118–121, 1939.

[GL96] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins Press, third edition, 1996.

[HKH+01] V. Hatzivassiloglou, J. Klavans, M. Holcombe, R. Barzilay, M. Kan, and K. McKeown. Simfinder: A flexible clustering tool for summarization. In *NAACL Workshop on Automatic Summarization*, pages 41–49, 2001.

[KSV04] R. Kannan, H. Salmasian, and S. Vempala. The spectral method for mixture models. In *Electronic Colloquium on Computational Complexity*, pages Report TR04–067, 2004.

[KVV04] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[Lan] K. Lang. 20 newsgroups data set. `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`.

[LMO04] T. Li, S. Ma, and M. Ogihara. Document clustering via adaptive subspace iteration. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 218–225, 2004.

[McS01] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd IEEE Annual Symposium on Foundations of Computer Science*, pages 529–537, 2001.

[NJW01] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems*, pages 849–856, 2001.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[PRTV00] C. Papdimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabalistic analysis. *Journal of Computer and System Sciences*, 61:217–235, 2000.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SS97] H. Schutze and C. Silverstein. Projections for efficient document clustering. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81, 1997.

[VW04] S. Vempala and G. Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.

[XLG03] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 267–273, 2003.

[ZDG+01] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Neural Information Processing Systems*, pages 1057–1064, 2001.

[ZK02] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 515–524, 2002.

Improving Self Organizing Map Performance for Network Intrusion Detection

Stefano Zanero*

Abstract

The continuous evolution of the types of attacks against computer networks suggests a paradigmatic shift from misuse based intrusion detection system to anomaly based systems. Unsupervised learning algorithms are natural candidates for this task, but while they have been successfully applied in host-based intrusion detection, network-based applications are more difficult, for a variety of reasons, including performance. We propose an architecture which implements a network-based, anomaly based intrusion detection system, which uses unsupervised learning algorithms. In this paper we describe the improvements and modifications needed in order to increase the throughput of a Self Organizing Map algorithm and make it able to handle high dimensional input data at a rate suitable for Intrusion Detection purposes at network speed.

1 Introduction and motivations

The continuous evolution of the threats against computer networks requires a paradigmatic shift from misuse based intrusion detection system to anomaly based systems. The "misuse detection" approach, which tries to define what constitutes an attack in order to detect it directly, has been widely successful. Most modern intrusion detection tools are misuse based, but they are increasingly showing the limits of this paradigm.

Whenever a new attack is discovered, the knowledge base of misuse IDSs must be updated in order to keep the systems effective. In addition, there is also an unknown number of discovered, but undisclosed, vulnerabilities (the so called "zero-days") that are not available to the experts for analysis and inclusion in the knowledge base [1]. Most attacks are also polymorph, and skilled attackers can exploit this polymorphism to evade detection [2, 3].

Since, by their own nature, Intrusion Detection Systems are intended to be a complementary security measure, which can detect the failures of other measures, the inability to detect unknown attacks or new ways to exploit an old vulnerability is an unacceptable limitation. The obvious solution seems then to implement an anomaly detection approach, modeling what is *normal* instead than what is *anomalous*. This is surprisingly similar to the earliest conceptions of what an IDS should do [4].

However, while a number of host based intrusion detection systems have been proposed and implemented, both in literature and in practice, network based anomaly detection is still an open field for research. In a previous work [5], we proposed a novel architecture for applying unsupervised learning and data mining techniques to a network based IDS. Unsupervised learning techniques are natural candidates for this type of task, but while they have been successfully applied in host based intrusion detection [6], their application to network based systems is still troublesome, mainly due to the problems of input selection, data dimensionality and throughput.

In this paper we describe the improvements and modifications we applied to a Self Organizing Map algorithm in order to increase its throughput to handle high dimensional data at a speed suitable for Network Intrusion Detection purposes. We describe various heuristics that can be used, and their effect on the accuracy of the algorithms. We also propose some performance tests that demonstrate that our heuristics do not diminish the overall effectiveness of the IDS.

The remainder of the paper is organized as follows: in Section 2 we describe the proposed architecture of our IDS; in Section 3 we describe how the curse of dimensionality affects the performance of the first stage of the architecture; in Section 4 we describe our heuristics and the associated performance improvements; in Section 5 we discuss the problem of significance of euclidean metrics in our high-dimensional space; in Section 6 we report on the experimental validation results for the improved algorithms; finally, in Section 7 we draw our conclusions and outline some future work.

2 The proposed architecture

In [5] we proposed an innovative architecture for a network based anomaly detection system, based on unsupervised learning algorithms. We chose to focus on this class of algorithms because they exhibit properties that are particularly well suited for anomaly detection, in particular the ability of detecting outliers and of building a model of "normality" without the need of

*Dipartimento di Elettronica e Informazione - Politecnico di Milano. E-mail: zanero@elet.polimi.it

a priori knowledge input.

If we think of the network packet flow as a stream of observations (packets), then anomaly detection is an instance of the outlier detection problem. An outlier is classically defined as follows: "an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [7]. Here, the "different mechanism" is an attacker who is trying to subvert a network service.

However, outlier detection on the flow of TCP/IP packets is not an easy task. Each packet has a variable dimension (which over an Ethernet link for instance varies between 20 and 1500 bytes), while unsupervised learning algorithms work well on multivariate data with a fixed number of features. The network and transport layer headers can be easily normalized and translated to a fixed number of features. It is important to note, however, that in the case of connection-oriented protocols (most notably TCP) the transport layer headers may need inter-correlation in order to be fully deciphered.

On the contrary, the data carried by the packet (the payload) cannot be easily translated into a fixed set of features, since each different application layer protocol would require its own set of features, increasing complexity and decreasing generality. In addition it would require a full reconstruction of traffic sessions, which would expose the IDS to reconstruction problems, possibly leading to attack windows [2].

In addition, the computational complexity of unsupervised learning algorithms scales up steeply with the number of considered features, and the detection capabilities decrease correspondingly (this is known as *the curse of dimensionality*). Only a few algorithms can be optimized to treat data with many thousands of dimensions, but only in the case that they are sparse (for instance, a word/document incidence matrix in a document classification and retrieval problem [8]). We are instead dealing with dense data.

Most of the existing researches on the use of unsupervised learning algorithms for network intrusion detection purposes solve this problem by discarding the payload and retaining only the information in the packet header (e.g. [9, 10, 11, 12]) . This is clearly not an optimal solution, since it leads to an unacceptable information loss: most attacks, in fact, are detectable only by analyzing the payload of a packet, not the headers alone. These algorithms show nevertheless interesting, albeit obviously limited, intrusion detection properties.

In order to solve this problem, we developed the concept of a two-tier intrusion detection system (shown in Figure 1), which allows us to retain at least part of the information related to the payload content. In the first tier of the system, an unsupervised clustering algorithm classifies the payload of the packets, observing one packet at a time and "compressing" it into a single byte of information. This classification can be added to the information decoded from the packet header (or to a subset of this information), and passed on to the second tier. On most networks, the traffic belongs to a relatively small number of services and protocols, regularly used, and a good learning algorithm can map it onto a relatively small number of clusters.

The second tier algorithm instead takes into consideration the anomalies, both in each single packet and in a sequence of packets. It is worth noting that most of the solutions proposed by previous researchers in order to analyze the sequence of data extracted by the packet headers could be used as a second tier algorithm, complemented by our first tier of unsupervised clustering.

3 The curse of dimensionality in the first stage

The first tier algorithm receives in input the payload of a TCP or UDP over IP packet: on an Ethernet segment this means up to 1460 bytes of data. Its role is to classify such information in a "sensible" way, which means that it should, in principle, keep as much information as possible for the second tier algorithm about the "similarity" between packets. Obviously, since our end goal is to detect intrusions, the classification should show mainly the property to separate packets with anomalous or malformed payload from normal packets, and should also divide the payloads reflecting the divisions between protocols as closely as possible. "The grouping of similar objects from a given set of inputs" [13] is obviously a typical clustering problem.

We have shown [5] that a Self Organizing Map (SOM) algorithm [14] is indeed able to sensibly cluster payload data, discovering interesting information in an unsupervised manner. Our research is the first attempt to cluster packet payloads to obtain meaningful results. A previous research showed that neural algorithms can recognize protocols automatically [15], while another paper later independently confirmed that the payload of the packets indeed shows some interesting statistical properties [16].

There are multiple reasons for choosing a SOM for this purpose. A SOM is a hard-competitive, neural based algorithm, which is capable to map a high-dimensional input space onto a low-dimensional (usually bi-dimensional) neuron space. The algorithm is robust with regard to the choice of the number of classes to divide the data into, and is also resistant to the presence of outliers in the training data, which is a desirable property: in real-world situations, the training data could already contain attacks or anomalies and the algorithm must be capable of learning regular patterns

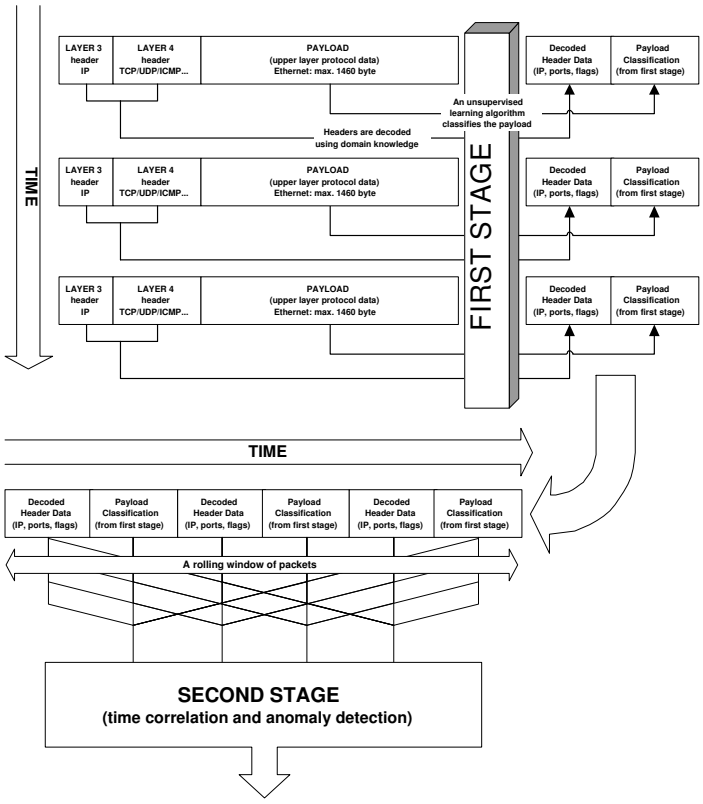


Figure 1: The two-tier architecture of the IDS, comprising a tier of unsupervised clustering followed by a tier of outlier detection.

out of a "dirty" training set. In addition, we have compared various algorithms and shown that the SOM had the best performance trade-off between speed and classification quality.

Unluckily, the curse of dimensionality hits heavily against the first tier. The second tier is not a problem, being required to handle a multivariate time series with a comparably small number of features (up to a maximum of about 30). There are alternative algorithms for clustering which are much faster in the learning phase than SOM; for example, the well known K-means algorithm is one of the fastest. But at runtime even K-means is not more efficient than a SOM, so we cannot solve the problem by choosing a different algorithm.

A traditional approach to the problem would use dimension reduction techniques such as dimension scaling algorithms [17] or Principal Component Analysis [18]. But our experiments demonstrated that they are quite ineffective in this particular situation, since by their nature they tend to "compress" outliers onto normal data, and this is exactly the opposite of what we want to achieve.

4 Improving the performance of the SOM algorithm

Since no alternative solution was viable, we developed various approximate techniques to speed up the SOM algorithm. The reference machine for our tests is an Athlon-XP 3200 based computer with 1 GB of DDR RAM, running GNU/Linux with a 2.6 kernel. All the tests, unless otherwise stated, refer to a SOM with square topology, and a size in the space of neurons of 10×10. The test are conducted on TCP packets, as they constitute over 85% of Internet traffic.

The data used for training and testing the prototype are subsets of the "1998 DARPA IDS Evaluation dataset", which is well commented and described by a master's thesis [19]. In [20] there is a detailed analysis of the shortcomings of this traffic sample set, and we share many of the author's observations: no detail is available on the generation methods, there is no evidence that the traffic is actually realistic, and that spurious packets, so common on the Internet today, are not taken into account. On the other hand, whenever we need to test the capability of our prototype of detecting attacks mixed in background data, we need to do this under test condi-

tions, with clearly labeled background data, and in spite of its shortcomings the DARPA dataset fulfills this function very well. However, we positively validated most of our results using also smaller dumps collected on our own internal network.

As we can see from the first line of values in Table 1, the throughput of an implementation of the Kohonen algorithm on our hardware and software configuration is on average of 3400 packets per second, which is not acceptable for an IDS monitoring a modern network.

We tried to develop heuristics for speeding up the computation, introducing minimal errors in the classification. The idea behind our heuristic is simple. Let N be the number of classes, and d the number of dimensions of the data. At runtime, the SOM algorithm consists simply of N evaluations of the distance function: in our test implementation, an euclidean distance function over d dimensions. Since the number of computations is $N \cdot d$, in order to speed up the computation we can try to reduce d by applying any dimensionality reduction technique: this, as we said before, cannot be done meaningfully via dimensionality reduction techniques. However, since just a few packets contain a high number of bytes of payload, we can try to use just the first $d' < d$ dimensions. Further experimental evaluation would then of course be required in order to understand if the "reduced" payloads carry the same information value as the complete packets.

If we do not want to reduce d, we must try to reduce the number of evaluations N. A way to do this is to pre-compute a grouping of the N centroids of the classes in $K < N$ super-clusters, and then to select the winning neuron in a two-step procedure. First, we determine which of the super-clusters the observation belongs to; and then we evaluate the distance function just over the $N' < N$ neurons belonging to the winning super-cluster. The algorithm is heuristic, since it can happen that the best matching neuron is not in the best matching super-cluster, but as we will see the error rate is very low. Obviously the best performance gain with this heuristic happens if each of the K super-clusters is formed by $\sim N/K$ neurons, since the average number of computations becomes $d \cdot (K + N/K)$ which has a minimum for $n = \sqrt{N}$. If the clusters are not balanced the worst case computational cost is higher, and this leads to a lower overall throughput. For smaller values of K the algorithm would be on average slower, and the error rate statistically would be slightly lower.

To form the super-clusters, a first naïve idea would be to exploit the map structure, which tends to keep "close" to each other the neurons which are close in the map space. However, this does not work very well experimentally, probably because of the high dimensionality of the feature space, causing a 35% error rate with $N = 3$, and even 60% with $N = 10$. Thus we resort to a K-means approach.

However, we must overcome two different issues in doing this. A first issue has to do with the nature of K-means, which is inherently initialization dependent, and prone to create very unbalanced cluster. Experimentally, with $N = 100$, using $K \geq 4$ does not create a balanced structure of clusters, unless we correct the randomness of the algorithm. Some authors proposed, in order to eliminate these weaknesses, the "global K-means" algorithm [21], which repeats K-means with all the possible initializations. We use a different and faster approach, by using the algorithm a fixed number m of times, and choosing the distribution in classes which minimizes the average expected number of operations, roughly approximating the probability that an observation falls into the i-th super-cluster as proportional to the fraction N_i/N (where N_i is the number of neurons in the i-th super-cluster). In Table 1 we refer to our variant of the K-means algorithm as "K-means+", and the column labeled "Crossv." reports the parameter m (number of runs of the K-means algorithm).

A second, more difficult issue, is how to deal with the training phase. During the training phase the neurons change their position, so theoretically we should repeat the K-means algorithm once for each training step. We can avoid to do so, and fix an arbitrary update frequency, a number of step after which we will recalculate the position of the centroid. As an additional attempt to reduce the cost of the K-means step, we decided to initialize the position of the K centroids to the same position they held before, even if this could lead the convergence to a local optimum, creating a non-optimal clustering. Our tests showed that in each case the cumulative approximations introduced by the algorithm make the training very unstable, leading to results which are not compatible with the ones obtained by normal training, and in which the properties of outlier resilience and robustness of the SOM algorithm are impaired. We are working to find a way to overcome these problems without sacrificing the throughput gain, but for now, the only way to speedup the throughput is to lower the number of dimensions.

In Table 1 we report the runtime throughput and the error rate of the algorithm, evaluated in packets per second, depending on different combination of the parameters, namely the number of bytes considered for each packet, the usage of an heuristic, the parameter K for the K-means algorithm used in the heuristic and the use of cross-validation repetitions. The results have been validated over multiple "days" of the DARPA dataset.

Max bytes per packet	Heuristics	K	Crossv.	Packets/sec.	Error %
1460	None	-	-	3464.81	-
1460	K-means	10	No	8724.65	0.8
1460	K-means+	5	10	5485.95	0.4
1460	K-means+	10	10	10649.20	0.8
800	None	-	-	4764.11	-
800	K-means+	5	10	9528.26	0.5
800	K-means+	10	10	15407.36	1.0
400	None	-	-	8400.45	-
400	K-means+	5	10	28965.84	0.6
400	K-means+	10	10	30172.65	1.2
200	None	-	-	10494.87	-
200	K-means+	5	10	51724.70	0.8
200	K-means+	10	10	65831.45	2.3

Table 1: Throughput and errors during runtime phase, calculated over multiple runs of the algorithm over different days of the DARPA dataset.

In order to evaluate the results, we refer to a well known study of the statistical properties of Internet traffic [22]. Analyzing the traffic flowing through an Internet Exchange datacenter, they show that approximately 85% of the traffic is constituted by TCP packets, and that a large proportion of TCP packets are 40 bytes long acknowledgments which carry no payload (30% to 40% of the total TCP traffic). Zero-size UDP packets, on the contrary, are almost non-existant. Since the first stage analyzes only packets with a non-null payload, almost 30% of the total traffic on the wire will not even enter it. The average size of a TCP packet is 471 bytes, of a UDP packet 157, and the overall average is approximately 420 bytes. It is also known from theoretical modeling and practical experience that an Ethernet network offers approximately 2/3 of its nominal capacity as its peak capacity. This means that a saturated 10 Mbps Ethernet LAN carries about 2.000 packets per second. Other statistics suggest that this value could be higher, up to 2.500 pps.

From Table 1, we can see that the original SOM algorithm, considering the full payload of 1460 maximum bytes per packet, with no heuristics, operates at a speed that is acceptable for use on a 10 Mb/s Ethernet network, but insufficient for a 100 MB/s network. However, using the K-means algorithm with 10 classes and no cross-validation, we obtain a much higher throughput (more than three times higher than the original one) but also a 0.7% error rate. Introducing K-means+ and crossvalidation, we obtain a better tradeoff between throughput and error rate, improving the former without compromising the latter. A speed of 10.500 packets/second is enough to handle a normal 100 Mbps link (considering also the presence of empty packets).

If necessary, performance could also be improved by reducing the number of bytes of the payload. This could evidently impact heavily against the recognition capabilities of the algorithm.

5 Choosing a meaningful metric

In recent researches, the effect of the curse of dimensionality on the concept of "distance metrics" has been studied in detail. In high dimensional spaces such as the one we are considering, the data becomes very sparse. Recent research results [23, 24] show that in high dimensional spaces the concept of proximity and distance may not be meaningful, even qualitatively.

Let $Dmax_d$ be the maximum distance of a query point to the points in a d-dimensional dataset, and $Dmin_d$ the minimum distance, and let X_d be the random variable describing the data points. It has been shown, under broad conditions, that if

$$lim_{d \to \infty} var \left(\frac{\| X_d \|}{E[\| X_d \|]} \right)$$

then

$$\frac{Dmax_d - Dmin_d}{Dmin_d}$$

This means, plainly, that in high dimensional space the difference between the distance of a query point to the farthest and to the nearest point in the dataset tends to be of a smaller order of magnitude than the minimum distance: in other words, the nearest neighbor identification is unstable and does not give much information.

Most of the hypotheses used in the demonstration do not hold for our type of variables. We have experimentally observed that in our setup the described ef-

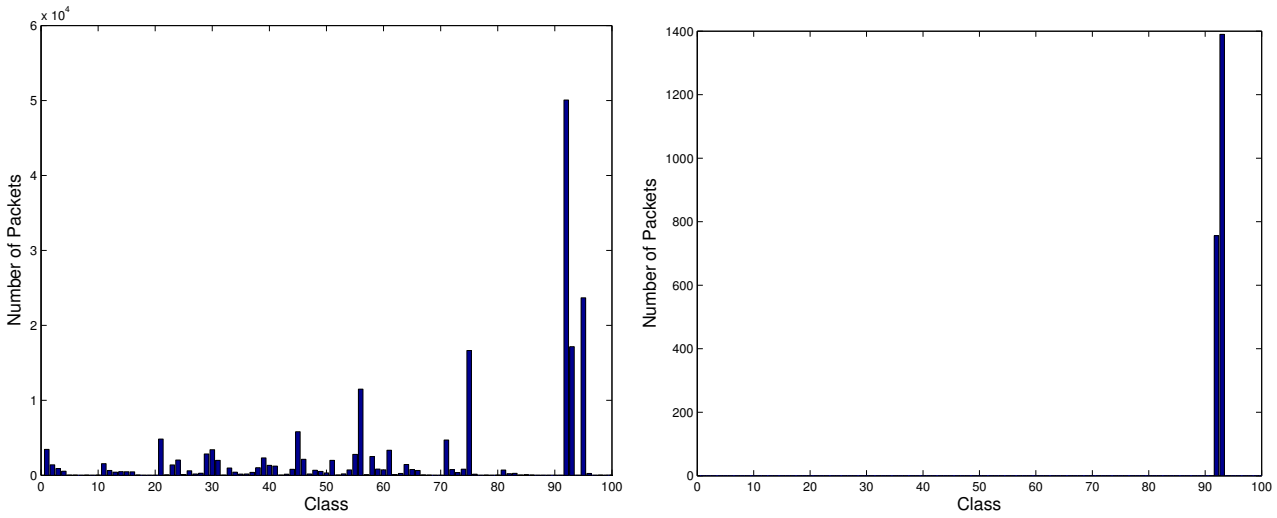


Figure 2: Comparison between the classification of a window of traffic and the traffic destined to port 21/TCP by a 10x10 SOM

fect does not happen: most points are extremely well characterized into dense and compact clusters. In order to better understand if this condition applied to our dataset, we recursively filtered out the most compact clusters and the "farthest" centroids, and analyzed the results, and in each case the difference between $Dmin$ and $Dmax$ was still significant. We thus concluded that the effect observed in the cited articles does not apply to our particular situation, probably because we are working in a compact region where the maximum possible distance between two different points is $\sqrt{255^2 \times 1460}$.

It has also been reported that in high dimensional spaces the L_1 metric, or "Manhattan distance", behaves considerably better than the usual euclidean metric we applied [24]. In [25] distance metrics with a fractional index $f \epsilon (0,1)$ are also proposed.

On the basis of these suggestions, we explored the application of different distance metrics and their effects on the classification of packets. However, in our particular application the use of these alternate distance seems to lump all the data in a few cluster, diminishing the overall recognition capabilities of the algorithm.

6 Recognition capabilities of the modified algorithm

In order to evaluate the recognition capabilities of the new algorithm, we must see if it can still usefully characterize traffic payloads for different protocols, and detect anomalous attack payloads from normal payloads. In Figure 2 we present a demonstration of the recognition capabilities of a 10×10 Self Organizing Map that creates a division of the data in 100 clusters. The network was trained for 10.000 epochs on TCP packet payloads. The histograms represent the number of packets (on y-axis) present in each cluster (on x-axis). Here and in the following, for graphical reasons, the number of packets on y-axis may be differently scaled in the various graphs. In Figure 2 we suppressed from the output the representation of classes 90 and 93, which are the most crowded and less characterized clusters in the classification, for better display.

On the left handside, we can see the classification of a whole window of traffic. On the right handside, we can see how the network classifies the subset of the packets with the destination port set to 21/TCP (FTP service command channel). It can be observed how all the packets fall in a narrow group of classes, demonstrating a strong, unsupervised characterization of the protocol.

In Figure 3 we present the result of the same test using the modified algorithm for runtime recognition. Also in this case, we can see the same strong characterization of the protocol (the similarity between the graphs is striking, but not surprising, since the error rate is approximately 1%). The same situation happens for all the cases we examined and compared, granting that the protocol characterization property is well preserved by the heuristics.

The capability to detect anomalous packets is also preserved. We analyzed how the SOM classifies packets from the attacks contained in the DARPA datasets. For example, let us discuss the case of a race condition and buffer overflow bug in the "ps" command, which is exploited over a perfectly legitimate telnet connection. 99.76 % of packets destined to TCP port 23 fall in classes

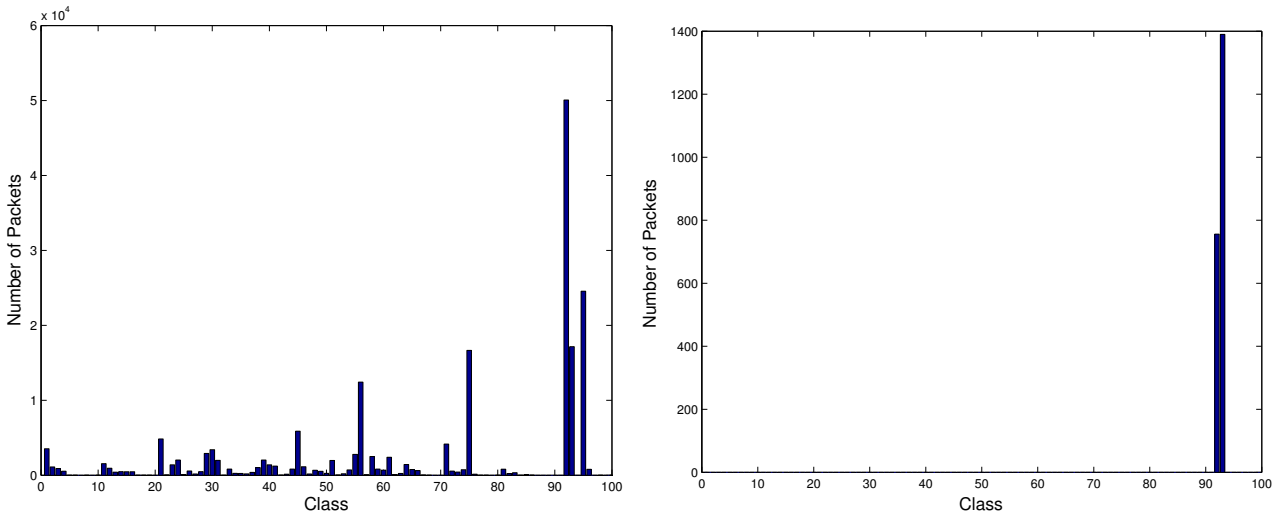


Figure 3: Comparison between the classification of a window of traffic and the traffic destined to port 21/TCP by a 10x10 SOM with our modified algorithm.

91 and 95, and all of them fall between class 90 and 95. The packets containing the attack fall instead in classes 45, 54, 55, 65, 71, 73 and 82, which are not normally associated with DPORT 23. This happens consistently over each instance of the attack.

A similar, albeit less defined, situation happens in the case of a buffer overflow in the "sendmail" MTA daemon. The packets destined to port 25 are less characterized, but over 90 % of them fall into 7 classes. The attack packets fall instead into three different classes that contain less than 3% of the normal packets destined to port 25. This helps us to understand that an important requirement for the second stage detection algorithm will be to keep track of anomaly scores in the recent past.

In order to test the algorithms on newer attacks, we ran the same SOM on the packet captures of some FTP server attacks (a format string wu-ftpd bug exploit, a globbing denial-of-service, a buffer overflow attack). In each case the anomalous payloads fall outside of the narrow characterization we have seen in Figures 2 and 3. The results we presented in [5] are thus preserved while using the modified, faster algorithm.

7 Conclusions and future work

We have described the challenges we met while implementing an innovative model of anomaly based network intrusion detection system, completely based on unsupervised learning techniques. We have described the overall architecture of the system and how the curse of dimensionality requires an appropriate resolution for a working implementation of the first stage of unsupervised clustering. By the means of various techniques, we improved the runtime efficiency of the algorithm, obtaining a throughput rate almost three times higher than the original one, if we are willing to accept a misclassification rate of about 0.8%, and twice as high than the original one with a very small misclassification rate of 0.4%, without truncating the number of the bytes of the payload examined by the algorithm. We have studied how these errors affect the algorithm detection capabilities, and concluded that our heuristically modified implementation works as well as the original version of the SOM. Having thus solved most of the challenges for the design of the first tier, our future work will focus on the choice and implementation of the second tier of learning, and on the empirical evaluation of the IDS under practical workloads.

Acknowledgments

This work was partially supported by the Italian FIRB Project "Performance evaluation for complex systems". We need to thank prof. Sergio M. Savaresi, prof. Salvatore J. Stolfo and the colleagues Giuliano Casale and Roberto Turrin for their precious suggestions for improvement over our previous work. We also need to thank warmly our student Matteo F. Zazzetta for his invaluable support in software development and lab testing, and the anonymous reviewers for their helpful and knowledgeable comments.

References

[1] Stefano Zanero. Detecting 0-day attacks with learning

intrusion detection systems. In *Blackhat Briefings USA 2004*, 2004.

[2] Thomas H. Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical Report T2R-0Y6, Secure Networks, Calgary, Canada, 1998.

[3] Giovanni Vigna, William Robertson, and Davide Balzarotti. Testing network-based intrusion detection signatures using mutant exploits. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 21–30. ACM Press, 2004.

[4] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, J. P. Anderson Co., Ft. Washington, Pennsylvania, Apr 1980.

[5] Stefano Zanero and Sergio Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 14th Symposium on Applied Computing, ACM SAC 2004*, 2004.

[6] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna. On the detection of anomalous system call arguments. In *Proceedings of ESORICS 2003*, Oct. 2003.

[7] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[8] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[9] M.V. Mahoney and P.K. Chan. Detecting novel attacks by identifying anomalous network packet headers. Technical Report CS-2001-2, Florida Institute of Technology, 2001.

[10] Dit-Yan Yeung and Calvin Chow. Parzen-window network intrusion detectors. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 4, pages 385–388, aug 2002.

[11] K. Labib and R. Vemuri. NSOM: A real-time network-based intrusion detection system using self-organizing maps. Technical report, Dept. of Applied Science, University of California, Davis, 2002.

[12] T. Lane and C.E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Trans. on Information and System Security*, 2(3):295–331, 1999.

[13] J. A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[14] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berling, 3 edition, 2001.

[15] K.M.C. Tan and B.S. Collie. Detection and classification of TCP/IP network services. In *Proc. of the Computer Security Applications Conf.*, pages 99–107, 1997.

[16] Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *RAID Symposium*, September 2004.

[17] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1995.

[18] I. T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[19] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachussets Institute of Technology, 1998.

[20] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, 2000.

[21] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2), 2003.

[22] S. McCreary and K. Claffy. Trends in wide area ip traffic patterns - a view from ames internet exchange. In *Proceedings of ITC'2000*, 2000.

[23] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[24] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. What is the nearest neighbor in high dimensional spaces? In *The VLDB Journal*, pages 506–515, 2000.

[25] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973, 2001.

Design of a MATLAB toolbox for term-document matrix generation*

Dimitrios Zeimpekis† Efstratios Gallopoulos‡

Abstract

Data clustering and many other fundamental operations in data mining and information retrieval are built using computational kernels from numerical linear algebra and operate on very large, sparse term-document matrices. To facilitate these tasks, we have built TMG, a toolbox for the generation and incremental modification of term-document matrices from text collections. The toolbox is written entirely in MATLAB, a powerful and popular problem solving environment for linear algebra. It can be used in research and education contexts to streamline document preprocessing and prototyping of algorithms for information retrieval. We outline the design and use of TMG and present results, from standard and readily available datasets of moderate size, concerning the effect of stemming and term weighting on clustering algorithms based on the vector space model.

Keywords: Indexing, term-document matrix, Vector Space Model, Clustering, MATLAB

1 Introduction

Two central tasks in Data Mining (DM) and Information Retrieval (IR) models are a preprocessing, "indexing" phase, in which the index of terms is built, and a "search" phase, during which the index is used in the course of queries, clustering and other operations. This paper outlines TMG, a toolbox that *i*) preprocesses documents to construct an index in the form of a sparse "term-document matrix", hereafter abbreviated by "tdm", and *ii*) preprocesses user queries so as to make them ready for further IR and DM. TMG is written entirely in MATLAB [4] and runs on any computer system that supports this environment. TMG parses single or multiple files or entire directories containing text, performs the necessary preprocessing, such as stopword removal and stemming and constructs a tdm according to parameters set by the user. It is also able to renew existing tdm's by performing efficient updates or downdates corresponding to the incorporation of new or deletion of existing documents. Therefore, TMG is expected to be particularly useful in the context of algorithm development for research and instruction in the blossoming area of Text Mining [31].

There exist already several tools for constructing tdm's since IR systems that are based on vector space techniques (e.g. Latent Semantic Indexing) are obliged to operate on rows and columns of such matrices; see e.g. [3, 25, 29]. The *Telcordia LSI Engine*, the *General Text Parser* (GTP) ([2, 18]), the PGTP (an MPI-based parallel version of GTP), the DOC2MAT [1] developed in the context of the CLUTO IR package [21] and the scripts `mc` [5, 16] and `countallwords` (included in the PDDP package [13]), are examples of useful tools for the construction of tdm's, implemented in high level or scripting languages (e.g. `C`, `C++`, `Java`, `Perl`). In view of the significant presence of computational linear algebra (CLA) kernels in vector space techniques for IR, we felt that there was a "market need" for a MATLAB-based tdm generation system, as MATLAB is a highly popular problem solving environment for CLA that enables the rapid prototyping of novel IR algorithms.

TMG assumes that documents are to be represented as term vectors and can be used to complement algorithms and tools that work with tdm's, e.g. [9, 12, 16]. For example, TMG was used in recent experiments included in [23, 28], and in our work with a generalization ([33]) of the Principal Direction Divisive Partitioning (PDDP) clustering algorithm [12, 13] and in [32]. Even though TMG is not tied to specific algorithms of the vector space model, it includes related MATLAB templates to facilitate the development of routines for querying and clustering. The toolbox makes heavy use of the sparse matrix infrastructure of MATLAB; on the other hand, in its current form, TMG does not employ term data compression techniques. As a result, we expect that the system will be most useful for datasets of moderate size, whereas, a system such as GTP would probably be a better choice for much larger ones. TMG and its documentation are available from

`http://scgroup.hpclab.ceid.upatras.gr/scgroup/Projects/TMG/`

This paper is organized as follows. Section 2 outlines

*Work conducted in the context of and supported in part by a University of Patras KARATHEODORI grant. Consult also [22] for an extended version of this work.

†Computer Engineering & Informatics Department, University of Patras, Greece. Supported in part by a Bodossaki Foundation graduate fellowship. Email: dsz@hpclab.ceid.upatras.gr.

‡Computer Engineering & Informatics Department, University of Patras, Greece. Also supported in part, as external collaborator, by a Research Academic Computer Technology Institute travel grant. Email: stratis@ceid.upatras.gr.

TMG, its core functions and the various options provided in the package. Section 3 discusses implementation issues, in particular the utilization of MATLAB's sparse matrix technology. In Section 4 we demonstrate the use of TMG while providing experimental results concerning the effect of stemming and term weighting in clustering. Finally, Section 5 provides concluding remarks.

2 The Text to Matrix Generator

2.1 Outline. TMG is designed to perform the preprocessing and filtering steps that are typically performed in the context of IR applications [6]. Specifically, TMG can be used for the following typical IR steps (in parentheses are the names of the relevant MATLAB `m`-functions):

- creation of the tdm corresponding to a set of documents (`tmg`);
- creation of query vectors from user input (`tmp_query`);
- update existing tdm by incorporation of new documents (`tdm_update`);
- downdate existing tdm by deletion of specified documents (`tdm_downdate`).

The document preprocessing steps encoded by TMG are the following: *i*) Lexical analysis; *ii*) stopword elimination, that is the removal of very frequent words such as articles, prepositions, conjunctions, etc. that carry little information about the contents of the processed documents; *iii*) stemming, that is the replacement of all variants of a word with a single common stem; *iv*) index-term selection, that is the selection of a subset of words encountered in the documents to form the document index; *v*) index construction. These steps are tabulated in Table 1.

2.2 User interface. The user interacts with TMG by means of any of the aforementioned basic functions or a graphical interface (GUI) implemented in function `tmg_gui`. The GUI facilitates user selection of the appropriate options amongst the many alternatives available at the command-line level.

A user that desires to construct a tdm from text will either use `tmg` or `tmg_gui`. The specific invocation of the former is of the form:

```
[outargs]=tmg('fname', OPTIONS);
```

where *outargs* is the output list:

```
[A, dictionary, global_wts, norml_factors,
words_per_doc, titles, files, update_struct].
```

The tdm is stored in `A`, while `dictionary` is a char array containing the collection's distinct words, and `update_struct`

Function `tmg`
Input: filename, `OPTIONS`
Output: tdm, dictionary, and several optional outputs;
parse files or input directory;
read the stoplist;
for each input file,
parse the file (construct dictionary);
end
normalize the dictionary (remove stopwords and too long or too short terms, stemming);
construct tdm;
remove terms as per frequency parameters;
compute global weights;
apply local weighting function;
form final tdm;

Table 1: Steps in function `tmg`

contains the essential information for the collection's renewal (see Section 3.2). The other output arguments store statistics for the collection.

Argument `'fname'` specifies the file or directory that contains the raw ASCII text. If the argument is a directory, TMG processes all files in the directory and recursively in all its subdirectories. TMG assumes that all files are valid inputs. Files suffixed with `html[s]` are assumed to be ASCII files written in the implied markup languages and are filtered to eliminate their corresponding tags. If a utility such as Ghostscript's `ps2ascii` is available, TMG is also able to process Adobe Acrobat PDF and POSTSCRIPT documents. Furthermore, it is not difficult for the user of TMG to enable it to process other types of documents as long as the necessary filters are in place.

The options available at the command line are set via the fields of the MATLAB `OPTIONS` structure; these are tabulated in Table 2 together with their default values. `OPTIONS` field `delimiter` specifies the delimiter that separates individual documents within the same file. The default delimiter is a blank line, in which case TMG is likely to generate more "documents" than the number of files given as input. `OPTIONS` field `line_delimiter` specifies if the `delimiter` takes a whole line of text. Field `stoplist` specifies the file containing the stopwords, i.e. the terms excluded from the collection's dictionary [10]. We note that the current release of TMG contains the sample stoplist contained in the GTP [2]. Field `stemming` controls the application of stemming, currently performed via a MATLAB implementation of a version of Porter's stemmer [26, 27].

The next two parameters are thresholds used to exclude from the dictionary's construction terms that exceed or fall below them; this is typical in IR filtering. Terms that are too short are likely to be articles and conjunctions that are

delimiter	String specifying the "end-of-document" marker for tmg. Possible values are emptyline (default), none_delimiter (treats each file as a single document) or any other string
line_delimiter	Variable specifying if the delimiter takes a whole line of text (default, 1)
stoplist	Filename for stopwords (default no name, meaning no stopword removal)
stemming	A flag that indicates if stemming is to be applied (1) or not (0) (default stemming=0)
min_length	Minimum term length (default 3)
max_length	Maximum term length (default 30)
min_local_freq	Minimum term local frequency (default 1)
max_local_freq	Maximum term local frequency (default Inf)
min_global_freq	Minimum number of documents for a term to appear to insert it in the dictionary (default 1)
max_global_freq	Maximum number of documents for a term to appear to insert it in the dictionary (default Inf)
local_weight	Local term weighting function (default 't', possible values 't', 'b', 'l', 'a', 'n')
global_weight	Global term weighting function (default no global weighting used, 'x', possible values 'x', 'e', 'f', 'g', 'n', 'p')
normalization	Flag specifying if document vectors are to be normalized ('c') or not ('x') (default)
dsp	Flag specifying if results are to be printed in the command window (1, default) or not (other)

Table 2: OPTIONS fields.

of little value as indexing terms while terms that are too long are likely to be misprints. The next four parameters are also thresholds that serve a similar purpose, but using a "frequency of occurrence" criterion.

Each element α_{ij} of the tdm A measures the importance of term i in document j and in the entire collection. There has been proposed various term weighting schemes using alternative functions for the local and global weighting for a term. Another usual practice is the use of normalization for each document vector. This normalization factor is used for the obliteration of bias towards longer documents [30]. OPTIONS fields local_weight, global_weight and normalization define the term weighting and normalization schemes used for the construction of the tdm [10, 24]. The weighting functions available in TMG are listed in Table 3.

The last OPTIONS field, dsp indicates if the intermediate results are printed on the MATLAB command window. Function tmg_query uses the dictionary returned by tmg and constructs, using the same processing steps as TMG, a "term query" array whose columns are the (sparse) query vectors for the text collection. The function is invoked by means of the command:

```
[Q, wds_per_query, titles,files]=
tmg_query('fname',dictionary,OPTIONS);
```

2.2.1 Graphical User Interface. As described so far, the main toolbox functions tmg and tmg_query offer a large number of options. Moreover, it is anticipated that future releases will further increase this number to allow for additional flexibility in operations and filetypes handled by TMG. In view of this, a GUI, we would be calling TMG_GUI, was created to facilitate interaction with these functions. The interface window (see Fig. 1) is invoked at the command line by means of command tmg_gui. The GUI was constructed by means of the interactive MATLAB tool GUIDE. TMG_GUI consists of two frames: One provides an edit box so that the user can specify the input filename expected by tmg; the other provides radio buttons, edit boxes, lists and toggle buttons for the optional arguments of tmg, tmg_query and update routines. After specifying all necessary parameters, and the Continue button is clicked, TMG_GUI invokes the appropriate function. The progress of the program is shown on the screen; upon finishing the user is queried if and where he wants the results to be saved; results are saved in MATLAB-mat file(s), i.e. the file format used by MATLAB for saving and exchanging data.

3 Implementation issues

We next address some issues that relate to the design choices made regarding the algorithms and data structures used in TMG.

3.1 Sparse matrix infrastructure. Our objectives were to build data structures that would achieve *i*) efficient processing of the kernel operations in the expected IR tasks; *ii*) low cost of creating and updating them; *iii*) low storage requirements. The "end product" produced by TMG is the tdm representing the data provided to the tool. Tdm's are usually extremely sparse. Therefore, a natural object for representing tdm's are MATLAB sparse arrays. It is worth noting that with the popularity of vector space models (VSM's) such as LSI, sparse matrix representations have become popular for IR applications; for example, GTP and the Telcordia LSI Engine use the Harwell-Boeing format ([2, 15, 18]), while the mc

Symbol	*Name*	*Type*
	Local term-weighting (l_{ij})	
t	Term frequency	f_{ij}
b	Binary	$b(f_{ij})$
l	Logarithmic	$\log_2(1+f_{ij})$
a	Alternate log [24]	$b(f_{ij})(1+\log_2 f_{ij})$
n	Augmented normalized term frequency	$(b(f_{ij})+(f_{ij}/\max_k f_{kj}))/2$
	Global term-weighting (g_i)	
x	None	1
e	Entropy	$1+(\sum_j(p_{ij}\log_2(p_{ij}))/\log_2 n)$
f	Inverse document frequency(IDF)	$\log_2(n/\sum_j b(f_{ij}))$
g	GfIdf	$(\sum_j f_{ij})/(\sum_j b(f_{ij}))$
n	Normal	$1/\sqrt{\sum_j f_{ij}^2}$
p	Probabilistic Inverse	$\log_2((n-\sum_j b(f_{ij}))/\sum_j b(f_{ij}))$
	Normalization factor (n_{ij})	
x	None	1
c	Cosine	$(\sum_j(g_i l_{ij})^2)^{-1/2}$

Table 3: Term-weighting and normalization schemes [10, 24].

toolkit ([5]) uses the compressed sparse column format also used by TMG. Furthermore, recent studies indicate the advantage of sparse matrix over traditional IR representations such as inverted indexes [19].

Another data structure popular in the context of Boolean searches for representing document collections is the "inverted index". Such a structure typically consists of a term dictionary and a "posting list" associated with each term; for each term t, the posting list consists of pairs $(\texttt{doc\_id}, n_t)$, where doc_id is an integer document identifier and n_t is an integer recording the number of occurrences of the term in the document. Restricting the representation to an inverted index, however, complicates the implementation of non-Boolean searches (e.g. in the form of incremental computations, cf. [6]), especially dimensionality reduction transformations that are at the core of LSI. On the other hand, an inverted index becomes useful as a transient data structure, when building the sparse tdm. In particular, to build directly the target tdm as a MATLAB sparse array, one would require very fast concatenation and access to new rows and columns. This would become inefficient at the matrix creation phase since, typically, it is not possible to have a good a priori estimate of the matrix size and total number of nonzeros. We represent the inverted index by means of two MATLAB cell arrays, one for the dictionary and the other for the posting list. Each element of the latter cell array is a numeric array of size $2 \times d_t$, where d_t is the number of documents containing the corresponding term, containing the pairs $(\texttt{doc\_id}, n_t)$. After parsing the collection, cleaning and stemming the dictionary, each cell array for the posting list is copied to another cell array each element of which is a MATLAB sparse column vector of size n. Finally this cell array is converted to the sparse array that contains the tdm by means of the cell2mat command. We note that TMG uses MATLAB functions such as unique and ismember that improve significantly the efficiency of the parsing phase.

MATLAB provides an effective environment for sparse computations (cf. [17]) that is utilized by TMG. Matrices classified as sparse are stored in the compressed sparse column format (CSC) (cf. [7]). MATLAB 6.5 uses 8 byte reals and 4 byte integers, and the total workspace occupied by a sparse non-square tdm A is approximately $Mem(A) = 12nnz + 4(n+1)$ bytes (the exact value depends on a MATLAB estimate of the total number of nonzeros of the sparse matrix [17]. Therefore, for non-square matrices, the space requirements are asymmetric, in the sense that $Mem(A) \neq Mem(A^\top)$, though the storage difference is $4|m-n|$, which is small relative to the total storage required. MATLAB is known to be much more efficient when operations are expressed in matrix or vector format rather than using loops operating on each element[1]; single element access takes time at least proportional to the logarithm of the length of its containing column and inserting or removing a nonzero may require extensive data movement. By expressing and coding

[1] Note that this difference has been less pronounced in recent versions of MATLAB that implement more advanced compilation techniques.

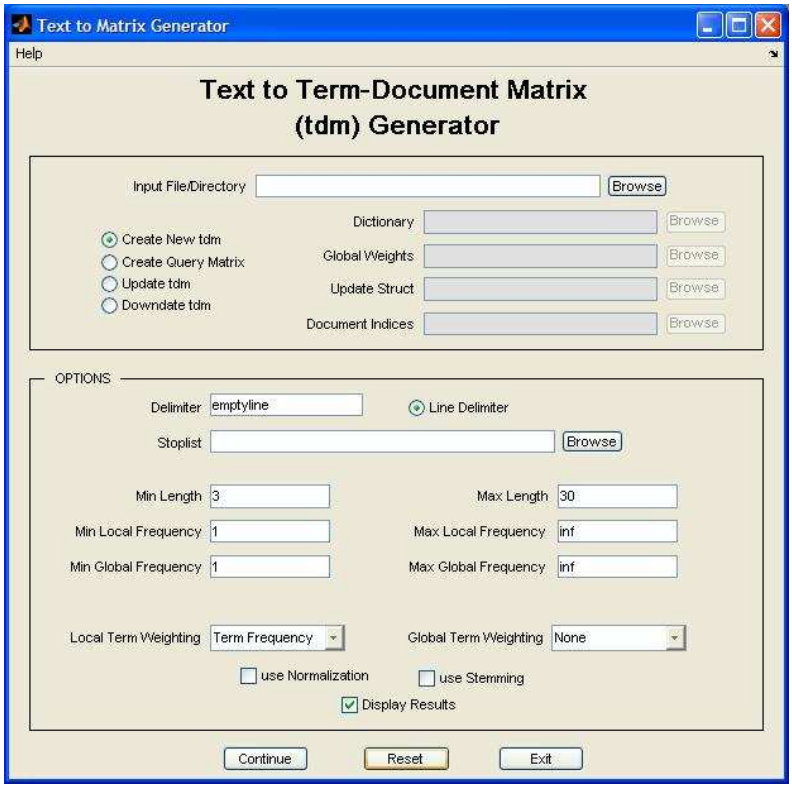


Figure 1: TMG_GUI.

the more intensive manipulations in TMG in terms of MATLAB sparse matrix operations, the cost of operating on tdm becomes proportional to the number of nonzeros in the tdm. We can demonstrate such a linearity for an operation such as the multiplication of the tdm with any vector, a kernel operation in vector space techniques, including LSI based on iterative methods. The plots in Figure 2 depict the time required by TMG for the construction of tdm's corresponding to three standard data collections, namely MEDLINE, CRANFIELD and CISI, relative to the number of their nonzeros[2].

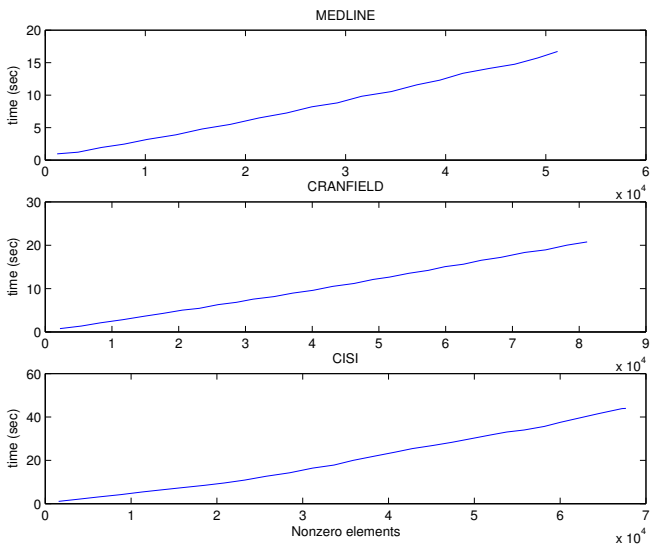


Figure 2: Runtimes (sec) of TMG versus the number of nonzeros in the tdm.

3.2 TMG **for document renewal.** The problem of effective updating or downdating tdm's is of importance as it is key to the effective maintenance of document collections in an IR system. For example, the problem of updating and downdating of a collection in an IR system based on LSI leads to interesting CLA questions related to the design of effective algorithms for the update of the SVD; see for example [25, 34]. In our case, we are interested in the simpler problem of modifying the tdm so as to retain independence from the underlying vector space technique. To that effect, TMG includes routines (functions `tdm_update`

[2]All experiments were conducted on a 2.8GHz Pentium 4 PC with 512MB RAM running Windows XP and MATLAB 6.5.

and tdm_downdate) for modifying an existing tdm so as to take into account the arrival or deletion of documents. Note that the addition of new documents, will not only cause an increase of the number of columns of the tdm, but can also cause an increase in the number of rows and a change of existing tdm entries. The latter can change because the parameters defining the weights change to reflect the new term frequencies, etc. The number of rows can increase either because *i*) new terms are encountered and satisfy the filtering requirements; and/or *ii*) terms in the original dictionary that were excluded by filtering suddenly become valid entries. This means that to update properly, we need to keep the entire dictionary of terms encountered during parsing (before any filtering steps have been performed) as well as the term frequencies of all terms. In that case, we would also be able to update the matrix when parameters such as the maximum and/or minimum global frequencies change; note that in the latter case, some terms might become invalid causing row removal from the tdm.

To perform updates, the first time TMG is run, it must keep the entire dictionary as well as the sparse tdm resulting before any filtering; this is done in a MATLAB structure array object (struct) denoted by update_struct. We avoid using one full and one normalized (post-filtering) dictionary by working only with the full dictionary and a vector of indices indicating those terms active in the normalized dictionary, stored in the same struct object. Function tdm_downdate operates in the same way, by removing documents from the collection. The user has to specify the update_struct and a vector of integers denoting the document indices of the normalized data matrix that he wants be remove.

4 TMG in clustering experiments

4.1 The BIBBENCH dataset To illustrate the use of TMG we created a new dataset, we call BIBBENCH consisting of three source files from publicly accessible bibliographies[3] in BIBTEX (the bibliography format for LATEX documents), with characteristics shown in Table 4. The first, we call BKN, is a 651 entry bibliography contained in the book [22], though loaded sometime before printing and therefore not corresponding exactly to the final edition. The major theme is clustering. The second bibliography, BEC is from http://jilawww.colorado.edu/bec/bib/, a repository of BIBTEX references from topics in Atomic and Condensed Matter Physics on the topic of Bose-Einstein condensation. When downloaded (Jan. 24, 2005) the bibliography contained 1,590 references. The last bibliography, GVL, was downloaded from http://www.netlib.org/bibnet/subjects/ and contains the full 861 item bibliography of the 2nd edition (1989) of a well-known treatise on Matrix Computations [20]. The file was edited to remove the first 350 lines of text that consisted of irrelevant header information. All files were stored in a directory named BibBench. It is worth noting that, at first approximation, the articles in BEC could be thought as belonging in one cluster ("physics") whereas those in BKN and GVL in another ("linear algebra and information retrieval").

We first used TMG to assemble the aforementioned bibliographies using term weighting, no global weighting, no normalization and stemming (txx_s) thus setting as non-default OPTIONS

```
OPTIONS.delimiter='@';
OPTIONS.line_delimiter=0;
OPTIONS.stoplist='bibcommon_words';
OPTIONS.stemming=1;
OPTIONS.min_global_freq =2; OPTIONS.dsp=
0
```

therefore, any words that appeared only once globally were eliminated (this had the effect of eliminating one document from GVL). The remaining packet had 3,101 bibliographical entries across three plain ASCII files KNB.bib, BEC.bib, GVL.bib. The stoplist file was selected to consist of the same terms found in [18] augmented by keywords utilized in BIBTEX and referring to items that are not useful for indexing, such as author, title, editor, year, abstract, keywords, etc.

The execution of the following commands

```
varargout=tmg('BibBench',OPTIONS);
```

has the following effect (compacting for some pretty-printing):

```
==========================================
Applying TMG for directory C:/TMG/BibBench...
==========================================
Results:
==========================================
Number of documents = 3101
Number of terms = 3154
Avg number terms per document (before
   normalization) = 36.9987
Avg number of indexing terms per
   document = 11.9074
Sparsity = 0.362853%
Removed 302 stopwords...
Removed 964 terms using the
   stemming algorithm...
Removed 2210 numbers...
Removed 288 terms using the
   term-length thresholds...
Removed 6196 terms using the
   global thresholds...
```

[3] The bibliographies are directly accessible from the TMG web site.

feature	BEC	BKN	GVL	BIBBENCH_txx_s
documents	1,590	651	860	3,101
terms (indexing)	1712	1159	720	3,154
stemmed terms	372	389	221	964
avg. terms/document	41	38	28	37
avg. terms/document (indexing)	13	13	8.40	12
tdm nonzeros (%)	0.74	1.00	1.15	0.36

Table 4: BIBBENCH dataset.

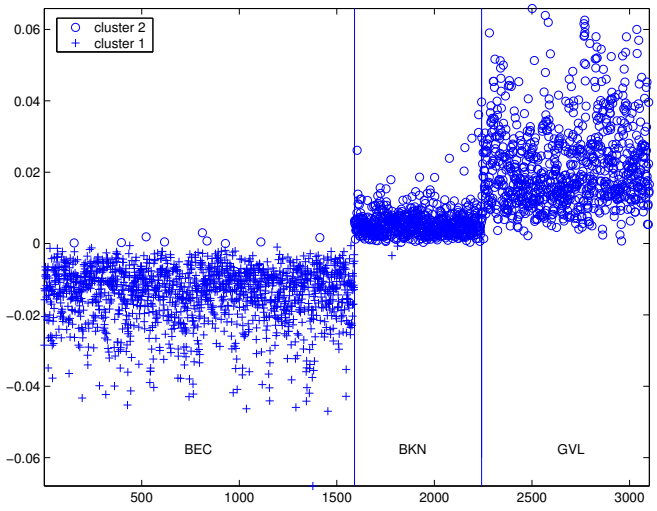


Figure 3: Values of each of the 3101 components of the maximum right singular vector $v_{\max}$ of the BIBBENCH dataset vs. their location in the set. The vertical lines that separate the three BIBTEXfiles and the labels were inserted manually.

```
Removed 0 elements using the
   local thresholds...
Removed 0 empty terms...
Removed 1 empty documents...
```

A simple combination of commands depicts the frequencies of the most frequently occurring terms. After running TMG as above, the commands

```
f=sum(A,2); plot(f,'.'); [F,I]=sort(f);
t=20; dictionary(I(end:-1:end-t+1),:)
```

plot the frequencies of each term (Fig. 4) and return the top $t = 20$ terms of highest frequency in the set, listed in decreasing order of occurrence below:

```
phy rev os condens instein lett atom
trap comput algorithm cluster method
data ga usa system matrix linear matric
mar
```

We next use TMG to modify the tdm so that it uses a different weighting scheme specifically `tnc` and stemming. This can be done economically with the `update_struct` computed earlier as follows:

```
update_struct.normalization='c';
update_struct.global_weight='n';
A=tdm_update([],update_struct);
```

Using MATLAB's `spy` command we visualize the sparsity structure of the tdm `A` in Fig. 4 (right). In the sequel we apply two MATLAB functions produced in-house, namely `pddp` and `block_diagonalize`. The former implements the PDDP(l) algorithm for clustering of term-document matrices [33]. We used $l = 1$ and partitioned in two clusters only, so that results are identical with the original PDDP algorithm [12]. In particular, classification of each document into one of the two clusters is performed on the basis of the sign of the corresponding element in the maximum right singular vector of matrix $A - Aee^{\top}/n$, where e is the vector of all 1's. The MATLAB commands and results are as follows:

```
>> clusters = pddp(A,'svds','normal','1',2);
Running PDDP(1) for k=2...
Using svds method for the SVD...
Splitting node #2 with 3101 documents
and 55.558 scatter value
    Leaf 3 with 1583 documents
    Leaf 4 with 1518 documents
Number of empty clusters = 0
PDDP(1) terminated with 2 clusters
```

Fig. 3 plots the maximum singular vector $v_{\max}$ corresponding to the BIBBENCH dataset. Even though our goal here is not to evaluate clustering algorithms, it is worth noting that PDDP was quite good at revealing the two "natural clusters". Fig. 3 shows that there are some documents from BEC (marked with '+') that were classified in the "clustering and matrix computations" cluster and very few documents from BKN and GVL that were classified in the "physics" cluster.

Finally, function `block_diagonalize` implements and plots the results from a simple heuristic for row reordering of the term-document matrix based on `pddp`. In particular, running

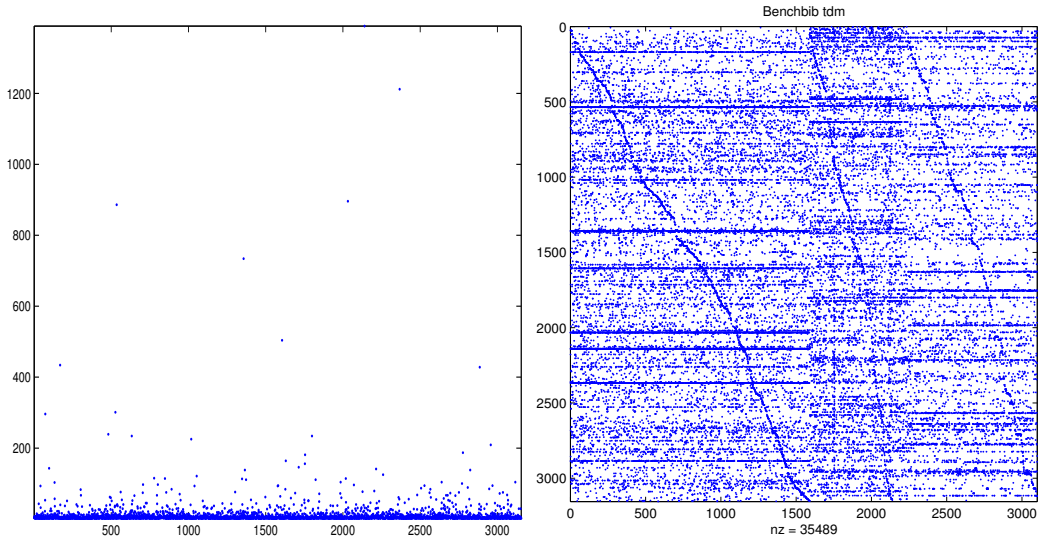


Figure 4: BIBBENCH term frequencies (left); tdm sparsity structure using `spy` (right).

```
>> block_diagonalize(A, clusters);
```

we obtain Fig. 5 (left), while the other plot of Fig. 5 depicts the same result when running PDDP(1) for $k = 4$. This illustrates the improvement made by the clustering procedure. We note here that experiments of this nature, in the spirit of work described in [11], are expected to be useful for instruction and research, e.g. to visualize the effect of novel reordering schemes. Finally, Table 5, shows the size and ten top most frequent terms (after stemming) for each of the four clusters obtained using PDDP(1) for $k = 4$. There were two "physics" clusters, the theme of another appears to be "linear algebra" while the theme of the last one is "data mining". The terms also reveal the need for better data cleaning ([14]), e.g. by normalizing or eliminating journal names, restoring terms, etc.: For instance, `numermath, siamnum` were generated because of non-standard abbreviations of the journals "Numerische Mathematik" and "SIAM Journal of Numerical Analysis". Terms `instein` and `os` were generated because of entries such as {E}`instein` and {B}`ose`, where the brackets were used in the BIBTEX to avoid automatic conversion to lower case.

I (1,033)	II (553)	III (633)	IV (885)
phy	phy	matric	cluster
rev	instein	numermath	usa
os	rev	matrix	data
condens	condens	eigenvalu	comput
trap	os	siamnuman	mine
instein	lett	symmetr	algorithm
ga	ketterl	linalgapp	york
atom	atom	problem	analysi
lett	optic	linear	parallel
interact	mar	solut	siam

Table 5: Ten most frequent terms for each of the four clusters of BIBBENCH using PDDP(1). In parentheses are the cluster sizes. We applied stemming but only minimal data cleaning.

4.2 Evaluating term-weighting and stemming We next use the flexibility of the toolbox to study the effect of different term weighting and normalization schemes and stemming on clustering text collections. We used part of collection REUTERS-21578, that is composed of 21,578 documents, 8,654 of which belong to a single topic. We applied TMG in three parts of REUTERS-21578 comprising of 22, 9 and 6 classes of documents. Table 6 gives the features of the three parts labeled as REUT1, REUT2 and REUT3, as reported by TMG. We experimented on the aforementioned collections with all possible schemes available in TMG and recorded the entropy values using two representative clustering algorithms: Spherical k-means (Skmeans) [16] as a representative partitioning method and PDDP [12], as a representative hierarchical partitioning technique based on spectral information.

The `stoplist` file was the default obtained from GTP, while stemming was enabled. As shown in Table 6, stemming causes a significant - up to 30% - reduction in dictionary size.

There are 60 possible combinations for term weighting and normalization in constructing the term document matrix. Taking into account the use or not of stemming, there are a total of 120 parameter combinations. We ran all of them on the aforementioned data collections and recorded the results

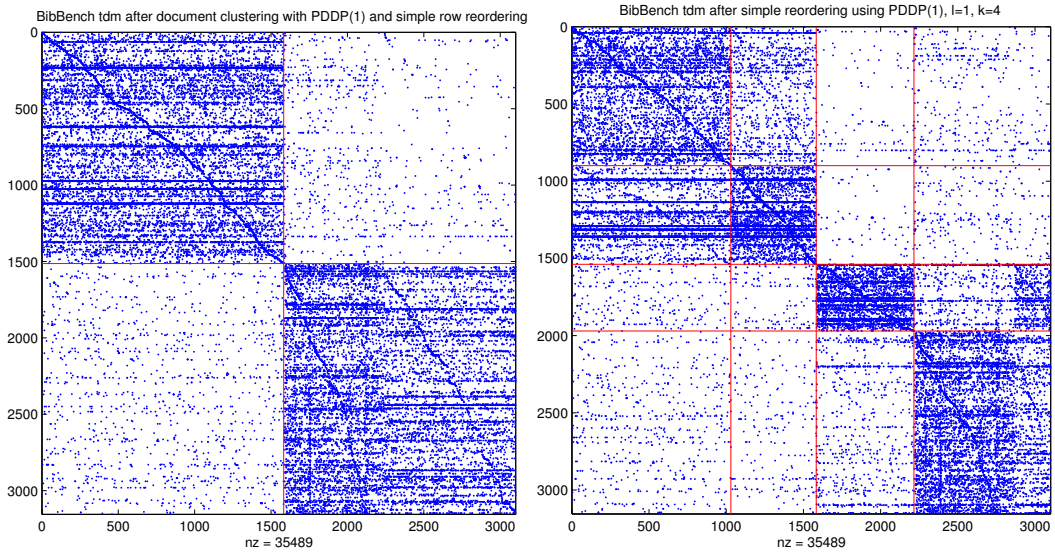


Figure 5: spy view of BIBBENCH tdm's for $k = 2$ (left) and $k = 4$ (right) clusters.

feature	REUT1	REUT2	REUT3
documents	880	900	1,200
terms (ind.)	4,522	4,734	5,279
avg. terms/doc.	179	180	175
avg. terms/doc. (ind.)	81	83	81
tdm nonzeros (%)	0.20	0.20	0.81
terms (after stemming)	3,228	3,393	3,691
dict. size reduction (%)	29	28	30

Table 6: Characteristics of document collections used in the clustering experiments

REUT1		REUT2		REUT3	
PDDP					
tpc_s	1.46	lec_ns	1.11	aec_s	0.85
tec_s	1.54	afc_ns	1.13	lec_s	0.90
tfc_s	1.58	aec_ns	1.14	tec_s	0.92
tec_ns	1.59	tec_s	1.17	tec_ns	0.93
lec_s	1.61	tfc_ns	1.18	bxc_ns	0.96
Skmeans					
tpc_s	1.18	axc_s	0.61	bxc_ns	0.66
tpc_ns	1.23	aec_s	0.73	lec_ns	0.67
tfc_s	1.28	lec_ns	0.73	lxc_ns	0.73
tec_s	1.30	lxc_s	0.73	axc_s	0.74
afc_s	1.31	tfc_s	0.73	bxc_s	0.74

Table 7: Entropy values for PDDP and Skmeans.

for PDDP and Skmeans. Table 7 lists the entropy values obtained using the 5 weighting and normalization schemes that returned the best results. An "_s" symbol implies the application of stemming whereas "_ns" means that no stemming was used.

Table 7 indicates some results, concerning the performance of PDDP and Skmeans in the context of stemming and the variety of term weighting options that are worth reporting. First, note that Skmeans returns good entropy values, about 45% better than PDDP for REUT2. Furthermore, stemming seems to improve quality in clustering in most cases resulting in lower entropy values, while it is clear from Table 7 that cosine normalization improves clustering in all cases (Skmeans normalizes the document vectors by default). Although Table 7 does not provide clear guidance regarding the selection of specific term weighting schemes we can see that 'logarithmic' and 'alternate log' local functions and 'entropy' and 'IDF' global functions appear to return good results. More interestingly, we get high quality results using the simple 'term frequency' local function, while weighting terms globally does not improve necessarily the quality of clustering.

5 Conclusions

We outlined the design of a MATLAB toolbox for the construction of tdm's from text collections. TMG makes heavy use of MATLAB's sparse matrix infrastructure. Our motivation was to facilitate users, such as researchers in computational linear algebra who use MATLAB to build algorithms for textual information retrieval and are interested in the rapid preparation of test data. We demonstrate the use and performance of our system for several publicly available

datasets, including REUTERS-21578 and on BIBBENCH, an easy to construct dataset of bibliography entries in BIBTEX format. Extensive experiments with TMG over the REUTERS-21578 collection and an exhaustive combination of term weighting and normalization schemes and stemming indicate the success of specific weighting schemes. They show that stemming can improve quality and efficiency of clustering but also underline that such benchmarking becomes easy with TMG. To the best of our knowledge, TMG is the first MATLAB toolbox possessing the range of capabilities described herein. We expect the system to become useful in several research and educational contexts and point users to [22] and the tool's website for more information. In addition to operating the tool for our experiments in the field of IR, we are currently working in enabling the tool to process a variety of other document types as well as in distributed implementations. Finally, we expect that MATLAB 7.0 will permit an even more efficient implementation because of its support for integer and single-precision floating-point arithmetic.

Acknowledgments TMG was conceived after a motivating discussion with Andrew Knyazev regarding a collection of MATLAB tools we had put together to aid in our clustering experiments. We thank Michael Berry for discussions and for including the software in the LSI web site [3], Efi Kokiopoulou and Constantine Bekas for many helpful suggestions, Dan Boley for his help regarding preprocessing in PDDP. We also thank Inderjit Dhillon, Pavel Berkhin and Jacob Kogan for providing us with a component of BIBBENCH and Elias Houstis for his help in the initial phases of this research. Special thanks are due to many of the users for constructive criticism of the tool and to the reviewers of this submission for their comments that not only helped us improve the paper but constitute useful advice for further enhancements.

References

[1] Doc2mat. Available from www-users.cs.umn.edu/~karypis/cluto/files/doc2mat-1.0.tar.gz.

[2] General Text Parser. Available from www.cs.utk.edu/~lsi/soft.html.

[3] Latent Semantic Indexing Web Site. Maintained by M.W. Berry and S. Dumais at, www.cs.utk.edu/~lsi/.

[4] MATLAB: The Language of Technical Computing. In http://www.mathworks.com/products/matlab/.

[5] Mc Toolkit. Available from, www.cs.utexas.edu/users/dml/software/mc/.

[6] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[7] R. Barrett, M. Berry, T. Chan, J. Demmell, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1993.

[8] M.W. Berry, editor. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer Verlag, New York, 2004.

[9] M. Berry, Z. Drmac, and E. Jessup. Matrices, Vector Spaces, and Information Retrieval. *SIAM Review*, 41:335–362, 1998.

[10] M.W. Berry and M. Brown. *Understanding Search Engines*. SIAM, Philadelphia, 1999.

[11] M.W. Berry, B. Hendrickson, and P. Raghavan. Sparse matrix reordering schemes for browsing hypertext. In J. Renegar, M. Shub, and S. Smale, editors, *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Mathematics (LAM)*, pages 99–123. American Mathematical Society, 1996.

[12] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[13] D.L. Boley. Principal direction divisive partitioning software (experimental software, version 2-beta), Feb. 2003. Available from www-users.cs.umn.edu/~boley/Distribution/PDDP2.html.

[14] M. Castellanos. Hot-Miner: Discovering hot topics from dirty text. In Berry [8], pages 123–157.

[15] C. Chen, N. Stoffel, M. Post, C. Basu, D. Bassu, and C. Behrens. Telcordia LSI engine: Implementation and scalability issues. In *Proc. 11th Workshop on Research Issues in Data Engineering (RIDE 2001): Doc. Management for Data Intensive Business and Scientific Applications*, Heidelberg, Germany, Apr. 2001.

[16] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[17] J.R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. Technical Report Tech. Report CSL 91-4, Xerox Palo Alto Research Center, 1991. This work was later published revised in *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992.

[18] J.T. Giles, L. Wo, and M. Berry. GTP (General Text Parser) software for text mining. *Statistical Data Mining and Knowledge Discovery*, pages 455–471, 2003.

[19] N. Goharian, T. El-Ghazawi, and D. Grossman. Enterprise text processing: A sparse matrix approach. In *Proc. IEEE International Conference on Information Techniques on Coding & Computing (ITCC 2001)*, Las Vegas, 2001.

[20] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2nd edition, 1989.

[21] G. Karypis. CLUTO. A clustering toolkit. Technical Report 02-017, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, Aug. 2002.

[22] J. Kogan, C.Nicolas and M. Teboulle, eds. *Grouping Multidimensional Data: Recent Advances in Clustering*. Springer, to appear.

[23] E. Kokiopoulou and Y. Saad. Polynomial filtering in latent semantic indexing for information retrieval. In *Proc. ACM SIGIR'04*, pages 104–111, 2004.

[24] T.G. Kolda. *Limited-Memory Matrix Methods with Applications*. PhD thesis, University of Maryland, College Park, 1997. Technical Report CS-TR-3806.

[25] T.A. Letsche and M.W. Berry. Large-scale information re-

trieval with latent semantic indexing. *Information Sciences*, 100(1-4):105–137, 1997.

[26] M.F. Porter. The Porter stemming algorithm. See `www.tartarus.org/~martin/PorterStemmer`.

[27] M.F. Porter. An algorithm for suffix stripping. *Program*, (3):130–137, 1980.

[28] J. Quesada. Creating your own LSA space. In T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Anlysis: A road to meaning*. Erlbaum, in press.

[29] G. Salton, J. Allan, and C. Buckley. Automatic structuring and retrieval of large text files. *Comm. ACM*, 37(2):97–108, 1994.

[30] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. *ACM SIGIR'96*, pages 21–29, 1996.

[31] S. Sirmakessis, editor. *Text Mining and its Applications (Results of the NEMIS Launch Conference)*. Springer, Berlin, 2004.

[32] D. Zeimpekis and E. Gallopoulos. CLSI: A flexible approximation scheme from clustered term-document matrices. In *Proc. SIAM 2005 Data Mining Conf. (to appear)*, Philadelphia, April 2005.

[33] D. Zeimpekis and E. Gallopoulos. PDDP(l): Towards a flexible principal direction divisive partitioning clustering algorithm. In D. Boley, I. Dhillon, J. Ghosh, and J. Kogan, editors, *Proc. Workshop on Clustering Large Data Sets (held in conjunction with the Third IEEE Int'l. Conf. Data Min.)*, pages 26–35, Melbourne, FL, Nov. 2003.

[34] H. Zha and H. Simon. On updating problems in latent semantic indexing. *SIAM J. Sci. Comput.*, 21:782–791, 1999.