

A Kernel-Based Semi-Naive Bayesian Classifier Using P-Trees¹

Anne Denton and William Perrizo

Department of Computer Science,
North Dakota State University
Fargo, ND 58105-5164, USA
{anne.denton, william.perrizo}@ndsu.nodak.edu

Abstract

A novel semi-naive Bayesian classifier is introduced that is particularly suitable to data with many attributes. The naive Bayesian classifier is taken as a starting point and correlations are reduced through joining of highly correlated attributes. Our technique differs from related work in its use of kernel-functions that systematically include continuous attributes rather than relying on discretization as a preprocessing step. This retains distance information within the attribute domains and ensures that attributes are joined based on their correlation for the particular values of the test sample. We implement a kernel-based semi-naive Bayesian classifier using P-Trees and demonstrate that it generally outperforms the naive Bayesian classifier as well as a discrete semi-naive Bayesian classifier.

Keywords: Bayesian classifiers, Semi-naive Bayes, Scalable Algorithms, Correlations, Kernel Methods, P-Trees.

1. INTRODUCTION

One of the main challenges in data mining is handling data with many attributes. The volume of the space that is spanned by all attributes grows exponentially with the number of attributes, and the density of training points decreases accordingly. This phenomenon is also termed the curse of dimensionality [1]. Many current problems, such as DNA sequence analysis, and text analysis suffer from the problem (see "spam" data set from [2] discussed below). A classifier that suffers relatively little from high dimensionality is the naive Bayesian classifier. Other classifiers, namely generalized additive models [4], have been developed that make similar use of the predictive power of large numbers of attributes and improve on the naive Bayesian classifier. These classifiers require an optimization procedure that is computationally unacceptable in settings in which the training data changes continuously, such as for sliding window approaches in data streams, in which old data is discarded at the rate at which new data arrives [5].

We introduce a lazy classifier that does not require a training phase. Our classifier improves on the accuracy of

the Naive Bayesian classifier by treating strongly correlated attributes as one. Approaches that aim at improving on the validity of the naive assumption through joining of attributes are commonly referred to as semi-naive Bayesian classifiers [6-8]. Kononenko originally proposed this idea [6] and Pazzani [7], more recently, evaluated Cartesian product attributes in a wrapper approach. In previous work continuous attributes were intervalized as a preprocessing step, significantly limiting the usefulness of the classifier for continuous data. Other classifiers that improve on the naive Bayesian classifier include Bayesian network and augmented Bayesian classifiers [9-11]. These classifiers commonly assume that correlations are determined for attributes as a whole, but generalizations that consider specific instances are also discussed [9]. They do, however, all discretize continuous attributes, and thereby lose distance information within attributes.

Our approach is founded on a general definition of the naive Bayesian classifier that involves kernel density estimators to compute probabilities [4]. We introduce a kernel-based correlation function and join attributes when the value of the correlation function at the location of the test sample exceeds a predefined threshold. No information is lost in the joining process. The benefits of a kernel-based definition of density estimators are thereby fully extended to the elimination of correlations. In contrast to most other techniques attributes are only joined if their values are correlated at the location of the test sample. An example of the impact of a local correlation definition could be the classification of e-mail messages based on author age and message length. These two attributes are probably highly correlated if the author is a young child, i.e. they should be joined if age and message length of the test sample are very small. For other age groups there is probably little basis for considering those attributes combined. Evaluation of kernel functions requires the fast computation of counts, i.e. of the number of records that satisfy a given condition. We use compressed, bit-column-oriented data structures, namely P-Trees to represent the data [12-16].

2. NAIVE AND SEMI-NAIVE BAYESIAN CLASSIFIER USING KERNEL DENSITY ESTIMATION

Bayes' theorem for a constant prior distribution can be stated as follows. Given a class label C with m classes c_l ,

¹ Patents are pending on the P-tree technology. This work is partially supported by GSA Grant ACT#: K96130308.

c_2, \dots, c_m and an attribute vector \mathbf{x} of all other attributes, the conditional probability of class label c_i can be expressed as follows

$$P(C = c_i | \mathbf{x}) = \frac{P(\mathbf{x} | C = c_i)P(C = c_i)}{P(\mathbf{x})} \quad (1)$$

$P(C = c_i)$ is the probability of class label c_i and can be estimated from the data directly. We use a representation that is favored by the statistics community and that is based on one-dimensional kernel density estimates as discussed in [4].

The conditional probability $P(\mathbf{x} | C = c_i)$ can be written as a kernel density estimate for class c_i

$$P(\mathbf{x} | C = c_i) = f_i(\mathbf{x}) \quad \text{with} \quad f_i(\mathbf{x}) = \frac{1}{N_i} \sum_{t=1}^N K_i(\mathbf{x}, \mathbf{x}_t) \quad (2)$$

where \mathbf{x}_t are training points, $K_i(\mathbf{x}, \mathbf{x}_t)$ is a kernel function and N_i is the number of training points with class label c_i . N is the total number of training points. The naive Bayesian model assumes that for a given class the probabilities for individual attributes are independent

$$P(\mathbf{x} | C = c_i) = \prod_{k=1}^M P(x_k | C = c_i) \quad (3)$$

where x_k is the k^{th} attribute of a total of M attributes. The conditional probability $P(x_k | C = c_i)$ can, for categorical attributes, simply be derived from the sample proportions. For numerical attributes several alternatives exist. We use a one-dimensional kernel density estimate that comes naturally from (2)

$$f_i(\mathbf{x}) = \prod_{k=1}^M f_i^{(k)}(x^{(k)}) = \frac{1}{N_i} \prod_{k=1}^M \left(\sum_{t=1}^N K_i^{(k)}(x^{(k)}, x_t^{(k)}) \right) \quad (4)$$

where the one-dimensional Gaussian kernel function is given by

$$K_{i \text{ Gauss}}^{(k)}(x^{(k)}, x_t^{(k)}) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x^{(k)} - x_t^{(k)})^2}{2\sigma_k^2}\right) [C = c_i] \quad (5)$$

with σ_k selected for good over all prediction accuracy. We chose σ_k as half of the the standard deviation of attribute k . Categorical attributes can be discussed within the same framework. The kernel function for categorical attributes is

$$K_{i \text{ Cat}}^{(k)}(x^{(k)}, x_t^{(k)}) = [x^{(k)} = x_t^{(k)}] [C = c_i] \quad (6)$$

where $[\pi]$ indicates that the term is 1 if the predicate π is true and 0 if it is false. We use this notation throughout the paper.

2.1. Correlation function of attributes

We will now go beyond the naive Bayesian approximation by joining attributes if they are highly correlated. Attributes are considered highly correlated if the product assumption in (3) can be shown to be a poor approximation.

The validity of the product assumption can be verified for any two attributes a and b individually by calculating the following correlation function

$$\text{Corr}(a, b) = \frac{N \sum_{t=1}^N \prod_{k=a, b} K^{(k)}(x^{(k)}, x_t^{(k)})}{\prod_{k=a, b} \sum_{t=1}^N K^{(k)}(x^{(k)}, x_t^{(k)})} - 1 \quad (7)$$

where the kernel function is a Gaussian function (5) for continuous data and (6) for categorical data. If the product assumption is shown to be poor, i.e., if the correlation function (7) exceeds a threshold, typically 0.05-1, then the two attributes will be considered together.

The kernel function for joined attributes is

$$K^{(a, b)}(x^{(a)}, x^{(b)}, x_t^{(a)}, x_t^{(b)}) = \sum_{t=1}^N \prod_{k=a, b} K^{(k)}(x^{(k)}, x_t^{(k)}) \quad (8)$$

With this definition it is possible to work with joined attributes in the same way as with the original attributes. It is important to observe that the criterion for correlation is a function of the attribute values. If attributes are correlated over all but not for the value of a particular test sample no join is performed.

2.2. P-Trees

The P-tree data structure was originally developed for spatial data [10] but has been successfully applied in many contexts [14,15] and is in describe in detail in those publications. P-Trees store bit-columns of the data in sequence to allow compression as well as the fast evaluation of counts of records that satisfy a particular condition. A tree-based structure replaces subtrees that consist entirely of 0 values by a higher level "pure 0" node, and subtrees that consist entirely of 1 values by higher level "pure 1" nodes. The number of records that satisfy a particular condition is now evaluated by a bit-wise AND on the compressed bit-sequences. Figure 1 illustrates the storage of a table with 2 integer and one Boolean attribute. The number of records with $A_1 = 12$ (i.e. the bit sequence 1100) is evaluated as a bit-wise AND of the two P-Trees corresponding to the higher order bits of A_1 and the complements of the two P-Trees corresponding to the lower order bits. This AND operation can be done very efficiently for the first half of the data set, since the single high-level 0-bit already indicates that the condition is not satisfied for any of the records. This is the basis for a scaling better than $O(N)$ for such operations.

The efficiency of P-tree operations depends strongly on the compression of the bit sequences, and thereby on the ordering of rows. For data that shows inherent continuity, such as spatial or multimedia data, such an ordering can be easily constructed. If data shows no natural continuity it may be beneficial to sort it. We sort according to all highest order bits first. Figure 1 indicates at the bottom the sequence in which bits are used for sorting.

A ₁	A ₂	A ₃
2	5	0
3	7	0
5	4	0
6	15	0
3	13	1
11	9	1
13	8	1
12	14	1

A ₁	A ₂	A ₃
0	0	1
1	0	1
0	1	0
1	1	0
0	1	1
1	0	1
1	0	0
1	1	1
0	1	1
1	0	1
1	0	0
1	1	1
0	1	1
1	1	0
0	1	1

Figure 1: Storage of tables as hierarchically compressed bit columns

2.3. HOBbit Distance

The nature of a P-tree-based data representation with its bit-column structure has a strong impact on the kinds of algorithms that will be efficient. P-Trees allow easy evaluation of the number of data points in a neighborhood that can be represented by a single bit pattern. The HOBbit distance [12] has the desired property. It is defined as

$$d_{HOBbit}(x_s^{(k)}, x_t^{(k)}) = \begin{cases} 0 & \text{for } x_s^{(k)} = x_t^{(k)} \\ \max_{j=0}^{\infty} \left(j + 1 \left\lfloor \left\lfloor \frac{x_s^{(k)}}{2^j} \right\rfloor \neq \left\lfloor \frac{x_t^{(k)}}{2^j} \right\rfloor \right) \right) & \text{for } x_s^{(k)} \neq x_t^{(k)} \end{cases} \quad (9)$$

where $x_s^{(k)}$ and $x_t^{(k)}$ are the values of attribute k for points \mathbf{x}_s and \mathbf{x}_t , and $\lfloor \cdot \rfloor$ denotes the floor function.

We would like to approximate functions that are defined for Euclidean distances by the HOBbit distance. The exponential HOBbit distance corresponds to the average Euclidean distance of all values within a neighborhood of a particular HOBbit distance

$$d_{EH}(x_s^{(k)}, x_t^{(k)}) = \begin{cases} 0 & \text{for } x_s^{(k)} = x_t^{(k)} \\ 2^{d_{HOBbit}(x_s^{(k)}, x_t^{(k)})-1} & \text{for } x_s^{(k)} \neq x_t^{(k)} \end{cases} \quad (10)$$

2.4. Algorithm

Our classification algorithm uses the background that has been developed as follows. Based on the attribute

values of each test sample we evaluate kernel functions for all attributes. For continuous attributes we evaluate (5) and for categorical attributes (6) for all values of the class labels. Note that these kernel functions only have to be evaluated once for each attribute value and can be reused as long as the training data is unchanged.

Kernel functions for pairs of attributes are then evaluated to determine the correlation function (7). For this purpose we use kernel functions that are independent of the class label rather than doing the analysis for different classes separately to make the solution numerically more stable. Attributes are joined if the correlation function exceeds a given threshold that is commonly chosen to be in the range 0.05 to 1. Joining of attributes consists in computing the joined kernel functions (8) and using the result to replace the individual kernel functions of the respective two attributes in (4). Evaluation of kernel functions for continuous attributes involves evaluating the number of data points within each of the HOBbit ranges of the attribute(s). This is done through AND operations on the P-Trees that correspond to the respective highest order bits. The number of points in each range is then weighted according to the value of the Gaussian function (5) using the exponential HOBbit distance (10) that corresponds to the given HOBbit range.

The products of kernel functions of single and combined attributes (4) are evaluated for all class label values. It can be seen from (2) that they correspond to the probabilities $P(\mathbf{x} | C = c_i)$. These probabilities are used in (1) together with the total probabilities of class label values $P(C = c_i)$ that are determined from the training set. The probability of the unknown sample is constant for all classes and does not have to be evaluated. The class label with the highest probability $P(\mathbf{x} | C = c_i) P(C = c_i)$ is chosen as prediction.

3. IMPLEMENTATION AND RESULTS

We implemented all algorithms in Java and evaluated them on 4 data sets. Data sets were selected to have at least 3000 data points and to contain continuous attributes. Two thirds of the data were taken as training set and one third as test set. Due to the consistently large size of data sets cross-validation was considered unnecessary. All experiments were done using the same parameter values for all data sets.

3.1. Data Sets

Three of the data sets were obtained from the UCI machine learning library [1] where full documentation on the data sets is available:

- spam data set: word and letter frequencies are used to classify e-mail as spam
- adult data set: census data is used to predict whether income is greater \$50000
- sick-euthyroid data set: medical data is used to predict sickness from thyroid disease

An additional data set was taken from the spatial data mining domain (crop data set). The RGB colors in the photograph of a cornfield are used to predict the yield of the field [17]. Class label is the first bit of the 8-bit yield information, i.e. the class label is 1 if yield is higher than 128 for a given pixel.

No preprocessing of the data was done, but some attributes, were identified as being logarithmic in nature, and the logarithm was encoded in P-Trees. The following attributes were chosen as logarithmic: "capital-gain" and "capital-loss" of the adult data set, and all attributes of the "spam" data set.

3.2. Results

We will now compare results of our semi-naïve algorithm with three more traditional implementations. The traditional Naïve Bayesian algorithm uses a Gaussian distribution function. The P-tree Naïve Bayesian algorithm uses HOBbit-based kernel density estimation, but does not check for correlations. The Discrete Semi-Naïve Bayesian algorithm does eliminate correlations but discretizes continuous data as a preprocessing step. Table 1 summarizes the results.

3.3. P-Tree Naive Bayesian Classifier

Before using the semi-naïve Bayesian classifier we will evaluate the performance of a simple naïve Bayesian classifier that uses kernel density estimates based on the HOBbit distance. Table 1 shows that for three of the data sets the P-tree naïve Bayesian algorithm constitutes an improvement over traditional naïve Bayesian.

3.4. Semi-Naive Bayesian Classifier

The semi-naïve Bayesian classifier was evaluated using two parameter combinations.

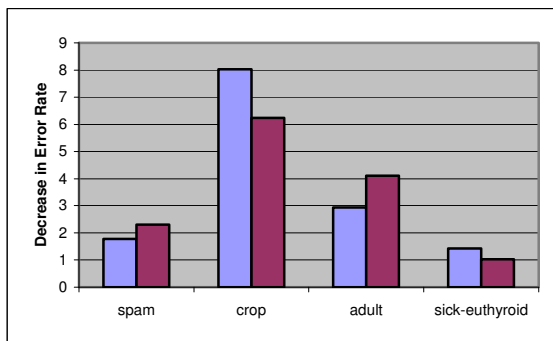


Figure 2: Decrease in error rate for the kernel-based semi-naïve Bayesian classifier compared with the P-tree naïve classifier in units of the standard error. Two parameter combinations were used: (1) $t = 0.3$, anti-correlations eliminated (left), (2) $t = 0.05$ only correlations eliminated (right).

Figure 2 shows the decrease in error rate compared with the P-tree naïve Bayesian classifier, which is the

relevant comparison when evaluating the benefit of combining attributes. It can be clearly seen that for the chosen parameters accuracy is increased over the P-tree naïve Bayesian algorithm. Run (1) used a cut-off of threshold $t = 0.3$ while run (2) used $t = 0.05$. Run (1) eliminates not only correlations, i.e., attributes for which $Corr(a,b) > t$ but also anti-correlations, i.e., attributes for which $Corr(a,b) < -t$. We then compared our approach with the alternative strategy of discretizing continuous attributes as a preprocessing step. Figure 3 shows the decrease in error rate of the kernel-based implementation compared with discretizing attributes as a preprocessing step. The improvement of accuracy for the kernel-based representation is clearly evident.

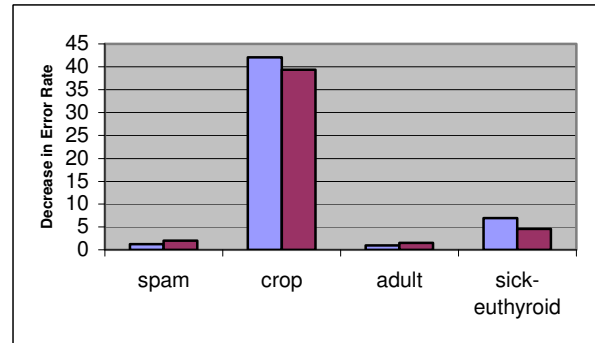


Figure 3: Decrease in error rate for two parameter combinations for the kernel-based semi-naïve Bayesian classifier compared with a semi-naïve Bayesian classifier using discretization in units of the standard error. See Figure 2 for explanation of parameter combinations (1) and (2).

3.5. Performance

It is important for data mining algorithms to be efficient for large data sets. Figure 4 shows that the P-tree-based semi-naïve Bayesian algorithm shows a better scaling than $O(N)$ as a function of the training points. This scaling is closely related to the P-tree storage concept that benefits increasingly from compression for increasing data set size.

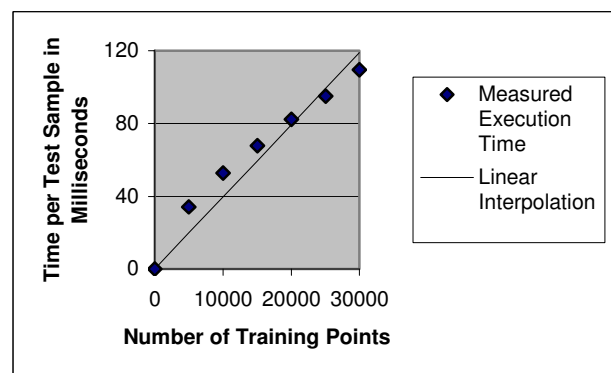


Figure 4: Scaling of execution time as a function of training set size

Table 1: Error Rates for all algorithms described in the text. (+/-) values indicate variance. See Figure 2 for explanation of parameter combinations (1) and (2).

	Traditional Naïve Bayes	(+/-)	P-Tree Naïve Bayes	(+/-)	Discrete Semi-Naïve Bayes (1)	(+/-)	Discrete Semi-Naïve Bayes (2)	(+/-)	Kernel Semi-Naïve Bayes (1)	(+/-)	Kernel Semi-Naïve Bayes (2)	(+/-)
spam	11.9	0.9	10.0	0.8	9.4	0.8	9.5	0.8	8.4	0.7	8.0	0.7
crop	21.6	0.2	22.0	0.2	28.8	0.2	28.5	0.2	20.7	0.2	21.0	0.2
adult	18.3	0.4	18.0	0.3	17.3	0.3	17.1	0.3	17.0	0.3	16.6	0.3
sick-euthyroid	15.2	1.2	5.9	0.7	11.4	1.0	8.8	1.0	4.2	0.6	4.6	0.7

4. CONCLUSIONS

We have presented a semi-naïve Bayesian algorithm that treats continuous data through kernel density estimates rather than discretization. We were able to show that it increases accuracy for data sets from a wide range of domains both from the UCI machine learning repository as well as from an independent source. By avoiding discretization our algorithm ensures that distance information within numerical attributes will be represented accurately and improvements in accuracy could clearly be demonstrated. Categorical and continuous data are thereby treated on an equally strong footing, which is unusual since classification algorithms tend to favor one or the other type of data. Our algorithm is particularly valuable for the classification of data sets with many attributes. It does not require training of a classifier and is thereby suitable to such settings as data streams. The implementation using P-Trees has an efficient sub-linear scaling with respect to training set size. We have thereby introduced a tool equally interesting from a theoretical and a practical perspective.

References

[1] D. Hand, H. Mannila, P. Smyth, "Principles of Data Mining", The MIT Press, Cambridge, Massachusetts, 2001.
 [2] <http://www.ics.uci.edu/mllearn/MLSummary.html>
 [3] P. Domingos, M. Pazzani, "Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier", 13th International Conference on Machine Learning, 105-112, 1996.
 [4] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction", Springer-Verlag, New York, 2001.
 [5] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining Stream Statistics over sliding windows," in ACM-SIAM Symposium on Discrete Algorithms (SODA), 2002.
 [6] I. Kononenko, "Semi-Naïve Bayesian Classifier", In Proceedings of the sixth European Working Session on Learning, 206-219, 1991.

[7] M. Pazzani, "Constructive Induction of Cartesian Product Attributes", Information, Statistics and Induction in Science, Melbourne, Australia, 1996.
 [8] Z. Zheng, G. Webb, K.-M. Ting, "Lazy Bayesian Rules: A lazy semi-naïve Bayesian learning technique competitive to boosting decision trees", Proc. 16th Int. Conf. on Machine Learning, 493-502, 1999.
 [9] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers", Machine Learning, 29, 131-163, 1997.
 [10] E. J. Keogh and M. J. Pazzani, "Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches", Uncertainty 99: The Seventh International Workshop on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 1999.
 [11] J. Cheng and R. Greiner, "Comparing Bayesian network classifiers", in Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI99), 101-107, Morgan Kaufmann Publishers, August 1999.
 [12] M. Khan, Q. Ding, W. Perrizo, "K-Nearest Neighbor classification of spatial data streams using P-Trees", PAKDD-2002, Taipei, Taiwan, May 2002.
 [13] W. Perrizo, Q. Ding, A. Denton, K. Scott, Q. Ding, M. Khan, "PINE - podium incremental neighbor evaluator for spatial data using P-Trees", Symposium on Applied Computing (SAC'03), Melbourne, Florida, USA, 2003.
 [14] Q. Ding, W. Perrizo, Q. Ding, "On Mining Satellite and other Remotely Sensed Images", DMKD-2001, pp. 33-40, Santa Barbara, CA, 2001.
 [15] W. Perrizo, W. Jockheck, A. Perera, D. Ren, W. Wu, Y. Zhang, "Multimedia data mining using P-Trees", Multimedia Data Mining Workshop, KDD, Sept. 2002.
 [16] A. Perera, A. Denton, P. Kotala, W. Jockheck, W. Valdivia Granda, W. Perrizo, "P-tree Classification of Yeast Gene Deletion Data", SIGKDD Explorations, Dec. 2002.
 [17] http://midas10.cs.ndsu.nodak.edu/data/images/data_set_94/
 [18] R. Kohavi, G. John, "Wrappers for feature subset selection", Artificial Intelligence, 1-2, 273-324, 1997.