CHAPTER 6. PAPER 2:

A KERNEL-BASED SEMI-NAIVE BAYES CLASSIFIER USING P-TREES

6.1. Abstract

The Naive Bayes classifier, despite its simplistic nature, has remained popular over the years and outperforms more complex classifiers in many settings, in particular where the feature space dimension is high. We take the Naive Bayes classifier as a starting point and reduce correlations by joining highly correlated attributes. Each joined attribute is treated as one attribute within an otherwise Naive Bayes classifier. Continuous attributes are integrated through approximate kernel density estimates that remove the need to discretize attributes as a preprocessing step as has been done in previous work. We implement a lazy Semi-naive Bayes classifier using P-trees and demonstrate that it generally outperforms the Naive Bayes classifier.

6.2. Introduction

One of the main challenges in data mining is to handle 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" and "splice" data sets from [2] discussed later.) Classification techniques that estimate a class label based on the density of training points such as nearest neighbor and Parzen window classifiers have to use a large range for numerical attributes in order to find any neighbors, thereby decreasing accuracy. Decision tree algorithms can only use a few attributes before the small number of data points in each branch makes results insignificant. These constraints mean that the potential predictive power of all other attributes is lost. A classifier that suffers relatively little from high dimensionality is the Naive Bayes classifier. Despites its name and its reputation of being trivial, it leads to a surprisingly high accuracy, even in cases in which the assumption of independence of attributes, on which it is based, is no longer valid [3].

We make use of the benefits of the Naive Bayes classifier but eliminate some of its problems by treating attributes that are strongly correlated as one attribute. Similar ideas have been pursued by Kononenko [4] and more recently Pazzani [5] who evaluated Cartesian product attributes in a wrapper approach. Other Semi-naive Bayes classifiers are defined by Zheng et al. [6] who use the naive assumption in conjunction with rule construction. Our algorithm differs from previous work in its treatment of continuous attributes. Rather than intervalizing continuous attributes as a preprocessing step and then considering them as categorical ones, we use kernel density estimators [7] to compute probabilities and correlations. Joined attributes are defined by their joined kernel function. No information is lost in the process of joining attributes. The HOBbit distance [8] is used to define intervals that are centered on a particular test point. Intervals are weighted according a Gaussian function. Previous work [4,5] allowed attributes to be joined once. Our algorithm allows for multiple iterations that further join attributes if the correlation is still too high. Since our approach evaluates correlations and density estimators lazily, and does not make use of the wrapper concept, it is suitable to data stream problems in which new data arrive continuously and old data may become invalid.

Section 6.3 presents the concept of kernel density estimation in the context of Bayes' theorem and the Naive Bayes classifier. Section 6.3.1 defines correlations; Section 6.3.2 summarizes P-tree properties; and Section 6.3.3 shows how the HOBbit distance can be used in kernel functions. Section 6.4 presents our results, with Section 6.4.1 introducing the data sets, Section 6.4.2 showing results in general, Section 6.4.3 focusing on a kernel-density-based Naive algorithm, and Section 6.4.4 discussing different parameter choices for the Semi-naive algorithm. Section 6.4.5 demonstrates the performance of our algorithm, and Section 6.5 concludes the paper.

6.3. Naive and Semi-naive Bayes Classifier Using Kernel Density Estimation

Bayes theorem is used in different contexts. In what is called Bayesian estimation in statistics, Bayes theorem allows consistent use of prior knowledge of the data, in form of a prior distribution, and transforming it into a posterior distribution through use of empirical knowledge of the data [1,5]. Using the simplest assumption of a constant prior distribution, Bayes theorem leads to a straightforward relationship between conditional probabilities. Given a class label C with *m* classes, $c_1, c_2, ..., c_m$, and an attribute vector **x** of all other attributes, the conditional probability of class label c_i can be expressed as follows:

$$P(C = c_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C = c_i)P(C = c_i)}{P(\mathbf{x})} \qquad (1)$$

$P(C = c_i)$ is the probability of class label c_i and can be estimated from the data directly. The probability of a particular unknown sample, $P(\mathbf{x})$, does not have to be calculated because it does not depend on the class label and the class with highest probability can be determined without its knowledge. Probabilities for discrete attributes can be calculated based on frequencies of occurrences. For continuous attributes, three alternatives are common. Semi-naive Bayes classifiers are commonly based on a discretization that converts numerical attributes into categorical ones [5]. The most common alternative for Naive Bayes classifiers is a function that collectively summarizes the training points conditional on the class label; see equation (5). This alternative will be called the traditional Naive Bayes classifier. We make use of a third alternative that is based on one-dimensional kernel density estimates as discussed in [7]. Conditional probability $P(\mathbf{x} \mid C = c_i)$ can be written as a kernel density estimate for class c_i

$$P(\mathbf{x} \mid C = c_i) = f_i(\mathbf{x}) \qquad (2)$$

with

$$f_i(\mathbf{x}) = \sum_{t=1}^{N} K_i(\mathbf{x}, \mathbf{x}_t), \qquad (3)$$

where $\mathbf{x}_t$ are training points and $K_i(\mathbf{x}, \mathbf{x}_t)$ is a kernel function.

Estimation of this quantity from the data would be equivalent density-based estimation [1], also called Parzen window classification [7] or, for HOBbit distance-based classification, Podium classification [9]. The Naive Bayes model assumes that, for a given class, the probabilities for individual attributes are independent

$$P(\mathbf{x} \mid C = c_i) = \prod_{k=1}^{M} P(x_k \mid C = c_i), \qquad (4)$$

where x_k is the kth attribute of a total of M attributes. Conditional probability $P(x_k \mid C = c_i)$ can, for categorical attributes, simply be derived from the sample proportions. For numerical attributes, we already mentioned three alternatives. Intervalization leads to the same formalism as categorical attributes and does not need to be treated separately. The traditional Naive Bayes classifier estimates probabilities by an approximation of the data through a function, such as a Gaussian distribution

$$P(x_k \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_k - \mu_i)^2}{2\sigma_i^{\,2}} \right), \tag{5}$$

where μ_i is the mean of the values of attribute x_k averaged over training points with class label c_i and σ_i is the standard deviation. The third alternative is to use a one-dimensional kernel density estimate that comes naturally from (2)

$$f_i(\mathbf{x}) = \prod_{k=1}^{M} f_i^{(k)}(x^{(k)}) = \prod_{k=1}^{M} \left(\sum_{t=1}^{N} K_i^{(k)}(x^{(k)}, x_t^{(k)}) \right), \tag{6}$$

where the one-dimensional Gaussian kernel function is given by

$$K_{i\,Gauss}^{(k)}(x^{(k)}, x_t^{(k)}) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x^{(k)} - x_t^{(k)})^2}{2\sigma_k^2} \right) [C = c_i], \tag{7}$$

where σ_k is the standard deviation of attribute k.

Note the difference to the kernel density estimate even in the case of a Gaussian kernel function. The Gaussian kernel function evaluates the density of training points within a range of the sample point, x. A Gaussian kernel function does not imply that the distribution of weighted training points must be Gaussian. We show that the density-based approximation is commonly more accurate. A toy configuration can be seen in Figure 6.1

in which the Gaussian kernel density estimate has two peaks, whereas the Gaussian distribution function has — as always — one peak only.

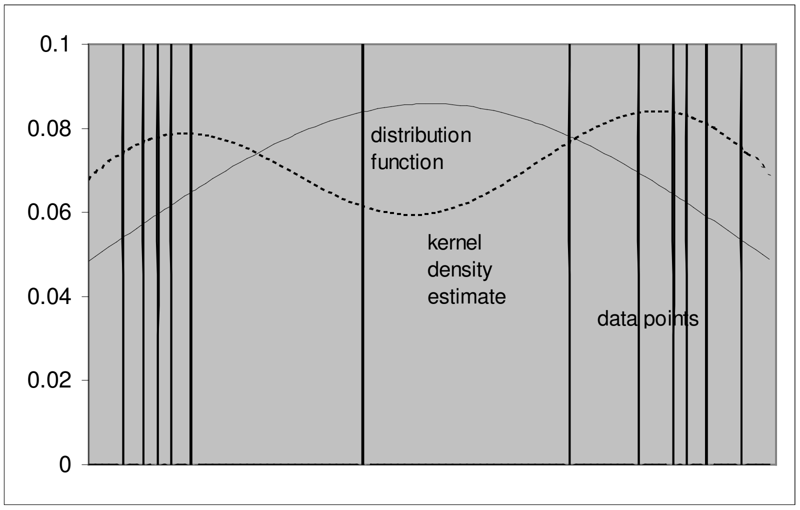

Figure 6.1. Comparison between Gaussian distribution function and kernel density estimate.

Categorical attributes can be discussed within the same framework. The kernel function for categorical attributes is

$$K_{i\,Cat}^{(k)}(x^{(k)}, x_t^{(k)}) = \frac{1}{N_i}[x^{(k)} = x_t^{(k)}], \qquad (8)$$

where N_i is the number of training points with class label c_i. $[\pi]$ 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. The kernel density for categorical attributes is identical to the conditional probability

$$f_{i\,Cat}(x^{(k)}) = \sum_{t=1}^{N} K_{i\,Cat}(x^{(k)}, x_t^{(k)}) = \sum_{t=1}^{N} \frac{1}{N_i}\left[x^{(k)} = x_t^{(k)}\right]\left[C = c_i\right] = P\left(x^{(k)} \middle| C = c_i\right) \qquad (9)$$

The kernel functions in the previous section could, in principle, be evaluated without intervalization. The Naive Bayes classifier only requires the evaluation of one-dimensional kernel densities. These densities could, therefore, be calculated exactly, either in a lazy fashion or by performing a convolution of the kernel function with the data in an eager preprocessing step. When going beyond the naive assumption, the kernel density evaluation has to be done in two or more dimensions, and an exact computation of densities becomes unreasonable in most settings. P-trees and the HOBbit distance that provide a means of making the computation efficient will be introduced.

6.3.1. Correlation of Attributes

We will now go beyond the Naive Bayesian approximation by joining attributes if they are highly correlated. We use correlation functions that are independent of the class label for this purpose rather than doing the analysis for different classes separately. This approach makes the solution numerically more stable by avoiding situations in which two attributes are joined for one class label and not for another. Based on the kernel-density-based formulation, we will define the correlation between two numerical attributes, *a* and *b*, as

$$Corr(a,b) = \frac{\sum_{t=1}^{N} \prod_{k=a,b} K^{(k)}(x^{(k)}, x_t^{(k)})}{\prod_{k=a,b} \sum_{t=1}^{N} K^{(k)}(x^{(k)}, x_t^{(k)})} - 1 \qquad (10)$$

where the kernel function is a Gaussian function for continuous data (7) and for categorical data (8). Note that the traditional Naive Bayesian classifier (5) could not be used in this context since it is based on collective properties. The sum over all data points that is explicit in the above equation is part of the definition of the mean and standard deviation for the traditional Naive Bayes classifier. Without this sum, the numerator and denominator would be equal, and the correlation would vanish by definition.

If the correlation between two attributes exceeds a threshold, typically 0.05-1, the attributes are joined. The kernel function for joined attributes is

$$K^{(a,b)}\left(x^{(a)}, x^{(b)}, x_t^{(a)}, x_t^{(b)}\right) = \sum_{t=1}^{N} \prod_{k=a,b} K^{(k)}(x^{(k)}, x_t^{(k)}) \tag{11}$$

With this definition, it is possible to work with joined attributes in the same way as with the original attributes. Since all probability calculations are based on kernel function, there is no need to state the format of the joined variables themselves explicitly. It also becomes clear how multiple attributes can be joined. The kernel function of joined attributes can be used in the correlation calculation (10) without further modification. In fact, it is possible to look at the joining of attributes as successively interchanging the summation and product in (6). The correlation calculation (10) compares the kernel density estimate at the location of the unknown sample after a potential join with that before a join. If the difference is great, a join is performed. The time complexity for further iterations increases since the kernel function for joined attributes has to be evaluated on the product space of simple attribute values. We, therefore, commonly limit the number of iterations to 3, after which theoretically up to 8 attributes could be joined. In practice, this maximum is rarely reached. An essential ingredient to limiting the complexity of the algorithm is the observation that training point counts do not have to be evaluated for attribute combinations for which the kernel function is small, with a typical threshold taken as 10^{-3}. We will show that multiple iterations can improve accuracy. We will now proceed to discuss the P-tree data structure and the HOBbit distance that make the calculation efficient.

6.3.2. P-trees

The P-tree data structure was originally developed for spatial data [10] but has been successfully applied in many contexts [11,12]. The key ideas behind P-trees are to store data column-wise in hierarchically compressed bit-sequences that allow efficient evaluation of bit-wise AND operations, based on an ordering of rows that benefits compression. The row-ordering aspect can be addressed most easily for data that show inherent continuity such as spatial and multi-media data. For spatial data, for example, neighboring pixels will often represent similar color values. Traversing an image in a sequence that keeps close points close will preserve the continuity when moving to the one-dimensional storage representation. An example of a suitable ordering is Peano, or recursive raster, ordering. If data show no natural continuity, sorting them according to generalized Peano order can lead to significant speed improvements. The idea of generalized Peano order sorting is to traverse the space of data mining-relevant attributes in Peano order while including only existing data points. Whereas spatial data do not require storing spatial coordinates, all attributes have to be represented if generalized Peano sorting is used, as is the case in this paper. Only integer data types are traversed in Peano order since the concept of proximity does not apply to categorical data. The relevance of categorical data for the ordering depends on the number of attributes needed to represent it. Figure 6.2 shows how two numerical attributes are traversed, together with the data sequence and P-tree that is constructed from the sequence of the highest-order bit of x.

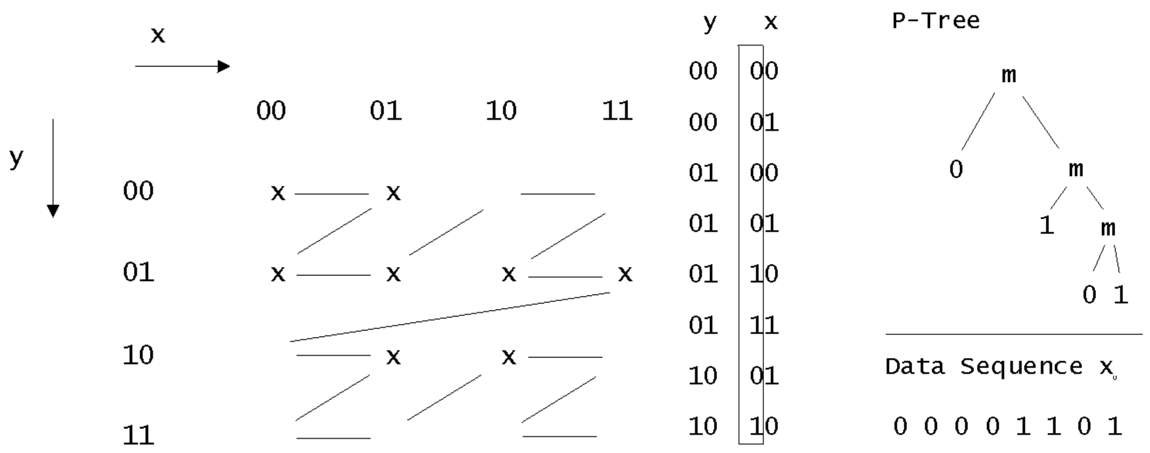

Figure 6.2. Peano order sorting and P-tree construction.

P-trees are data structures that use hierarchical compression. The P-tree graph shows a tree with fan-out 2 in which 1 stands for nodes that represent only 1 values, 0 stands for nodes that represent only 0 values, and m ("mixed") stands for nodes that represent a combination of 0 and 1 values. Only "mixed" nodes have children. For other nodes, the data sequence can be reconstructed based on the purity information and node level alone. The P-tree example is kept simple for demonstration purposes. The implementation has a fan-out of 16 and uses an array rather than pointers to represent the tree structure. It, furthermore, stores the count of 1-bits for all nodes to improve ANDing speed. For data mining purposes, we are mainly interested in the total number of 1-bits for the entire tree, which we will call root count in the following sections.

6.3.3. 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 has the desired property. It is defined as

$$d_{HOBbit}(x_s^{(k)}, x_t^{(k)}) = \begin{cases} 0 & \text{for} \quad x_s^{(k)} = x_t^{(k)} \\ \max\limits_{j=0}^{\infty}\left(j+1 \,\middle|\, \left\lfloor x_s^{(k)} \big/ 2^j \right\rfloor \neq \left\lfloor x_t^{(k)} \big/ 2^j \right\rfloor \right) & \text{for} \quad x_s^{(k)} \neq x_t^{(k)}, \end{cases} \tag{10}$$

where $x_s^{(k)}$ and $x_t^{(k)}$ are the values of attribute k for points $\mathbf{x}_s$ and $\mathbf{x}_t$, and $\lfloor \ \rfloor$ denotes the floor function. The HOBbit distance is the number of bits by which two values have to be right-shifted to make them equal. The numbers 32 (10000) and 37 (10101) have HOBbit distance 3 because only the first two digits are equal, and the numbers, consequently, have to be right-shifted by 3 bits. The first two bits (10) define the neighborhood of 32 (or 37) with a HOBbit distance of no more than 3.

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

$$d_{EH}(x_s^{(k)}, x_t^{(k)}) = \begin{cases} 0 & for \quad x_s^{(k)} = x_t^{(k)} \\ 2^{d_{HOBbit}(x_s^{(k)}, x_t^{(k)})-1} & for \quad x_s^{(k)} \neq x_t^{(k)} \end{cases} \tag{11}$$

With this definition, we can write the one-dimensional Gaussian kernel function in HOBbit approximation as

$$K_{i\,Gauss}^{(k)}(x_s^{(k)}, x_t^{(k)}) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{d_{EH}(x_s^{(k)}, x_t^{(k)})^2}{2\sigma_k^2} \right) [C = c_i] \tag{12}$$

Note that this representation can be seen as a type of intervalization due to the fact that a large number of points will have the same HOBbit distance from the sample. The small number of HOBbit intervals provides a reason why the HOBbit-/P-tree-based algorithm can be efficient despite the high computational effort that multi-dimensional kernel density functions, in particular correlation functions, would otherwise involve.

6.4. Implementation and Results

We implemented all algorithms in Java and evaluated them on 9 data sets. Data sets were selected to have at least 3000 data points and a binary class label. Two-thirds of the data were taken as a training set and one-third as a 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.

6.4.1. Data Sets

Seven of the data sets were obtained from the UCI machine learning library [1] where full documentation on the data sets is available. These data sets include the following:

- spam data set: Word and letter frequencies are used to classify e-mail as spam.
- adult data set: Census data are used to predict whether income is greater than \$50000.
- sick-euthyroid data set: Medical data are used to predict sickness from thyroid disease.
- mushroom data set: Physical characteristics are used to classify mushrooms as edible or poisonous.

- splice data set: A piece of DNA is classified as exon-intron, intron-exon boundary or neither based on 60 nucleotides.
- waveform data set: Waves that have been generated as the combination of 2 of 3 base waves with artificially added noise are to be identified.
- kr-vs.-kp (king-rook-vs.-king-pawn) data set: Configurations on a chess board are used to predict if "white can win".

Two additional data sets were used. A gene-function data set was generated from yeast data available at the web site of the Munich Information Center for Protein Sequences [13]. This site contains hierarchical categorical information on gene function, localization, protein class, complexes, pathways, and phenotypes. One function was picked as a class label, "metabolism." The highest level of the hierarchical information of all properties except function was used to predict whether the protein was involved in "metabolism." Since proteins can have multiple values for localization and other properties, each domain value was taken as a Boolean attribute that was 1 if the protein is known to have the localization and 0 otherwise. A second data set was generated from spatial data. The RGB colors in the photograph of a cornfield are used to predict the yield of the field. The data corresponded to the top half of the data set available at [14]. 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. Table 6.1 summarizes the properties of the data sets.

No preprocessing of the data was done. Some attributes, however, 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.

Table 6.1. Summary of data set properties

	Number of attributes without class label			Number of classes	Number of instances		Un-known values	Proba-bility of minority class label
	total	cate-gori-cal	nume-rical		training	test		
spam	57	0	57	2	3067	1534	no	37.68
crop	3	0	3	2	784080	87120	no	24.69
adult	14	8	6	2	30162	15060	no	24.57
sick-euthyroid	25	19	6	2	2108	1055	yes	9.29
mushroom	22	22	0	2	5416	2708	yes	47.05
gene-function	146	146	0	2	4312	2156	no	17.86
splice	60	60	0	3	2126	1064	no	51.88
waveform	21	0	21	3	3333	1667	no	65.57
kr-vs.-kp	36	36	0	2	2130	1066	no	47.75

6.4.2. Results

We will now compare results of our Semi-naive algorithm with an implementation of the traditional Naive Bayes algorithm that uses a Gaussian distribution function, cf. equation (5), and a P-tree Naive Bayes algorithm that uses HOBbit-based kernel density estimation, cf. equation (12), but does not check for correlations. Table 6.2 gives a summary of the results. Note that the data sets waveform and kr-vs.-kp are artificial datasets. It is well known that artificial data set are not always amenable to the same techniques as real data sets, and results that are gained on them may not be representative [15].

Table 6.2. Results of different Naive and Semi-naive Bayes algorithms.

	Traditional Naïve Bayes	(+/-)	P-tree Naïve Bayes	(+/-)	Semi-naïve Bayes (a)	(+/-)	Semi-naïve Bayes (b)	(+/-)	Semi-naïve Bayes (c)	(+/-)
spam	11.9	0.9	10.0	0.8	8.9	0.8	8.4	0.7	8.0	0.7
crop	21.6	0.2	22.0	0.2	20.7	0.2	20.7	0.2	21.0	0.2
adult	18.3	0.4	18.0	0.3	16.9	0.3	17.0	6.1	16.6	0.3
sick-euthyroid	15.2	1.2	5.9	0.7	4.3	0.6	4.2	0.6	4.6	0.7
mushroom	21.6	0.9	21.6	0.9	0	0	0.1	0.1	0.2	0.1
gene-function	15.2	0.8	15.2	0.8	14.8	0.8	15.0	0.8	15.0	0.8
splice	5.5	0.7	5.5	0.7	5.5	0.7	5.1	0.7	6.3	0.8
waveform	18.2	1.0	18.9	1.1	24.0	1.2	20.4	1.1	18.5	1.1
kr-vs-kp	12.0	1.1	12.0	1.1	10.2	1.0	11.2	1.0	10.0	1.0

6.4.3. P-tree Naive Bayes Classifier

Before using the Semi-naive Bayes classifier, we will evaluate the performance of a Simple Naive Bayes classifier that uses kernel density estimates based on the HOBbit distance. Whether this classifier improves performance depends on two factors.

Classification accuracy should benefit from the fact that a kernel density estimate, in general, gives more detailed information on an attribute than the choice of a simple distribution function. Our implementation does, however, use the HOBbit distance when determining the kernel density estimate. The approximation involved in the use of the HOBbit distance may make classification worse over all. Figure 6.3 shows that for three of the data sets that contain continuous data, the P-tree Naive Bayes algorithm constitutes an improvement over traditional Naive Bayes. It should be noted that the difference between the traditional and the P-tree Naive Bayes algorithm affects exclusively numerical attributes. We will, therefore, get identical results for data sets that are entirely composed of categorical attributes.

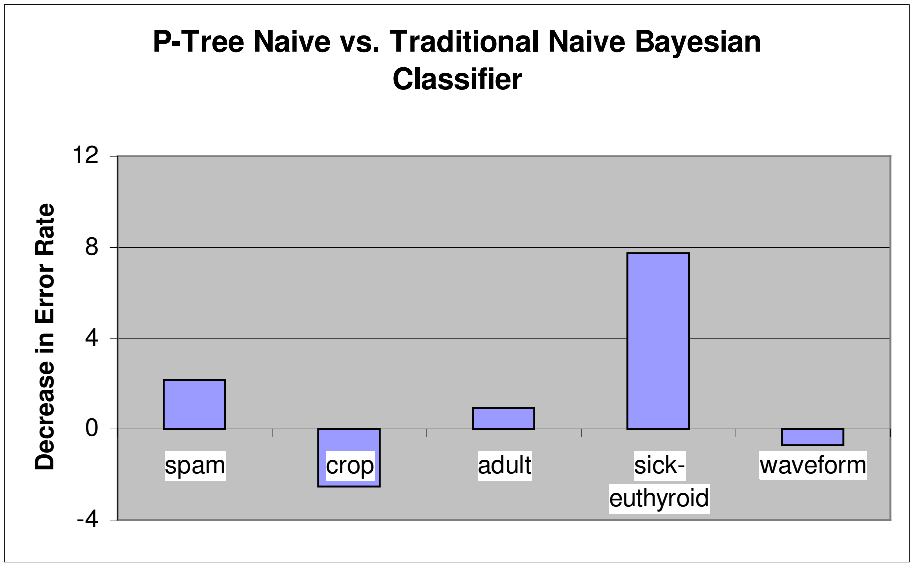

Figure 6.3. Decrease in error rate for the P-tree Naive Bayes classifier compared with the traditional Naive Bayes classifier in units of the standard error.

6.4.4. Semi-naive Bayes Classifier

The Semi-naive Bayes classifier was evaluated using three parameter combinations. Figure 6.4 shows the decrease in error rate compared with the P-tree Naive Bayes classifier, which is the relevant comparison when evaluating the benefit of combining attributes.

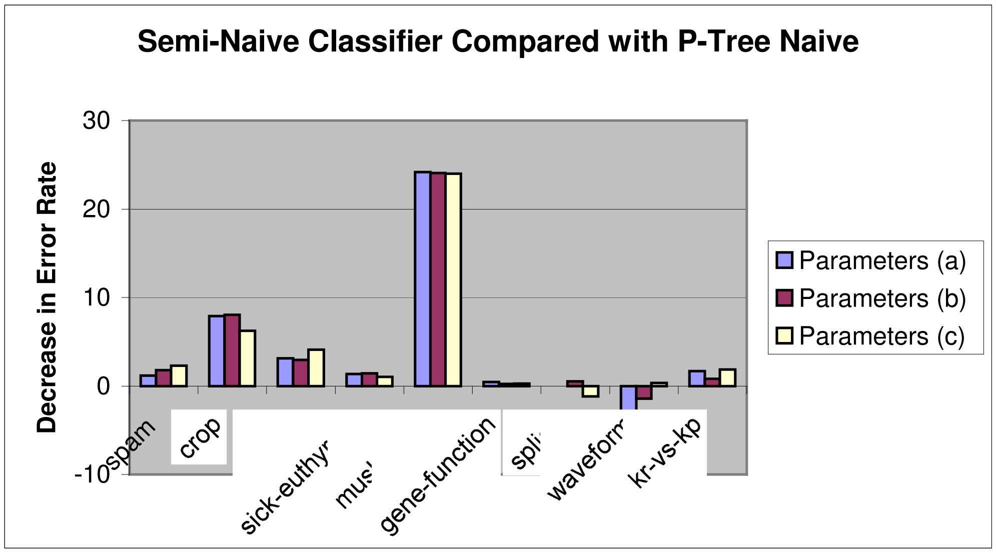

Figure 6.4. Decrease in error rate for 3 parameter combinations for the Semi-naive Bayes classifier compared with the P-tree Naive classifier.

As a first observation it may be interesting to note that, for the majority of data sets, all three parameter combinations lead to a significant improvement. It may seem surprising how little the results depend on the parameter setting, given that the cutoff value for correlations that are considered varies from 0.05 to 1, and runs differ in their treatment of anti-correlations as well as the number of iterations. It can, therefore, be concluded that the benefits of correlation elimination are fairly robust with respect to parameter choice. The only data set that shows a clear decrease in accuracy is the waveform dataset, which is an artificial data set that should be considered with caution.

Run (a) uses a cut-off of threshold t = 1 for correlations that are being eliminated and does 3 iterations using that cut-off. Both other runs use lower thresholds (t = 0.3 for run (b) and t = 0.05 for run (c)) but only do one iteration. Run (b) eliminates not only correlations, i.e., attributes for which $Corr(a,b) > t$, but also anti-correlations, i.e., attributes for which $Corr(a,b) < -t$. Additional runs showed that multiple iterations lead to the best results when the threshold is relatively high, which suggests that joining attributes can be overdone, resulting in lower classification accuracy for decreasing correlation threshold or an increasing number of iterations. It is has been noted that, even correlated attributes, can benefit the Naive Bayes classifier by providing more information [15]. The mechanism by which correlation elimination can lead to a decrease in accuracy in the Semi-naive Bayes algorithm is the following sections. Correlation elimination was introduced as a simple interchanging of summation and multiplication in equation (6). For categorical attributes, the kernel function is a step function. The product of step functions will quickly result in a small volume in attribute space, leading to correspondingly few training points and poor statistical significance. By combining many attributes, the problems of the curse of dimensionality can recur. Similarly, it turned out not to be beneficial to eliminate anti-correlations as extensively as correlations because they will even more quickly lead to a decrease in the number of points that are considered.

The parameter values that were chosen based on these considerations improved accuracy over the P-tree Naive Bayes algorithm for both for data sets with continuous attributes and ones with categorical attributes. This improvement is in contrast to early results of Kononenko [4], which showed little or no improvement. It should be noted that we do include a data set that comes from the same domain as Kononenko's, medical data that are represented by our sick-euthyroid data set. It can be concluded that the kernel-based treatment of continuous variables, which maintains distance information that would be lost by discretization, is indeed very beneficial to the Semi-naive Bayes classifier.

6.4.5. Performance

Data mining has to be efficient for large data sets. Figure 6.5 shows that the P-tree-based Semi-naive 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.

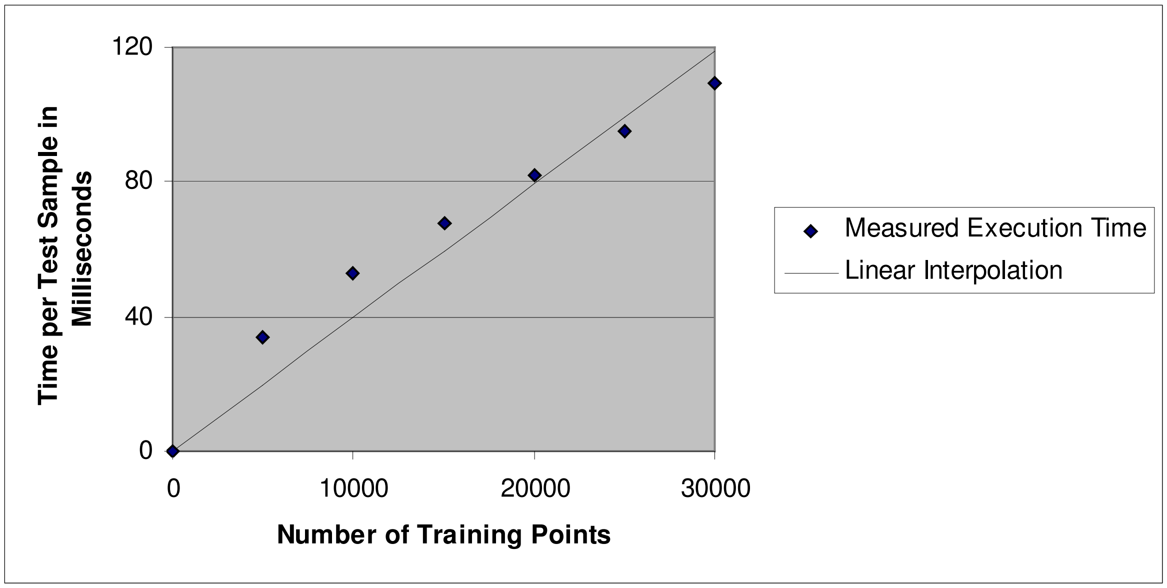

Figure 6.5. Scaling of execution time as a function of training set size.

6.5. Conclusions

We have presented a Semi-naive 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 independent sources. By avoiding discretization, our algorithm ensures that distance information within numerical attributes will be represented accurately. Categorical and continuous data are thereby treated on an equally strong footing, which is unusual for classification algorithms that tend to favor one type of data. The implementation using P-trees has an efficient sub-linear scaling with respect to training set size. Our algorithm is, therefore, not only theoretically interesting, but also has a practical implementation.

6.6. References

[1] D. Hand, H. Mannila, and P. Smyth, "Principles of Data Mining," The MIT Press, Cambridge, MA, 2001.

[2] C.L. Blake and C.J. Merz, "(UCI) Repository of Machine Learning Databases," Irvine, CA, 1998 http://www.ics.uci.edu/~mlearn/MLSummary.html, accessed Oct. 2002.

[3] P. Domingos, and M. Pazzani, "Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier," 13th Int. Conf. on Machine Learning, Morgan Kaufmann, pp. 105-112, 1996.

[4] I. Kononenko, "Semi-naive Bayesian Classifier," 6th European Working Session on Learning, pp. 206-219, 1991.

[5] M. Pazzani, "Constructive Induction of Cartesian Product Attributes," Information, Statistics and Induction in Science, Melbourne, Australia, 1996.

[6] Z. Zheng, G. Webb, and K.M. Ting, "Lazy Bayesian Rules: A Lazy Semi-naive Bayesian Learning Technique Competitive to Boosting Decision Trees," 16th Int. Conf. on Machine Learning, pp. 493-502, 1999.

[7] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," Springer-Verlag, New York, 2001.

[8] M. Khan, Q. Ding, and W. Perrizo, "K-nearest Neighbor Classification of Spatial Data Streams Using P-trees," Pacific Asian Conf. on Knowledge Discovery and Data Mining (PAKDD-2002), Taipei, Taiwan, May 2002.

[9] W. Perrizo, Q. Ding, A. Denton, K. Scott, Q. Ding, and M. Khan, "PINE-Podium Incremental Neighbor Evaluator for Spatial Data Using P-trees," Symposium on Applied Computing (SAC'03), Melbourne, FL, 2003.

[10] Q. Ding, W. Perrizo, and Q. Ding, "On Mining Satellite and Other Remotely Sensed Images," Workshop on Data Mining and Knowledge Discovery (DMKD-2001), pp. 33-40, Santa Barbara, CA, 2001.

[11] W. Perrizo, W. Jockheck, A. Perera, D. Ren, W. Wu, and Y. Zhang, "Multimedia Data Mining Using P-trees," Multimedia Data Mining Workshop with the ACM Conf. on Knowledge Discovery and Data Mining (KDD-2002), Edmonton, Canada, Sept. 2002.

[12] A. Perera, A. Denton, P. Kotala, W. Jockheck, W. Valdivia Granda, and William Perrizo, "P-tree Classification of Yeast Gene Deletion Data," SIGKDD Explorations, Dec. 2002.

[13] Munich Information Center for Protein Sequences, "Comprehensive Yeast Genome Database," http://mips.gsf.de/genre/proj/yeast/index.jsp, accessed Jul. 2002.

[14] W. Perrizo, "Satellite Image Data Repository," Fargo, ND http://midas-10.cs.ndsu.nodak.edu/data/images/data_set_94/, accessed Dec. 2002.

[15] R. Kohavi and G. John, "Wrappers for Feature Subset Selection," Artificial Intelligence, Vol. 1-2, pp. 273-324, 1997.