CHAPTER 4

LAZY KERNEL-DENSITY-BASED CLASSIFICATION USING P-TREES

4.1. Introduction

Kernel functions can be seen as the unifying concept behind many techniques in classification and clustering. This chapter introduces kernel functions and describes a simple implementation based on the concept, which is related to a technique that is known in machine learning as Parzen Window classification [1] and extends previous work using P-trees [2]. Chapters 5 and 6 present two different, related algorithms that can be explained in the kernel-based framework. Section 4.4 compares the results of the classification algorithms of Chapters 5 and 6 with the straightforward kernel-based algorithm of this chapter.

In Chapter 5, a very simple kernel function, a step function, is used and combined with an entropy-based attribute selection scheme that is similar to decision tree induction. It is described in the decision tree/rule-based framework that is well established independently of kernel functions [3,4]. It does, however, differ from those other methods in its treatment of continuous data. Comparable methods take the attitude that continuous data must be discretized before classification starts. Our kernel-based derivation leads to a fundamentally different strategy, in which neighborhoods are determined on the fly. This strategy can be followed efficiently using the HOBBit distance [5] which is designed to suit P-trees well.

A second kernel-based algorithm is the subject of Chapter 6. It assumes attribute independence improves the validity of this assumption by joining attributes, which makes it

a Semi-naive Bayesian classifier. Other Semi-naive Bayes classifiers have been discussed in the literature [6,7] and show many similarities except for, again, the treatment of continuous data. Our kernel-based derivation naturally includes continuous data and does not, therefore, require initial intervalization. This derivation is in accordance with a view of Naive Bayes classifier in the framework of kernel methods [8], which is established in the statistics community but is not commonly used in data mining.

Chapter 7 discusses kernel-based clustering and derives it as an approximation of a technique that is not commonly viewed in this context. It differs from Chapters 5 and 6 in treating a kernel-based representation as a result rather than the starting point.

4.2. Kernel Methods

Kernel methods are commonly explained in the context of support vector machines, renormalization networks such as radial-basis functions, Kriging, and kernel-ridge regression [9]. All of these techniques have in common that a loss-function is minimized to result in a classifier that is optimal according to some definition [10]. This optimization step is not, however, the defining element of kernel methods. Clustering techniques that are well-established in statistics use kernel functions to derive a density estimate [11]. The idea of these techniques is to estimate the data point density from a convolution with a kernel-function that is non-local in attribute space. These techniques have recently been adopted by the data mining community [12], and Chapter 7 of this thesis can be seen as a P-tree based implementation of such an algorithm with some modifications.

Kernel-based density estimates cannot only be used for clustering, but also for classification. A class label value can be predicted from the kernel density estimate of each

of the potential class label values. The class label value that corresponds to the highest density is chosen. This procedure is very similar to k-nearest-neighbor classification (k-NN), in which the density of points is evaluated based on a window that contains precisely *k* neighbors, all of which are weighted equally. Other windows that have a fixed size and may have distance-dependent weighting have been used. The machine learning calls such windows Parzen Windows [1] while the statistics community refers to them as kernels [10].

In general, classification can be used to predict the value of a class label attribute with any discrete domain. Prediction of a continuous attribute is commonly referred to as regression. For the purpose of this thesis, we limit ourselves to binary classification in which the class label attribute can have one of two values, $y = 1$ and $y = -1$, that are represented as 1 and 0 in P-trees. All classification problems can be mapped to binary classification problems [9]. The following function, $f_{simple\_kernel}(x)$, weights the class label values, y_t, of the training points at x_t by the value of the kernel function at point x and takes the difference between the predicted density of training points with $y = 1$ and those with $y = -1$.

$$f_{simple\_kernel}(\mathbf{x}) = \sum_{t=1}^{N} y_t K(\mathbf{x}, \mathbf{x}_t) \qquad (1)$$

$f_{simple\_kernel}(\mathbf{x})$ will be positive if a sample at position **x** is expected to be 1 and negative if it is expected to be -1. One alternative of a kernel is a simple step-like shape in *d* dimensions

$$K(\mathbf{x}_s, \mathbf{x}_t) = \prod_{i=1}^{d} \frac{1}{l^{(i)}} \theta\left(\frac{l^{(i)}}{2} - | x_s^{(i)} - x_t^{(i)} |\right), \qquad (2)$$

where l_i is the length of the box for the i^{th} attribute, $x_s^{(i)}$ is the i^{th} attribute value of the unknown sample, and $x_t^{(i)}$ the respective value of a training point. $\theta(x)$ is the Heaviside

step function, which is 1 for $x > 0$ and 0 for $x < 0$. Alternatively, a smooth function can be chosen such as a Gaussian function with width σ_i.

$$K(\mathbf{x}_s, \mathbf{x}_t) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_s^{(i)} - x_t^{(i)})^2}{2\sigma_i^2}\right) \quad (3)$$

This kernel is a special type of a translation invariant kernel, $K(x,z) = K_{inv}(x-z)$.

Both k-NN and kernel classification rely on the definition of a distance metric. This distance function can be the HOBbit distance, which is particularly suitable to an implementation that uses P-trees, see section 6.2.3. This form of a kernel approach is called Podium classification [2].

4.2.1. Relationship with Support Vector Machines and Other Kernel Methods

k-NN and Parzen Window/Podium classification do not have any mechanism to ensure that even the training points themselves are classified correctly. A loss function can be defined to quantify the degree to which training samples are misclassified. Note that minimization of a loss function requires additional degrees of freedom. To that aim, training points are weighted by a constant, c_t, that can be optimized

$$f(\mathbf{x}) = \sum_{t=1}^{N} c_t K(\mathbf{x}, \mathbf{x}_t) \quad (4)$$

The product, $y_t f(x_t)$, is called the margin of the example with respect to $f(x)$ [9]. For regression problems, the square error of the predicted value is traditionally chosen, but other alternatives were developed by Vapnik for support vector machines, in particular the

ε-insensitive loss function [8]. A common choice for the loss function in classification problems is the misclassification loss function

$$V(y_t, f(\mathbf{x}_t)) = \theta(-y_t f(\mathbf{x}_t)), \tag{5}$$

where θ is the Heaviside step function. The classification error is

$$E = \frac{1}{N}\sum_{t=1}^{N} V(y_t, f(\mathbf{x}_t)) \tag{6}$$

In support vector machines, the error is minimized by an optimization of the objective function [8]

$$H[f] = \frac{1}{N}\sum_{i=1}^{N} V(y_i, f(x_i)) + \lambda \| f \|_K^2 \tag{7}$$

where $\| f \|_K^2$ is the norm of the function, *f*, and λ is a Lagrange multiplier. The choice of λ determines sensitivity to noise. If the training set were noise free, correct classification of training points would be the only goal. In the presence of noise, overfitting can decrease prediction accuracy, and a simple solution, i.e., one with a small norm, may outperform a more complicated one even if the simpler solution misclassifies some training points. Other classification techniques, such as regularization networks, result in very similar optimization problems but differ algorithmically.

A closely related technique is boosting [13,14]. If boosting is performed with Parzen Window classification as base classifier, the resulting algorithm is very similar to algorithms derived from a hill climbing approximation to support vector machine optimization (compare [14] to algorithm 7.1 in [9]). Differences are limited to the precise updating strategies. From a mathematical perspective, support vector machines are derived through a transformation into a high-dimensional space in which the problem is linear.

4.3. Classification Strategies

Different P-tree-based classification algorithms can be derived from the observations in the introduction. We will start by assuming that we use a Podium or kernel function to determine training points that are similar to a test point. This approach immediately implies that different attributes are treated as if they had the same impact on the classification. Such an assumption may be limiting if some attributes do not affect the class label attribute. Note that support vector machines and other kernel techniques also consider attributes as equivalent unless the additional step of kernel optimization is used [15]. Attribute weighting can be introduced as part of the algorithm [16], or independently irrelevant attributes can be eliminated independently in a wrapper approach, such as feature sub-set selection [17].

For the purpose of this thesis, we assume that the kernel function itself is fixed. k-NN does not make this assumption but rather keeps the number of training points in the range of the window fixed. A P-tree-based implementation of k-NN is also possible [5]. The simplest similarity-based classification uses a window that has the shape of a step function; see equation (2). The step-function approach faces the problem that the volume of interest is likely to be empty except if the small number of attributes that are used for classification is very small. It can easily be seen why this problem should occur. To get a rough estimate on the number of training points within the window, we can assume that each attribute independently splits the training set into two equally large subsets. If a data set has 25 attributes, which is the median of attribute numbers used in Chapters 5 and 6, only the fraction $1/2^{25} \sim 3*10^{-8}$ of the data set could be expected to be within the volume. Even for 10^6 training points, which is more than were available in any data set considered,

it would be unlikely to find even a single training point within the range of the window function. This problem is also called the curse of dimensionality [8]. Three solutions to this problem were considered in this thesis. One alternative is to use a Podium function that decreases gradually, such as a Gaussian function, as will be discussed in the remainder of this chapter. The second alternative is to select attributes according to some measure of importance as will be done in Chapter 5. The third approach is to assume some degree of attribute independence, which allows using a factorization approach that is discussed in Chapter 6.

4.3.1. Podium Classification

In this thesis, we will focus on Podium functions that can be written as the product of one-dimensional Podium functions for each dimension. Such a limitation is not necessary. In fact, the original paper on Podium classification [2] did not make that assumption, but rather evaluated the distance between test point and training points by applying the MAX metric to the individual HOBbit distances of the attributes. Note that the Gaussian function using the Euclidean distance between points is equal to the product of Gaussian functions of distances in each dimension.

$$\exp\left(\sum_{i=1}^{d} -\frac{(x_s^{(i)} - x_t^{(i)})^2}{2\sigma^2}\right) = \prod_{i=1}^{d} \exp\left(-\frac{(x_s^{(i)} - x_t^{(i)})^2}{2\sigma^2}\right), \quad (8)$$

where $x_s^{(i)}$ is the i^{th} attribute of a sample point and $x_t^{(i)}$ the corresponding attribute of a training point.

Section 5.2.2 motivates the use of the HOBbit distance, pointing out its benefits in the context of P-tree algorithms

$$d_{HOBbit}(a_s, a_t) = \begin{cases} 0 & \text{for} \quad a_s = a_t \\ \max\limits_{j=0}^{\infty}(j+1 \mid \left\lfloor a_s / 2^j \right\rfloor \neq \left\lfloor a_t / 2^j \right\rfloor) & \text{for} \quad a_s \neq a_t \end{cases} \tag{9}$$

where a_s and a_t are attribute values, and $\lfloor \ \rfloor$ denotes the floor function. The HOBbit distance is, thereby, the number of bits by which two values have to be right-shifted to make them equal. The numbers 32 (10000) and 37 (10101), for example, have HOBbit distance 3 because only the first 2 digits are equal and the numbers, consequently, have to be right-shifted by 3 bits.

When moving to HOBbit distances, we have to make sure that we maintain the general behavior of the Gaussian function, which has been shown to work well as a kernel function in many contexts. The average Euclidean distance over the interval of equal HOBbit distance should, therefore, be used in any Podium calculation. In one dimension, the average Euclidean distance between two values, a_0 and a_1, that have HOBbit distance d_{HOBbit} can be calculated as follows. We know from the definition of the HOBbit distance (9) that $\left\lfloor a_0 / 2^{d_{HOBbit}} \right\rfloor = \left\lfloor a_1 / 2^{d_{HOBbit}} \right\rfloor = a_H$. For $d_{HOBbit} = 0$, both values are identical and, therefore, have Euclidean distance 0. For $d_{HOBbit} \neq 0$, we assume without loss of generality that value a_0 satisfies $\left\lfloor a_0 / 2^{d_{HOBbit}-1} \right\rfloor - 2a_H = 0$ and a_1 satisfies $\left\lfloor a_1 / 2^{d_{HOBbit}-1} \right\rfloor - 2a_H = 1$. With $m = 2^{d_{HOBbit}-1}$, a_0 can be written as $a_0 = 2^{d_{HOBbit}} a_H + k$ with $k \geq 0 \wedge k < m$ and a_1 as $a_1 = 2^{d_{HOBbit}} a_H + m + l$ with $l \geq 0 \wedge l < m$. The average distance is, therefore,

$$d_{Average} = \frac{1}{m^2} \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} (a_1 - a_0) = \frac{1}{m^2} \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} (m + l - k) = m = 2^{d_{HOBbit}-1} \tag{10}$$

Note that we only have to evaluate the one-dimensional average since the multi-dimensional kernel function is defined as the product of one-dimensional kernel functions.

It is useful to introduce an exponential HOBbit distance, d_{EH}, that approximates a Euclidean distance and is defined as

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

Note that the values of d_{EH} and d_{HOBbit} are identical for distances 0 and 1. We can, therefore, easily discuss categorical data in the same framework. Data points for which training and sample values of a categorical attribute are equal (different) are considered to have the distance $d_{EH}(a_s,a_t) = d_{HOBbit}(a_s,a_t) = 0$ ($d_{EH}(a_s,a_t) = d_{HOBbit}(a_s,a_t) = 1$) for that attribute. Treating categorical attributes in the same framework means that training points with differing categorical attribute values will contribute to the classification due to the fact that the Gaussian function is non-zero at finite distances. This property is important because we would otherwise still suffer from the curse of dimensionality. We can regain a kernel function that is 0 for differing categorical attribute values by taking the limit $\sigma \to 0$ for the Gaussian kernel.

The exponential HOBbit distance can be used in place of the Euclidean distance for individual dimensions without changing the equality in (8)

$$\exp\left(\sum_{i=1}^{d} -\frac{d_{EH}(x_s^{(i)}, x_t^{(i)})^2}{2\sigma^2} \right) = \prod_{i=1}^{d} \exp\left(-\frac{d_{EH}(x_s^{(i)}, x_t^{(i)})^2}{2\sigma^2} \right) \tag{12}$$

It is sometimes preferable to use the Manhattan distance instead of the Euclidean distance, especially in high dimensions. Since distance functions are not explicitly considered, using the Manhattan distance requires finding a function for which an equivalent equality holds; i.e., the product of its evaluation on individual dimensions should be equal to the function when evaluated for the total Manhattan distance. The

exponential function has the desired property. We can again use the exponential HOBbit distance instead of the Euclidean distance within individual dimensions

$$\exp\left(\sum_{i=1}^{d} -\frac{d_{EH}(x_s^{(i)}, x_t^{(i)})}{\tau}\right) = \prod_{i=1}^{d} \exp\left(-\frac{d_{EH}(x_s^{(i)}, x_t^{(i)})}{\tau}\right) \tag{13}$$

We generalize our approach slightly by allowing the width of the Gaussian function, σ, and exponential function, τ, respectively, to take different values for each dimension. This generalization can be seen as independent scaling of different dimensions. For a Gaussian window, it leads to a kernel function

$$K_{Gauss}(\mathbf{x}_s, \mathbf{x}_t) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_i} \exp(-\frac{d_{EH}(x_s^{(i)}, x_t^{(i)})^2}{2\sigma_i^2}) \tag{14}$$

For an exponential decay window, the kernel function is

$$K_{Exp}(\mathbf{x}_s, \mathbf{x}_t) = \prod_{i=1}^{d} \frac{1}{\tau_i} \exp(-\frac{d_{EH}(x_s^{(i)}, x_t^{(i)})}{\tau_i}) \tag{15}$$

The prediction is done using

$$f_{Podium}(\mathbf{x}) = \sum_{t=1}^{N} y_t K(\mathbf{x}, \mathbf{x}_t) \tag{16}$$

with the predicted class label being 1 if $f_{Podium}(x)$ is positive and -1 one if it is negative.

This approach successfully resolves the problem of possibly not having data points in the volume that is used to decide on the class label. For weight functions that are non-zero in the entire space, all training points would contribute to f_{Podium}, albeit with varying prefactors. In practice, the kernel function is commonly chosen to be 0 when it falls below some threshold to limit the computational effort. Unfortunately, this choice does not fully resolve the problem of computational inefficiency. When discussing the possible choice of a step function as a kernel function, we made an estimate on how space would be split up

for a typical scenario of 25 attributes and a kernel function that divides space in 2 sections. We can now reinterpret the result as stating that, in a setting with 25 binary attributes, the attribute space would consist of about $3*10^7$ small volumes. Many of those volumes will not require the computation of an AND because the Podium function falls below the threshold. Many others will have no training points in them, resulting in a count of 0 that can commonly be evaluated much faster than a non-zero count due to the structure of P-trees. The resulting P-tree algorithm nevertheless shows inherently exponential scaling with the number of attributes, which requires some modifications for any data set that has more than very few attributes, typically 8-10. Note that this problem does not occur when the MAX metric is used to combine HOBbit dimensions instead of the Manhattan or the Euclidean metric as is the case in [2]. For categorical data, the MAX metric leads to only two volumes, one with points for which all attributes match the unknown sample and one for which that is not the case. The use of the MAX metric without an additional attribute selection scheme is, therefore, limited to continuous data.

The complexity considerations require limiting the number of attributes to the most important ones. Which attributes are considered most important depends on the test point. Information gain is used based on the weighted density of points. Note that the derivation of information gain described in the next chapter cannot strictly be applied to a setting with continuous attribute densities. The exact formulation that is derived in Appendix B can, therefore, not be used. The standard definition (See "*Info*" and "*InfoGain*" in Section 5.2.4.) can, however, be applied successfully even in this general case.

4.4. Experimental Results

Sections 5.3 and 6.3 describe data sets and experimental setup in some detail. At this point, it is interesting to see how the Podium method described earlier and the two other techniques compare. Figure 4.1 compares the accuracy of all three algorithms. For the Podium implementation, eight attributes with the highest information gain were selected. The width of the Gaussian function, σ, was chosen to be one-half of the standard deviation of each attribute for continuous attributes. For categorical attributes, σ was chosen to be 1. The results are compared with the rule-based results for 20 paths (Section 5.3.3.) and the Semi-naive Bayes classifier of Section 6.3.4. It can be seen that, for most data sets, the rule-based algorithm performs best. The straightforward Podium implementation is most competitive on data sets with few attributes, such as the crop data set with only three attributes.

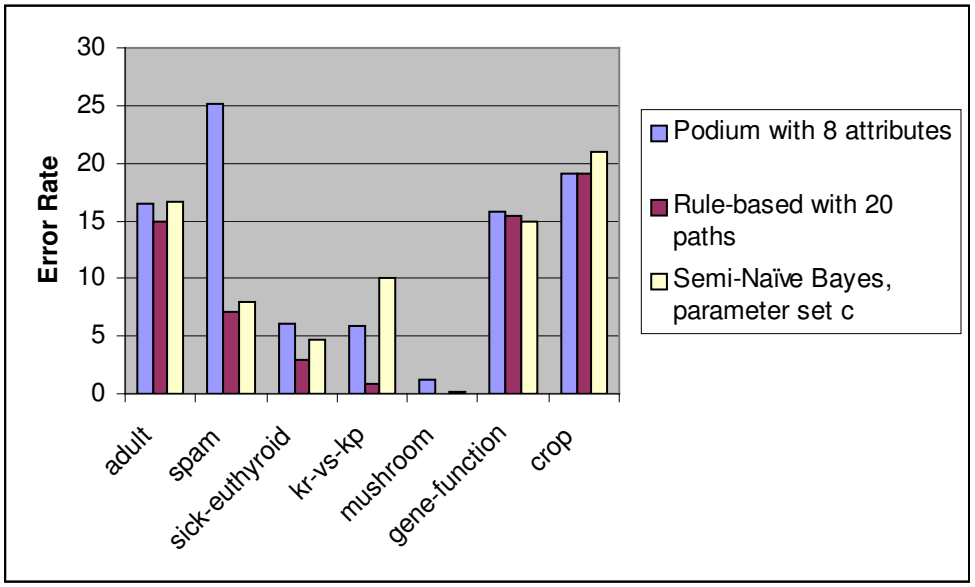

Figure 4.1. Error rate comparison among the algorithms of Chapters 4-6.

The following three chapters represent papers that have been published or are intended for publication. The paper-style organization made a certain amount of redundancy unavoidable and also motivated a choice of notation that seemed clearest in the given context. To assist in the transition between chapters, some remarks have been collected in the following section.

4.5. Remarks to Simplify the Transition Between Chapters 4, 5, and 6

The following sections are redundant or subsets of each other

- Sections 5.3.1 and 6.3.2 on P-trees are identical.
- Section 6.3.3 repeats material that has been covered in Sections 4.5 and 5.3.2.
- Section 5.4.1 on data sets is a subset of Section 6.4.1. Section 6.4.1 describes two additional data sets, the splice and the waveform data set, which have a class label domain of size three.

4.5.1. Kernel Formulation for Chapter 5

The kernel function notation of Chapters 4 and 6 could be used in Chapter 5 as well. The kernel function that is used for n attributes that have been considered relevant by information gain comparison and information gain cutoff is

$$K(\mathbf{x}_s, \mathbf{x}_t) = \prod_{i=1}^{n} \frac{1}{2^{l_{HOBbit}^{(i)}}} [l_{HOBbit}^{(i)} \geq d_{HOBbit}(x_s^{(i)}, x_t^{(i)})], \quad (17)$$

where $[\pi]$ is 1 if predicate π holds and 0 otherwise.

4.5.2. Class Label Domain

Whereas Chapters 4 and 5 assume binary class labels, this assumption is not made for Chapter 6, where class labels with larger domains are also used. Based on equation (2) of Chapter 6,

$$P(\mathbf{x} \mid C = c_i) = f_i^{chapter6}(\mathbf{x}) = \sum_{t=1}^{N} K_i(\mathbf{x}, \mathbf{x}_t) \tag{18}$$

equation (1) in this chapter would be written as

$$f_{simple\_kernel}(\mathbf{x}) = f_1^{chapter6}(\mathbf{x}) - f_{-1}^{chapter6}(\mathbf{x}) \tag{19}$$

4.5.3. Treatment of Categorical Data

Chapter 4 assumed that categorical data use the same Gaussian kernel function as continuous data, with differing categorical attributes having distance 1 and identical ones having distance 0. This choice leads to a non-zero contribution to the kernel function for differing categorical attributes, which is desirable in the light of the curse of dimensionality. In Chapter 6, we do not have to use this computationally intensive choice but can use a simple kernel function that is 0 for differing attributes because only a few attributes are combined. The corresponding kernel function for categorical data can be seen the limiting case of $\sigma \to 0$ in (13).

4.6. References

[1] R. Duda, P. Hart, and D. Stork, "Pattern Classification," 2nd Edition, John Wiley and Sons, New York, 2001.

[2] 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.

[3] J. Ross Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[4] J. Friedman, R. Kohavi, and Y. Yun, "Lazy Decision Trees," 13th National Conf. on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference, 1996.

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

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

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

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

[9] N. Cristianini and J. Shawe-Taylor, "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods," Cambridge University Press, Cambridge, England, 2000.

[10] E. Allwein, R. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000.

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

[12] A. Hinneburg and D. A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98), New York, pp. 58-65, Aug. 1998.

[13] R. Shapire, "The Strength of Weak Learnability," Machine Learning, Vol. 5 No.6, pp. 197-227, 1990.

[14] Y. Freund and R. Schapire, "Experiments with a New Boosting Algorithm," 13th Int. Conf. on Machine Learning, pp. 148-156, Morgan Kaufmann, 1996.

[15] K. Bennett, M. Momma, and M. Embrechts, "A Boosting Algorithm for Heterogeneous Kernel Models," ACM Conf. for Knowledge Discovery and Data Mining (SIGKDD '02), Edmonton, Alberta, Canada, 2002.

[16] S. Cost and S. Salzberg, "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," Machine Learning, Vol. 10, pp. 57-78, 1993.

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