To the Graduate Council:

I am submitting herewith a thesis written by David Lawrence Beck entitled "Multiple Resolution Nonparametric Classifiers." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Jens Gregor
Major Professor

We have read this thesis
and recommend its acceptance:

Michael Berry

Michael Thomason

Accepted for the Council:

Linda Painter
Interim Dean of Graduate Studies

(Original signatures are on file with student records.)

MULTIPLE RESOLUTION NONPARAMETRIC CLASSIFIERS

A Thesis Presented
for the Master of Science
Degree
The University of Tennessee, Knoxville

David Lawrence Beck

December 2006

Copyright © 2006 by David Lawrence Beck
All rights reserved

Acknowledgments

I wish to express my appreciation to my wife and family for their patience and support and to my parents for inspiring me to enjoy learning. My thanks also goes to Dr. Jens Gregor for his guidance and instruction and Dr.'s Berry and Thomason for serving on my committee. This work was funded in part by the United States Naval Warfare Center Weapons Division, China Lake, to whom I express my gratitude.

Abstract

Bayesian discriminant functions provide optimal classification decision boundaries in the sense of minimizing the average error rate. An operational assumption is that the probability density functions for the individual classes are either known *a priori* or can be estimated from the data through the use of estimating techniques. The use of Parzen-windows is a popular and theoretically sound choice for such estimation. However, while the minimal average error rate can be achieved when combining Bayes Rule with Parzen-window density estimation, the latter is computationally costly to the point where it may lead to unacceptable run-time performance. We present the Multiple Resolution Nonparametric (MRN) classifier as a new approach for significantly reducing the computational cost of using Parzen-window density estimates without sacrificing the virtues of Bayesian discriminant functions. Performance is evaluated against a standard Parzen-window classifier on several common datasets.

TABLE OF CONTENTS

Chapter Page

I. **INTRODUCTION**.....1
Overview of Classification Techniques.....1
Probabilistic Approaches to Classification.....2
Bayes Rule.....4
Overview.....8

II. **PROBABILITY DENSITY ESTIMATING TECHNIQUES**.....9
Parametric Approaches.....9
Nonparametric Approaches.....10
Parzen-windows.....11
Nearest Neighbor.....15

III. **MULTIPLE RESOLUTION NONPARAMETRIC CLASSIFICATION**.....17
Mathematical Foundations.....17
Multiple Resolution Editing.....20
Multiple Resolution Condensing.....24

IV. **RESULTS**.....36
Two Dimensional Examples.....36
Ripley's Data Set.....36
Banana Data Set.....43
High Dimensional Examples.....49
Summary of Results.....50

V. **CONCLUSION**.....55

BIBLIOGRAPHY.....57

VITA.....61

LIST OF FIGURES

Figure Page

1. The Decision Boundary of a Two Class Problem.....6
2. The Discriminant Function of a Two Class Problem.....7
3. The Discriminant Function of a Two Class problem, View 2.....7
4. Parzen-Window Classification with a Resolution of 3.125.....13
5. Parzen-Window Classification with a Resolution of 7.41.....13
6. Parzen-Window Classification with a Resolution of 25.....14
7. Parzen-Window Classification with a Resolution of 200.....14
8. MRN Output with 3 Samples.....28
9. MRN Output with 4 Samples.....28
10. MRN Output with 5 Samples.....29
11. MRN Output with 6 Samples.....29
12. MRN Output with 7 Samples.....30
13. MRN Output with 8 Samples.....30
14. MRN Output with 9 Samples.....31
15. MRN Output with 10 Samples.....31
16. MRN Final Output with 10 Samples.....32
17. Schematic Showing Resolutions and Alpha Values.....34
18. MRN Output on 200 Samples of Ripley's Data Set.....37
19. MRN Output on 400 Samples of Ripley's Data Set.....38
20. MRN Output on 600 Samples of Ripley's Data Set.....39
21. MRN Output on 800 Samples of Ripley's Data Set.....40
22. MRN Output on 1000 Samples of Ripley's Data Set.....41
23. MRN Output on 200 Samples of the Banana Data Set.....44
24. MRN Output on 400 Samples of the Banana Data Set.....45
25. MRN Output on 600 Samples of the Banana Data Set.....46

26. MRN Output on 800 Samples of the Banana Data Set.....47
27. MRN Output on 1000 Samples of the Banana Data Set.....48

CHAPTER I
INTRODUCTION

Overview of Classification Techniques

The use of computers in pattern recognition and classification is prevalent and has become an issue of widespread visibility. Classification problems vary widely in scope and objective. What is being classified? How many classes exist? How quickly must the objects be classified? How accurately must they be classified? How many samples are available? What kinds of samples are available and how are the objects represented? No single solution will be appropriate for all classification tasks and, because of this, numerous classification algorithms exist.

In general there are two broad categories of feature-based classification algorithms: probabilistic and non-probabilistic. However, the delineation between these two is often fuzzy. Many non-probabilistic algorithms are closely tied to probabilistic algorithms and vary mainly in the computational approach used to optimize the solution.

A particularly popular non-probabilistic approach is the support vector machine. Under this paradigm, the classification problem is viewed from the perspective of finding a boundary that separates or nearly separates all of the samples. Support vector machines allow for the use of nonlinear boundary surfaces through the use of kernels. Support vectors are samples that lie near the decision boundary. The decision boundary is determined by a weighted summation of distance measurements between the sample and the support vectors (Cortes and Vapnik, 1995) giving us the following general equation for the support vector machine discriminant:

$$f(x)=\sum_{i=1}^{N} y_i a_i K(x, x_i) \tag{1}$$

where $y_i \in [+1, -1]$ represents the class label, N is the number of support vectors, K is the kernel and a_i is the weighting factor for support vector i. Recognizing each support vector is fixed at the end of the training phase, for a two-class problem we can rewrite this as:

$$f(x)=\sum_{i \in C1} a_i f_i(x) - \sum_{i \in C2} a_i f_i(x) \tag{2}$$

where $f_i(x)=K(x, x_i)$ and *C1* and *C2* consist of all support vectors for class one and class two respectively. If the support vector kernel is a valid probability density function (such as the popular Gaussian kernel), this is remarkably similar to the general formula for the Bayesian discriminant given a fixed sample, two-class Parzen-window probability density estimate (Duda et al, 2001). Similarly, a finite mixture model is represented by the same general mathematical formula (Archambeau et al, 2004).

Obviously probabilistic and non-probabilistic approaches share similarities. The differences then lie in the constraints placed upon the weights, functions and samples, the selection and number of functions involved and how the weight optimization is performed.

Probabilistic Approaches To Classification

Probabilistic approaches typically view the object to be classified as a random variable that can assume one of several discrete values (the classes). The behavior of a discrete random variable can be described by a probability mass function, *P(x)*. To be valid a

probability mass function must meet the following criteria:

$$P(x) \geq 0 \tag{3}$$

and

$$\sum_{x \in X} P(x) = 1 \tag{4}$$

The statistical independence of two variables is then, by definition:

$$P(x, y) = P(x) P(y) \tag{5}$$

Intuitively, this indicates that prior knowledge of one variable gives no information with respect to the occurrence of the second variable.

Conditional probability on the other hand is defined by:

$$P(x|y) = \frac{P(x, y)}{P(y)} \tag{6}$$

Intuitively, this is the likelihood of x occurring given that y has already occurred. Assuming the two variables are not independent, knowledge of *y* presents useful information with respect to *x*.

The law of total probability states:

$$P(y) = \sum_{x \in X} P(x, y) \tag{7}$$

This indicates the total likelihood of y occurring is equivalent to the sum of the likelihood of *x* and *y* occurring for all *x*.

Typically we discuss variables that assume a finite set of discrete values in terms of probability mass functions (as above). When the random variable is continuous we describe it using probability density functions (PDF). The previous definitions hold for PDF's as well by merely replacing the summations with integrals. We distinguish between probability mass functions and probability density function by the use of uppercase for the former and lowercase for the latter. However, for the remainder of the paper we will, for simplicity, assume all variables are continuous and can be described by a PDF.

An intuitive choice for classification would be to select the most probable class given the available information. Hence, to perform classification of an object, various quantifiable features of the object are selected and these features, along with the class, are assumed to be random variables associated with a given PDF. The variable of most interest in a classification problem is the class and the goal then becomes to determine the likelihood that an object belongs to a certain class given the features of the object.

Bayes Rule

Training of a classifier then becomes the search for $p(c|y)$ where y is a feature of the object to be classified and c is the class. If $p(c,y)$ and $p(y)$ were known, this calculation would be straightforward. Unfortunately, the joint probability $p(c,y)$ is often difficult to calculate in practice. However, by rearranging terms in the law of total probability and the definition of conditional probability we can derive Bayes rule:

$$p(c|y)=\frac{p(y|c)\,p(c)}{p(y)} \tag{8}$$

This is one of the most crucial theorems when performing probabilistic classification since it allows us to calculate the probability of an object's class given its features.

Given a class whose feature PDF is known, we can easily calculate the likelihood that a new sample with a given feature value will be a member of that class. It can be proven (Duda et al, 2001) that the minimum error rate for a classification problem is given when each object is classified according to its most probable class. The mathematical equation that assigns new samples to their most probable class is the Bayesian discriminant.

The Bayesian discriminant, given equal priors, is:

$$g(x) = p(c_1|y) - p(c_2|y) \tag{9}$$

The decision boundary is defined as the collection of points in feature space at which the classes are equally likely. Hence the decision boundary is given by:

$$p(c_1|y) - p(c_2|y) = 0 \tag{10}$$

Intuitively, the decision boundary is the point at which our classification decision changes. Figure 1 shows the decision boundary for a two-class problem where the feature PDFs are known to be Gaussian. In this example the PDF for each class has a different mean and covariance matrix. Figures 2 and 3 graph the discriminant function in 3-D form. The 3-D graphs more clearly show the difference between the class variances and give greater information regarding each class PDF.

Obviously, if we know the exact feature PDF for each class in a classification problem, it is simple to calculate Bayes rule and the decision boundary. Unfortunately, it is often the case that the feature PDF for each class is unknown which means the search for a probabilistic classification algorithm often leads to the search for a probability density estimation algorithm.

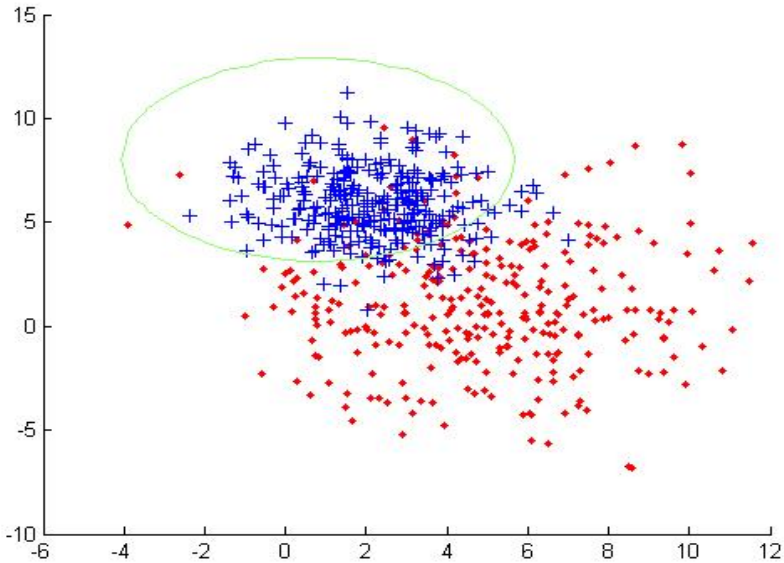

Figure 1 - The Decision Boundary of a Two Class Problem. Each class is determined by a bi-variate Gaussian PDF. Class 1 (red) is $N(\mu = (5,1); \sigma^2 = 3)$. Class 2 (blue) is $N(\mu = (2,6); \sigma^2 = 1.5)$.

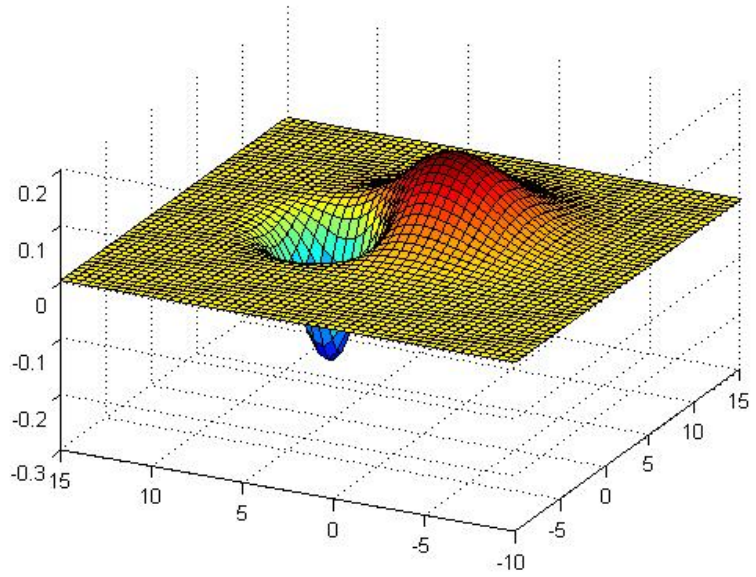

Figure 2 – The Discriminant Function of a Two Class Problem.

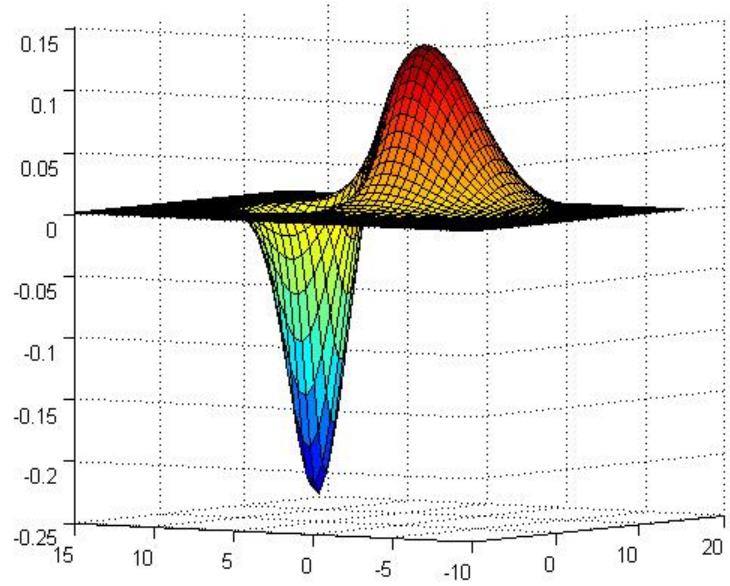

Figure 3 – The Discriminant Function of a Two Class problem, View 2. This view clearly shows the difference in PDF between the two classes.

Overview

This search for effective density estimation has led to the development of many different algorithms. One of the most promising, from a theoretical standpoint, is the Parzen-window approach. Unfortunately, Parzen-window classifiers suffer from poor model-complexity and run-time efficiency. The Multiple Resolution Nonparametric (MRN) classifier we introduce in Chapter III eliminates the issues of model-complexity and run-time performance while preserving the theoretical and practical advantages of Parzen-window density estimation and the ensuing Bayesian discriminant. In Chapter II we discuss the background behind density estimation in general and discuss the nonparametric estimating techniques that form the basis for MRN classifiers in detail. Chapter III demonstrates the mathematical foundations and details the algorithm. Chapter IV compares the results of an MRN classifier with a standard Parzen-window classifier on several data sets and Chapter V provides a summary of results and conclusions.

CHAPTER II
PROBABILITY DENSITY ESTIMATING TECHNIQUES

Parametric Approaches

Probability density estimating techniques can also be grouped into two main categories: parametric and non-parametric techniques. Again, there is some crossover in that, typically, non-parametric techniques use a sum of functions to estimate the probability. Often these functions are the same probability functions used in parametric techniques. The major difference being that in parametric density estimation we assume the entire class is governed by a known distribution form.

Parametric techniques assume the general form of the feature PDF is known. Only the specific parameters of the PDF are unknown. Often these techniques will assume PDFs with desirable computational attributes such as Gaussian, Rayleigh or Gamma distributions.

A parametric probabilistic classification algorithm then proceeds by attempting to estimate the PDFs governing parameters based on a training set. The Bayesian discriminant is then used to create a decision rule and decision boundary.

Several techniques exist for parameter estimation: Maximum-Likelihood, Bayesian, Kalman filter, etc. However, each of these techniques assumes the PDF form is known *a priori*. When this assumption is incorrect, the resulting error rates may be much higher than expected since the true form of the PDF may vary greatly from the assumed form.

Nonparametric Approaches

Because the true form of the PDF is typically unknown, non-parametric techniques are frequently used in probabilistic classification algorithms as a means to estimate the true PDFs.

Several non-parametric techniques exist for estimating probability densities. Most of these techniques are based upon the observation that the probability that a feature will fall within a certain range of values is given by:

$$P_r = \int_a^b p(x)dx \tag{11}$$

Here we define P_r as the probability that x will fall in region R where R=(a,b). If we assume the region R is small enough that the probability within R does not vary significantly the following estimate can be used for P_r:

$$P_r \approx p(x)V \tag{12}$$

Here, *p(x)* is the probability of some point *x* in R and V is the volume enclosed by R. It can be shown (Duda et al, 2001) that *k/n* where *k* is the number of samples falling in R and *n* is the total number of samples is an accurate estimate of P_r which leads to the following estimate for *p(x)*:

$$p_n(x) \simeq \frac{k/n}{V} \tag{13}$$

Likewise, it can be proven (Duda et al, 2001) that this estimate will converge to the true

probability *p(x)* in the limit as $n \to \infty$ if the following conditions hold:

$$* \quad \lim_{n \to \infty} V_n = 0 \tag{14}$$

$$* \quad \lim_{n \to \infty} k_n = \infty \tag{15}$$

$$* \quad \lim_{n \to \infty} k_n / n = 0 \tag{16}$$

The most common means of ensuring these conditions are by specifying an initial volume, V_0 which shrinks as some function of n or by specifying the number of samples, k_0, in the initial volume and increasing k as some function of n.

Parzen-windows

Classical Parzen-windows proceed by using a fixed volume "window function" to determine k / n. Often, Gaussian kernels are used to approximate k / n and, in fact, any valid probability density function could be used as the window function (Duda et al, 2001). A window size or resolution is used to determine the range of effect each sample will have on its neighbors. Resolution is defined as $1/h$ where h is the window size. Hence, higher resolutions (smaller window sizes) will diminish the effect each sample has on it's neighbors. Likewise, lower resolutions (larger window sizes) increase the effect of each sample upon its neighbors.

We note that, to guarantee convergence, Eqn. (14) demands that window size must decrease as sample size increases. This means resolution increases as sample size increases. Hence, in theory the selection of a resolution should be trivial since the estimate will converge at some point regardless of how slowly the resolution increases. However, with finite sample sizes this is not the case and the resolution dramatically affects the resulting discriminant function and decision boundary.

Figures 4-7 illustrate a progression of resolution from low to high and its effect on discriminant functions and decision boundaries derived from Parzen-window density estimates. The plot on the left displays samples from two classes and the Bayesian decision boundary. The plot on the right gives a three-dimensional view of the Bayesian discriminant function.

It can be seen that all training samples are correctly classified in figure 7 whereas several are incorrectly classified in figure 4. This observation that, on finite sets, increasing resolution tends to increase the number of correctly classified training samples helped motivate the Multiple Resolution Nonparametric classifier approach presented here. The details of Parzen-window density estimation and resolution are discussed in more depth in Chapter III.

The computational complexity associated with Parzen-window based PDF estimation is $O(n)$ for classification where n is the size of the training set. This is intuitive since the estimate is accomplished by calculating the summation of a distance calculation between the new sample and every sample in the training set. The classification time thus increases linearly with respect to the size of the training set. However, as the sample size increases the largest number of samples will be located in the areas of least interest (areas of high single-class density), not areas of high interest (near classification boundaries). Unfortunately this means that, as the sample size increases, the majority of samples do not provide useful information with respect to the boundary and, in practice, the model complexity quickly overwhelms any benefit to the theoretical computation complexity.

Several methods have been suggested to reduce the size of the Parzen-window training set. One method proposed reducing the Parzen training set while maintaining an overall accuracy by minimizing L_2 error (Girolami & He, 2003). However, this does not ensure any optimality with respect to classification error rate and may place undue emphasis on regions distant to the decision boundary, requiring both greater computation

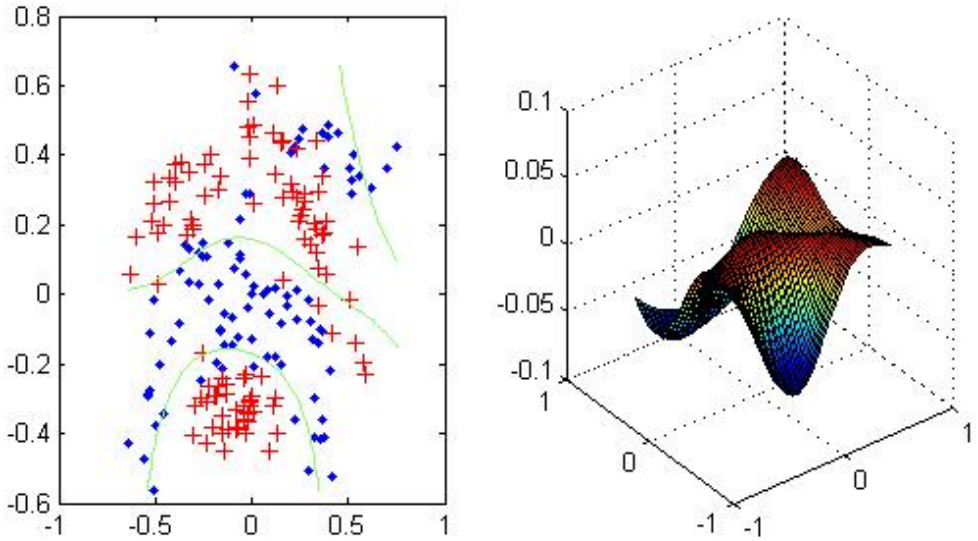

Figure 4 – Parzen-Window Classification with a Resolution of 3.125

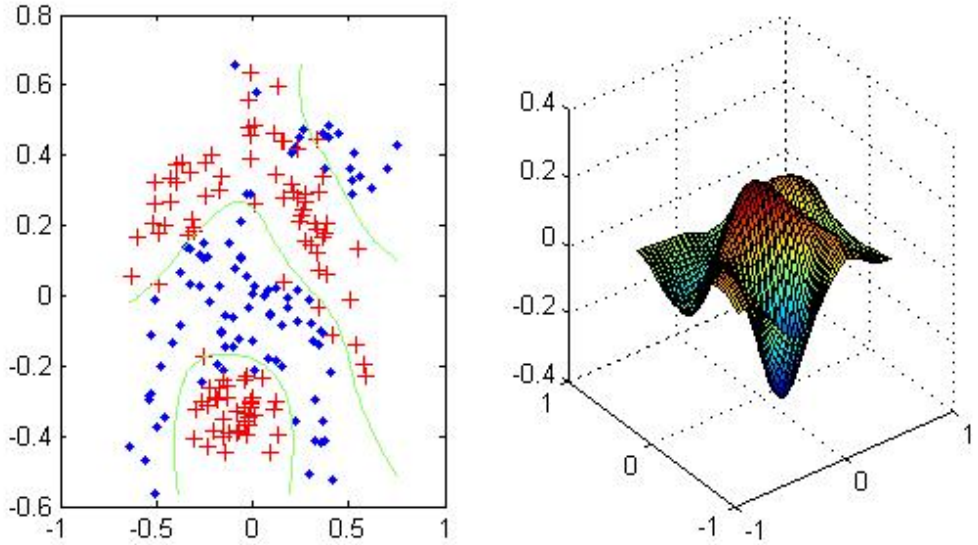

Figure 5 – Parzen-Window Classification with a Resolution of 7.41

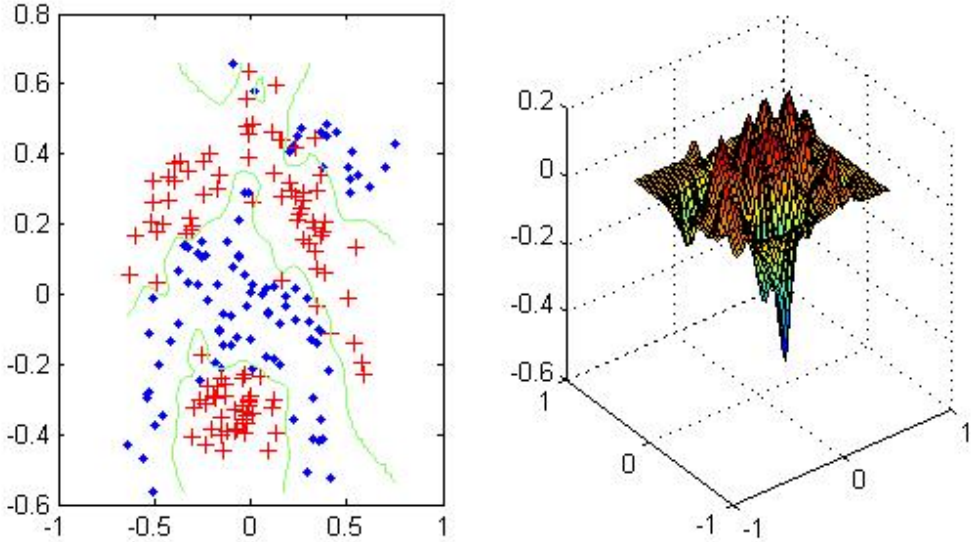

Figure 6 – Parzen-Window Classification with a Resolution of 25

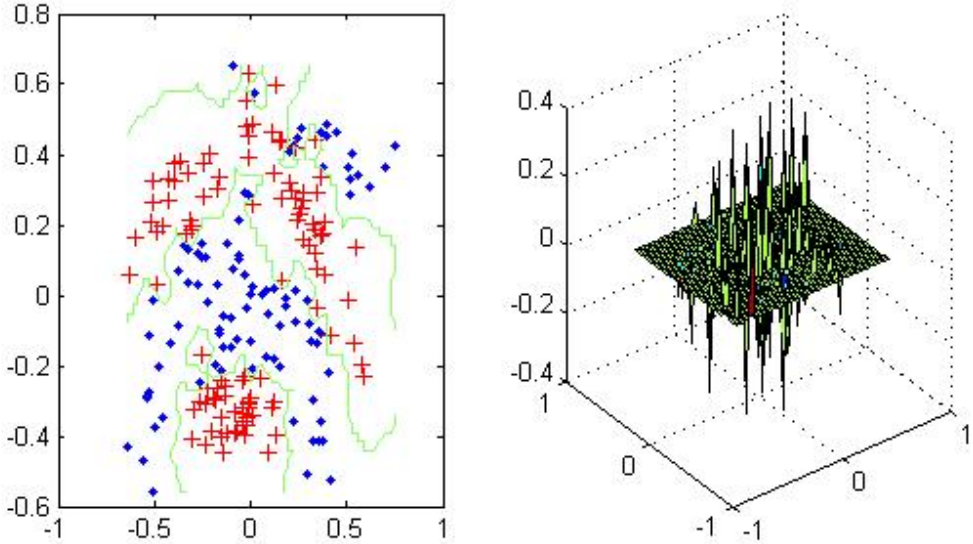

Figure 7 – Parzen-Window Classification with a Resolution of 200

and less reduction. Placing samples in bins and using some bins as samples has been suggested. However, as Hall and Wand (1994) point out, the accuracy of these algorithms is heavily dependent on bin size and the specific binning strategy. Mitra, et al. (2002) proposed a multiple resolution (or multi-scale) density based condensing technique. This technique clusters samples in variable-sized discs, using the k-nearest neighbor to obtain a density estimate. This technique was not intended to provide an optimally reduced set for a classifier, and comparisons to classification based reduction methods were specifically excluded from the paper because, as the authors stated, "...error in density estimates is not the optimality criterion for such methods."

Nearest Neighbor

Ideally, we would like to be able to find a reduction technique that provides a reduced set with training-set consistency, i.e. the reduced set classifies all training-set samples consistent with the original non-reduced set. In the past such training set consistent reduction algorithms have only been applicable to nearest neighbor algorithms. Nearest neighbor classifiers are similar to Parzen-window classifiers in many respects. The major difference being that classical Parzen-window calculations are based on an initial fixed volume whereas the nearest neighbor algorithms fix the number of samples to be included and allow the volume to vary. The simplest of the nearest neighbor algorithms is the 1-nearest neighbor (1-NN).

The 1-NN rule states that a new sample is classified based solely on the class of the nearest known training sample. It is important to note that for some applications, the nearest neighbor algorithm works quite well. If the underlying class PDFs allow for a small classification error rate, meaning there is little overlap between densities, the Bayes error rate and 1-NN error rate are similar. In less ideal situations, the 1-NN error rate can produce error rates equal to twice the Bayes error rate. Unfortunately the ratio of 1-NN error rate to Bayes error rate increases as the Bayes error rate increases, indicating that for problems with moderately high error rate, the 1-NN classifier may perform quite

poorly.

In an attempt to alleviate this problem k-nearest neighbor (k-NN) algorithms expand the volume of feature space to be used for classifying a new sample to include k training samples where $k \geq 1$. A voting system is then used to determine the classification of a new sample. Obviously, this shares many similarities to Parzen-window estimates. Indeed, if k is equal to the number of samples, and a hypercube function with resolution equal to the entire feature space is used for the Parzen-window estimate, both algorithms degenerate to the *a priori* probabilities and are equivalent.

The naive approach to nearest neighbor classifiers is to simply store all training samples and when a new sample is encountered, calculate the distance between the new sample and all training samples. The new sample is then classified according to the nearest k training sample's classes. The new sample can then be added to the set of training samples. Since there is no time involved in training the classifier and the classification complexity is $O(n)$, these algorithms have several theoretical advantages. In practice though, they suffer from the same problem as Parzen-window classifiers: model complexity.

There have been several attempts to resolve this issue for nearest neighbor classifiers. Nearest neighbor algorithms are especially well adapted to parallelization which can allow for $O(1)$ time complexity given n processors (Duda, et al, 2002). One of the first approaches to obtaining a reduced set of training-consistent samples was the condensed nearest neighbor rule (Hart, 1968). This condensed nearest neighbor rule helped motivate the discovery of the Multiple Resolution Nonparametric (MRN) described next.

CHAPTER III
MULTIPLE RESOLUTION NONPARAMETRIC CLASSIFICATION

Mathematical Foundations

While support vector machines, Parzen-window classifiers and nearest neighbor classifiers have valuable theoretical properties, their practical application has often been marred by computational complexity during training and/or classification. Some data mining and classification problems may have millions of samples available. However, most algorithms do not provide a means of handling massive amounts of data effectively in both the training and classification phases.

Despite these technological advances, the inverse problem also exists. Samples may be very difficult or costly to obtain and only a small number of samples is available. To compound this problem, feature selection may be difficult which often leads to high-dimension feature spaces. As Dasgupta (1999) illustrated, the number of samples required for accurate probability density estimation increases exponentially with the dimension of the feature space. Support vector machines (SVM) attempt to alleviate this problem by only retaining samples near the boundary and seeking the largest possible margin.

However, it can be shown that the Bayesian discriminant (given the true PDF) guarantees the lowest average error rate and is, in this sense, optimal (Duda et al, 2001). Unfortunately the computational requirements for typical PDF estimation through non-parametric techniques are prohibitively expensive for most applications.

Here we present a method that trains in $O(n^2)$ time, classifies in $O(k)$ where k is the model complexity (number of retained samples), and has model complexity similar to support vector machines. The method provides training-set consistent decision boundaries to the Parzen-window classifier while also guaranteeing a minimization of mean squared error (MSE) given the reduced model complexity. Due to its reduced model complexity and minimization of MSE, this algorithm should improve performance and generalization in high-dimension, low-sample problems as well, while providing optimal Bayesian error rates as sample sizes increase. We also note that the Multiple Resolution Nonparametric classifier is particularly suitable to parallelization in both the training and classification stages since the majority of computation is spent in summing distance calculations between samples.

The algorithm follows two basic steps: an estimation step and an approximation step. The estimation step computes an estimate of the Bayesian discriminant while the approximation step is an iterative approach to optimizing model complexity. In our description of the algorithm we will assume a two class problem although this could easily be generalized to any multi-class problem.

First, we examine the theoretical underpinnings of the algorithm. To begin, we must find an estimate of the Bayesian discriminant. Obviously, we must guarantee that our estimate will converge to the true Bayesian discriminant as n approaches infinity since we wish to minimize classification error rate. A Parzen-window approach, using the Gaussian distribution as the window function such a guarantee of convergence as well as some well-desired computational attributes.

The multi-variate Gaussian distribution is defined by:

$$p(x)=\frac{1}{(2\pi)^{1/d}\left|\Sigma^{1/2}\right|}e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^t\Sigma^{-1}(\vec{x}-\vec{\mu})} \tag{17}$$

where d is the feature space dimension. We can simplify the computation of each sample by making the following assumption:

$$\Sigma = \sigma^2 I \tag{18}$$

where I is the identity matrix. It is important to note that the assumption of a scalar matrix does not invalidate the convergence of the approach to the true PDF since the simplified Gaussian function is still a legitimate density function (Duda et. al, 2001). For notational simplicity, we present derivations here based on a univariate density function. The extension to a multivariate density function is straightforward.

The Parzen-window density estimation proceeds by centering a Gaussian window function at the new sample and summing the probability across all samples, giving us:

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi h_n}} e^{-\frac{1}{2} \frac{(x - x_i)^2}{h_n}} \tag{19}$$

Where h_n is meant to indicate that the variance, h, will vary as a function of sample size.

The discriminant is given by:

$$g(x) = p(c_1|x) - p(c_2|x) \tag{20}$$

Using Bayes theorem this becomes:

$$g(x) = p(x|c_1)\, p(c_1) - p(x|c_2)\, p(c_2) \tag{21}$$

By combining (19) and (21), performing some simple algebra and assuming equal priors

we arrive at the following:

$$g(x)=\sum_{i=1}^{n} \alpha_i f(x, x_i) \tag{22}$$

where

$$f(x, x_i)=e^{-\frac{1}{2}\frac{(x-x_i)^2}{h_n}} \tag{23}$$

and

$$\alpha_i=\frac{y_i}{n_i\sqrt{2\pi h_n}} \tag{24}$$

where y_i is the class label for the given sample (i.e. -1 or +1) and n_i is the number of samples in the class.

Multiple Resolution Editing

The classification of a new sample is determined by the resultant sign of Eqn. (22). Recognizing this, we can reduce the model complexity by eliminating samples that would cause a change in magnitude but not sign.

There is a plethora of literature discussing such "training-set consistent" optimization approaches – reduction techniques that ensure the reduced training set classifies the original samples identically to the non-reduced training set. Interested readers are referred to the excellent article by Toussiant (2002).

The edit technique is a popular approach to eliminating samples that do not contribute to correct classification. Devijver and Kittler (1982) apply this algorithm to nearest neighbor or k-NN algorithms. However, by modifying the editing technique, we can apply it to out current approach and achieve a reduced sample set that is training-set consistent.

We begin the training of our classifier by classifying all of the samples in our training set according to our Bayesian discriminant estimate. This can be accomplished in $O(n^2)$ time. The samples are then labeled as correctly classified or incorrectly classified. By separating the correct and incorrect samples from Eqn. (21) we arrive at the following:

$$g(x) = p(c_1)\, p_{correct}(x|c_1) - p(c_2)\, p_{correct}(x|c_2) - p(c_1)\, p_{incorrect}(x|c_1) + p(c_2)\, p_{incorrect}(x|c_2) \tag{25}$$

where each estimate $p_{correct}$ and $p_{incorrect}$ is the summation estimates from Eqn. (22) performed on the correctly and incorrectly labeled training samples respectively.

We can rewrite this as:

$$g(x) = p(c_1)\, p_{correct}(x|c_1) - p(c_2)\, p_{correct}(x|c_2) + E(x) \tag{26}$$

Here $E(x)$ represents the contribution of all misclassified samples and is equivalent to:

$$E(x) = \sum_{y \in Y} \alpha_i f(x, y) \tag{27}$$

where Y is the set of incorrectly classified training samples.

Consider the creation of a reduced set consisting only of the correctly classified samples

from the original set. From Eqn. (26) we can see that any sample that was correctly classified with the entire sample set will only be misclassified using this reduced set if:

$$sign[p_{reduced}(x)+E(x)] \neq sign[p_{reduced}(x)] \tag{28}$$

where

$$p_{reduced}(x)=p(c_1)p_{correct}(x|c_1)-p(c_2)p_{correct}(x|c_2) \tag{29}$$

and *E(x)* is as defined previously.

At this point the importance of the Gaussian window function becomes apparent since we can allow each sample's variance to decrease towards zero. As the variance approaches zero the Gaussian begins to approximate a delta function where its influence is infinite at its center and zero at all other points. Hence, we arrive at the following:

$$\lim_{h_i \to 0} p_{reduced}(x_i) > E(x_i) \tag{30}$$

$$\lim_{h_i \to 0} sign[p_{reduced}(x_i)] = y_i \tag{31}$$

where $y_i = 1 \; \forall \; x_i \in c_1$ and $y_i = -1 \; \forall \; x_i \in c_2$.

This proves we can create a reduced set that eliminates misclassified samples and adjust the variance of individual samples in the reduced set to ensure all samples are classified with training-set consistency.

The resolution of a sample is defined as the inverse of its variance. As the variance decreases, the resolution increases and the effect of neighboring samples decreases. Likewise as the variance increases, the resolution decreases and the effect of neighboring

samples increases. This represents a sort of smoothing effect at low resolutions, intuitively similar to the smoothing of an image when going from a high resolution to a low resolution. The term resolution is thus used to imply this visual effect of smoothing an image at low resolution, versus increasing granularity at higher resolution.

Incorporating multiple resolutions into (23) and (24) and performing some simple algebra leads to the following:

$$f(x, x_i) = e^{-\frac{H_i^2}{2}(x - x_i)^2} \tag{32}$$

and

$$\alpha_i = y_i \frac{H_i}{n_i \sqrt{2\pi}} \tag{33}$$

where H_i , the resolution for sample x_i , is equivalent to $\sqrt{1/h_i}$ where h_i is the variance for sample x_i .

In practice only a few samples require different resolutions, however without the addition of these few separate resolutions, the algorithm would fall apart and we would be relegated to approximating the nearest neighbor algorithm with its increased error rates.

To further improve our reduced estimate of the Bayesian discriminant, we can observe from Eqn. (22) that a simple minimization of mean square error (MSE) can be accomplished by adjusting each α_i through a pseudo-inverse or gradient descent approach. It has been shown that by setting a consistent margin (i.e. all samples are equidistant from the decision boundary), minimizing MSE approximates the Bayesian discriminant (Duda et al, 2001). This MSE approximation to the Bayesian discriminant's accuracy is bounded by the ability of the underlying function to approximate the discriminant. In our case, as is shown above, we have a guarantee that a training-set consistent approximation is

achievable.

Minimizing mean square error allows for an extra degree of freedom in the multiple resolution nonparametric classifier algorithm. While we can achieve a training set consistent subset without minimizing MSE, minimization allows us to more closely approximate the discriminant and reduces the number of samples retained. In the future, we will refer to the number of retained samples after reduction as model complexity, since it determines the computational complexity and is directly related to how the algorithm represents the final decision boundary.

To this point we have proved we can develop a theoretically sound estimate of the Bayesian discriminant and maintain a training-set consistent approximation to that estimate by removing misclassified samples from the original training set while adjusting the resolution of the remaining samples and minimizing MSE.

Multiple Resolution Condensing

We can extend this approach and further minimize the final reduced set by removing *all* samples not necessary to correct classification of the original training set, not just those that are originally misclassified. Through the use of a technique somewhat similar to the condensing technique described by Devijver and Kittler (1982) we can thereby drastically reduce the model complexity. The training algorithm for the Multiple Resolution Nonparametric classifier then becomes:

1. *Classify all training samples according to the Bayesian discriminant using Parzen-window density estimates.*

2. *Remove any samples that are incorrectly classified and call the set of all remaining samples TRAIN*
3. *Sort all samples in TRAIN according to discriminant function magnitude*
4. *Set the initial resolution H_i for each sample*
5. *Add the first sample in TRAIN to an otherwise empty set. Call this set TEST*
6. *Set K=1*
7. *Set all $\alpha_1 = 1$*
8. *Set N equal to the number of samples in TRAIN*
9. *DO*
10. *Set Current_Sample = TRAIN(K % N)*
11. *IF Current_Sample is misclassified THEN*
12. *IF Current_Sample already exists in TEST THEN*
13. *Increase the resolution H_i for Current_Sample*
14. *ELSE*
15. *Add Current_Sample to TEST*
16. *END IF*
17. *Calculate new α_i's that minimize MSE*
18. *END IF*
19. *Set K=K+1*
20. *UNTIL all samples in TRAIN are correctly classified.*
21. *End of algorithm*

In Step 3, we note that when classifying the samples in Step 1, the distance to the decision boundary is related to the magnitude (regardless of sign) of the output of the discriminant function. Hence, sorting the samples by the absolute value of the discriminant function provides an intuitive measure of the sample's proximity to the decision boundary – which allows us to avoid explicitly calculating the distance between each sample and the decision boundary.

We note that classification is performed in Step 11 using the TEST set according to the following:

$$class = sign[\sum_{x_i \in TRAIN} \alpha_i f(x, x_i)] \tag{34}$$

where α_i and $f(x,x_i)$ are as defined in (32) and (33).

It is important to note that in Step 17 we are minimizing MSE through a pseudo-inverse approach. For strict adherence to the $O(n^2)$ complexity claim, a Widrow-Hoff or Ho-Kashyap gradient descent approach could be used with some limiting number of iterations. However, in our experience, since the algorithm begins with an empty matrix and this matrix never exceeds the final size of the reduced set and the reduced training sets are so small, the computational complexity and numerical issues of a pseudo-inverse approach are negligible. This is in contrast to most algorithms where the inverse is done on the entire training set and hence the computational impact of matrix inverses can be exorbitant. If the underlying PDF and sample size demanded it, any of the aforementioned gradient descent approaches could be used.

Additionally, average case complexity for the training algorithm, excluding the original Parzen-window classification appears to grow linearly. This is discussed more thoroughly in the results section, however, it is somewhat intuitive since the final number of samples in TEST is most closely related to the underlying densities, not the number of samples in TRAIN. For applications where the $O(n^2)$ Parzen-classification step is prohibitive, a smaller, representative subset could be used in lieu of the entire set while the entire data set could be used throughout the rest of the algorithm. This would allow the algorithm to be tailored to the computational demands of the situation while still making use of all available data in later steps. However, such a discussion is beyond the scope of this paper and should be considered an area for future research.

It can be shown that when minimizing mean square error using a consistent margin, the result is an approximation to the Bayesian discriminant (Duda, et al, 2001). This approximation is only limited by the underlying function's ability to approximate the form of the discriminant. In our case, the underlying function can perfectly approximate our discriminant, given the entire training set, since the discriminant was derived from the training set originally.

Figures 8-16 show the MRN classifier at different stages of the algorithm. The dots and crosses represent samples from different classes. Samples displayed in black are samples that were removed in Step 2. Samples retained in TEST are circled in green. The solid cyan line is the current MRN decision boundary, the dashed magenta line is the Parzen-window decision boundary. The discriminant function is also shown to the right of the decision boundary plot.

Figure 8 shows the MRN output with three samples in TEST and each sequential figure illustrates the inclusion of an additional sample in the TEST set. The training set consisted of 200 randomly selected samples from the Ripley data set.

It can be seen in the discriminant function plot of figure 9 that the sign of the α_i value for the sample at -.25, .25 is not representative of the blue class. Indeed, it is possible for MSE to be minimized when the sign of α_i for some samples do not correspond to those samples' class. This may also lead to some intermediate decision boundaries that seem counterintuitive such as in Figure 10. Minimization of MSE gives us a 'best' approximation to the Bayesian discriminant given the current test set. However, as Duda, et. al. state: "...the discriminant function that 'best' approximates the Bayes discriminant does not necessarily minimize the probability of error." (Duda, et al, 2001). Typically this is a serious disadvantage to using MSE as a criterion function for classification algorithms. However, this problem is eliminated in MRN classifiers through the use of multiple resolutions.

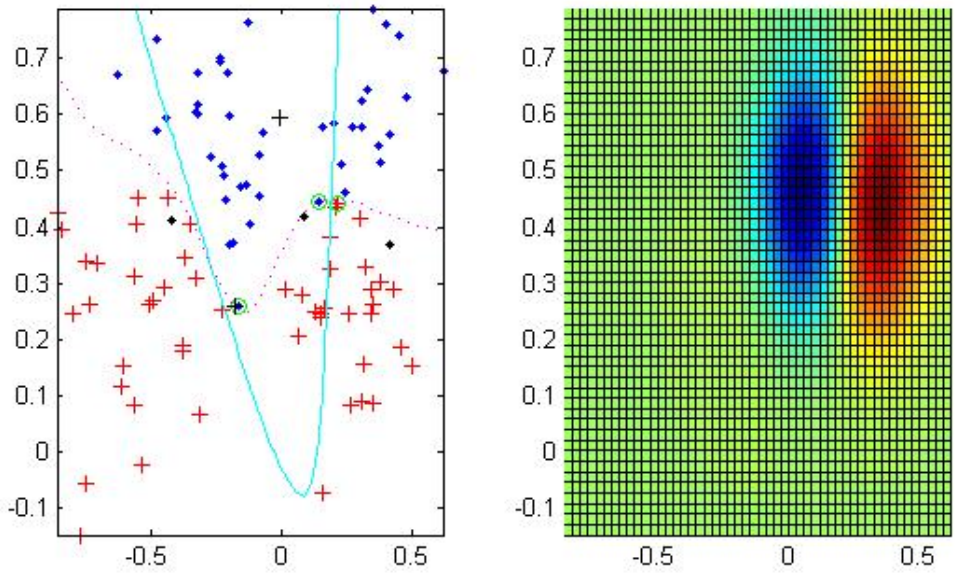

Figure 8 – MRN Output with 3 Samples

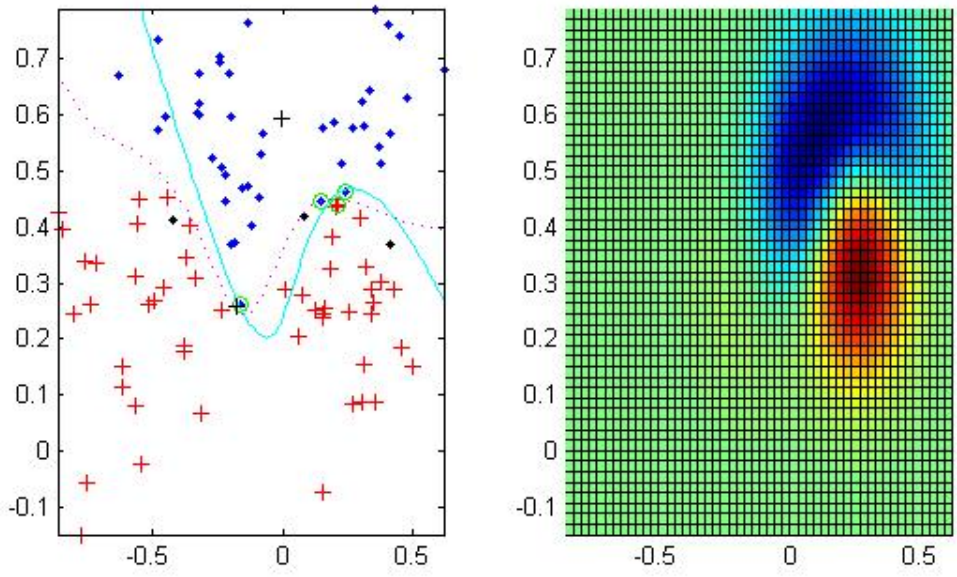

Figure 9 – MRN Output with 4 Samples

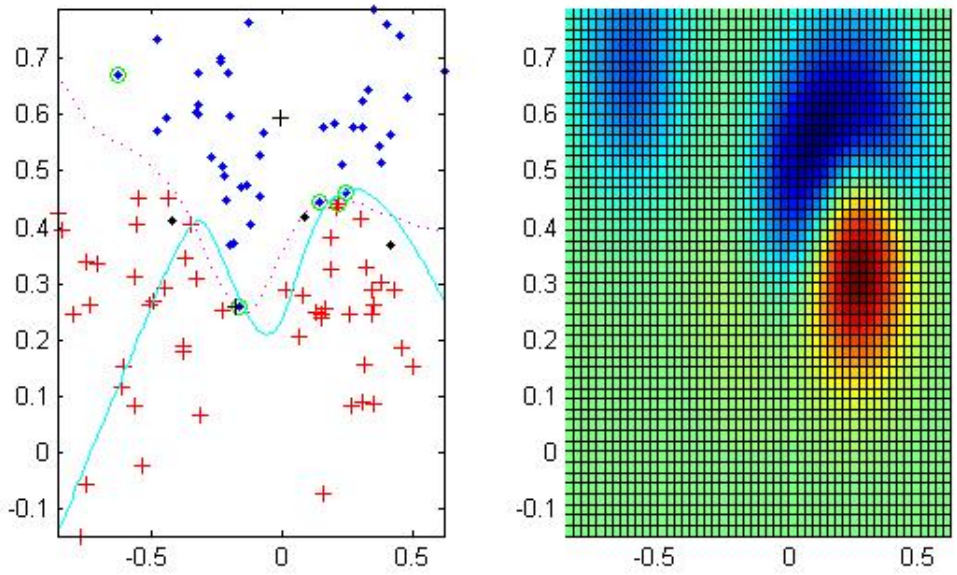

Figure 10 – MRN Output with 5 Samples

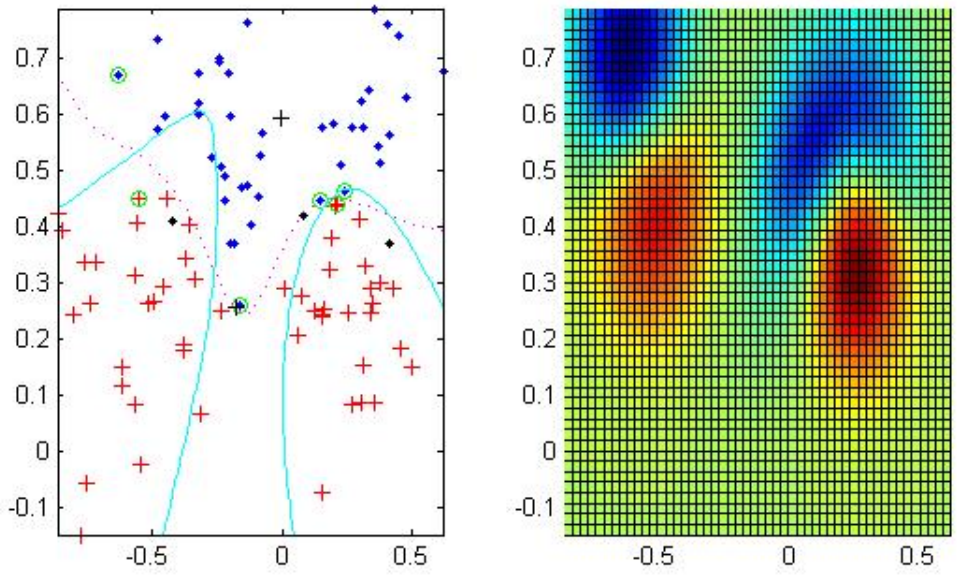

Figure 11 – MRN Output with 6 Samples

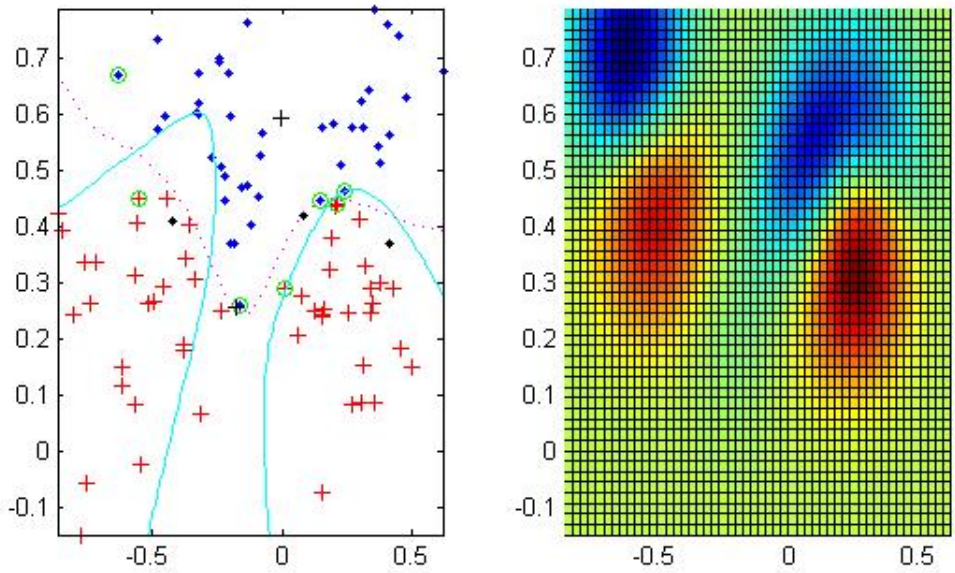

Figure 12 – MRN Output with 7 Samples

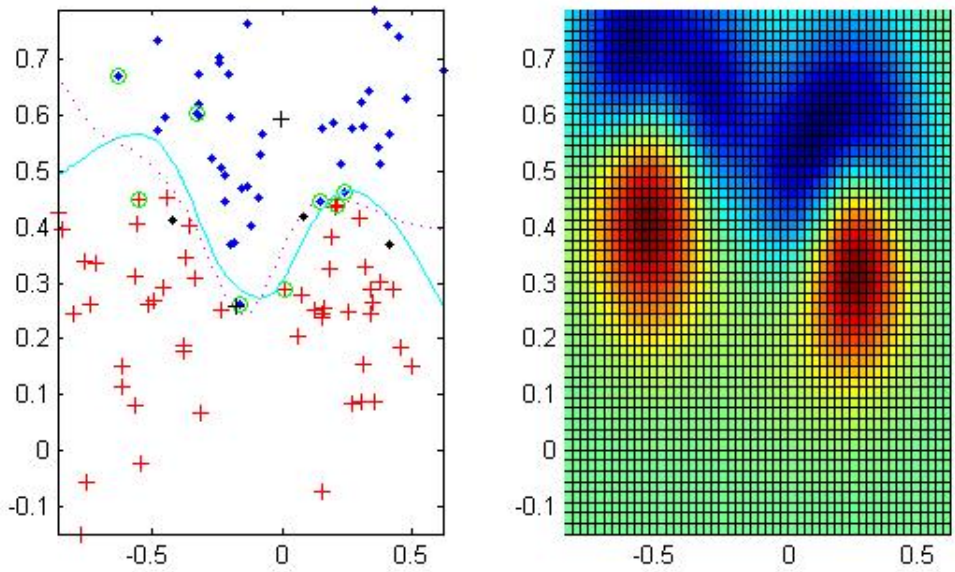

Figure 13 – MRN Output with 8 Samples

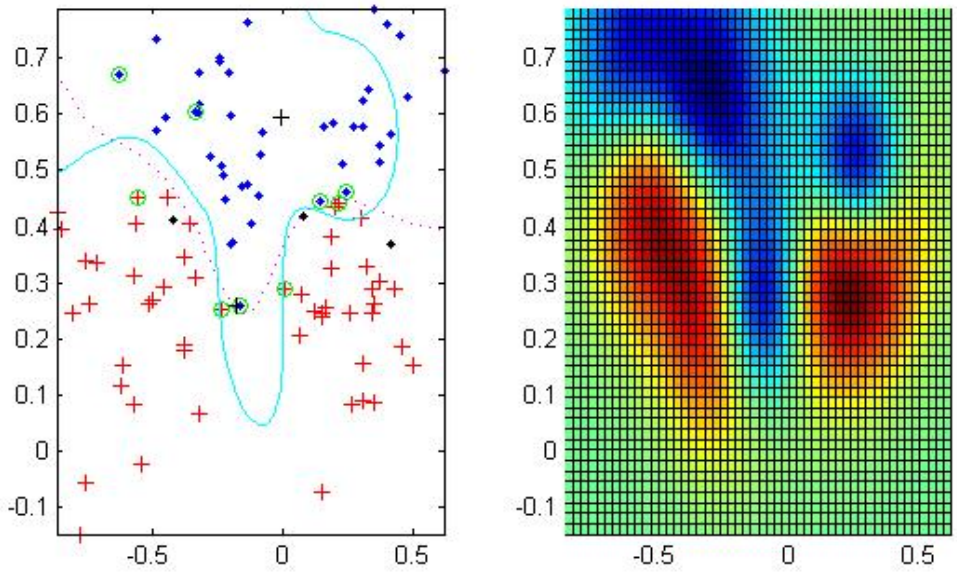

Figure 14 – MRN Output with 9 Samples

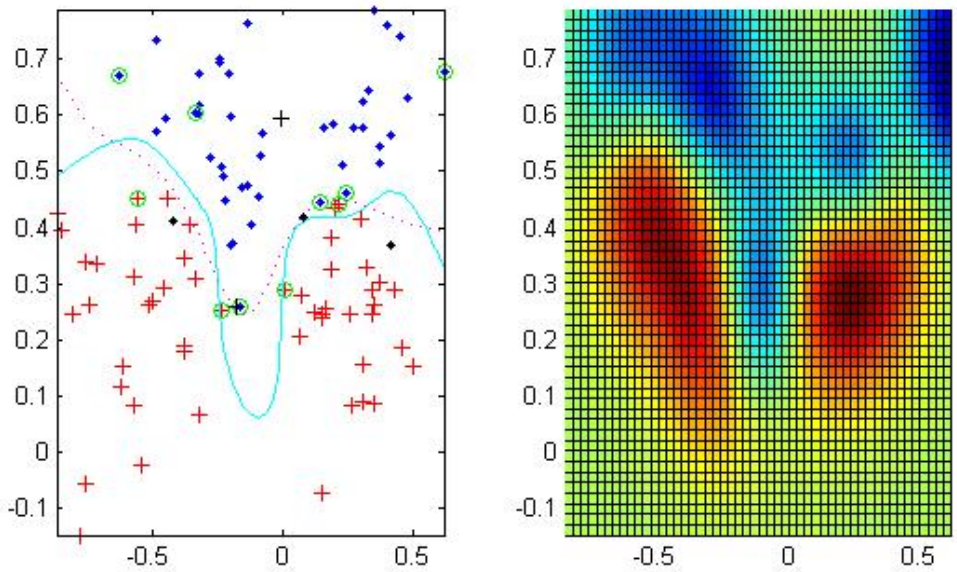

Figure 15 – MRN Output with 10 Samples

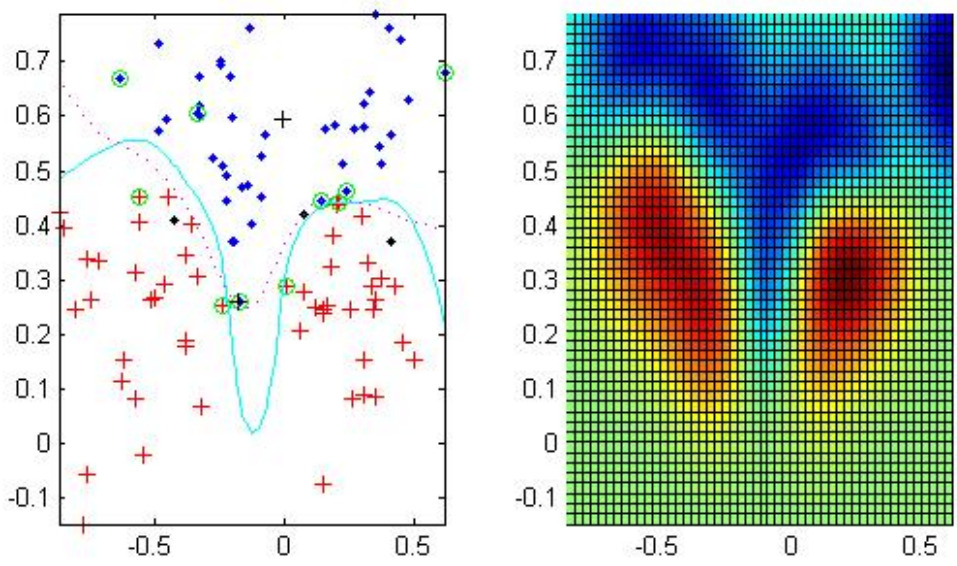

Figure 16 – MRN Final Output with 10 Samples

In Figure 11 we see that the region at -.55, .7 has increased in magnitude whereas the region at .25, .6 has decreased in magnitude. This illustrates the fact that the probability regions at each step can either grow or diminish. This can also lead to some apparent "outliers" being selected. This can occur when, at step i, a sample is classified incorrectly and hence added to the test set. At step $i+c$ it may be classified correctly due to the aforementioned diminution or growth of some other sample's influence. Since the sample was added at step i and samples cannot be removed once added, it may appear to be an unnecessary sample, or even an "outlier", at stage $i+c$. It is important to point out there is no guarantee of an optimally small subset of samples inherent to this algorithm. However, in general, optimal subset problems are NP-hard or NP-complete and the current approach gives excellent experimental results.

Figures 13 and 14 show the beneficial effects of multiple resolutions. In Figure 13, the only misclassified sample is located at -.25, .25 and has an α_i whose sign is not representative of the sample's true class. Without multiple resolutions we could not proceed at this point since there are no other misclassified samples to be added to our test set. In figure 14 we see that, per the algorithm, the resolution at this sample was increased. As the resolution increases, it's effect on its neighbors decreases and MSE is minimized when it's α_i value is representative of its true class. However, this change in resolution and sign affects the decision boundary and causes other samples to be misclassified. This can be seen in Figure 14.

With the addition of more samples in figures 15 and 16, we see that the algorithm terminates with a viable estimate to the discriminant function and a training-set consistent decision boundary.

Figure 17 shows a schematic diagram of the final result. The size of the green circles represent the resolution at each retained sample and the numeric values in black are the α_i values for the corresponding sample.

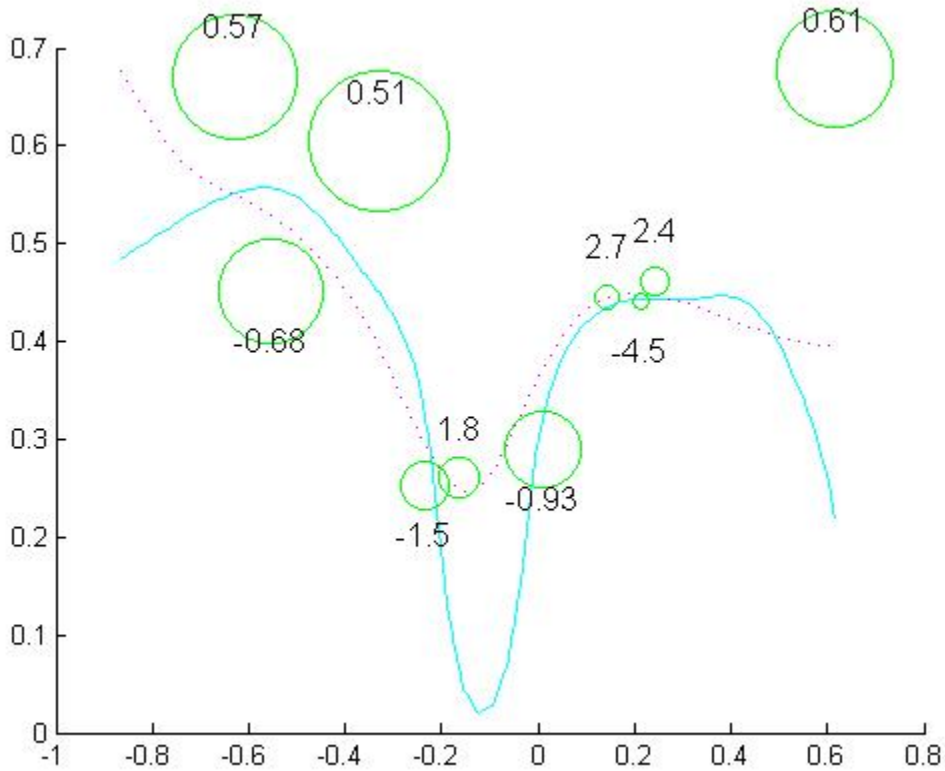

Figure 17 – Schematic Showing Resolutions and Alpha Values

The results of this approach share striking similarities to support vector machines (SVM). The postulation that the samples nearest to the boundary (support vectors) are the most "important" leads to a large reduction in model complexity. The final result in the MRN approach maintains training-set consistency while the SVM approach maximizes the margin. The SVM approach is based on the paradigm of a linear classifier in some high dimensional space. The inner-product of different bases in this space is the kernel, which must remain consistent. An oft overlooked disadvantage to SVM classifiers is training complexity, particularly with respect to memory requirements. Recent research has indicated the number of support vectors for non-separable problems may increase linearly with respect to the number of training samples (Steinwart, 2004). Memory requirements then scale exponential to both sample size and the number of support vectors (Bakir, et. al, 2004). The MRN approach has the added benefits of better training complexity, Bayesian decision boundaries, excellent applicability to parallelization and the added flexibility of multiple resolutions.

While the final result bears similarities to SVM classifiers, the foundations of the approach are based on Parzen-window density estimates. Thus, we are assured our approach is based in theoretically sound, probabilistic approaches that guarantee optimal error rate as sample sizes increase towards infinity.

CHAPTER IV
RESULTS

Two Dimensional Examples

The accuracy of the MRN algorithm is, predictably, similar to the Parzen-window estimation accuracy. Due to some additional generality introduced by the reduction algorithm one would expect to see some minor variation in error rates. Our results affirm that MRN error rates show minor deviations from Parzen-window estimates; however, the differences tend to be statistically insignificant.

All error rates are the result of tenfold cross validation. Experimental error rates were obtained by randomizing the original data set and splitting it into ten equal subsets. Training was performed on 90% of the data and classification testing was then performed on the remaining 10%. Ten runs were performed where a different subset was kept out to be used as the test set in each run.

To allow for visualization of the resulting decision boundaries and set reduction, five runs were completed using increasingly larger sets and the results were plotted. Parzen-window classifier decision boundaries were then calculated and superimposed on the MRN classifier output.

Ripley's Data Set

The MRN algorithm was run five times on a random subset of Ripley's data. After each run, 200 new random samples were added to the training set. Figures 18-22 illustrate the results of these runs. Each figure uses the same notation as Figures 8-16.

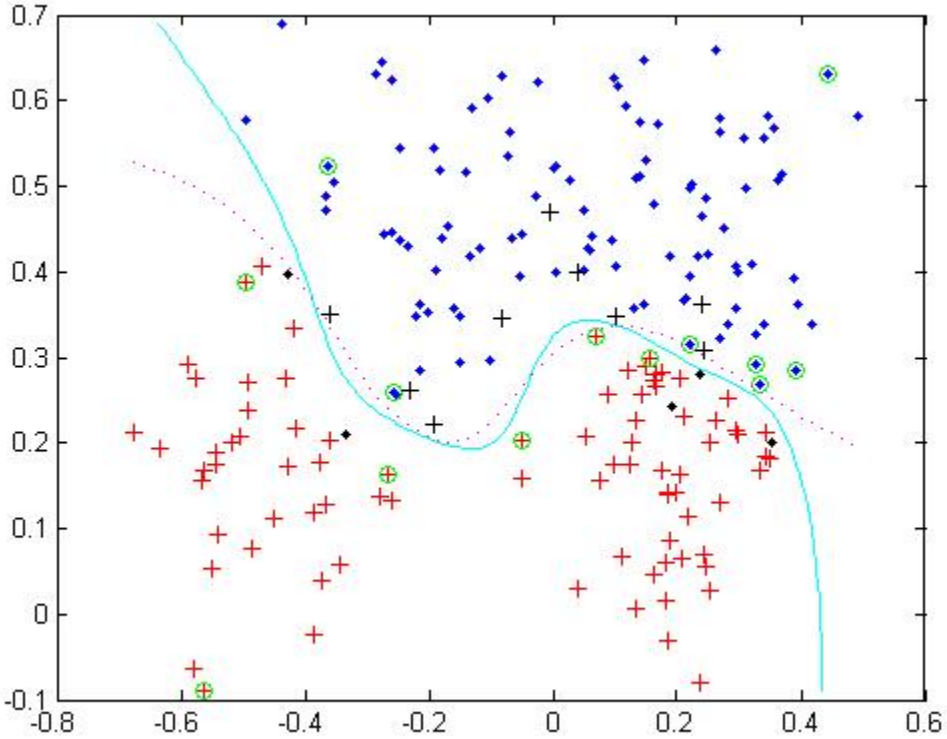

Figure 18 – MRN Output on 200 Samples of Ripley's Data Set

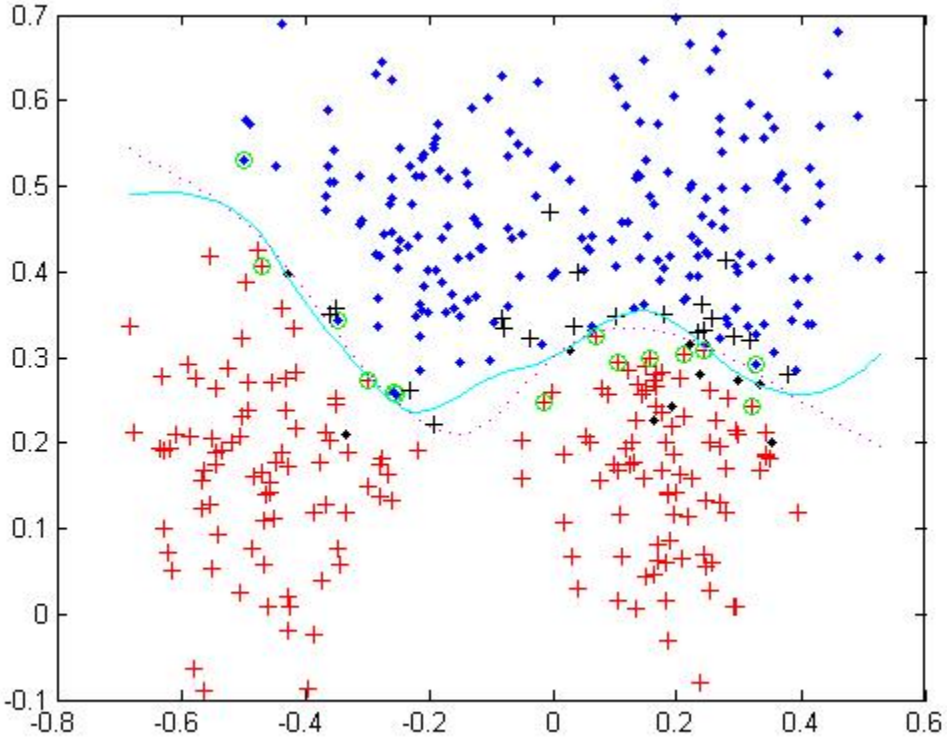

Figure 19 – MRN Output on 400 Samples of Ripley's Data Set

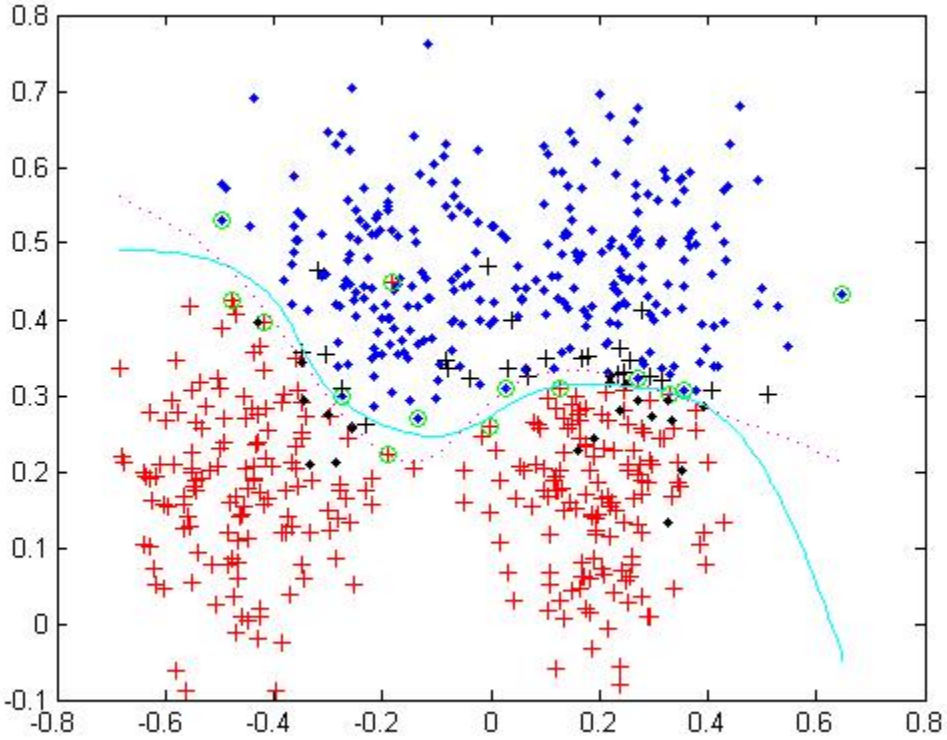

Figure 20 – MRN Output on 600 Samples of Ripley's Data Set

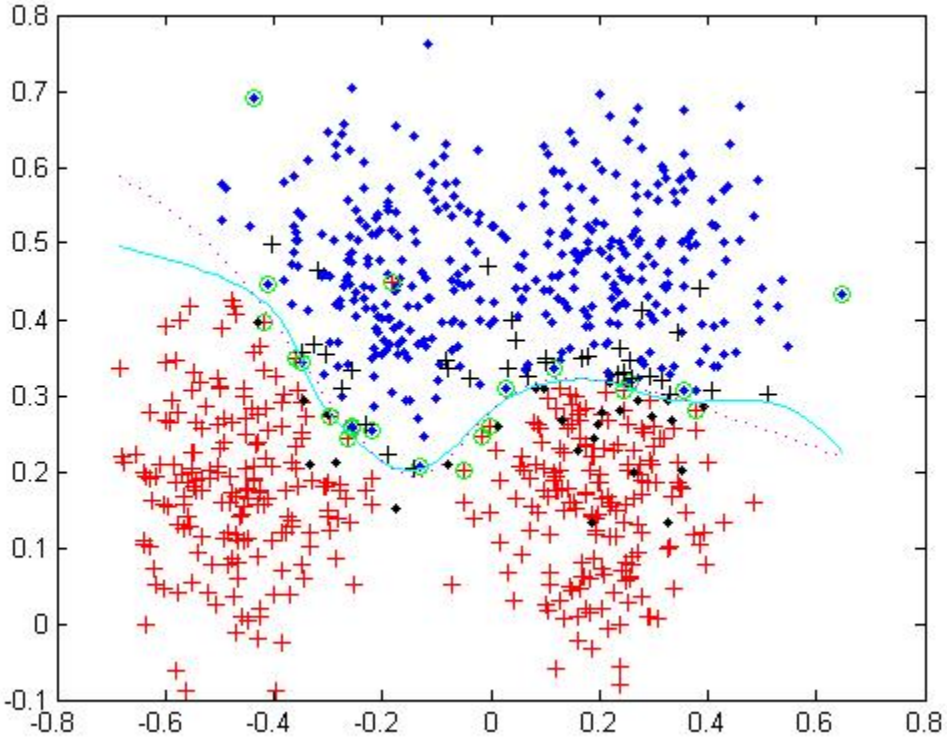

Figure 21 – MRN Output on 800 Samples of Ripley's Data Set

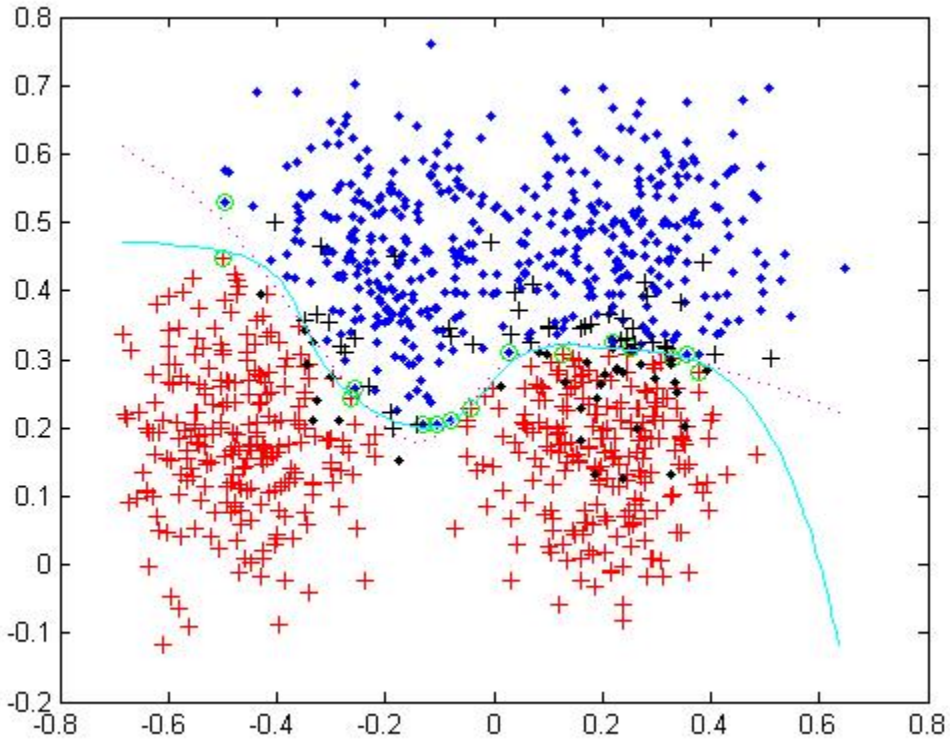

Figure 22 – MRN Output on 1000 Samples of Ripley's Data Set

It is important to note that in the MRN algorithm a training-set consistent reduced set is one which correctly classifies all of the samples that were correctly classified under the initial training set. Using the reduced set, some samples may be correctly classified that were incorrectly classified under the original training set since these samples are essentially discarded in the editing step. This is most evident in Figure 18 where the black cross (indicating a misclassification by the Parzen-window classifier) near the first inflection point of the decision boundary lies on the correct side of the MRN decision boundary.

Figures 18-22 demonstrate how closely the final reduced set decision boundary approximates the Bayesian decision boundary of the Parzen-window estimates as sample sizes increase. The reduced set decision boundary deviates most from the Parzen-window boundary in areas of extremely low probability (the tails of the distributions). Fortunately these are the areas where samples are most unlikely to fall, and hence tend to have negligible affects on error rates. This is a common issue with virtually all probabilistic approaches.

The average error rate of the MRN classifier for the tenfold cross-validation tests on the Ripley data set, using a random training subset of 900 samples and a testing subset of 100 samples was identical to the error rates for the Parzen-window classifier. This was achieved while reducing the training set by 97.6%.

As was previously alluded to, an important advantage to the MRN classifier is that the reduction in training set size is not as much a function of the original training set size as it is the form of the underlying distribution. To illustrate this we ran tenfold cross-validation runs on the MRN and Parzen-window classifiers with a smaller subset of 405 samples and compared the average reduced set size. The smaller sample size created an average of 16.4 training samples versus the larger training set's 21.3 average retained samples. The larger training set consists of more than twice as many samples whereas the

reduced (MRN) set consists of only 1.3 times as many samples as the reduced MRN set from the smaller sample problem set. This tends to indicate the reduction ratio will increase as the sample size increases since the number of samples required for a given degree of accuracy is largely determined by the shape of the underlying class densities.

Training time, excluding the Parzen-window classification performed in step 1, averaged .1953 seconds for the 405-sample set versus .6811 seconds for the 900-sample set. Hence, a training set 2.22 times the original size requires only 3.49 times more computation time. It should be remembered these times were achieved using the simplistic, and computationally expensive pseudo-inverse approach. This strengthens the assertion that, minus the Parzen-window classification step, average complexity is substantially lower than $O(n^2)$. As was alluded to previously, this fact could be used to tailor the computational complexity of the algorithm to the available processing power.

Banana Data Set

The MRN algorithm was run five times on a random subset of the banana data set. After each run 200 new, random samples were added to the training set. Figures 23-27 illustrate the results of these runs. Each figure uses the same notation as the output plots in Figures 8-16.

The MRN classifier results in a reduced set consisting of an average of 37.5 samples, giving a 95.8% reduction in model complexity and classification computation time. Again we see that error rates are nearly identical (12.31% for MRN versus 11.91% for Parzen). The underlying distributions are obviously more complex and hence we see the increase in retained samples in the reduced sets for the banana data. Average training time, excluding Step 1, was 1.15 seconds for 900 samples. The increase in training time can be directly attributed to the increased complexity of the underlying PDFs and hence the increase in the number of retained samples.

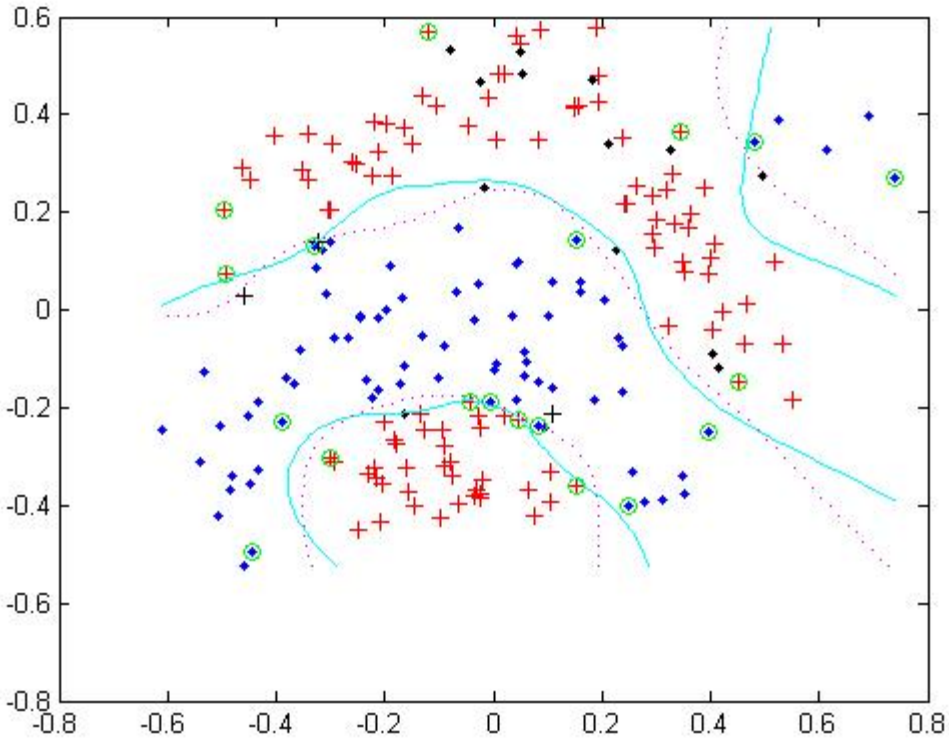

Figure 23 – MRN Output on 200 Samples of the Banana Data Set

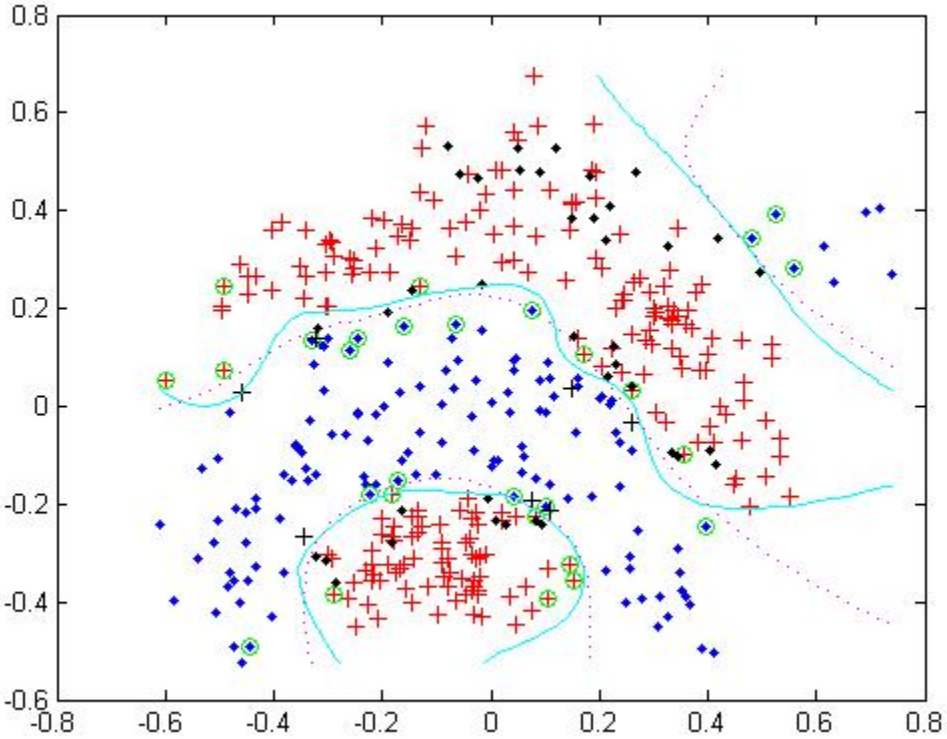

Figure 24 – MRN Output on 400 Samples of the Banana Data Set

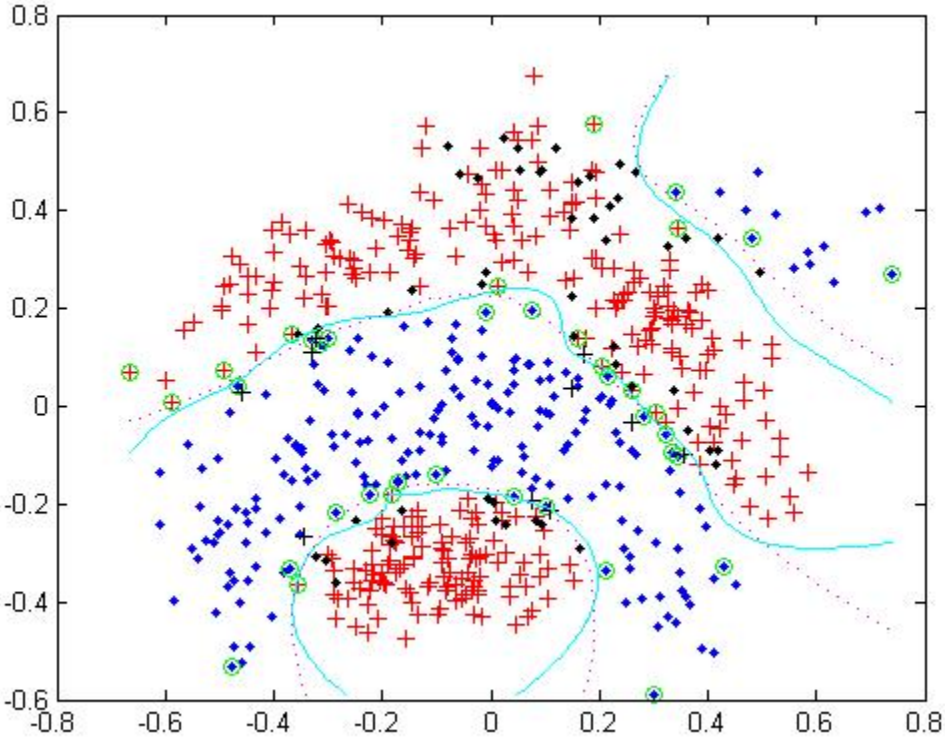

Figure 25 – MRN Output on 600 Samples of the Banana Data Set

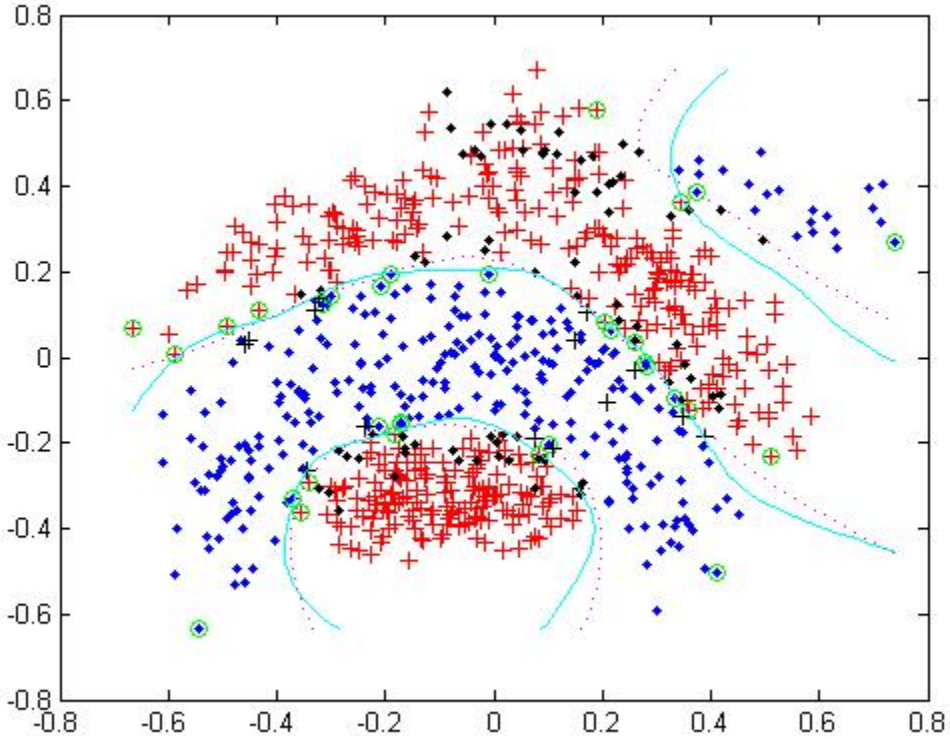

Figure 26 – MRN Output on 800 Samples of the Banana Data Set

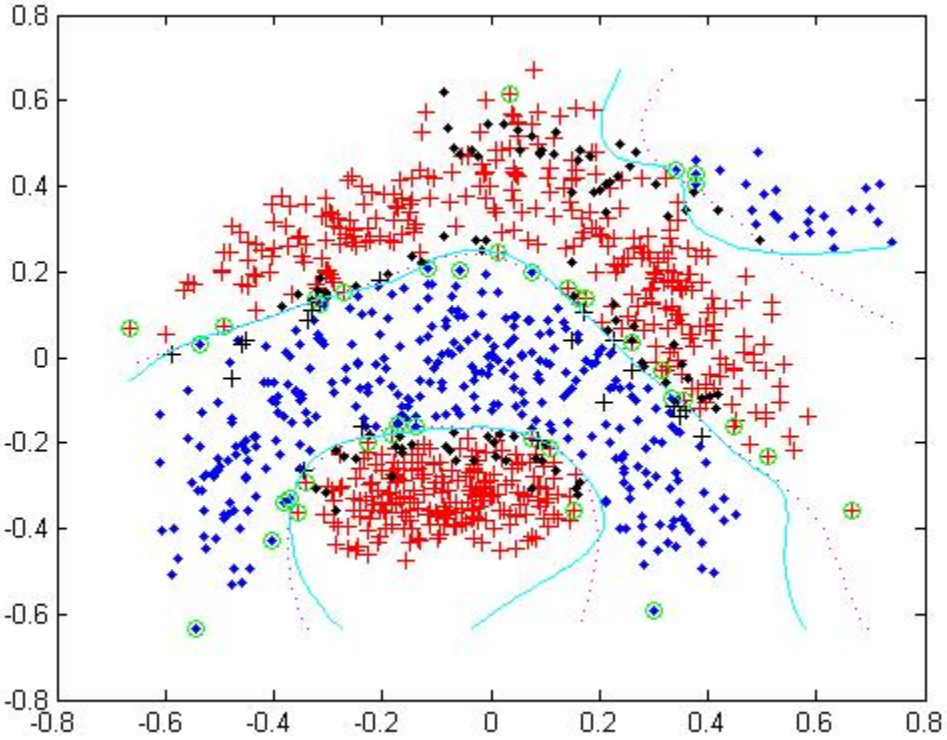

Figure 27 – MRN Output on 1000 Samples of the Banana Data Set

High Dimensional Examples

Performance on high-dimensional classification problems was evaluated by testing on the Pima Indians data set and the Ionosphere data set, both available from the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html).

The Pima Indians data set consists of 692 samples where each sample has 8 features. When run on the Pima Indians data set the MRN classifier creates a reduced set consisting of an average of 47.2 samples, giving a 93% reduction in model complexity and classification computation time. The ten-fold cross-validation error rate was 24.97% versus 24.58% for the Parzen classifier.

The Ionosphere data set consists of 316 samples where each sample has 34 featuress. The MRN classifier creates a reduced set consisting of an average of 72.7 samples, giving a 77% reduction in model complexity and classification computation time. The ten-fold cross-validation error rate was 10.87% versus 11.71% for the Parzen classifier. The extremely small sample size (especially with respect to feature space dimension) is sufficient to explain the lower reduction rate. As was shown earlier, the ratio of reduced set samples to original training-set samples is more dependent upon the underlying PDFs and feature space than the size of the original training-set. While the reduction rate in this example is substantially lower than any of the other examples it is our belief that, given a larger original training set, the reduction ratio would improve.

In general, probabilistic methods have been viewed as suboptimal for high-dimension, low-sample problems. However, it is important to keep in mind that, when performing ten-fold cross-validation on such small sample sizes, the failure to correctly classify one sample can make an extremely large difference.

For example, in the paper by Gregor and Liu (2006), a Bayesian framework for SVM parameter estimation and the best published error rates for SVM classification on both the Pima Indians and Ionosphere data sets are presented. These error rates are 22% and 5.5% respectively.

While the Pima Indians error rates are similar to probabilistic error rates, the Ionosphere error rates appear substantially lower. However, using ten-fold cross-validation on the Ionosphere data set generates 10 training sets of 316 samples and 10 testing sets of 35 samples. The difference in error rate between the MRN classifier (10.87%) and the SVM classifier (5.5%) equates to the misclassification of 1.88 samples. In addition 3 of the 10 runs on the MRN classifier achieve an error rate of 5.6%. With such small sample sizes it is difficult to meaningfully compare error rates between approaches.

Table 1 lists the results of each of the ten runs of the Pima Indians data set and compares them to the Parzen classifier. Table 2 lists the results of each of the ten runs of the Ionosphere data set and compares them to the Parzen classifier.

Summary of Results

The MRN classifier gives results equivalent to classical Parzen-window classifiers with substantially reduced (up to 97.6%) model complexity. Likewise, these results are achieved while maintaining a worst-case training complexity of $O(n^2)$ while classification is $O(k)$ where k is the cardinality of the reduced set and is no more than and typically much less than n. The results of the various ten-fold cross-validation tests are summarized in Table 3.

When compared to non-probabilistic techniques such as support vector machines, MRN

Table 1 – Comparison of Results Between the MRN and Parzen Classifiers on the Pima Indians Data Set

Run	***MRN Set Size***	***MRN Error Rate***	***Parzen Error Rate***
1	49	27.63%	27.63%
2	49	23.38%	22.08%
3	45	31.17%	28.57%
4	49	20.78%	23.38%
5	49	25.97%	25.97%
6	44	32.47%	29.87%
7	49	20.78%	22.08%
8	41	22.08%	19.48%
9	45	22.08%	23.38%
10	52	23.38%	23.38%

Table 2 – Comparison of Results Between the MRN and Parzen Classifiers on the Ionosphere Data Set

Run	***MRN Set Size***	***MRN Error Rate***	***Parzen Error Rate***
1	74	11.43%	14.29%
2	63	5.56%	11.11%
3	70	5.56%	8.33%
4	60	11.11%	8.33%
5	90	13.89%	11.11%
6	77	11.11%	11.11%
7	73	11.11%	8.33%
8	72	22.22%	19.44%
9	77	5.6%	13.89%
10	71	11.11%	11.11%

Table 3 – Summary of Results

Data Set	***MRN Best Error***	***Parzen Best Error***	***MRN Worst Error***	***Parzen Worst Error***	***MRN Average Error***	***Parzen Average Error***	***MRN Ave. Model Size (Samples)***	***Parzen Ave. Model Size (Samples)***
Ripley	.050	.040	.110	.109	.081	.081	21.3	900
Banana	.060	.030	.220	.210	.123	.119	37.5	900
Ionosphere	.056	.083	.222	.194	.109	.117	72.7	316
Pima Indians	.208	.195	.325	.299	.250	.246	47.2	692

classifiers provide comparable results in all but the most sparse applications. However, results from such small-sample applications should be tempered by the very fact that such small samples may not be a reliable measure of accuracy.

CHAPTER V
CONCLUSION

Multiple Resolution Nonparametric classifiers offer excellent training complexity, high applicability to parallelization, Bayesian error rates and significant model complexity reduction.

In our implementation, the optimal initial resolution at the given sample size was found through exhaustive search. This may not be the most elegant approach to identifying an optimal window resolution for the initial Bayesian discriminant; however, in theory, as the sample size increases, the specific resolution used for the PDF estimation becomes less important. In our implementation we decided to leave the most pertinent aspects of the algorithm unencumbered by extraneous, peripheral approaches to initial resolution optimization, gradient descent optimization or code optimization.

In many modern applications massive amounts of data are available for training but, in the past, there has been no practical means of training classifiers using a theoretically sound estimation of the Bayesian discriminant on such large data sets. Due to its ease of parallelization, significant model-complexity reduction and improved computational complexity in training, MRN classifiers allow for the more widespread use of probabilistic methods on such data sets. This development may be especially applicable to data mining applications where large amounts of data are often available but training and/or classification computational complexity proscribes the use of current algorithms.

Future research may focus on minimizing the complexity of the Parzen-window classification step through the use of a smaller representative training set for density estimation while the entire data set could be used as the TRAIN set. This would allow

computational cost to be tailored to the available computational resources. Experimental results would indicate the amount of error introduced by such a technique.

Other research could include the use of criteria other than mean square error in step 7. The algorithm could also be modified to allow for the removal of samples in intermediate steps as well as for the reduction of sample resolutions. These techniques might improve reduction but would require careful consideration to ensure the algorithm remains provably correct. Some form of relaxation with respect to outliers, and a maximization of margin, similar to support vector machines may provide enhanced accuracy on high dimension, low sample problems.

BIBLIOGRAPHY

BIBLIOGRAPHY

Archambeau, C., Vrins, F., Verleysen, M. “Flexible and Robust Bayesian Classification by Finite Mixture Models.” European Symposium on Artificial Neural Networks, vol. 12, 2004, pp. 75-80.

Babich, G., Camps, O. “Weighted Parzen-windows for Pattern Classification.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, 1996, pp. 567-570.

Bakir, G. H., Bottou, L., Weston, J. “Breaking SVM Complexity with Cross-Training.” Advances in Neural Information Processing Systems, vol. 17, 2005, pp. 81-88.

Byeungwoo, J., Landgrebe, D. “Fast Parzen Density Estimation Using Clustering-Based Branch and Bound.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, 1995, pp. 950-954.

Dasgupta, S. “Learning Mixtures of Gaussians.” Report No. UCB/CSD-99-1047, University of California Berkeley, 1999.

Davies, S., Moore, A. "Mix-Nets: Factored Mixtures of Gaussians in Bayesian Networks with Mixed Continuous and Discrete Variables." Proceedings of the Conference on Uncertainty in Artificial Intelligence, vol. 15, 2000, pp. 168-175.

Davis, D., Hwang, J. "Expanding Gaussian Kernels for Multivariate Conditional Density Estimation." IEEE Transactions on Signal Processing, vol. 46, 1998, pp. 269-275.

Devijver, P., Kittler, J. “Pattern Recognition a Statistical Approach.” Prentice Hall

International, 1982.

Duda, R., Hart, P., Stork, D. “Pattern Classification.” Wiley-Interscience, 2001.

Duin, R., Pekalska, E., Tax, D. “The Characterization of Classification Problems by Classifier Disagreements.” International Conference on Pattern Recognition, vol. 2, 2004, pp. 140-143.

Fukunaga, K., Hayes, R. “The Reduced Parzen Classifier.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, 1989, pp. 423-425.

Girolami, M., Chao, H. “Probability Density Estimation from Optimally Condensed Data Samples.” IEEE Transactions on Pattern Analysis, vol. 25, 2003, pp. 1253-1264..

Gregor, J., Liu, Z. "A Bayesian Framework for Regularized SVM Parameter Estimation." IEEE International Conference on Data Mining, vol. 4, 2004, pp. 99-105.

Hall, P. and Wand, M.P. “On the accuracy of binned kernel density estimators.” Journal of Multivariate Analysis, vol. 56, 1996, pp. 165-184.

Holder, L., Russell, I., Markov, Z., Pipe, A., Carse, B. “Current and Future Trends in Feature Selection and Extraction for Classification Problems.” International Journal of Pattern Recognition and Artificial Intelligence, vol. 19, 2005, pp. 133-142.

Mitra, P., Murthy, C., Sankar, K. “Density-Based Multiscale Data Condensation.” IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 24, 2002, pp. 734-747.

Moore, A. “Very fast EM-based Mixture Model Clustering Using Multiresolution kd-trees.” Advances in Neural Information Processing Systems, vol. 11, 1998, pp. 543-549.

Steinwart, I. “Sparseness of Support Vector Machines.” Journal of Machine Learning Research, vol. 4, 2003, pp. 1071-1105.

Tipping, M., “The relevance vector machine.” Advances in Neural Information Processing Systems, vol. 12, 2000, pp. 652-658.

Toussaint, G. “Proximity Graphs for Nearest Neighbor Decision Rules: Recent Progress.” Symposium on Computing and Statistics, vol. 34, 2002, pp. 86-105.

Vita

Dave Beck was born in Washington D.C. and raised in Utah and Wyoming. He earned a Bachelor of Science degree in Computer Science from Utah State University in 2001. He received his Master of Science degree in Computer Science at the University of Tennessee, Knoxville in 2006. He is employed at the Naval Air Warfare Center, Weapons Division, China Lake, Ca.