Classification via distance profile nearest neighbors

by

Ashley M. Moraski

A Project

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Applied Statistics

by

May 2006

APPROVED:

Jayson D. Wilbur, Advisor

Bogdan M. Vernescu, Department Head

Abstract

Most classification rules can be expressed in terms of a distance (or dissimilarity) from the point to be classified to each of the candidate classes. For example, linear discriminant analysis classifies points into the class for which the (sample) Mahalanobis distance is smallest. However, dependence among these point-to-group distance measures is generally ignored. The primary goal of this project is to investigate the properties of a general non-parametric classification rule which takes this dependence structure into account. A review of classification procedures and applications is presented. The distance profile nearest-neighbor classification rule is defined. Properties of the rule are then explored via application to both real and simulated data and comparisons to other classification rules are discussed.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to Dr. Jayson Wilbur, for not only advising me on this project but also for his patience and encouragement while doing so.

I would also like to thank the rest of the Statistics professors at WPI for all of their guidance and support as well. Dr. Andrew Swift for his constant willingness to help, especially with R syntax, which lessened many nights of frustration. Dr. Balgobin Nandram for his belief in me, especially early on, and showing me that constant hard work can pay off. Dr. Joseph Petruccelli for not only challenging me to do the best I can, but always showing me how to clarify myself and to communicate better.

And to my entire family for their support, both financially and emotionally, I am extremely grateful. Without their constant understanding and compassion, I would not have been able to accomplish all that I have already.

Contents

List of Figures

List of Tables

Chapter 1

Background

1.1 Introduction

1.1.1 Classification

The statistical problems of discrimination and classification are concerned with characterizing differences between groups and constructing rules for assigning objects of unknown origin to the correct group. While the two terms are sometimes used interchangeably, *discrimination* refers to methods for characterizing group separation, and *classification* refers to methods for assigning objects to groups. Additionally, it is important to emphasize that in problems of classification, knowledge is available about group divisions and the characteristics associated with each group prior to the analysis. That is tosay, groups are predetermined. This factor distinguishes classification from *clustering*, a class of techniques sometimes referred to as unsupervised classification, in which no knowledge is assumed *a priori* about the groups. In fact, the goal of clustering is to identify a grouping for these unlabeled objects. For a review on clustering, see the following references, [Banks et al., 2004] and [Gnanadesikan et al., 1989].

Methods for classification are generally viewed in two ways. The first view considers a function which maps the multidimensional feature vector $\boldsymbol{X}$ to the classes. Under this view, the sample space is partitioned into regions and a class is associated with each region. The second approach is based on estimating the probability that an object with feature vector $\boldsymbol{X}$ belongs to each of the classes. With prior probabilities of group membership specified (or a assumed equal), classification is based on posterior probabilities of group memebership [Hand, 1997]. An associated loss or cost of misclassification can also be incorporated in a decision-theoretic approach. In all cases, the construction of classification rules requires objects for which the group is known. These objects are commonly divided into two subsets, the *training set* (or training data) which is used to define the rule and the *test set* (or test data) which is used for validation.

1.1.2 Notation and definitions

In the g-group classification problem there are $g \geq 2$ well-defined groups denoted by $G_1, G_2, \ldots, G_g$. A new object is classified into one of these groups based on p relevant predictor variables (or features) which are denoted by the feature vector $\boldsymbol{X} = \{X_1, X_2, \ldots, X_p\}$. The observed p-dimensional feature vector will be denoted $\boldsymbol{x}$ and its group by $G(\boldsymbol{x}) \in \{G_1, G_2, \ldots, G_g\}$ [Johnson & Wichern, 2002]. When the distribution of $\boldsymbol{X}$ for the population defined by each of the g groups can be assumed to follow a known distribution form, parametric techniques can be used to allocate new objects into each of the groups. The distribution of $\boldsymbol{x}$ for G_i will be denoted $f_i(\boldsymbol{x})$, the likelihood of observing $\boldsymbol{x}$ in group G_i.

In order to classify objects of unknown group origin, *classification rules* are constructed which define how objects are to be allocated. As was mentioned previously these rules can be considered a partitioning of the sample space, where each non-

overlapping region is associated with a mutually exclusive group. Thus a classification rule $\hat{G} : \boldsymbol{X} \rightarrow \{1, \ldots, g\}$ attempts to estimate the function $G : \boldsymbol{X} \rightarrow \{1, \ldots, g\}$ which maps points on the sample space to their correct group.

For example, if the distributions are known, then for the $g = 2$ case, a common allocation rule is to classify the object into G_1 if $f_1(\boldsymbol{x}) > f_2(\boldsymbol{x})$ (assuming equal prior probabilities and cost of misclassification for each group). As a result, parametric techniques can generally be investigated through the likelihood ratio.

Letting $c(1|2)$ be the cost associated with assigning an object to group 1 that is really in group 2 and $c(2|1)$ be the cost associated with assigning an object to group 2 that is really in group 1, as well as p_1 and p_2 be the prior probabilities for each group, then an object is classified into G_1 if $f_1(\boldsymbol{x}) * p_1 * c(2|1) > f_2(\boldsymbol{x}) * p_2 * c(1|2)$. This reduces to the classification rule mentioned above for equal costs and equal priors, by taking the ratio between costs to be 1 and the ratio between priors to be 1. This is all an example of the second approach to classification discussed above [Johnson & Wichern, 2002].

This can be generalized for a g-group classification. Letting p_i denote the prior probability of objects in group G_i, $P(j|i)$ be the probability of classifying an object from G_i into G_j, and $c(j|i)$ be the cost associated with classifying an object from group G_i into G_j, then the density ratio can be utilized where an object is classified into group i if $\frac{f_i(\boldsymbol{x})}{f_j(\boldsymbol{x})} \geq \frac{p_j}{p_i} * \frac{c(i|j)}{c(j|i)}$ for all competing groups j.

Errors

In constructing classification rules, the common objective is to identify a rule which minimizes the probability of misclassification. That is to say, assigning an object to an incorrect group. When costs of misclassification are not equal, a the choice of classification rule should take this into account by attempting to minimize *expected*

cost of misclassification, $ECM = \sum_{i=1}^{g} p_i \sum_{j=1 j \neq i}^{g} P(j|i)\, c(j|i)$. If the prior probabilities are unknown or complete knowledge is not available, it is common practice to assume that they are equal. When the cost of misclassification is equal this is equivalent to minimizing the *total probability of misclassification*, TPM, or actual error rate.

However, the distribution of $\boldsymbol{x}$ in each group is assumed to be known for these criteria, or equivalently that the size of the training and test sets are infinite. The expected value of this error rate over the sample of fixed size is referred to as the *expected error rate* [Hand, 1997]. And the *apparent error rate* is defined as the fraction of objects in the sample of known objects that are misclassified by the classifier and this is commonly used instead because it does not depend on the distributions [Johnson & Wichern, 2002].

Methods for counting errors, and in particular how to count them for the nearest-neighbor method, have been investigated thoroughly. Because simple counting may not be possible if all observations are used in the training data set in order to determine the best rule possible, other methods have been proposed.

Letting L_G be the probability of error for classifier G, then the best classifier denoted G^* is defined to be $G^* = argmin_{G:\boldsymbol{X} \to \{1,\ldots,g\}} L_G$. G^* is known as Bayes classifier or Bayes Rule because G^* depends on the distributions. When the minimal probability is found denoted L_{G^*} this is known as Bayes error. Thus the risk associated with Bayes error is denoted, R^*. As a result, Cover and Hart show that for large samples where the risk associated with these probabilities is as follows, R_n is the expectation of the incurred loss and for large samples $R = lim_{n\to\infty} R_n$, then $R^* \leq R \leq R^* (2 - gR^*/(g-1))$. The upper bound is important because it shows that for any number of groups, the probability of error for the nearest neighbor rule is bounded above by twice the Bayes error probability [Cover & Hart, 1967]

When comparing the error associated with a single class to that of a group classification analysis proves very difficult. Single class error depends mainly on noise, class distribution and distributional dimensions, which tends to be rather large, while the group classification error, although relying on similar properties, has a lower error due to overlap [Fukunaga & Flick, 1984].

1.1.3 Common parametric classification methods

The most common parametric classification technique is linear discriminant analysis (LDA). This technique is optimal when the data are multivariate normally distributed with equal covariance matrices and thus the log-likelihood ratio can be expressed as a linear function of the $\boldsymbol{x}$'s. It should be noted however that this does not have to be the case for LDA to perform well, but it is the best technique when given that type of data. Similarly quadratic discriminant analysis (QDA) is the optimal rule for multivariate normal populations with unequal covariances. In canonical discriminant analysis, or Fisher's Linear Discriminant Functions (LDF), common covariance matrices are again assumed and yields the same results as linear discriminant analysis in the case where $g = 2$. All of these techniques will perform reasonably well even if the assumptions about the population specified do not hold [Johnson & Wichern, 2002]. However, when a parametric distributional form cannot be assumed, non-parametric techniques for classification, such as nearest-neighbor methods should also be considered.

1.2 Nearest-neighbor methods

1.2.1 Initial development

Fix and Hodges are credited with being the first to propose the notion of a nearest-neighbor classification technique [Fix & Hodges, 1951, McLachlan, 1992]. The nearest-neighbor rule assigns an unclassified object to the group of the nearest set of already classified objects. In nearest-neighbor classification an estimate to the posterior distribution for the classes is based on the proportion of neighbors with a certain class located in the vicinity of the object under consideration [Hand, 1997]. To start, consider the case where there are two distinct groups ($g = 2$), denoted G_1 and G_2 into which $\boldsymbol{x}$ can be classified. Each of the groups can then have an associated p-variate probability distribution, say $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$. Using the idea expressed above that an object is classified into G_1 if the probability is greater for it to land in group 1 as opposed to group 2, $f_1(\boldsymbol{x}) > f_2(\boldsymbol{x})$, then this is simply an allocation into a group based on a majority vote. So, parametric techniques can be used when distributional assumptions are known [Fix & Hodges, 1951].

When the distributions are not known, one way to estimate them is to partition the sample space into non-overlapping tolerance regions, resulting in data reduction as in the first approach described above. Tolerance regions are found as functions of the training set instead. This idea is similar to representing a population through its sample parameters. By estimating the distributions through the training set, the data are instead represented through the parameters of the tolerance regions. This type of a nearest-neighbor method identifies the distance to the k^{th} nearest object in G_i and assigns the object to the group in which the distance is minimal or nearest [Patrick & Fischer, 1970].

The major distinction between the two methods described in the previous two

paragraphs is as follows. First, Fix and Hodges made comparisons between the linear discrimination rule to nonparametric discrimination, specifically for small samples. Second, Patrick and Fisher investigated tolerance intervals (regions), specifically nonparametric, where a region is defined as a function of the data to obtain the probability of the next observation being in this region, free from distribution. This resembles a generalized version of nearest neighbor.

In addition, it has been shown that the nearest-neighbor rule can be modified to no longer just classify based on the majority, but rather it is also possible to reject a class instead of just accepting a plausible one through the use of a threshold factor [Tomek, 1976]. In our study we do not consider a rejection option, but it is possible to allow for it in further study.

So there are two basic approaches to nearest neighbor described so far. The first identifies the k nearest objects to the unclassified object and fixes a radius surrounding them, then allocates the object to the group which contains the largest proportion of objects within the radius. The second identifies the distance to the kth nearest object in each group, then allocates the unclassified object into the group where the distance is the smallest. It should be noted that Goldstein was the one to restructure the nearest neighbor rule to allow for such comparison of distances of the test samples to the training samples and classifies based on this nearness [Dasarathy, 1991].

1.2.2 Further methods

Further generalizations of the nearest neighbor method consider a weighting technique on observations where closer sample observations are more influential and yields a smaller rates of misclassification. Thus each point has an associated influence based on its distance from the observation under consideration. Yet the

application of this technique to properties involving large sample sets allowed for further investigation [Dudani, 1976].

A suggestion was made for an alternative definition of neighborhood of a point in the feature space. By adding a pair of distance and directional constraints, the voting method can be updated within a contained sphere given a fixed radius, as opposed to the usual fixing of the radius based on the selection of nearest neighbors chosen [Dasarathy, 1991].

1.2.3 Selection of k

In estimation of any sort, there is always an issue with maintaining a balance between variance and bias. Since over fitting to the data will lead to more bias, however under fitting will increase the variance while decreasing the bias, it is preferable to find the case that allows for an even trade off of both. In nearest-neighbor methods, a larger k will cause less variance in estimating the probability but may lead to a higher bias in estimating $f_i(\boldsymbol{x})$ [Hand, 1997]. Most work began with simply the 1-NN rule but with the introduction of the 2-NN rule, it was demonstrated that in fact the 2-NN rule was much of an improvement in that it resulted in fewer errors and further provided advances to $2k$-NN rules and using more than just two classes [Dasarathy, 1991].

Previously, Fix and Hodges's method simply assigned classes arbitrarily when ties resulted; however, a technique was developed to handle the allocation of ties in a more systematic manner. Through exploration in the voting method under the $2k$-NN rule with two classes, the suggestion arose to choose a lower odd-numbered ($2k$-1)NN rule that can provide results as good as those for large samples [Dasarathy, 1991].

So, when the value of k is large, each individual sample observation within the

training set has less influence, and leads to a robust design; however, k is restricted and larger k require more computational time. It has been suggested that the best way to find k, is to first test many possible choices and then compare to already classified observations to observe how well it performs [Dasarathy, 1991]. While many of these techniques focus primarily on the two group case, application of these methods can be generalized to the multigroup case.

1.2.4 Sample size selection

Most of the techniques and properties mentioned hold when sample sizes are large. In fact, after their initial publication, Fix and Hodges implemented their procedures for smaller sample sizes, which called into question some of their initial assumptions [Fix & Hodges, 1952]. Sample size selection is always an important task in that too few sample observations will result in a poor rule, while obtaining a sample large enough to employ the notions above may not be feasible. In terms of storage and computational time, efficiency also needs to be considered. Certain constraints on data sets and how to employ sample size selection have been investigated. It has been suggested that how quickly the error rate converges to R and the loss associated are both fundamental aspects to consider in the selection of sample size of data to be used [Peterson, 1970]. The very specific case where small sample sets are drawn from uniform distributions has been explored. In this case, the errors are close to R^* even with small sample sizes, but results are best when sample sizes are nearly equal [Dasarathy, 1991]. When considering unequal sample sizes for different classes the bias created was considered and another weighting technique employed which in essence changes the metric [Dasarathy, 1991]. The idea of applying penalties has been seen for many other areas and not just classification on how to handle bias and variance issues, so this idea expands across other areas of statistics.

It was verified that in fact the training set should contain elements in direct proportion as those found under the practical situations and thus deemed that the training set should be fully representative, which is somewhat obvious [Dasarathy, 1991]. That is to say, taking unrepresentative samples will lead to poor results.

1.2.5 Dimensionality

Because it becomes very difficult to conceptualize classification problems in higher-dimensions and in many instances the groups cannot be easily separated, it is necessary to reduce nearest-neighbor problems to a realm that can be handled more easily and distinctly. Methods have already been inspected to estimate the required dimensions of the feature characteristics in order to determine the appropriate number of features to represent data and provide a possible reduction in dimensions [Pettis & Dubes, 1979]. At higher dimensions finding the Bayes error rate, R, may not be adequately performed simply by increasing the sample size. Instead, it has been suggested that the relationship between the bias and sample size should be used to estimate the error by first finding the mean error for several sample sizes and then extrapolating the Bayes estimate from the relationship [Fukunaga & Hummels, 1987].

1.2.6 Alternative methods

Discriminant Adaptive Nearest Neighbor

In higher dimensions, the bias results because the nearest neighbor may in fact be far away. Through adaptation of the distance metric, the bias may decrease while leaving the variance the same, and thus such a technique can be advantageous over previously considered methods. One such technique is the discriminant

adaptive nearest-neighbor (DANN) [Hastie & Tibshirani, 1996]. The metric is adjusted at each point under consideration denoted $\boldsymbol{x_0}$ based on the class distribution surrounding it developed from a neighborhood of some fixed set of points and then the adapted metric is used for the nearest-neighbor rule. The DANN metric at each point $\boldsymbol{x_0}$ can be defined as $D(\boldsymbol{x},\boldsymbol{x_0}) = (\boldsymbol{x}-\boldsymbol{x_0})^T \boldsymbol{\Sigma} (\boldsymbol{x}-\boldsymbol{x_0})$ where $\boldsymbol{\Sigma} = \boldsymbol{W}^{-1/2}\left[\boldsymbol{W}^{-1/2}\boldsymbol{B}\boldsymbol{W}^{-1/2} + \boldsymbol{\epsilon}\boldsymbol{I}\right]\boldsymbol{W}$ where $\boldsymbol{W}$ is the pooled within-class covariance matrix $\sum_{k=1}^{K} \pi_k \boldsymbol{W}_k$ and $\boldsymbol{B}$ is the between class covariance matrix $\sum_{k=1}^{K} \pi_k (\bar{\boldsymbol{x}}_k - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_k - \bar{\boldsymbol{x}})^T$, both computed only using the neighborhood of fixed points. This metric is the one that is then used in the nearest-neighbor rule for $\boldsymbol{x_0}$. In general $\epsilon = 1$ because it is the parameter used to prevent using points "too far away" from $\boldsymbol{x_0}$ [Hastie et al., 2001].

Support Vector Machines

When classes overlap it is difficult to apply a linear classification rule to separate them. With support vector machines (SVM) a nonlinear boundary is determined through construction of a linear boundary in a transformed feature space. In the g=2 case, the classification rule is constructed as follows: Let the training data consist of n pairs, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $x_i \in \Re^p$ and $y_i \in \{-1, 1\}$. Defining a hyperplane as $x : f(x) = x^T\beta + \beta_0 = 0$ where β is a unit vector. The classification rule is then $G(x) = sign\left[x^T\beta + \beta_0\right]$. It is desired that the hyperplane that allows for the largest possible margin between training points for class 1 and -1 be determined. When overlap amongst classes occurs, it is still desired to optimize the margin, however, the definition is relaxed and some points may be allowed to cross over into the margin region while still bounding the sum of all the proportional amounts by which the predicted observations are on the the wrong side of the margin. So with SVM, the observations that are clearly inside their boundary do not influence the formation of the boundary and instead the boundary is more

determined by the observations that are close to it where as with linear discriminant analysis the structure of the covariance matrix influences boundary formation [Hastie et al., 2001]. For a more thorough analysis of the comparison and contrast between support vector machines and linear discriminant analysis see the article by Gokcen and Peng [Gokcen & Peng, 2002].

Chapter 2

Distance profile nearest-neighbor classification

2.1 Nearest-neighbor classification rules

For a general g-group classification problem, $\boldsymbol{x}_{ij}$ denotes the observed p-dimensional feature vector for the j^{th} training sample in the i^{th} group where $i = 1, \ldots, g$, $j = 1, \ldots, n_i$. To clarify the two approaches to nearest neighbor discussed previously, with the first approach to identify the distance to the k^{th} nearest object in group i for each $i = 1, \ldots, g$. assign $\boldsymbol{x}$ to the group in which the distance is the smallest. Since nearest neighbor classification first identifies the nearest objects of previously determined classes, it is important to distinguish what qualifies an object as being "nearest" and which distance function is applied. The second is to identify the k^{th} closest $\boldsymbol{x}_{ij}$ and fix a radius surrounding the p-dimensional feature vector $\boldsymbol{x}$ that extends out as far as that k^{th} closest object. Next, assign $\boldsymbol{x}$ to the group that contains the most objects within the fixed radius.

For the first approach, first let $d_{ij}(\boldsymbol{X})$ denote the distance between each $\boldsymbol{x}_{ij}$ and

$\boldsymbol{x}$. For example, if distance in the p-dimensional feature space is measured using the Euclidean metric then

$$d_{ij}\left(\boldsymbol{X}\right) = \sqrt{\left(\boldsymbol{x}_{ij} - \boldsymbol{X}\right)'\left(\boldsymbol{x}_{ij} - \boldsymbol{X}\right)}.$$

So $\boldsymbol{x}$ is assigned to the group for which $d_{ij}(\boldsymbol{x})$ is the smallest.

For the second approach the distance is fixed within some sphere. In higher dimensions this approach gets more complicated as was discussed in the section on discriminant adaptive nearest-neighbor method. See Section 1.2.6. This time the distances are ordered within each group. So, let $d_{i(j)}\left(\boldsymbol{X}\right)$ denote the distance from $\boldsymbol{X}$ to its jth nearest neighbor in group i such that for each $i = 1, \ldots, g$

$$d_{i(1)}(\boldsymbol{X}) < d_{i(2)}(\boldsymbol{X}) < \cdots < d_{i(n_i)}(\boldsymbol{X})$$

are the order statistics of $d_{i1}(\boldsymbol{X}), d_{i2}(\boldsymbol{X}), \ldots, d_{in_i}(\boldsymbol{X})$.

This type of k-nearest-neighbor classifier identifies the k nearest training samples to $\boldsymbol{x}$ and classifies it into the group which contains the largest number of the k nearest neighbors. These methods can also be modified to allow for the situations where distance is measured to the k^{th} nearest neighbor in each group or the average distance to the k^{th} nearest neighbors in each group is found.

For R code that performs an algorithm for nearest-neighbor see Appendix A.

2.2 Advantages and disadvantages

However, these methods are not without their limitations and are based on some assumptions. Although, nearest neighbor is distribution free and the classifier then has no explicit functional form, it is very difficult to check the assumption that the

distribution is locally constant near $\boldsymbol{x}$. Also, the choice of the distance function must be taken into consideration. It must be appropriate and meaningful. For example, Euclidean distance is usually the default choice but may not be appropriate as in such cases where the variables are of very different magnitudes and must be standardized first. Also, distances in high dimensions becomes complicated and assigning one object to be nearer than other gets blurred because as p gets increasingly larger the ratio of nearest to furthest neighbors approaches 1.

2.3 Classification via interpoint distance profiles

To address these specific problems with the k-NN classifier an alternative classifier is proposed. The g-dimensional distribution group-distance space classification is made as classification via interpoint distance profiles. This will be referred to as distance profile nearest-neighbor (DPNN). For R code that performs the algorithm for distance profile nearest-neighbor exlpained below, see Appendix A.

Let $D_i(\boldsymbol{X})$ denote the distance between $\boldsymbol{X}$ and group i. For example, if the distance from an object,$\boldsymbol{x}$, to group i is defined as the distance to its first nearest neighbor in group i then $D_i(\boldsymbol{x}) = d_{i(1)}(\boldsymbol{x})$. This roughly corresponds to the concept of single linkage in cluster analysis.

Once D_i is defined for each of the groups $i = 1, \ldots, g$, the g-dimensional group distance profile for $\boldsymbol{X}$ is denoted by $\boldsymbol{D}(\boldsymbol{X}) = [D_1(\boldsymbol{X}), D_2(\boldsymbol{X}), \ldots, D_g(\boldsymbol{X})]$. Furthermore, the g-dimensional distance profiles for each of the training samples is denoted by $\boldsymbol{D}_{ij} = [D_1(\boldsymbol{x}_{ij}), D_2(\boldsymbol{x}_{ij}), \ldots, D_g(\boldsymbol{x}_{ij})]$, for $i = 1, \ldots, g$ and $j = 1, \ldots, n_i$, where $\boldsymbol{x}_{ij}$ is ignored when computing $D_i(\boldsymbol{x}_{ij})$.

An alternative to this is examined in order to draw inferences about this method. Instead of obtaining distance profiles based on measuring the distance to each group,

when the class is already known for an observation $\boldsymbol{x}_{ij}$, $\boldsymbol{x}_{ij}$ is included in the measurement to its own class creating a distance of 0 in that distance profile and thus $\boldsymbol{x}_{ij}$ is not ignored when computing $D_i(\boldsymbol{x}_{ij})$. This will be referred to as anchored distance profile nearest-neighbor (anchored DPNN). For R code that performs the algorithm for anchored distance profile nearest-neighbor see Appendix A.

So, $\delta_{ij}(\boldsymbol{X})$ denotes the distance between $\boldsymbol{D}_{ij}$ and $\boldsymbol{D}(\boldsymbol{X})$. For example, if distance in the g-dimensional distance-profile space is measure by the Euclidean metric then

$$\delta_{ij}\left(\boldsymbol{X}\right) = \sqrt{(\boldsymbol{D}_{ij} - \boldsymbol{D}(\boldsymbol{X}))'(\boldsymbol{D}_{ij} - \boldsymbol{D}(\boldsymbol{X}))}$$

Letting $\delta_{i(j)}\left(\boldsymbol{X}\right)$ denote the distance from $\boldsymbol{D}(\boldsymbol{X})$ to its jth nearest distance-profile neighbor in group i such that for each $i = 1, \ldots, g$

$$\delta_{i(1)}(\boldsymbol{X}) < \delta_{i(2)}(\boldsymbol{X}) < \cdots < \delta_{i(n_i)}(\boldsymbol{X})$$

are the order statistics of $\delta_{i1}(\boldsymbol{X}), \delta_{i2}(\boldsymbol{X}), \ldots, \delta_{in_i}(\boldsymbol{X})$ then the proposed k-nearest distance profile neighbor classifier would classify $\boldsymbol{X}$ into the group that contains the most k nearest distance-profile neighbors.

Figure 2.1: Graphical Representation of Distance Profile for unlabeled observation

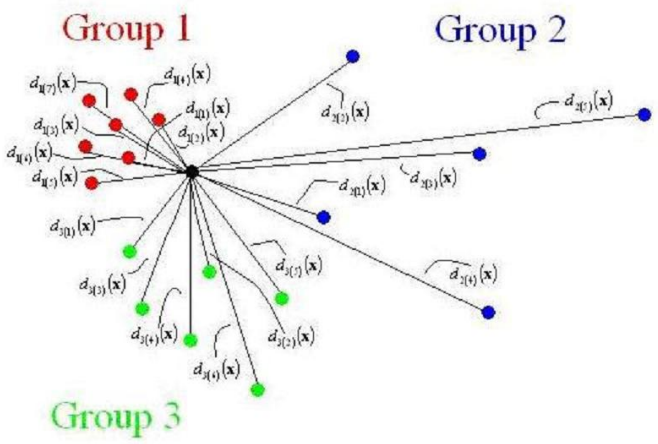
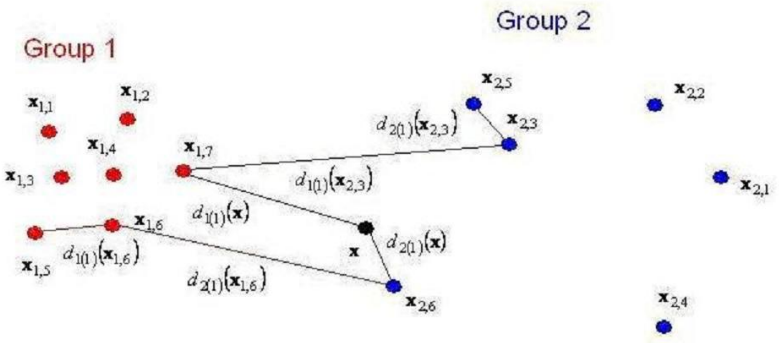


Figure 2.2: Graphical Representation of Distance Profile notation



Chapter 3

Applications of classification

Classification has many applications in the life sciences, social sciences, computer applications, business, and forensic investigations. Major applications include diagnostic testing, decision making, and pattern recognition. In this chapter, there will be a discussion on the following topics. In terms of diagnostic testing, the liver disorder data shows an application of classification into the diagnosis of medical conditions, in particular liver disorders. With the dermatology data, another medical application arises for classification in disease diagnosis. The diabetes data also shows another medical application for classification to determine the possiblility for disease development. Another medical application and the associated risk involved with development of cancer arises with the breast cancer data. The hepatitis data demonstrates a motivation for classification by how it is applied to determine the impact of a disease on life or death, thus tying into the next category of decision making. For decision making, the contraceptive method data demonstrates an application of classification as to the effectiveness of certain methods of birth control as a means of socioeconomic development and how it can be used to motivate decisions in marketing and product availability. From a biological view, classification

can be implemented as demonstrated with the iris data in species classification. The ionoshpere data shows how classification is a technique that can be applied to frequency transmission for communication and governmental defense to aid in decision for strategical purposes. The sonar data set is another application of classification for military purposes to aid in decisions concerning defense. Finally, the voting data demonstrate classification use in political aspects used to make decision in accordance with political agendas. Pattern recognition is a third form of motivation with applying classification. It is represented by the vowel data which shows how classification can be used to help improve aspects of communication through speech recognition.

3.1 Diagnostic testing

3.1.1 Liver disorders

The liver disorders data contains entries to determine if certain liver disorders arise due to excessive alcohol consumption. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 345 observations and no observations were removed. There are 2 classes with 145 observations from the {1} class, and 200 observations from the {2} class. It appears that consumption of more than 5 drinks daily is some sort of a selector for the classes. There are a total of 6 predictor variables all of which are continuous. The 6 predictors are (1) mean corpuscular volume - (mcv), (2) alkaline phosphotase - (alkphos), (3) alamine aminotransferase - (sgpt), (4) aspartate aminotransferase - (sgot), (5) gamma-glutamyl transpeptidase - (gammagt), and (6) number of half-pint equivalents of alcoholic beverages consumed per day - (drinks). For a complete description of this data set, see under PC-Beagle Users Guide the BUPA Medical Research Ltd.

database maintained by Richard S. Forsyth.

3.1.2 Dermatology

The dermatology data was obtained to identify the problem with diagnosing certain dermatological diseases because they all share the clinical features of erythema and scaling with very little differences. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 366 observations; however once all missing entries were removed only 358 remained. There are 6 classes with 111 observations from the psoriasis {1} class, 60 observations from the seboreic dermatitis {2} class, 71 observations from the lichen planus {3} class, 48 observations from the pityriasis rosea {4} class, 48 observations from the chronic dermatitis {5} class, and 20 observations from the pityriasis rubra pilaris {6} class. There are a total of 34 predictor variables, where 32 are categorical having integer values ranging from 0 to 3, 1 is binary, and 1 is continuous.

3.1.3 Diabetes

The Pima Indians diabetes data was obtained from the National Institute of Diabetes and Digestive and Kidney Diseases. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 768 observations and no observations were removed. There are 2 classes with 500 observations from the negative for diabetes {0} class, and 268 observations from the positive for diabetes {1} class. There are a total of 8 predictor variables all of which are continuous. The 8 predictors are (1) Number of times pregnant, (2) Plasma glucose concentration a 2 hours in an oral glucose tolerance test, (3) Diastolic blood pressure in mm Hg, (4) Triceps skin fold thickness in mm, (5) 2-Hour serum insulin in mu U/ml, (6) Body mass index in kg/m^2, (7) Diabetes pedigree function, and (8) Age in years. For a

complete description of this data set, see its past usage in [Smith & Johannes, 1988].

3.1.4 Breast cancer

The breast cancer data was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 699 observations; however once all missing entries were removed only 683 remained. There are 2 classes with 444 observations from the benign {2} class and 239 observations from the malignant {4} class. There are a total of 9 predictor variables, all of which are categorical and each has integer entries that range from 1 to 10. The 9 predictors are (1) Clump Thickness, (2) Uniformity of Cell Size, (3) Uniformity of Cell Shape, (4) Marginal Adhesion, (5) Single Epithelial Cell Size, (6) Bare Nuclei, (7) Bland Chromatin, (8) Normal Nucleoli, and (9) Mitoses. For a complete description of this data set, see its past usage in [Michalski & Lavrac, 1986].

3.1.5 Hepatitis

The hepatitis data was donated by G. Gong at CMU. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 155 observations; however once all missing entries were removed only 80 remained. There are 2 classes with 13 observations from the die {1} class and 67 observations from the live {2} class . There are a total of 19 predictor variables, where 13 are binary, 5 are categorical and 1 is continuous. For a complete description of this data set, see its past usage in [Diaconis & Efron, 1983] or [G. Cestnik & Bratko, 1987].

3.2 Decision making

3.2.1 Contraceptive method

This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The detail of interest is to examine the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics in order to examine economic development and make decisions accordingly. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 1473 observations and no observations were removed. There are 3 classes with 629 observations from the no use {1} class, 333 from the long term use {2} class and 511 observations from the short term use {3} class. There are a total of 9 predictor variables, some of which are categorical, some of which are binary and some of which are continuous. The 9 predictors are (1) Woman's age (continuous), (2) Woman's education (categorical having integer values ranging from 1 to 4), (3) Husband's education (categorical having integer values ranging from 1 to 4), (4) Number of children ever born (continuous), (5) Wife's religion (binary : Islam, Non-Islam), (6) Woman's work status (binary), (7) Husband's occupation (categorical having integer values ranging from 1 to 4), (8) Standard-of-living index (categorical having integer values ranging from 1 to 4), and (9) Media exposure (binary). For a complete description of this data set, see its past usage in [Lim & Shih, 1999].

3.2.2 Iris

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to

this day. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 150 observations and no observations were removed. There are 3 classes with 50 observations from the Setosa {1} class, 50 observations from the Versicolour {2} class, and 50 observations from the Virginica {3} class. There are a total of 4 predictor variables all of which are continuous. The 4 predictors are (1) sepal length in cm, (2) sepal width in cm, (3) petal length in cm, and (4) petal width in cm. For a complete description of this data set, see its original introduction in [Fisher, 1936].

3.2.3 Ionosphere

This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 351 observations and no observations were removed. There are 2 classes with 126 observations from the bad (b) {1} class and 225 observations from the good (g) {2} class. The were 34 predictor variables in the original data set; however, one was removed because all the entries were zero and caused problems. The remaining 33 predictors are all continuous. For a complete description of this data set, see its past usage in [Sigillito & Baker, 1989].

3.2.4 Sonar

This data was contributed by Terry Sejnowski, now at the Salk Institute and the University of California at San Deigo. The data set was developed in collaboration with R. Paul Gorman of Allied-Signal Aerospace Technology Center. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a

total of 208 observations and no observations were removed. There are 2 classes with 11 observations from the mine (M) {1} class and 97 observations from the rock (R) {2} class. There are a total of 60 predictor variables with values that lie in the range of 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. Two different extractions were performed based on the two different papers. The first was to use the 60% for training data and 40% for testing data, however the second was a strict allocation of 104 observations to each set. For a complete description of this data set, see its past usage in [Gorman & Sejnowski, 1988].

3.2.5 Voting

This data set includes votes for each of the 1984 U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA contributed by Jeff Schlimmer. It is one of many databases found at [D.J. Newman & Merz, 1998]. The dataset contains a total of 435 observations; however once all missing entries were removed only 232 remained. There are 2 classes with 124 observations from the democrat {1} class and 108 observations from the republic {2} class. There are a total of 16 predictor variables as mentioned above, all of which are binary. For a complete description of this data set, see its past usage in [Schlimmer, 1987].

3.3 Pattern recognition

3.3.1 Vowel

The vowel data was obtained through the connectionist benchmark of [D.J. Newman & Merz, 1998]. The dataset contains a total of 990 observations and no observations were removed.

There are 11 classes, as speaker independent recognition of the eleven steady state vowels of British English. For each utterance, there are ten floating-point input values, so there are a total of 10 predictor variables. The vowels appear in the following words: heed, hid, head, had, hard, hud, hod, hoard, hood, who'd, heard. For a complete description of this data set, see its past usage in [Rabiner & Schafer, 1978].

Table 3.1: Characteristics of data from UCI repository

Data	Number of Observations (without missing entries)	Number of Observations within each class	Type of predictor (categorical (binary,integer) or continuous
liver	345	1 - 145 2 - 200	All 6 predictors are continuous
derm	358	1 (psoriasis) - 111 2 (seboreic dermatitis) - 60 3 (lichen planus) - 71 4 (pityriasis rosea) - 48 5 (cronic dermatitis) - 48 6 (pityriasis rubra pilaris) - 20	33 predictors are categorical, 1 predictor (age) is numeric integer
diabetes	768	0 (negative for diabetes) - 500 1 (positive for diabetes) - 268	All 8 predictors are continuous
cancer	683	2 (benign) - 444 4 (malignant) - 239	All 9 predictors are categorical
hep	80	1 (die) - 13 2 (live) - 67	13 predictors are binary, 5 predictors are categorical, 1 predictor is continuous
contra	1473	1 (no use) - 629 2 (long term) - 333 3 (short term) - 511	4 predictors are categorical, 3 predictors are binary, 2 predictors are numeric integer
iris	150	1 Iris Setosa - 50 2 Iris Versicolour - 50 3 Iris Virginica - 50	All 4 predictors are continuous
ion	351	1 b (bad) - 126 2 g (good) - 225	All 33 predictors are continuous (2nd predictor from original data removed because all 0 entries)
sonar	208	1 M (mine) - 111 2 R (rock) - 97	Each observation is a set of 60 numbers in the range 0.0 to 1.0
vote	232	1 democrat - 124 2 republic - 108	All 16 predictors are binary
vowel	990 train: 528 test: 462	eleven steady state vowels of British English using a specified training set of lpc derived log area ratios	All 10 predictors are numeric floating point input values

Chapter 4

Simulations

In addition to the UCI data, simulations were run in order to compare different methodologies and see which technique performs best under different situations. The following simulations were again replicated to resemble those performed by Hastie and Tibshirani in their attempt to compare their adaptive nearest neighbor classification technique with other nearest neighbor methods available at that time [Hastie & Tibshirani, 1996]. As with the UCI data, training and testing samples were randomly extracted for each simulation 20 times. The goal was to see how the technique using distance profiles compares under the following conditions: (1) introduction of noise into the data sets and the results produced for each of the methods as compared to the results of the methods for the data sets containing only the given predictors, (2) increasing the number of predictors and the effect on method performance, and (3) the effect multimodality on the methods. Through these simulations, traits suggested by the UCI data could be examined more thoroughly as well as other characteristics often observed in practice not already recognized through this collection of data. See Appendix B for the generation of the data.

4.1 Two bivariate normal classes

This set contains two classes with two predictors, (X_1, X_2). Each class is bivariate normal separated by 2 units in X_1 and the predictors have variance $(1, 2)$ and correlation 0.75. Training set size was 200 total observations, 100 observations per class, and test set size was 500 total observations.

4.2 Two bivariate normal classes with noise

This set contains the same simulation as in Simulation 1, but with the introduction of noise. In addition to that contained in Simulation 1, 14 additional predictors were added each having an independent standard Normal distribution. Training set size was 200 total observations, 100 observations per class, and test set size was 500 total observations.

4.3 Four multimodal classes

This set contains four classes each with three independent subclasses that are bivariate normal thus containing two predictors. The means of the subclasses are chosen at random without replacement from the integers $[1, 2, \ldots, 5] \times [1, 2, \ldots, 5]$ and the standard deviation for each subclass is 0.25. Training set size was 240 total observations, 20 observations per subclass, and test set size was 480 total observations.

4.4 Four multimodal classes with noise

This set contains the same simulation as in Simulation 3, but with the introduction of noise. In addition to that contained in Simulation 3, 8 additional predictors were

added each having an independent standard Normal distribution. Training set size was 240 total observations, 20 observations per subclass, and test set size was 480 total observations.

4.5 Four dimensional Gaussian sphere with noise

This set contains two classes with ten predictors, with the last six predictors being noise variables each with an independent standard Normal distribution. The first four predictors of class 1 are independent standard Normal conditioned on the radius being greater than 3 while the first four predictors of class 2 are independent standard Normal without any restrictions. Training set size was 200 total observations, 100 observations per class, and test set size was 500 total observations.

4.6 Ten dimensional Gaussian sphere

This set contains the same simulation as in Simulation 5, but the restrictions on class 1 change. All of the predictors in class 1 are independent standard Normal conditioned on the radius being greater than $\sqrt{22.4}$ and less than $\sqrt{40}$ while all of the predictors of class 2 are independent standard Normal without any restrictions. Training set size was 200 total observations, 100 observations per class, and test set size was 500 total observations.

4.7 Four multivariate normal classes

This set contains four classes with six predictors. Each predictor was independent standard Normal, while the class probabilities were 0.1, 0.2, 0.2, and 0.5 respectively. Training set size was 100 total observations, and test set size was 500 total

observations.

4.8 Two completely separable classes

This set contains two classes with ten predictors. The predictors in class 1 were independent standard Normal, while the predictors in class 2 were independent normal with mean $\sqrt{j}/2$ and variance $1/j$ for $j = 1, 2, \ldots, 10$. Training set size was 200 total observations, 100 observations per class, and test set size was 500 total observations.

4.9 Comparison of Gaussian and uniform spheres

After observance of performance on the data sets and simulations suggested from the literature, further simulations were performed to make inferences about certain characteristic traits of the data under certain methods. In particular, nearest neighbor distance profile performance was measured to compare the distributional characteristics of the populations. Classes from normal distributions and classes with data uniformly spread within a sphere were compared. Under each distribution the influence of the number of predictors was examined as well as the distinguishability between classes at certain distances apart.

Chapter 5

Results

5.1 Data from UCI repository

This section summarizes and compares the results obtained by applying DPNN, NN, LDA, and SVM methods for data from the UCI Repository of Machine Learning [D.J. Newman & Merz, 1998]. Error rates were calculated based on the misclassification rate of each method to provide inferences as to the performance of each method.

From each dataset, training and testing samples were selected 20 times randomly, then standardized. Thus the boxplots for the error rates are based on the 20 errors obtained in each case. All missing observations were removed from each dataset and then standardized to the training data by subtracting the mean then dividing by the standard deviation of each variable in the training sample to obtain the standardized values. The training set was created by randomly allocating 200 observations from the original data and then randomly selecting 200 out of the remaining observations and placing those in the testing set for any dataset that contains over 400 observations. For datasets that contain fewer than 400 observations, 60% of the ob-

servations were randomly allocated to the training data and 40% of the observations to the testing data. The training and testing samples were randomly extracted 20 times from each of the UCI data. This was done in accordance with the methods performed by Hastie and Tibshirani [Hastie & Tibshirani, 1996] as well as Gokcen and Peng [Gokcen & Peng, 2002] on these particular datasets for the purpose of method comparisons.

The first graph in each section displays the error rates for the following methods as denoted by the following:

d denotes the method using anchored distance profiles in which the class is known for each element in the training set and the distance to that class is 0 for the training set, then used to compare the distance profiles for the testing set. This method was used solely as a basis for performance comparison and is not an actual method discussed. Thus, $d_{k_1k_2}$ denotes measuring distance from each object to group using the k_1 nearest neighbor from the other groups and then using the k_2 nearest distance profile. dp denotes the method using distance profiles in which the class is unknown for each element in the training set and the distance to that class is based on a nearest neighbor in that class for the training set, then used to compare the distance profiles for the testing set. This method is classification via DPNN. Thus, $dp_{k_1k_2}$ denotes measuring distance from each object to group using the k_1 nearest neighbor from the other groups and then using the k_2 nearest distance profile. knn denotes the method of NN. Thus, $knn1$ denotes classification based on the first nearest neighbor and $knn5$ denotes classification based on the fifth nearest neighbor. lda denotes the method of linear discriminant analysis. svm denotes the method of support vector machines. Then, a representation of the data is displayed both as a multidimensional scaling of the data in its the feature space based on its p-characteristics and as it appears in distance space based on its distance profiles.

5.1.1 Liver disorders

In the BUPA liver disorders data, it is obvious that LDA performs well. This results because each of the classes contain only continuous predictor elements and thus are from a continuous distribution model, possibly multivariate normal, which is, as stated earlier, when LDA works most effectively. It is surprising, however, that LDA works so well given that the data is not easily linearly separable. SVM does well also though which indicates that linear separation works well for data of this type. It is important to note that nearest neighbor methods, *knn*, perform in a similar manner to the distance profile methods, *dp*, in that larger nearest neighbors cause higher error rates. (See Figure 5.1). In the multidimensional scaling representation of this data, it can be observed that the two classes are not linearly separable; however, this may be a result of an outlying observation at around (-250,30). (See Figure 5.2). The distance profiles show no evidence that the data appears closer to any one group over another, and thus supports why LDA is still more effective in this instance. (See Figure 5.3).

5.1.2 Dermatology

For the dermatology data, the distance profile method works rather well compared to the other methods, especially the distance profile that uses the fifth nearest neighbor to measure point to group distance and then the fifth nearest neighbor to measure group to group distance. This is an instance in which there are multiple classes as well as a larger number of predictors that are primarily categorical. It is important to note that nearest neighbor methods, *knn*, perform in a similar manner to the distance profile methods, *dp*, in that smaller nearest neighbors cause higher error rates. (See Figure 5.4). In this dataset, it appears that the groups could be linearly

Figure 5.1: Boxplots of error rates for liver disorder data

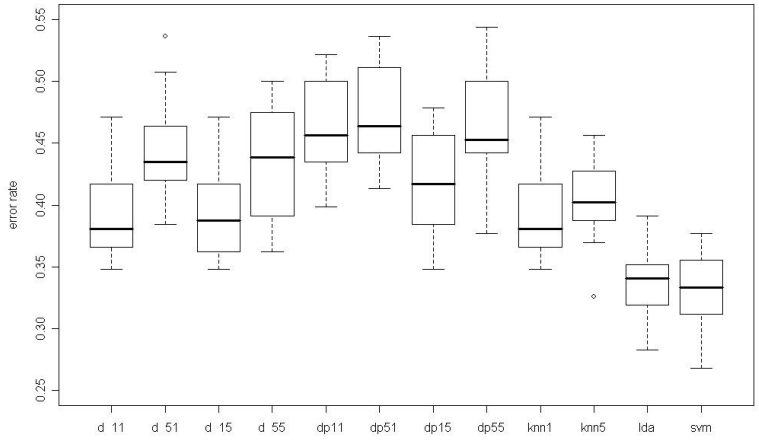


Figure 5.2: Scatterplot of multidimensional scaling representation of liver disorders data in feature space with points in black corresponding to group 1 and points in red corresponding to group 2

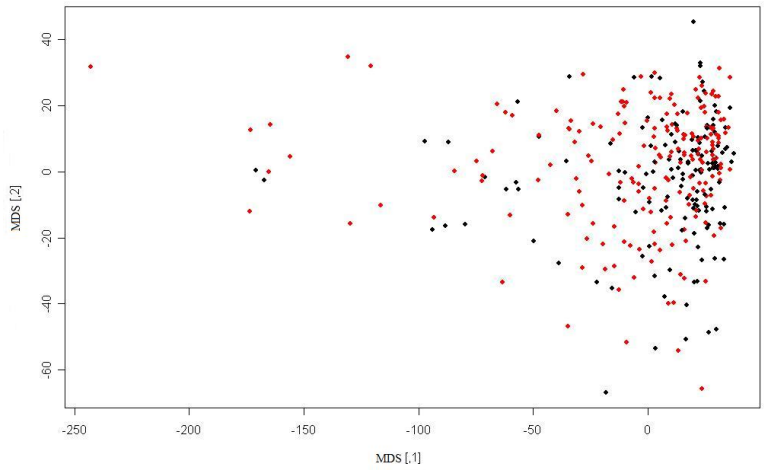
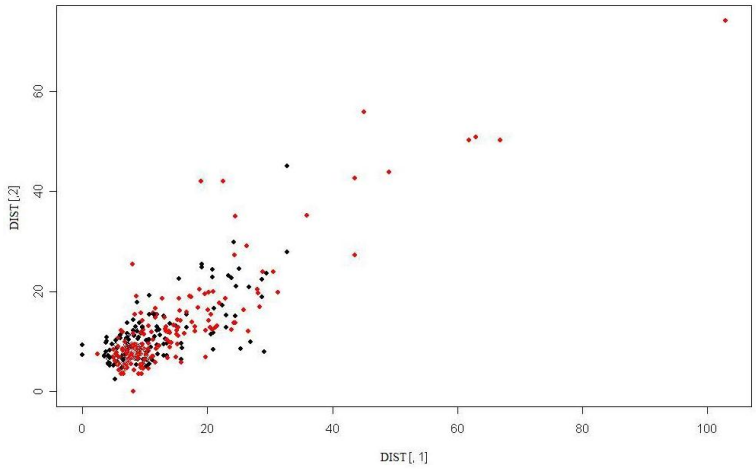


Figure 5.3: Scatterplot of liver disorders data in distance space with points in black corresponding to group 1 and points in red corresponding to group 2



separated from the rest, which is why LDA works relatively well also. However, there are multiple classes and with multiple dimensions, which hinders the effectiveness of the LDA method. (See Figure 5.5). In the distance profile representation of the data the groups are relatively differentiable with some overlap, hence why the distance profile method using first nearest neighbors for both distance measures is not as effective, but the higher order nearest neighbors work better. (See Figure 5.6). It is really difficult though to see any real differences between the methods though.

5.1.3 Diabetes

In the Pima Indians diabetes data the DPNN method performs relatively poorly. (See Figure 5.7). When removing the fifth predictor, it performs reasonably similar to the rest of the other methods, (see Figure 5.8), although this is not evident in the multidimensional scaling representation of the data due to outlying observations. However, if there is some overlap of the groups they are relatively differentiable.

Figure 5.4: Boxplots of error rates for dermatology data

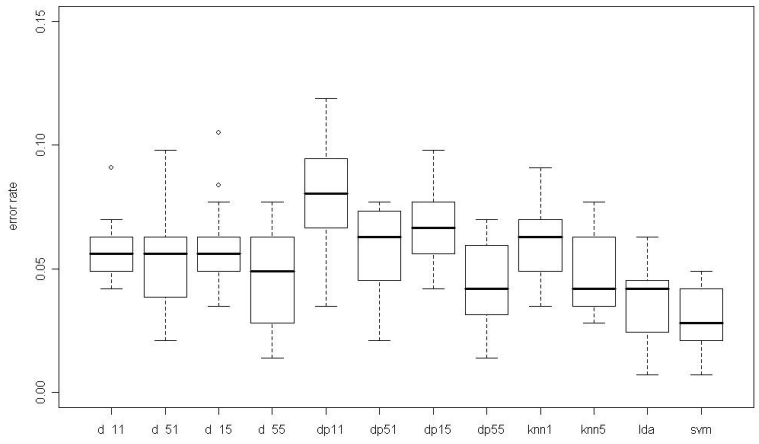


Figure 5.5: Scatterplot of multidimensional scaling representation of dermatology data in feature space with points in black corresponding to group 1, points in red corresponding to group 2, points in green corresponding to group 3, points in blue corresponding to group 4, points in cyan corresponding to group 5 and points in magenta corresponding to group 6

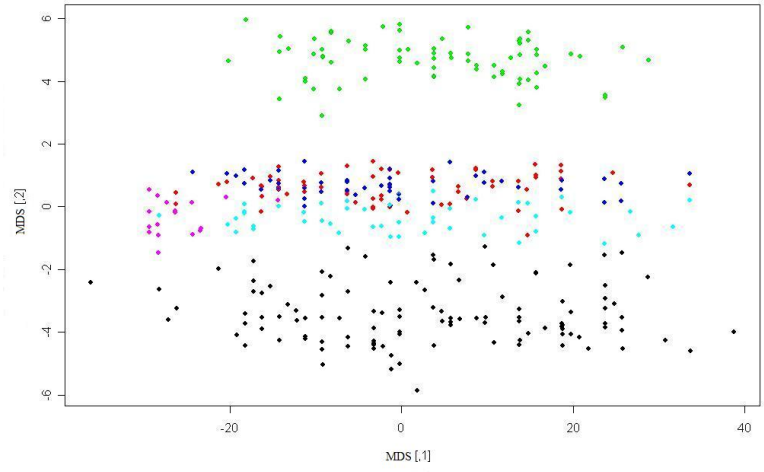
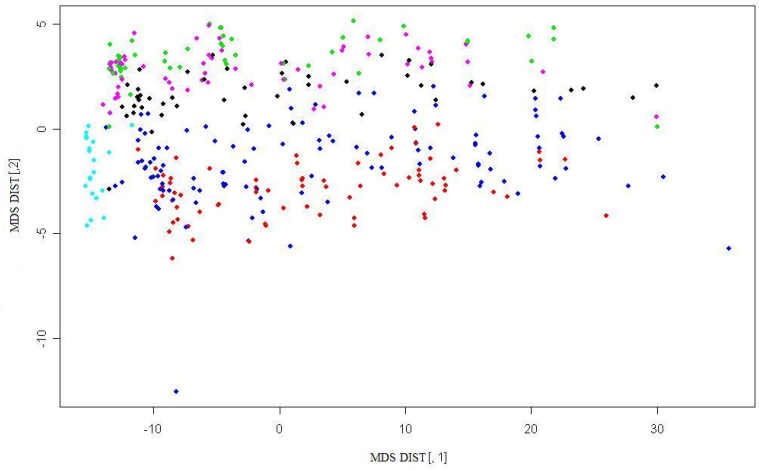


Figure 5.6: Scatterplot of multidimensional scaling representation of dermatology data in distance space with points in black corresponding to group 1, points in red corresponding to group 2, points in green corresponding to group 3, points in blue corresponding to group 4, points in cyan corresponding to group 5 and points in magenta corresponding to group 6



(See Figure 5.9). As observed from the graphical representation of the data in feature space, an oddity occurs where all of the data bunches up toward the end of the MDS[,1] axis. This is influenced by the fifth predictor in the data and can be remedied by its removal as seen in Figure 5.10. However, on the original data set, the distance profiles are sensitive to the overlap caused by the predictor and thus the two groups are essentially non distinct in the distance space, making the distance profile method perform poorly. There is an observation in distance space near (110, 250) which may in fact be a nearest neighbor distance profile causing the tailing observations to be pulled toward or away from a certain group. (See Figure 5.11). Even after removing the fifth predictor, there is not much improvement in distance space due to the presence of outliers. (See Figure 5.12).

Figure 5.7: Boxplots of error rates for diabetes data

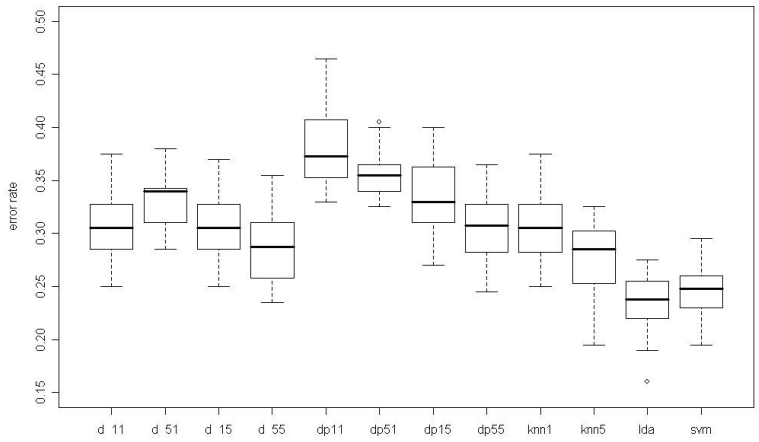


Figure 5.8: Boxplots of error rates for diabetes data with fifth predictor removed

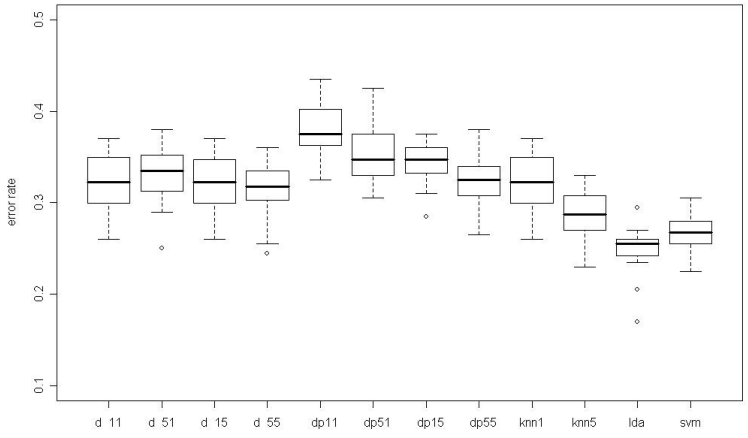


Figure 5.9: Scatterplot of multidimensional scaling representation of diabetes data in feature space with points in black corresponding to group 0 and points in red corresponding to group 1

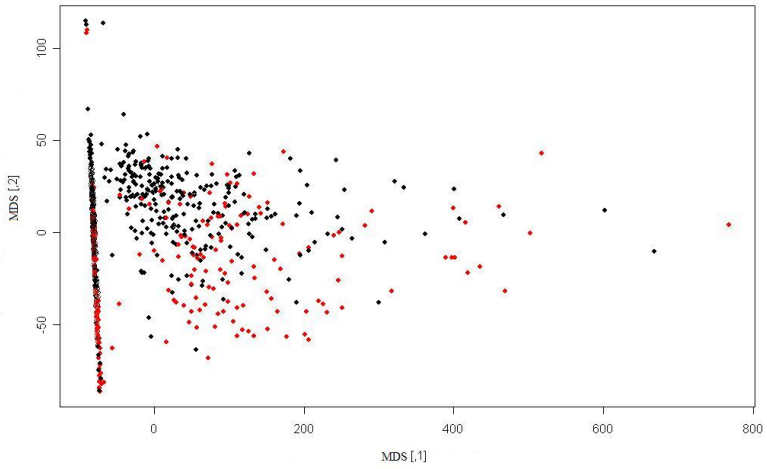


Figure 5.10: Scatterplot of multidimensional scaling representation of diabetes data with fifth predictor removed in feature space with points in black corresponding to group 0 and points in red corresponding to group 1

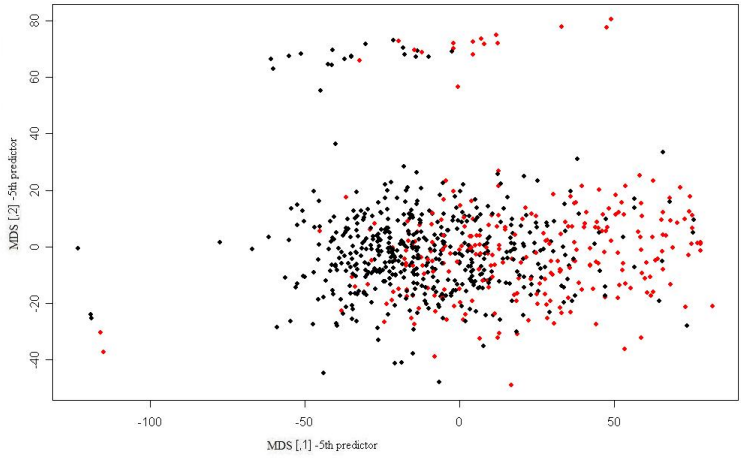


Figure 5.11: Scatterplot of diabetes data in distance space with points in black corresponding to group 0 and points in red corresponding to group 1

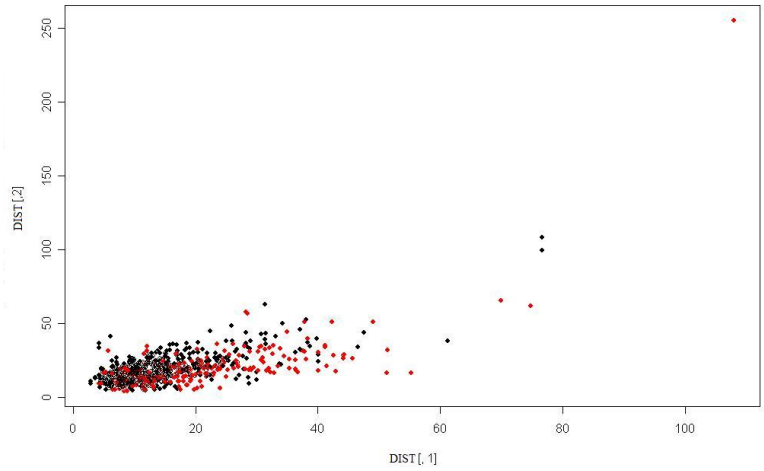
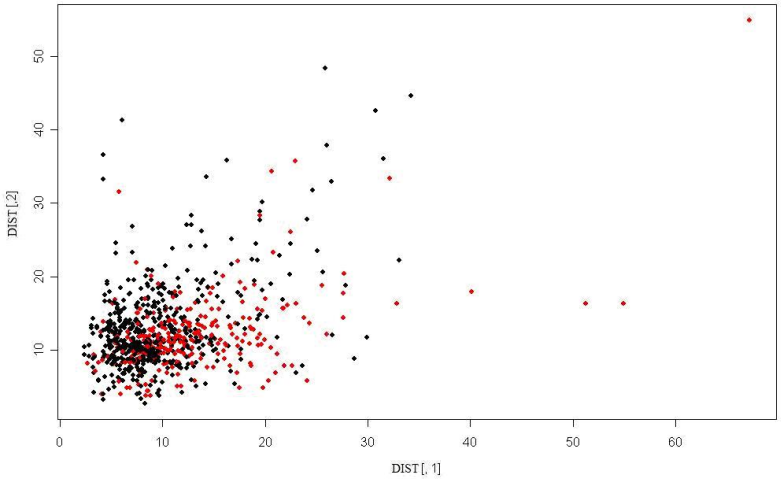


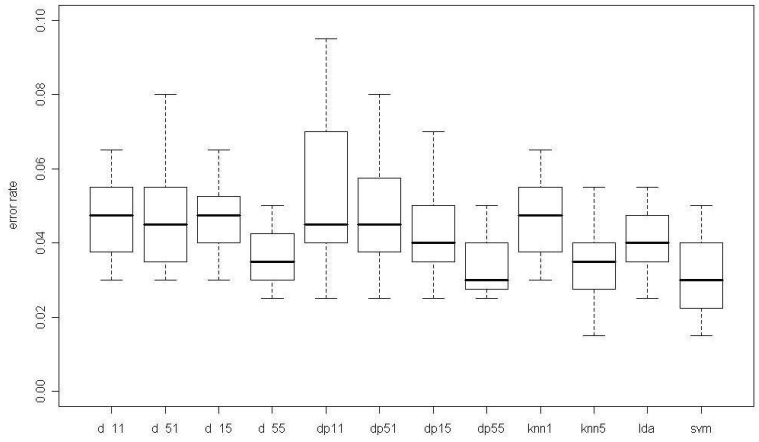
Figure 5.12: Scatterplot of diabetes data with fifth predictor removed in distance space with points in black corresponding to group 0 and points in red corresponding to group 1



5.1.4 Breast cancer

In the Wisconsin breast cancer data, all methods perform relatively the same. (See Figure 5.13. It is evident from the multidimensional scaling representation that the two classes are essentially separated, while the spread of the two classes differs. (See Figure 5.14). Representation of the data in distance space yields surprising results, in the structuring of one class into what appears to be levels. That may be caused by the fact that for the Benign group the mitosis predictor has integer values primarily assigned as category one. (See Figure 5.15).

Figure 5.13: Boxplots of error rates for breast cancer data



5.1.5 Hepatitis

In the hepatitis data, all methods are essentially equivalent except for LDA which performs poorly in comparison with the rest of the methods. For this dataset, the predictors are mostly binary and categorical. It is important to note that nearest neighbor methods, *knn*, perform in a similar manner to the distance profile methods,

Figure 5.14: Scatterplot of multidimensional scaling representation of breast cancer data in feature space with points in red corresponding to group 2 and points in blue corresponding to group 4

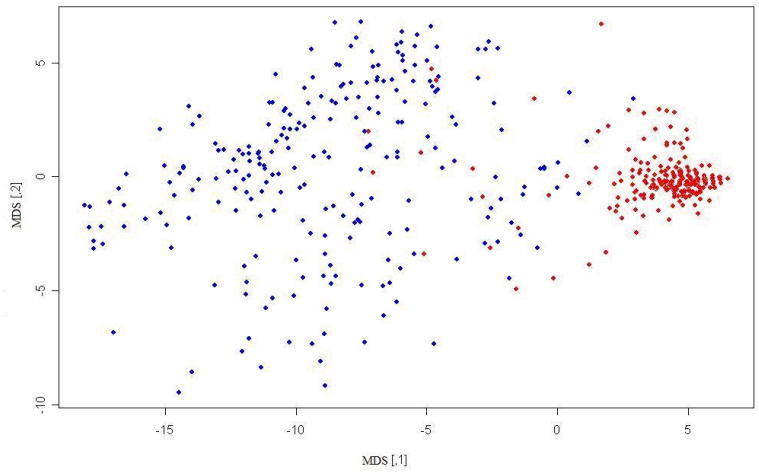
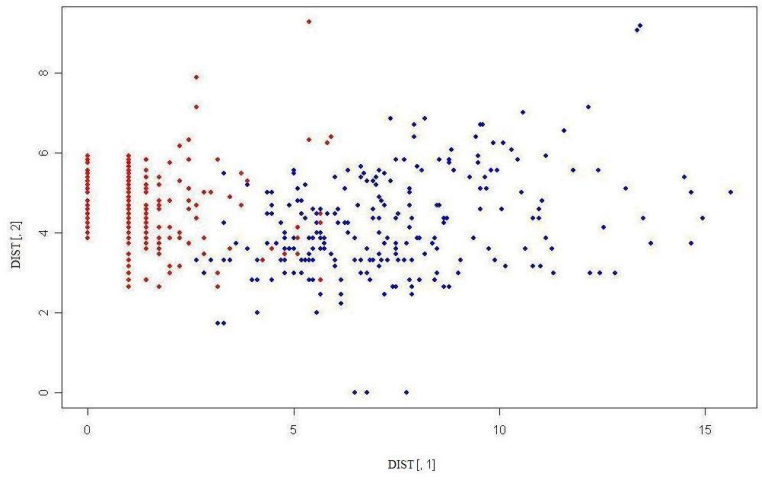
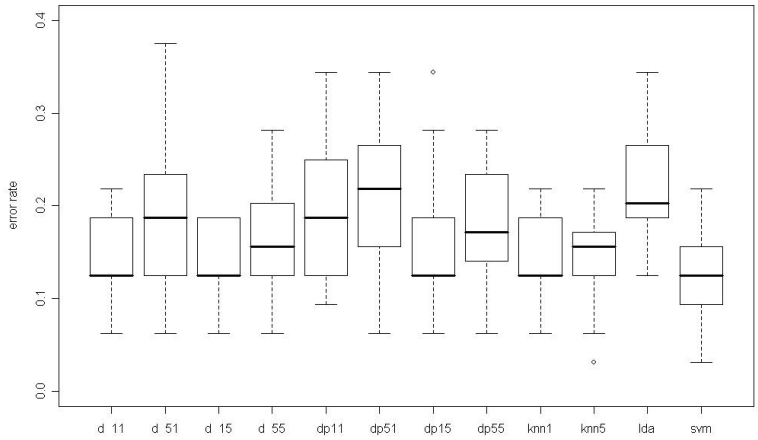


Figure 5.15: Scatterplot of breast cancer data in distance space with points in red corresponding to group 2 and points in blue corresponding to group 4



dp, in that larger nearest neighbors cause higher error rates. (See Figure 5.16). The distance profiles display that the distance to each group is almost the same regardless of which group the element actually belongs to. This is why the distance profile method offers no additional improvement over NN. (See Figure 5.18).

Figure 5.16: Boxplots of error rates for hepatitis data



5.1.6 Contraceptive method

For the contraceptive method choice data, it does not appear that using distance profiles offers much of an advantage over NN and in fact LDA is again a more effective method. While there are some categorical and binary elements to the predictors in this case, the continuous elements as well as the linear separability of the data may be the reason for gaining more advantageous results using LDA. (See Figure 5.19). From the multidimensional scaling representation of the data it is not clear that the three classes are linearly separable, however, due to the higher dimensions it is possible that they may just each be overlayed onto each other and actually exist

Figure 5.17: Scatterplot of multidimensional scaling representation of hepatitis data in feature space with points in black corresponding to group 1 and points in red corresponding to group 2

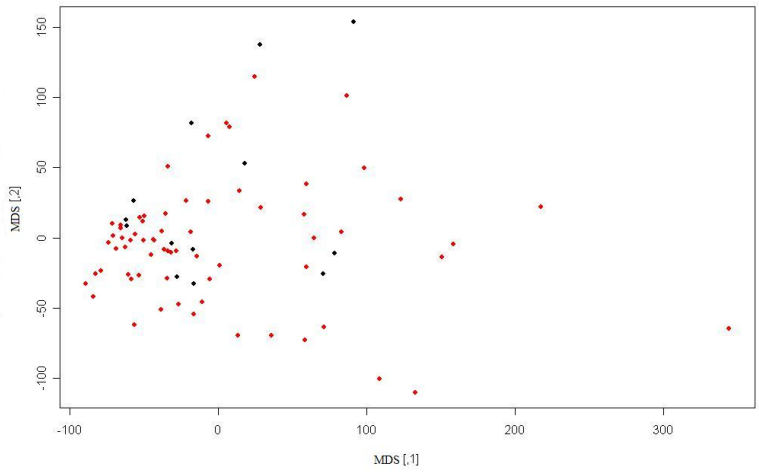
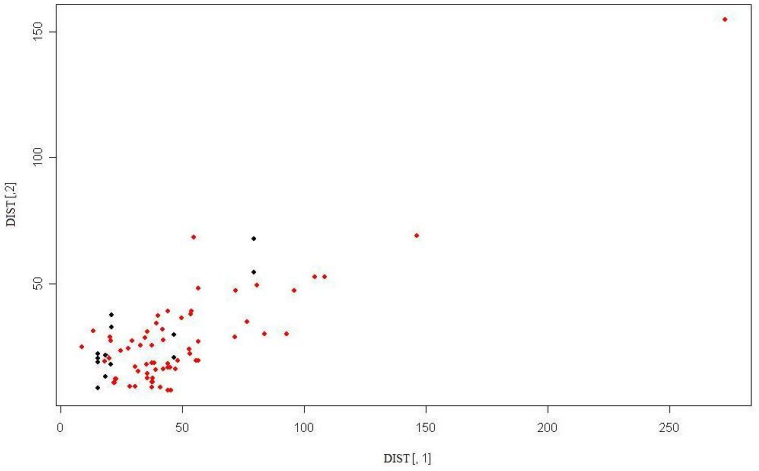
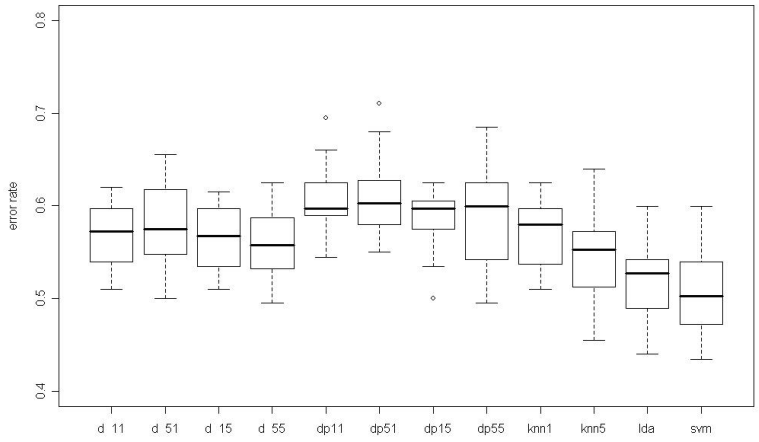


Figure 5.18: Scatterplot of hepatitis data in distance space with points in black corresponding to group 1 and points in red corresponding to group 2



on different planes. Also, this influence of the binary predictors may be causing difficulty. Thus LDA would work best in this situation. (See Figure 5.20). While the classes are more differentiable for this dataset, there is still a lot of overlap of the groups which is why distance profiles are still not as effective in this case. (See Figure 5.21).

Figure 5.19: Boxplots of error rates for contraceptive method data



5.1.7 Iris

For the iris data, LDA works the most effectively in classification. Here the three groups are generated from multivariate normal distributions and thus is a text book example of when to apply LDA. (See Figure 5.22). From the multidimensional scaling representation of the data, it is evident that the three groups are almost all linearly separable, giving more support to the use of LDA. (See Figure 5.23). The distance profiles do not offer any advantage to the rate of misclassification for data of this type. (See Figure 5.24).

Figure 5.20: Scatterplot of multidimensional scaling representation of contraceptive method data in feature space with points in black corresponding to group 1, points in red corresponding to group 2 and points in green corresponding to group 3

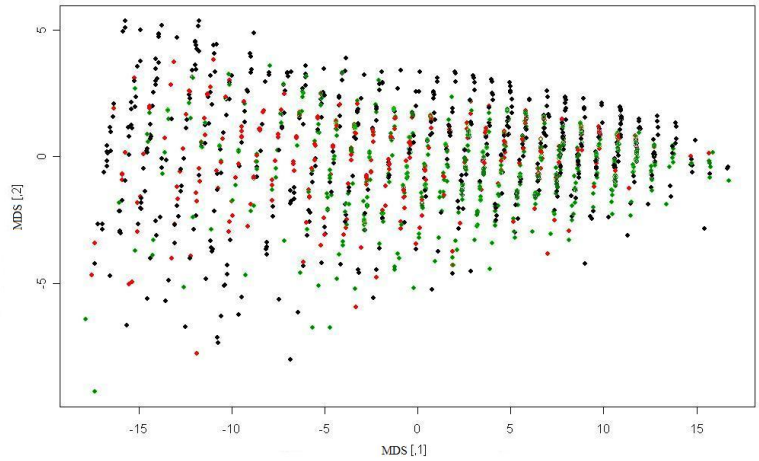


Figure 5.21: Scatterplot of multidimensional scaling representation of contraceptive method data in distance space with points in black corresponding to group 1, points in red corresponding to group 2 and points in green corresponding to group 3

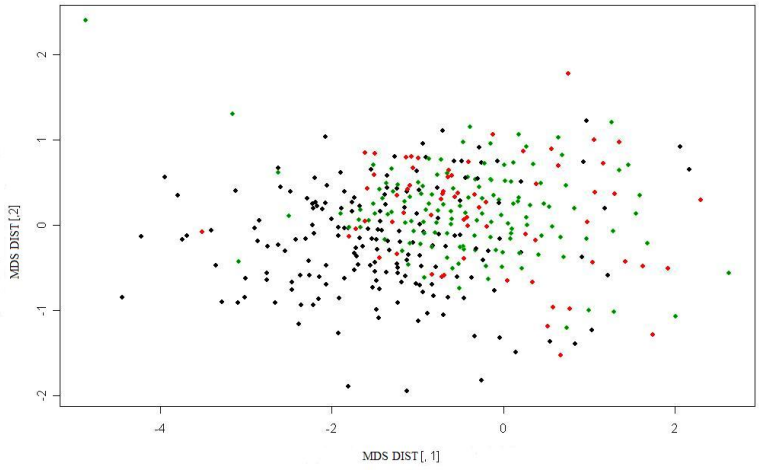


Figure 5.22: Boxplots of error rates for iris data

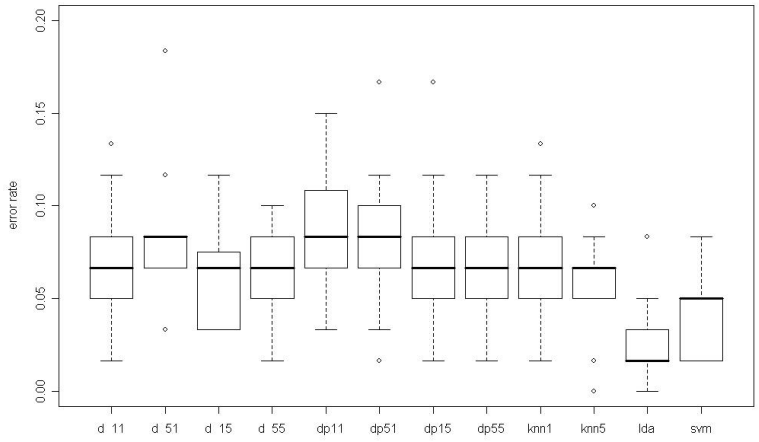


Figure 5.23: Scatterplot of multidimensional scaling representation of iris data in feature space with points in black corresponding to group 1, points in red corresponding to group 2 and points in green corresponding to group 3

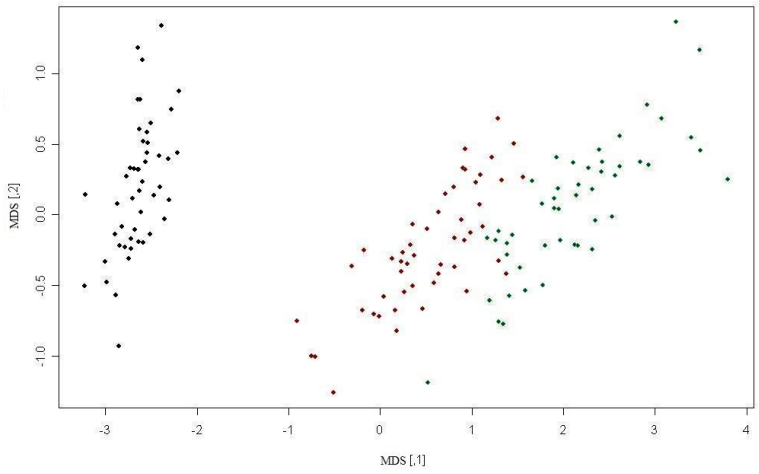
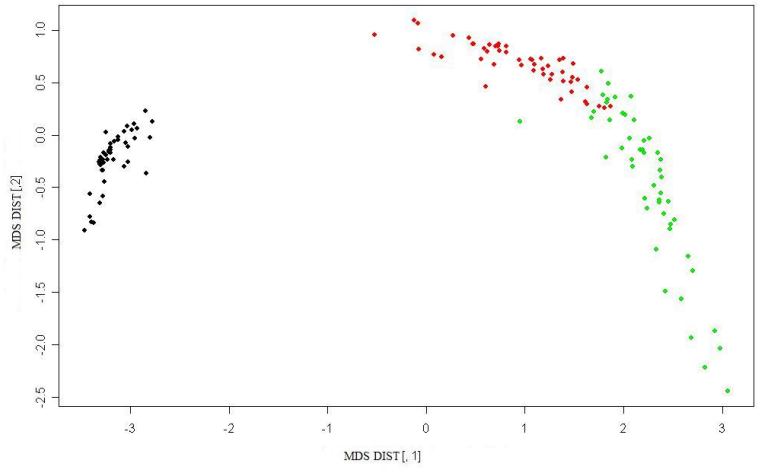


Figure 5.24: Scatterplot of multidimensional scaling representation of iris data in distance space with points in black corresponding to group 1, points in red corresponding to group 2 and points in green corresponding to group 3



5.1.8 Ionosphere

The ionosphere data clearly demonstrates an instance in which using distance profiles offers an advantage in that they lead to a reduced number of misclassification errors when compared with other methods. While there are only two classes with continuous predictors, it has another element that qualifies it as a good candidate for distance profile methods, it appears to be somewhat multimodal. It is important to note that with the anchored distance profiles using the first nearest neighbor to measure group-to-group distances does poorly, as in the case of NN. However, DPNN does very well using this measure, suggesting that classification for data with a dependence on point-to-group distance measures can be improved with this methodology (See Figure 5.25). The multimodality of the data can be observed in the multidimensional scaling of the data. There is a clear separation of one class into two subgroups. (See Figure 5.26). The multimodality is also supported in the

distance space and the other group is in fact equidistant from itself and the other group. (See Figure 5.27).

Figure 5.25: Boxplots of error rates for ionosphere data

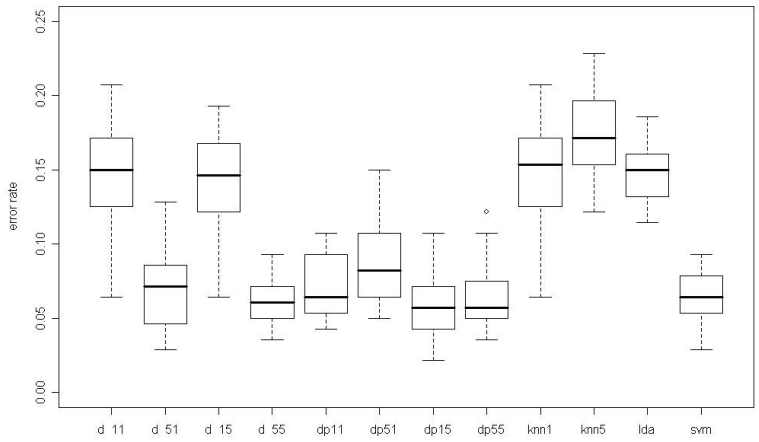


Figure 5.26: Scatterplot of multidimensional scaling representation of ionosphere data in feature space with points in black corresponding to group 1 and points in red corresponding to group 2

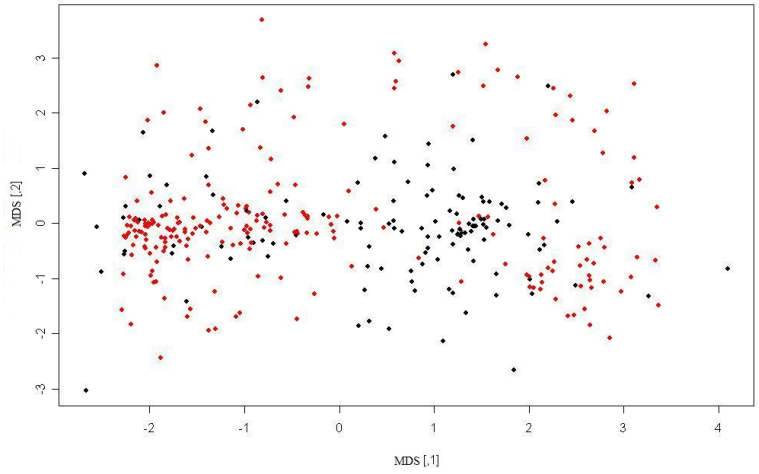
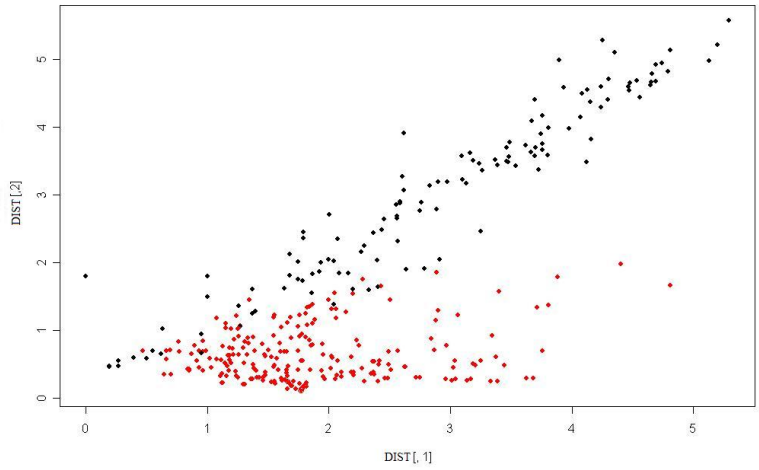


Figure 5.27: Scatterplot of ionosphere data in distance space with points in black corresponding to group 1 and points in red corresponding to group 2



5.1.9 Sonar

In the sonar data, it can be shown that error rates are approximately equivalent, regardless of how the training set and testing set are formed. Here NN performs best. (See Figures 5.28 and 5.29). It is important to note that in Figures 5.28 and 5.29 nearest neighbor methods, *knn*, perform in a similar manner to the distance profile methods, *dp*, in that larger nearest neighbors cause higher error rates. The two groups overlap greatly and are very difficult to differentiate amongst. (See Figure 5.30). While distance profiles offer an improvement in discerning the two classes, the spread is large and in many instances the opposite group is just as close to the object as its own group is for both groups. This is different than the other cases where group to group distances resulted in a tie because in those cases there was only one group that had elements equidistant from both classes where as this data has ties for both classes. (See Figure 5.31).

Figure 5.28: Boxplots of error rates for sonar data

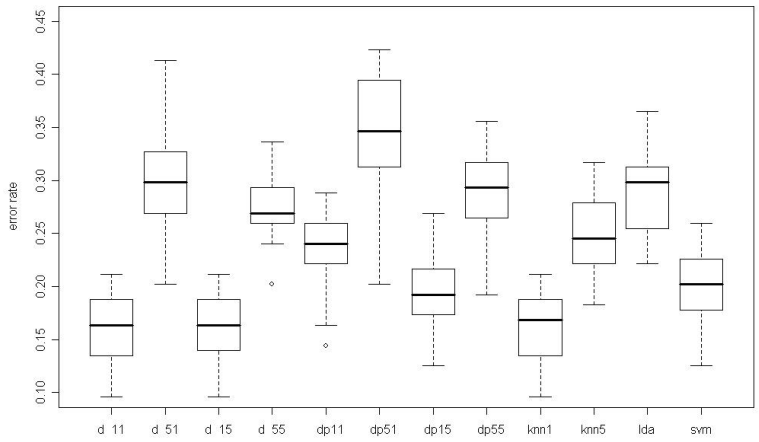


Figure 5.29: Boxplots of error rates for alternate version of sonar data

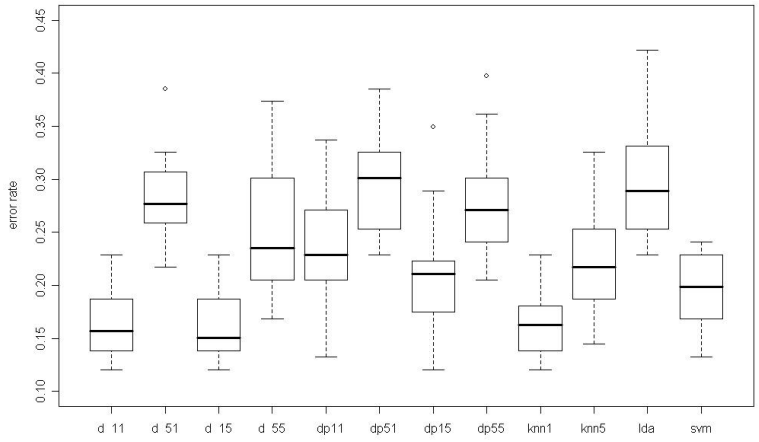


Figure 5.30: Scatterplot of multidimensional scaling representation of sonar data in feature space with points in black corresponding to group 1 and points in red corresponding to group 2

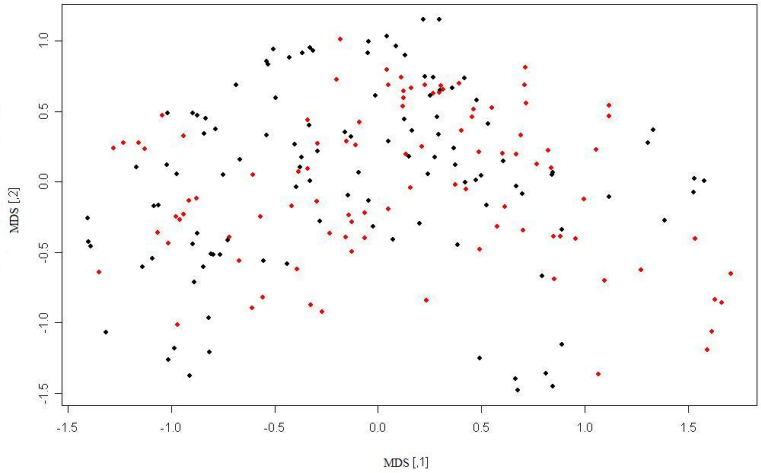
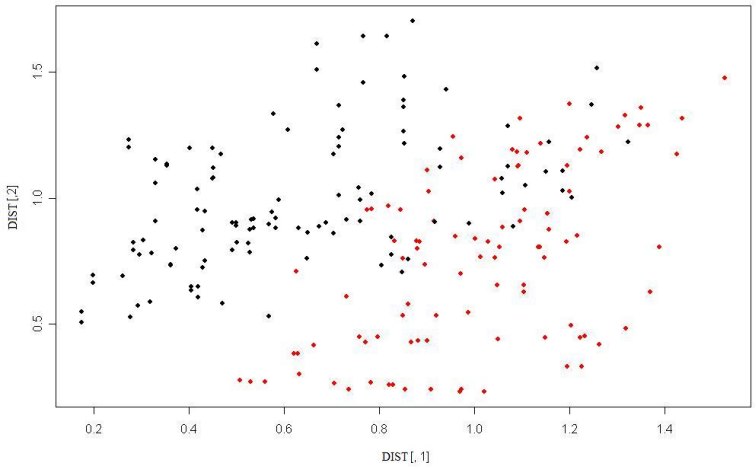


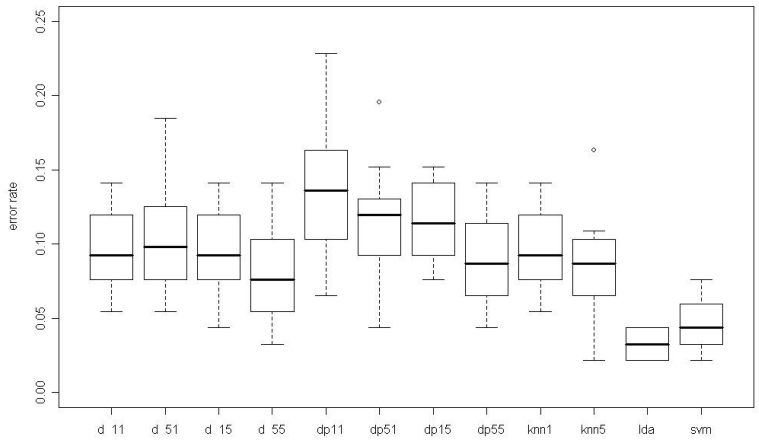
Figure 5.31: Scatterplot of sonar data in distance space with points in black corresponding to group 1 and points in red corresponding to group 2



5.1.10 Voting

The United States congressional voting records data again demonstrates an instance where LDA performs the best but has primarily binary predictors violating the normality assumption. (See Figure 5.32). Observing the multidimensional scaling representation of the data, there are two relatively distinct classes with a few points that cross over into the opposite region. (See Figure 5.33). As a result, the elements that cross over hinder the effectiveness of creating distance profiles that are adequate for using the distance profile method. This is caused by the binary traits in the predictors. (See Figure 5.34).

Figure 5.32: Boxplots of error rates for voting data



5.1.11 Vowel

For the vowel data, distance profiles offer no clear advantage over NN, however, LDA performs very poorly relative to the rest of the methods. It is important to note that dp_{51} does better than dp_{51} and $knn5$. (See Figure 5.35). In the multidimensional

Figure 5.33: Scatterplot of multidimensional scaling representation of voting data in feature space with points in black corresponding to group 1 and points in red corresponding to group 2

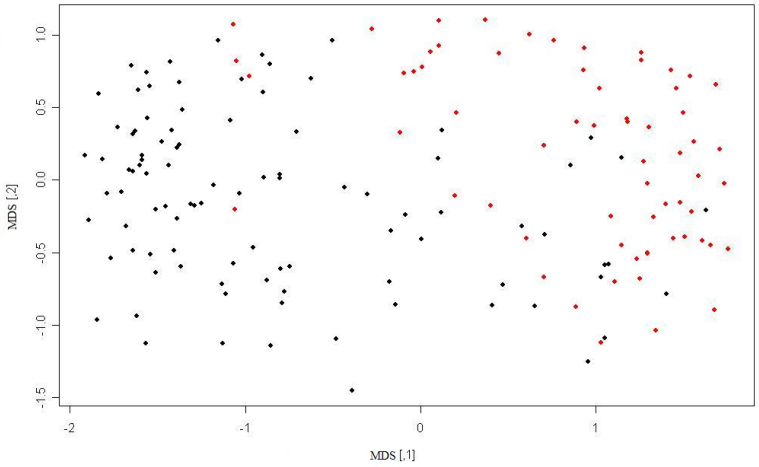
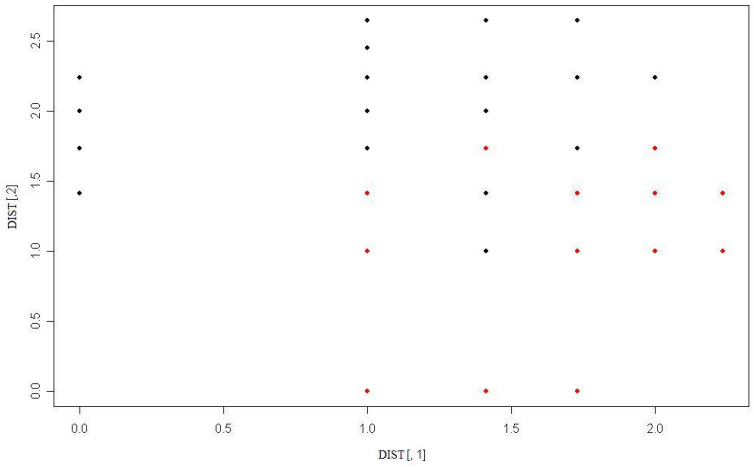
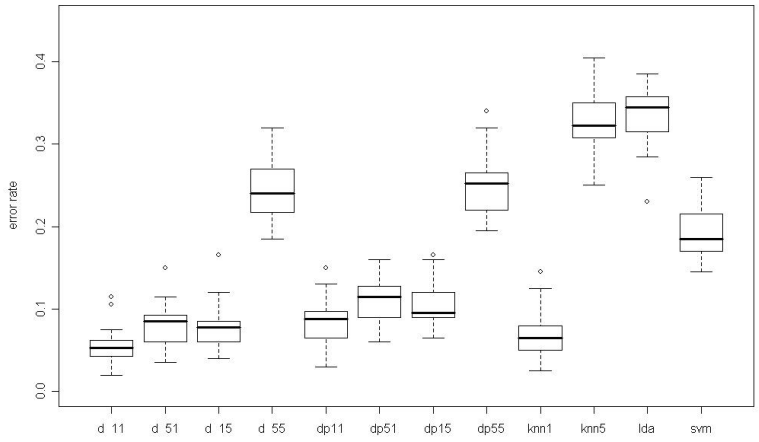


Figure 5.34: Scatterplot of voting data in distance space with points in black corresponding to group 1 and points in red corresponding to group 2



scaling representation of the data, it is evident that there is much overlap to the groups, hence the poor performance of LDA. (See Figure 5.36). In the distance profile representation of the data, the groups are more discernible, but offer no distinct separation of the classes providing limited improvement over the original representation. (See Figure 5.37).

Figure 5.35: Boxplots of error rates for vowel data



5.2 Simulations

For the simulations replicated from Hastie's paper [Hastie & Tibshirani, 1996], the following results were obtained using the different methods previously discussed. Error rates were computed based on the misclassifications for each method so comparisons can be drawn as to the accuracy of each method. For each dataset, training and testing samples were selected 20 times randomly then standardized, thus the boxplots for the error rates are based on the 20 errors obtained in each case. The first graph displays the error rates for the following methods as denoted by the follow-

Figure 5.36: Scatterplot of multidimensional scaling representation of vowel data in feature space with points in red corresponding to group 1, points in orange corresponding to group 2, points in yellow corresponding to group 3, points in green corresponding to group 4, points in blue corresponding to group 5, points in purple corresponding to group 6, points in brown corresponding to group 7, points in black corresponding to group 8, points in gray corresponding to group 9, points in magenta corresponding to group 10 and points in cyan corresponding to group 11

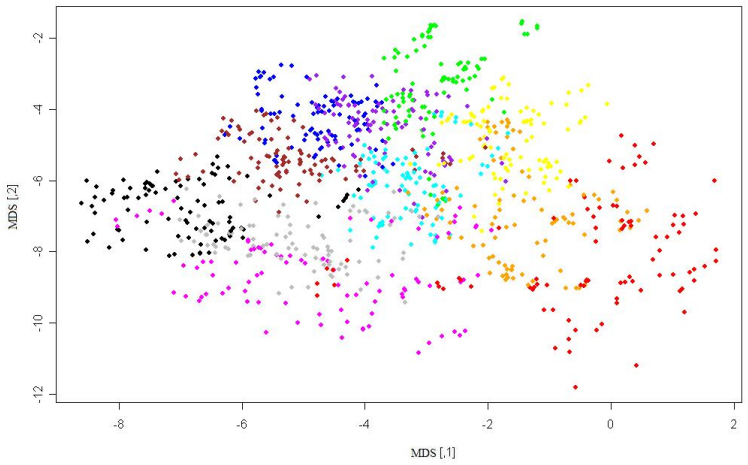
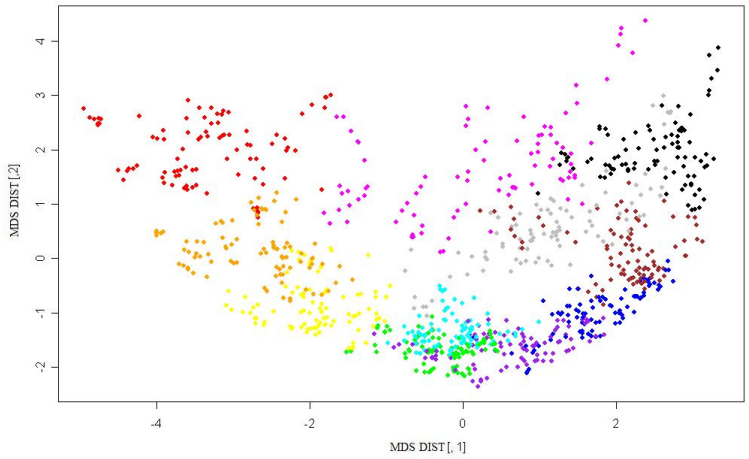


Figure 5.37: Scatterplot of multidimensional scaling representation of vowel data in distance space



ing: *dp* denotes the method using distance profiles in which the class is unknown for each element in the training set and the distance to that class is based on a nearest neighbor in that class for the training set, then used to compare the distance profiles for the testing set. This method is classification via DPNN. Thus, $dp_{k_1 k_2}$ denotes measuring distance from each object to group using the k_1 nearest neighbor from the other groups and then using the k_2 nearest distance profile. *knn* denotes the method of NN. Thus, $knn1$ denotes classification based on the first nearest neighbor and $knn5$ denotes classification based on the fifth nearest neighbor. *lda* denotes the method of linear discriminant analysis. *svm* denotes the method of support vector machines. Then, a representation of the data is displayed both as a multidimensional scaling of the data in its the feature space based on its p-characteristics and as it appears in distance space based on its distance profiles.

5.2.1 Two bivariate normal classes

For Simulation 4.1, there are two bivariate normal populations. As a result, LDA turns out to be the most effective method for classification. In comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. (See Figure 5.38). As seen in the Figure a linear separation between the two classes is entirely realistic and thus LDA performs well in this instance. (See Figure 5.39). The distance profile representation of the data then demonstrates the difficulty of handling the overlapping region for this case. (See Figure 5.40).

5.2.2 Two bivariate normal classes with noise

It can be noted that as in Simulation 4.1, for Simulation 4.2 having two bivariate normal populations although with the presence of noise will still result in LDA performing best. In comparison with Hastie's result in [Hastie & Tibshirani, 1996],

Figure 5.38: Boxplots of error rates for two bivariate normal classes

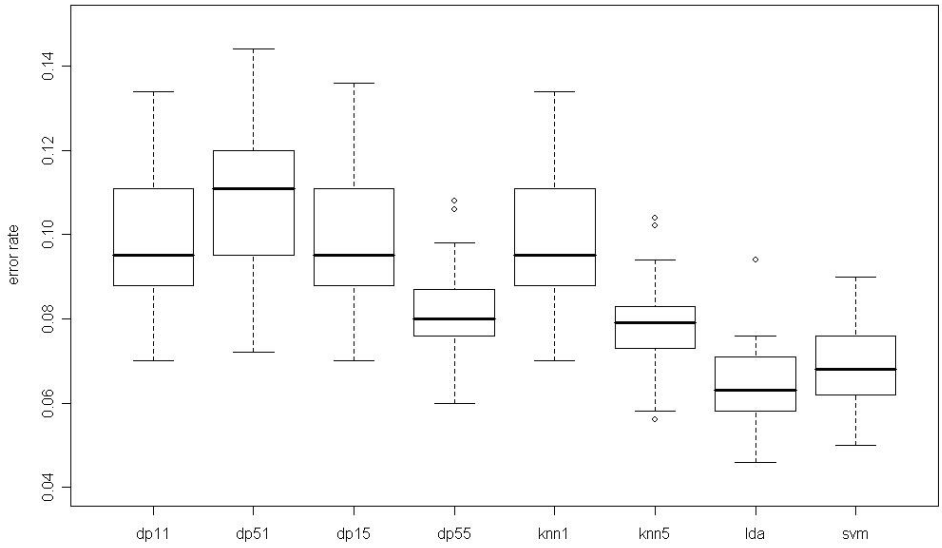


Figure 5.39: Scatterplot of two bivariate normal classes data in feature space

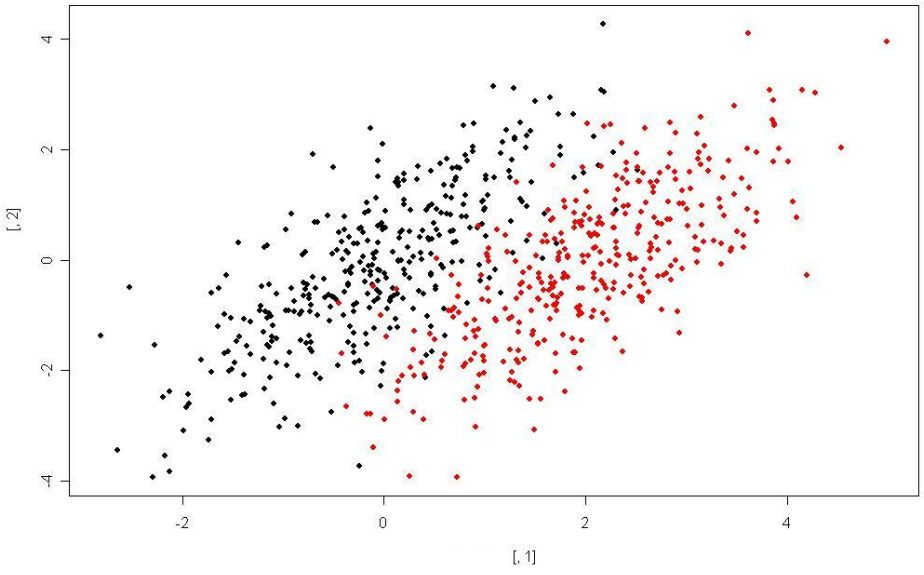
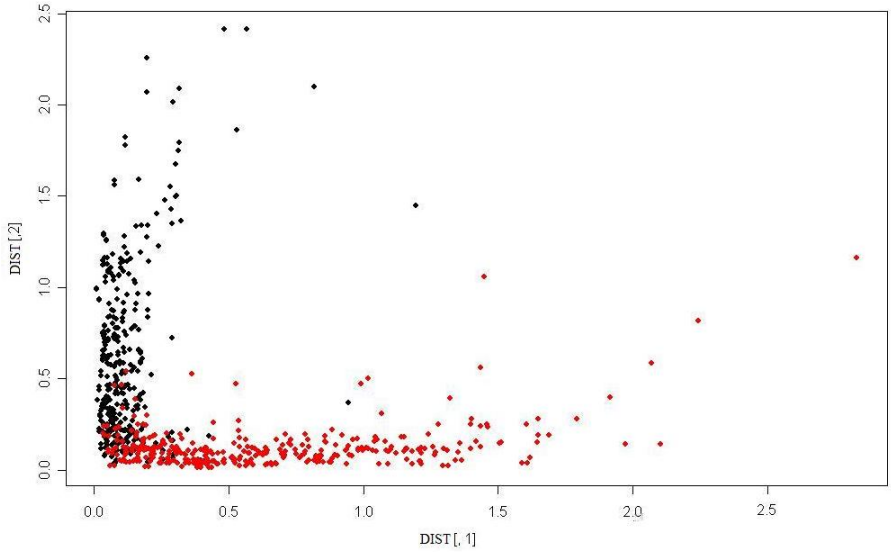


Figure 5.40: Scatterplot of two bivariate normal classes data in distance space



performance was generally worse in our case. (See Figure 5.41). Again although there appears to be overlap, it is possible to separate the two classes linearly. (See Figure 5.42). The introduction of noise actually makes the effectiveness of DPNN decrease because many observations cross over into the other region, preventing the observations from actually being more similar to its own class than the other. (See Figure 5.43).

5.2.3 Four multimodal classes

There has been evidence that using the DPNN method should vastly improve over other technique when the data appears to be generated from multimodal populations as suggested in the Ionosphere data discussed previously. In an attempt to analyze the impact of this effect on the efficiency of the method, the following simulations were performed. At first it was believed that looking at the boxplots, the distance profile method did not show vast improvements over nearest neighbor in

Figure 5.41: Boxplots of error rates for two bivariate normal classes with noise

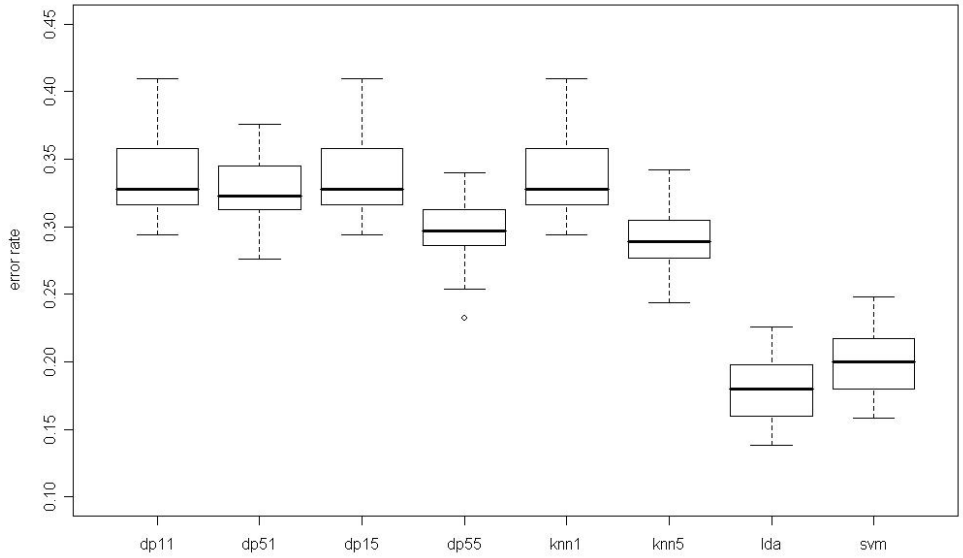


Figure 5.42: Scatterplot of two bivariate normal classes with noise data in feature space

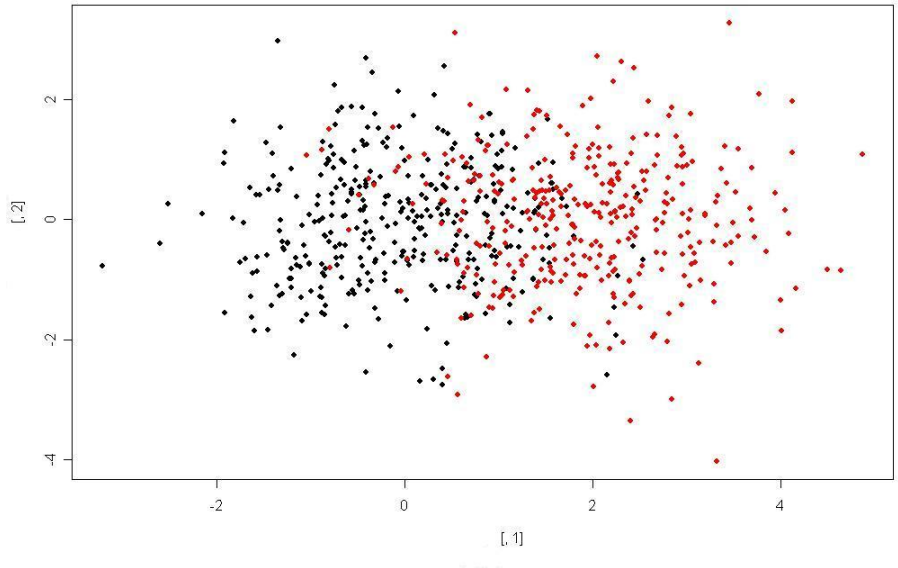
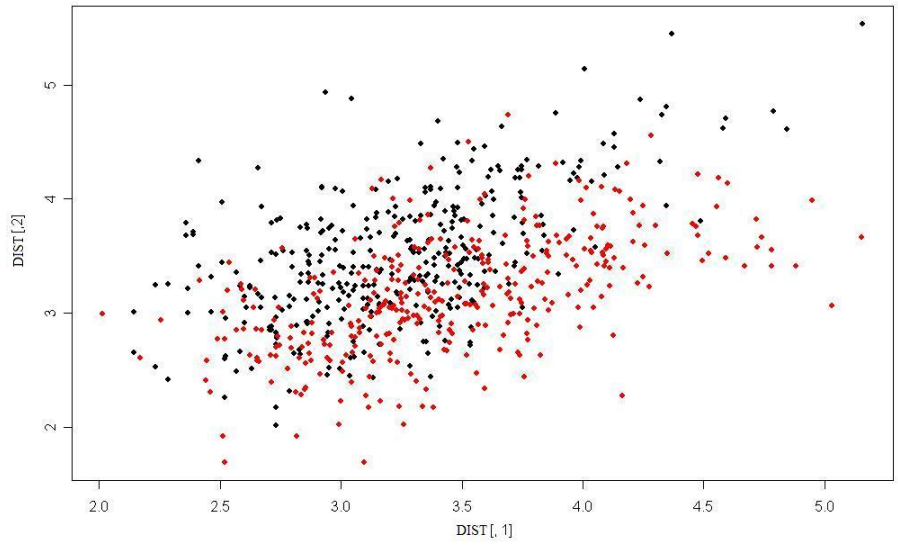


Figure 5.43: Scatterplot of two bivariate normal classes with noise data in distance space



this situation because the training sample size was small compared to the test sample size, thus the rule for classification was not as good as it could be. (See Figure 5.45). However, examination of the error rates after adjusting for this fact gave the same impression. (See Figure 5.47). However, in comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. By looking at the graphical representation of the data though it is clear why nearest neighbor performs just as well. All of the groups seem to have equal spread and are generally far from any other group. (See Figure 5.49). By inspecting the distance space, it appears that for the most part, each of the groups is equidistant from itself and any of the other groups. (See Figure 5.50). In an effort to understand this phenomenon, the distance space was broken down in order to observe each of the group combinations separately. (See Figures 5.51,5.52,5.53,5.54,5.55 and 5.56). Some unusual criss-crossing patterns appear that suggest further investigation.

Figure 5.44: Boxplots of error rates for four multimodal class data - small training samples

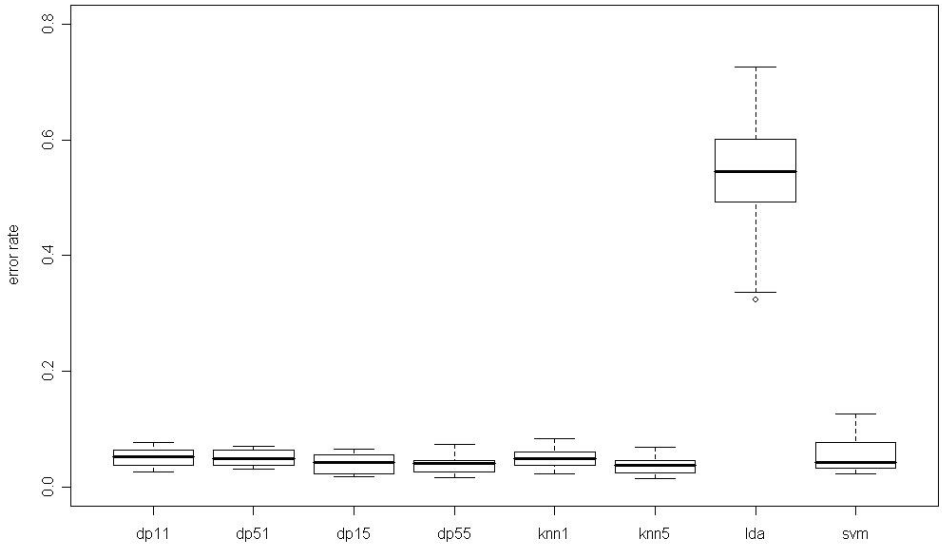


Figure 5.45: Boxplots of error rates for four multimodal class data excluding LDA - small training samples

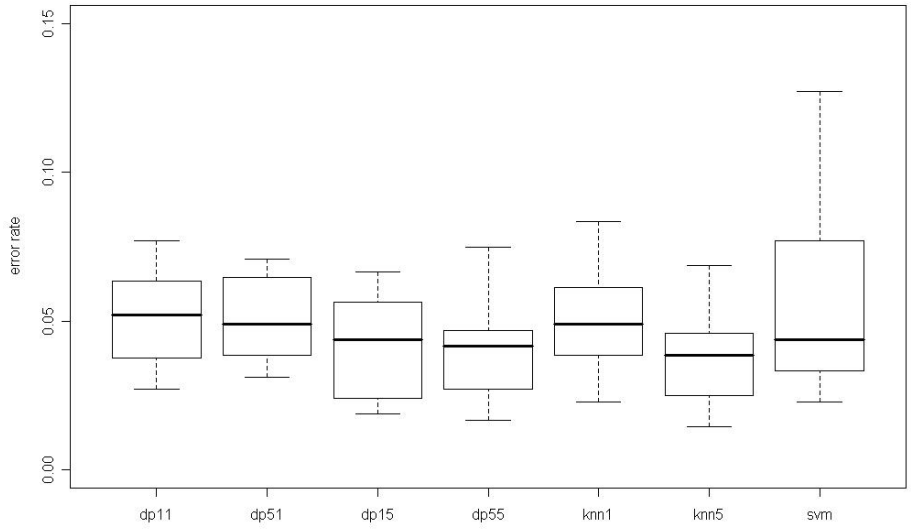


Figure 5.46: Boxplots of error rates for four multimodal class data - large training samples

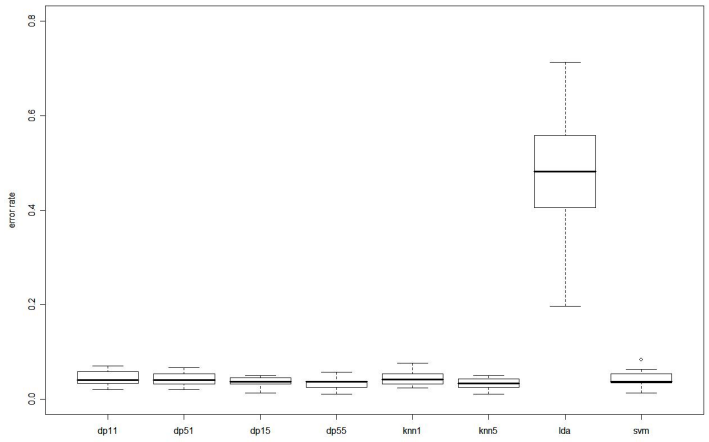


Figure 5.47: Boxplots of error rates for four multimodal class data excluding LDA - large training samples

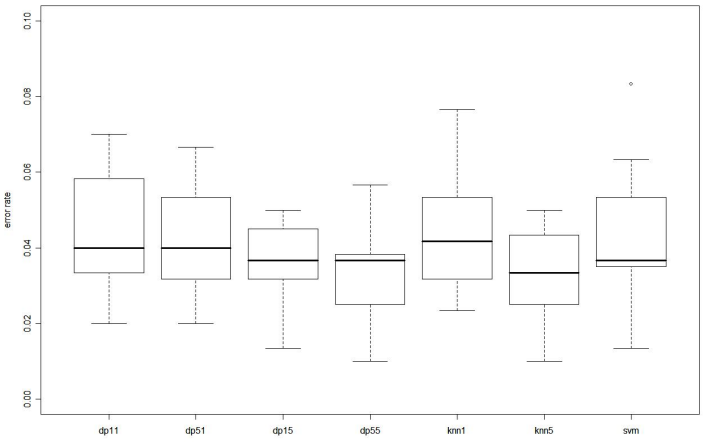


Figure 5.48: Scatterplot of four multimodal class data in feature space

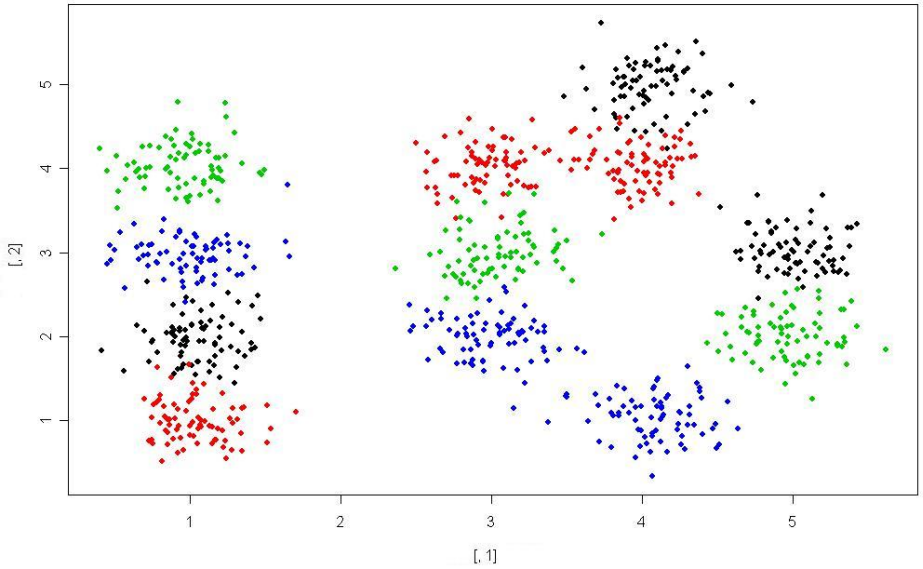


Figure 5.49: Scatterplot of multidimensional scaling representation of four multimodal class data in feature space

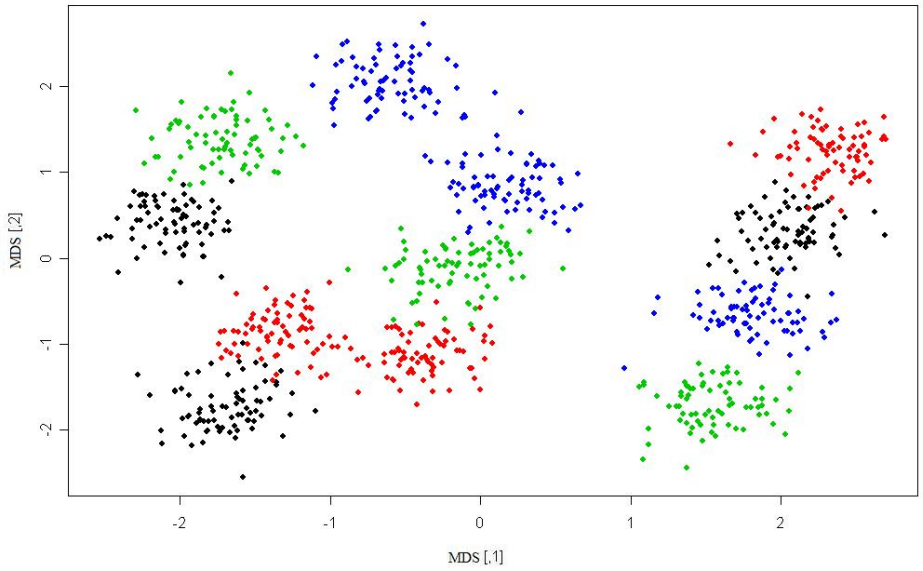


Figure 5.50: Scatterplot of multidimensional scaling representation of four multimodal class data in distance space

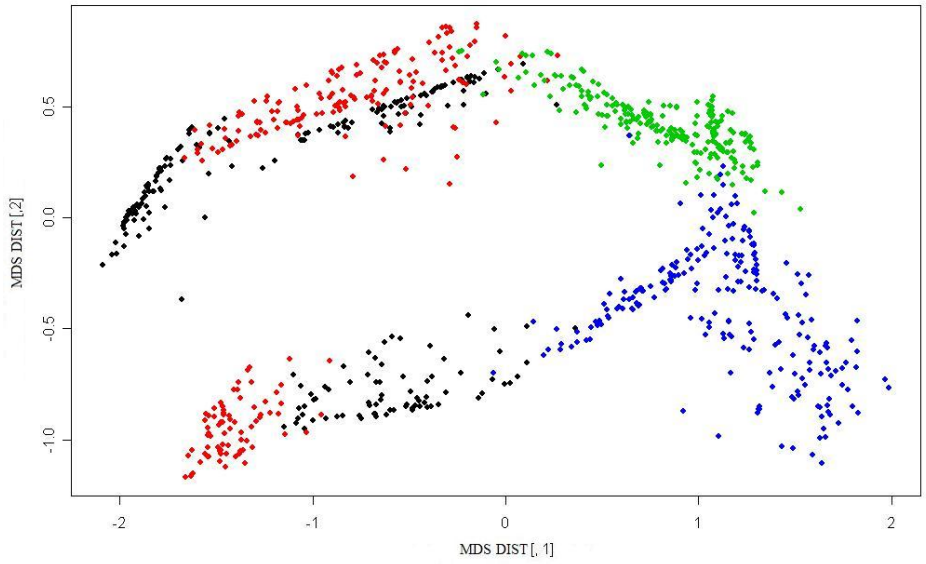


Figure 5.51: Scatterplot of groups one and two of the four multimodal classe data in distance space

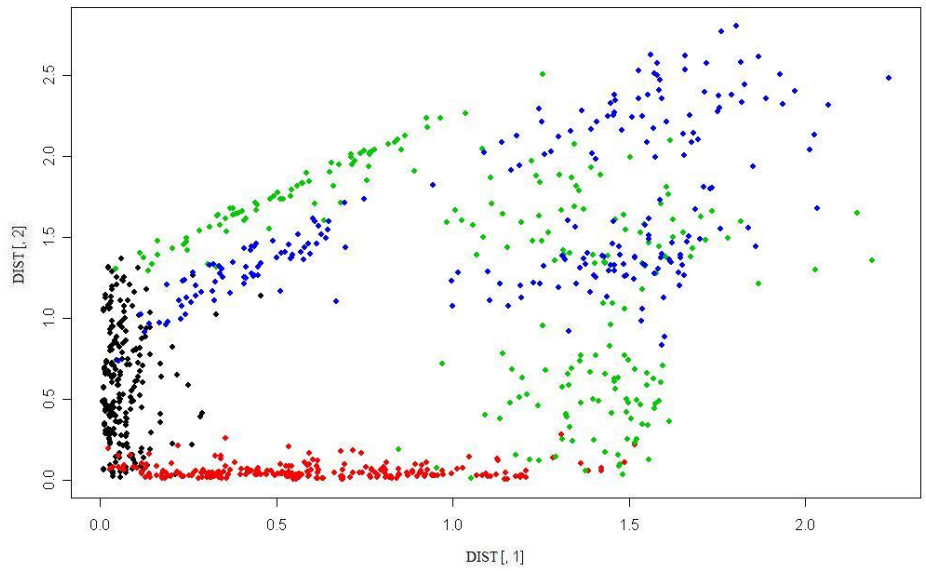


Figure 5.52: Scatterplot of groups one and three of the four multimodal class data in distance space

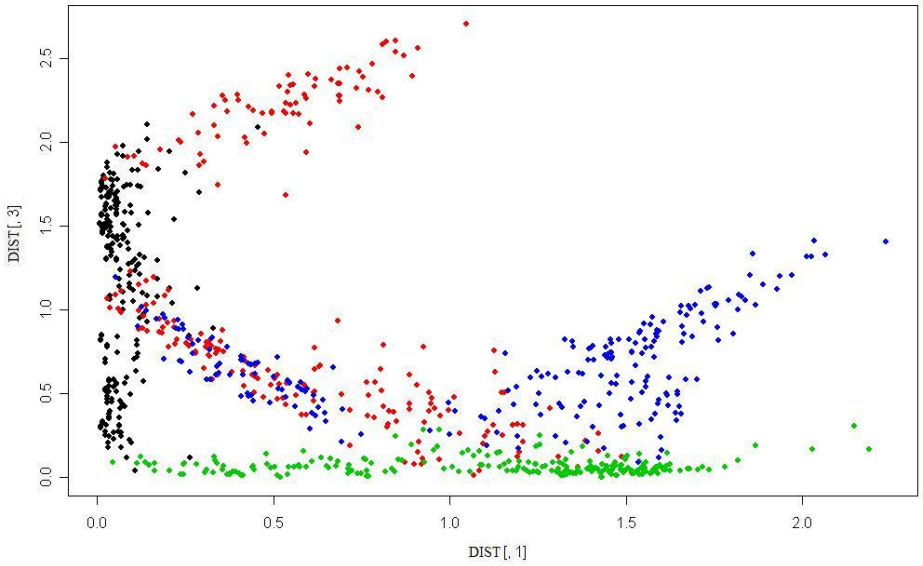


Figure 5.53: Scatterplot of groups one and four of the four multimodal class data in distance space

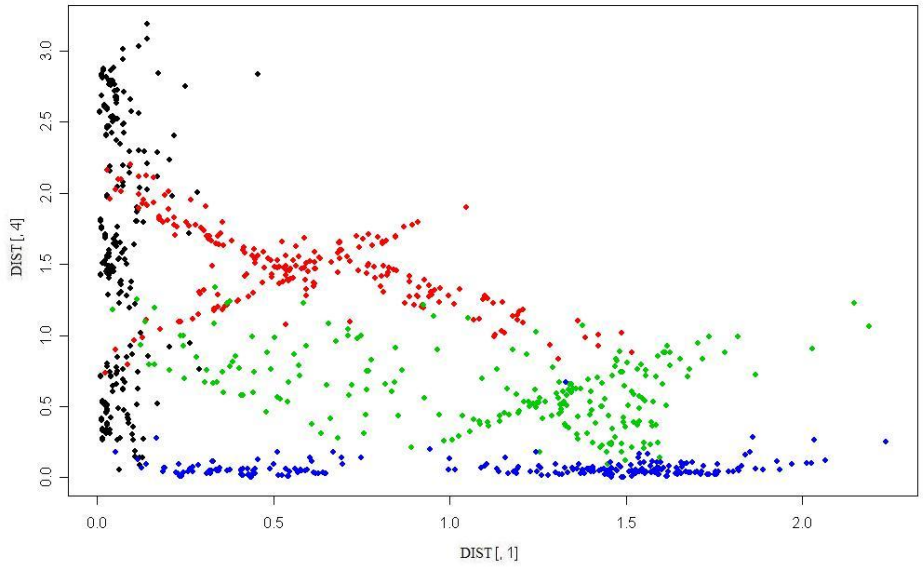


Figure 5.54: Scatterplot of groups two and three of the four multimodal class data in distance space

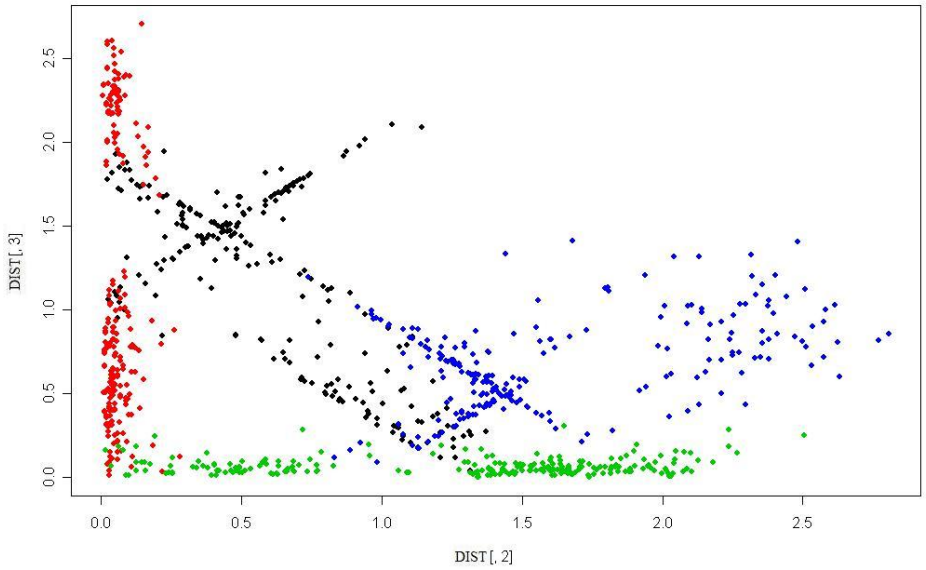


Figure 5.55: Scatterplot of groups two and four of the four multimodal class data in distance space

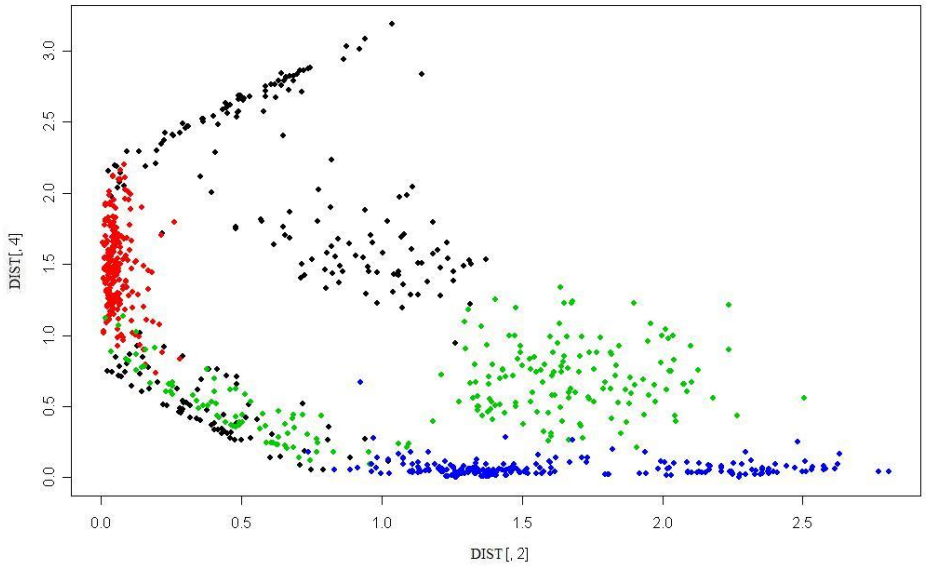
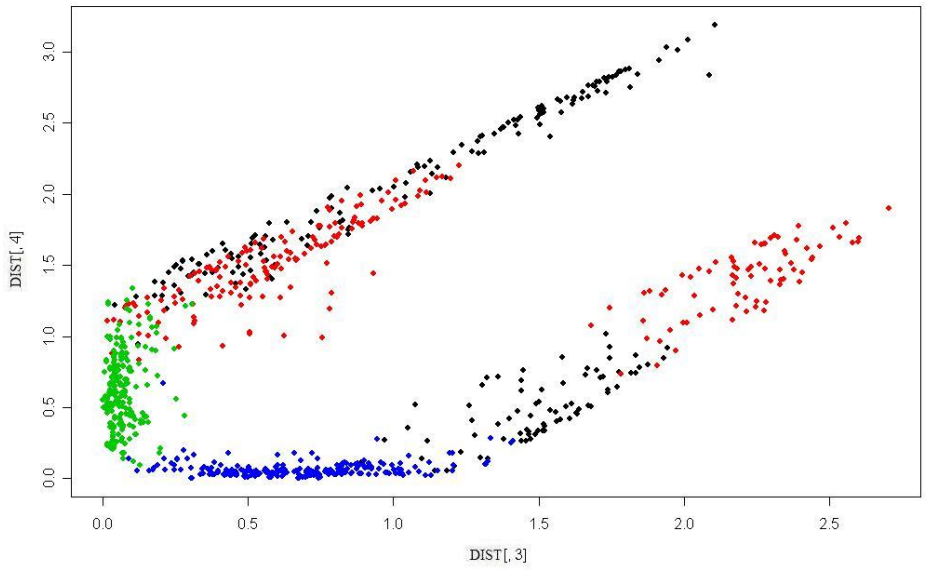


Figure 5.56: Scatterplot of groups three and four of the four multimodal class data in distance space

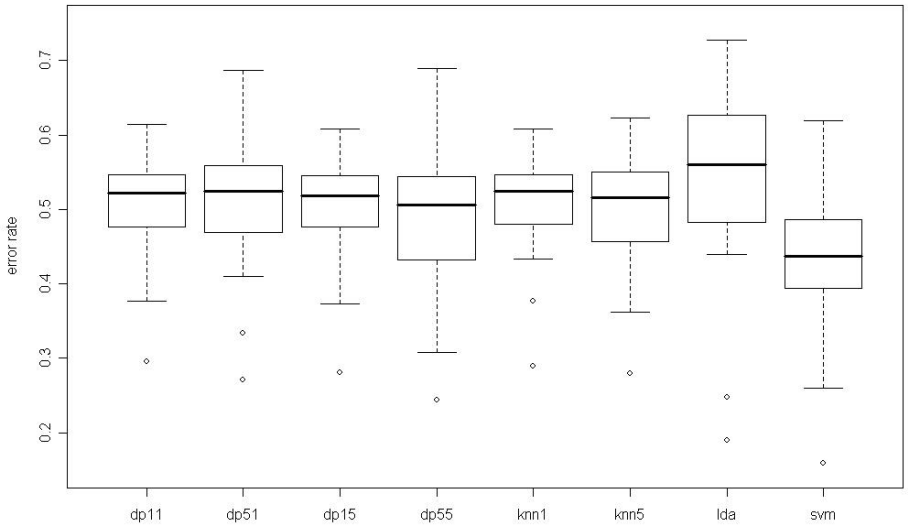


5.2.4 Four multimodal classes with noise

As DPNN did not show any improvement over nearest neighbor in the last case, noise was introduced to see if by changing the spreads, distance profiles would be more beneficial. In the boxplots of the error rates, it shows that for the smaller training sample size no improvement is made, but that for the larger training sample size a slight improvement is gained as was the case with the previous simulation. In comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. (See Figures 5.57 and 5.58). Examining the graphical representation of the data to see exactly how the noise influences the spread, it seems that there is very little overlap of groups and very few points are close to boundaries between any of the groups. (See Figure 5.59). Thus these do not show very good instances where distance profiles is helpful and motivated the investigation into different distributional circumstances. All of the groups seem to have equal spread and are generally far from any other group. (See Figure 5.60). By inspecting the

distance space, it appears that for the most part, each of the groups is equidistant from itself and any of the other groups. (See Figure 5.61). In an effort to understand this phenomenon, the distance space was broken down in order to observe each of the group combinations separately. (See Figures 5.62,5.63,5.64,5.65,5.66 and 5.67). All that was obtained from this was that there appears to be a lot of overlap in distance space, where none of the observation in a group are at all close to their own group hence the reason for poor performance in classification.

Figure 5.57: Boxplots of error rates for four multimodal class data with noise - small training samples



5.2.5 Four dimensional Gaussian sphere with noise

In Simulation 4.5, it is evident that this is an instance in which DPNN performs slightly better than NN while both outperform LDA. However, in this case SVM does perform the best. In comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. (See Figure 5.68). In the multidimensional scaling representation of the data, it is understand as to why LDA performs

Figure 5.58: Boxplots of error rates for four multimodal class data with noise - large training samples

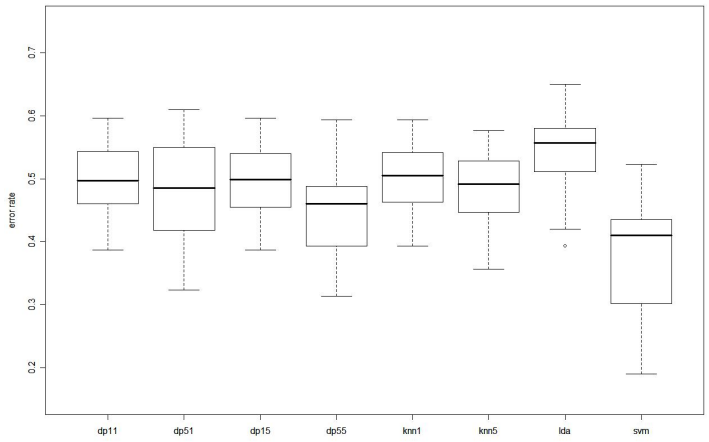


Figure 5.59: Scatterplot of four multimodal class data with noise in feature space

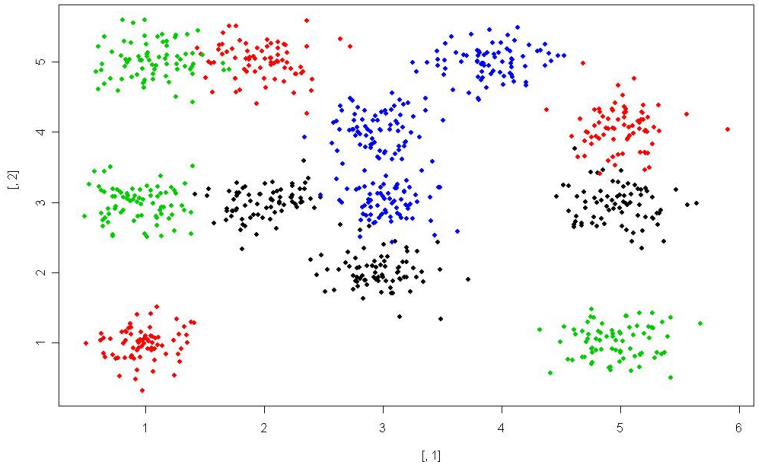


Figure 5.60: Scatterplot of multidimensional scaling representation of four multimodal class data with noise in feature space

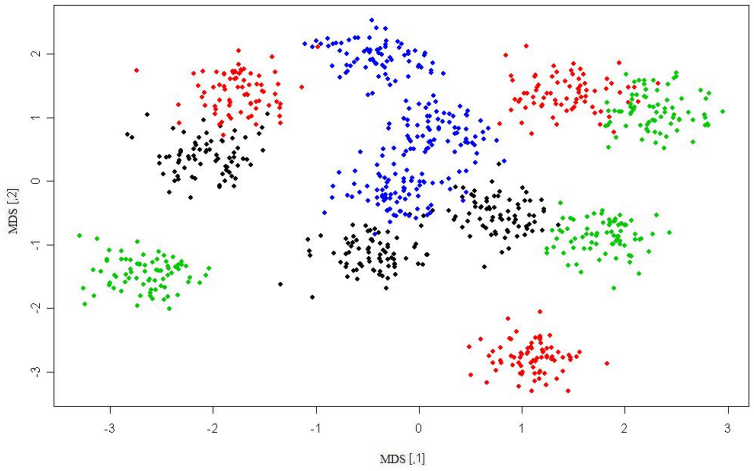


Figure 5.61: Scatterplot of multidimensional scaling representation of four multimodal class data with noise in distance space

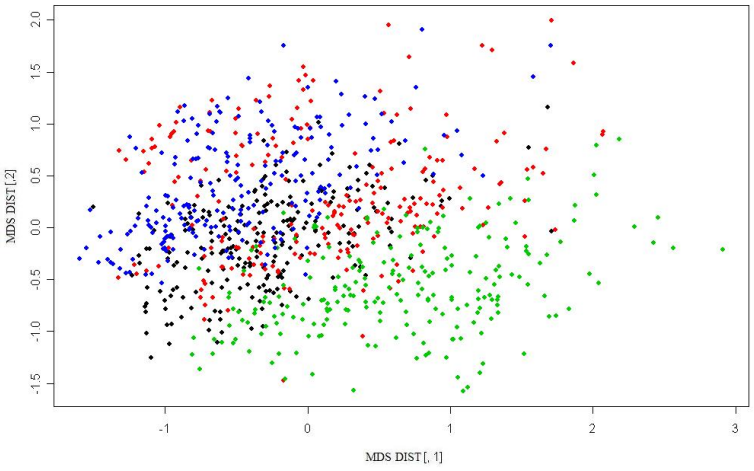


Figure 5.62: Scatterplot of groups one and two of four multimodal class data with noise in distance space

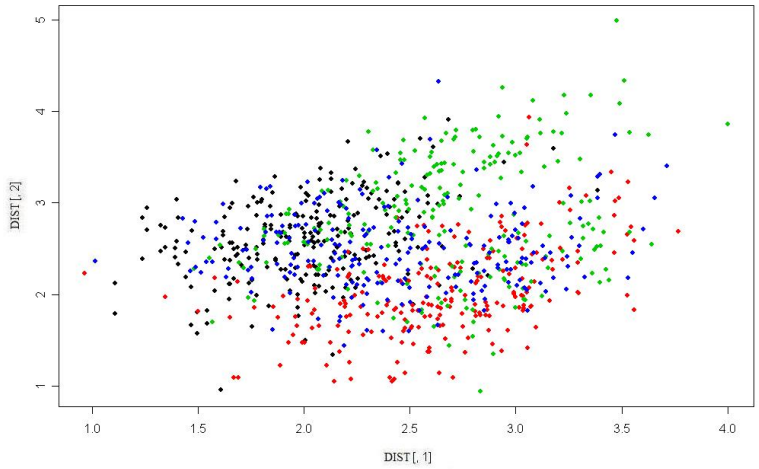


Figure 5.63: Scatterplot of groups one and three of four multimodal class data with noise in distance space

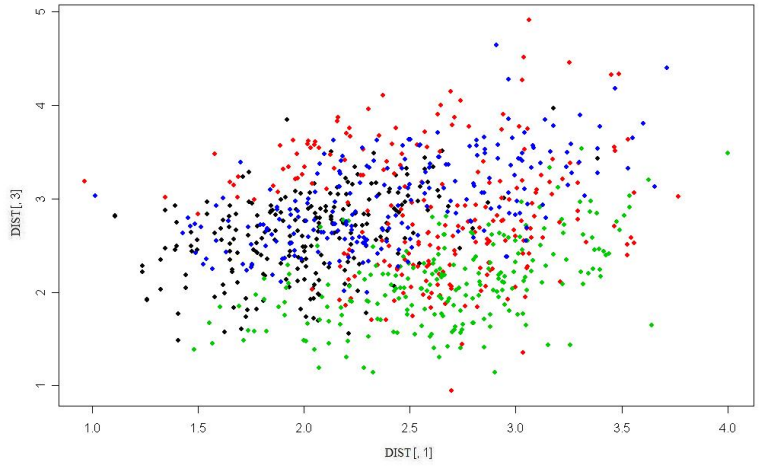


Figure 5.64: Scatterplot of groups one and four of four multimodal class data with noise in distance space

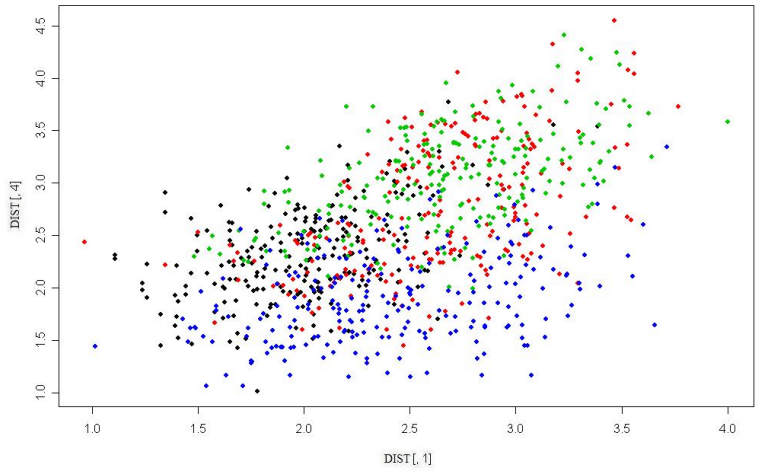


Figure 5.65: Scatterplot of groups two and three of four multimodal class data with noise in distance space

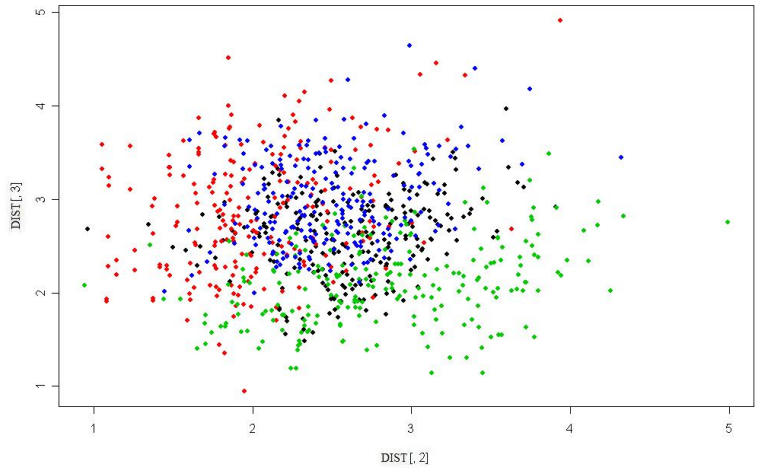


Figure 5.66: Scatterplot of groups two and four of four multimodal class data with noise in distance space

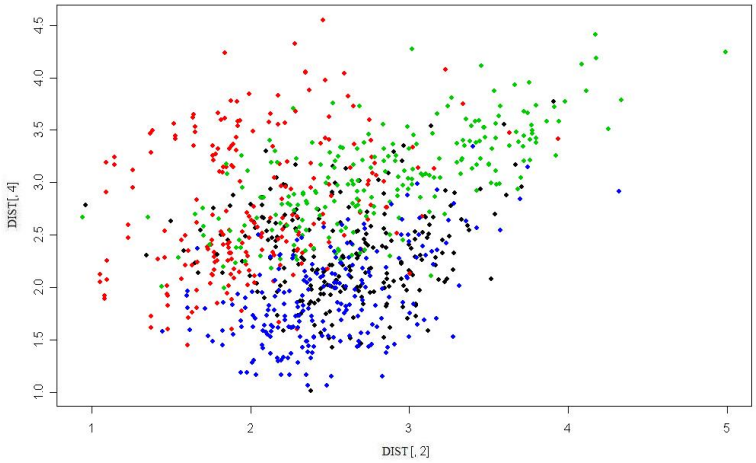
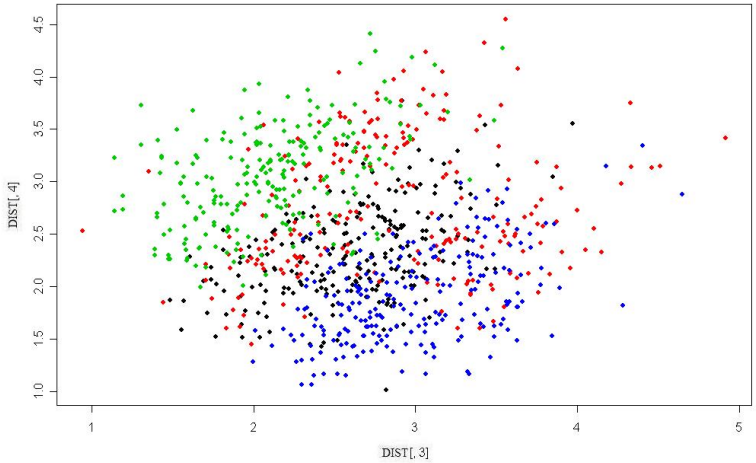
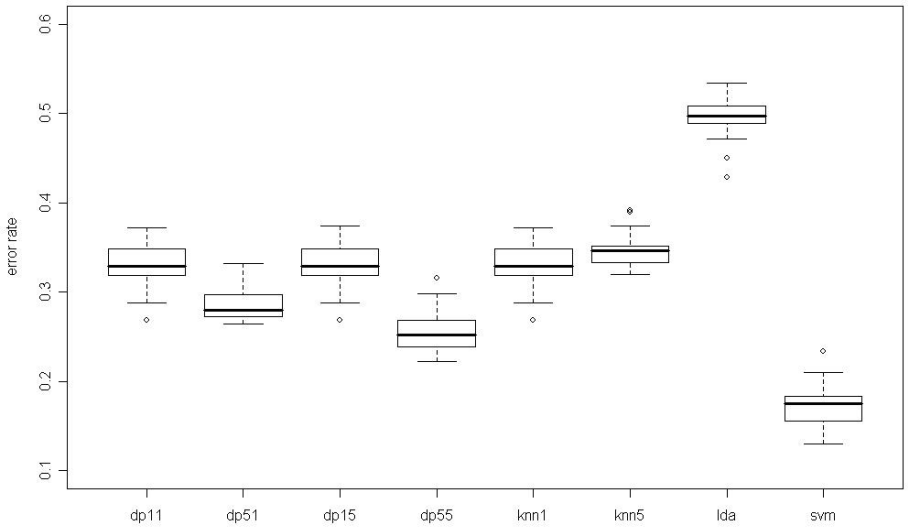


Figure 5.67: Scatterplot of groups three and four of four multimodal class data with noise in distance space



so poorly and why these other methods would be an improvement. There is no clear linear separation, however there does not seem to be any evidence to suggest that DPNN would out perform NN. (See Figure 5.69). This is further supported by the graphical representation of the data in distance space, where there is no clear separation of the two groups resulting from the large amounts of overlap. (See Figure 5.70).

Figure 5.68: Boxplots of error rates for four dimensional Gaussian sphere with noise



5.2.6 Ten dimensional Gaussian sphere

In Simulation 4.6, however, DPNN, specifically the ones that call for the fifth nearest neighbor for initial point to group distance measuring, performs very well comparatively with other methods. This suggests an investigation into the nature of choosing nearest neighbors for DPNN methods. It is important to note that $knn5$ performs differently than either dp_{55} or dp_{51}. In comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. (See

Figure 5.69: Scatterplot of multidimensional scaling representation of four dimensional Gaussian sphere with noise data in feature space

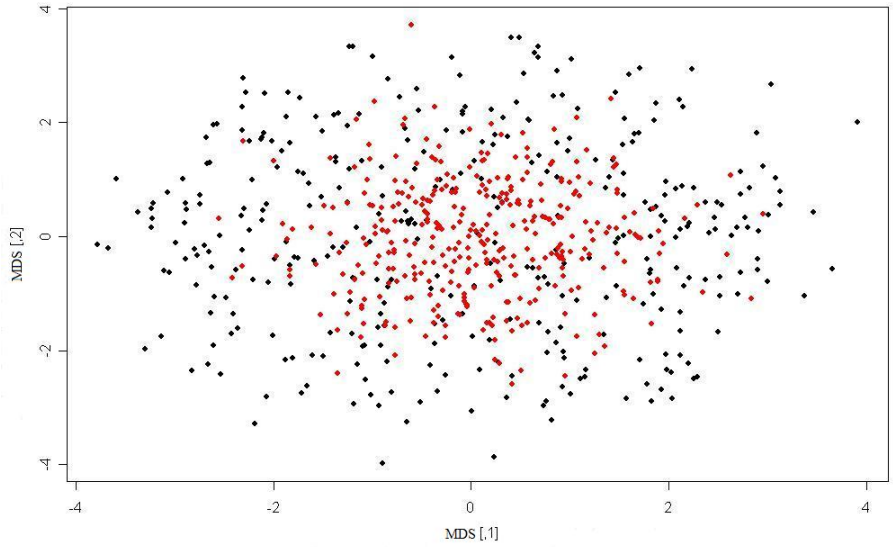


Figure 5.70: Scatterplot of multidimensional scaling representation of four dimensional Gaussian sphere with noise data in distance space

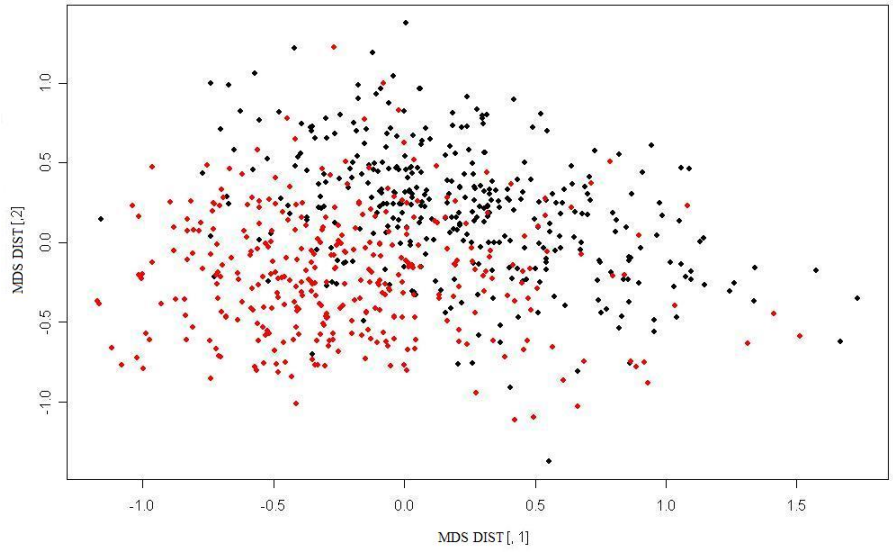
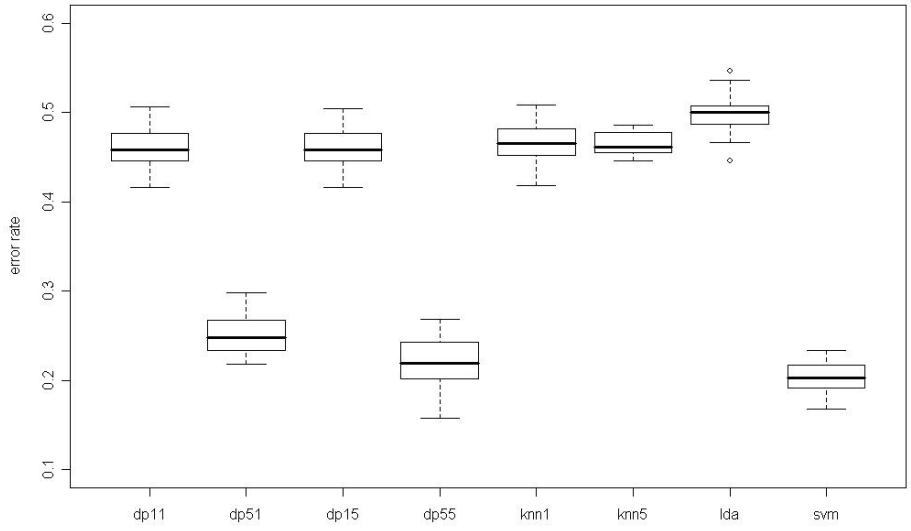


Figure 5.71). From the representation of the data in the feature space, it is clear why a distance profile method would work well. One group is much more spread out than the other. While there is overlap in generation of the MDS plot, it is not entirely likely for overlap in the true the feature space. (See Figure 5.72). The distance space representation of the data demonstrates why data of this nature would be sensitive to the choice of which nearest neighbor is used in developing and comparing distance profiles. (See Figure 5.73).

Figure 5.71: Boxplots of error rates for ten dimensional Gaussian sphere



5.2.7 Four multivariate normal classes

DPNN does not perform very well compared to LDA or SVM for Simulation 4.7. In comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. (See Figure 5.74). This is understandable after observing the data in the feature space where all classes seem to have very similar patterns for distance and overlap quite a bit. (See Figure 5.75). Representation of the data in

Figure 5.72: Scatterplot of multidimensional scaling representation of ten dimensional Gaussian sphere data in feature space

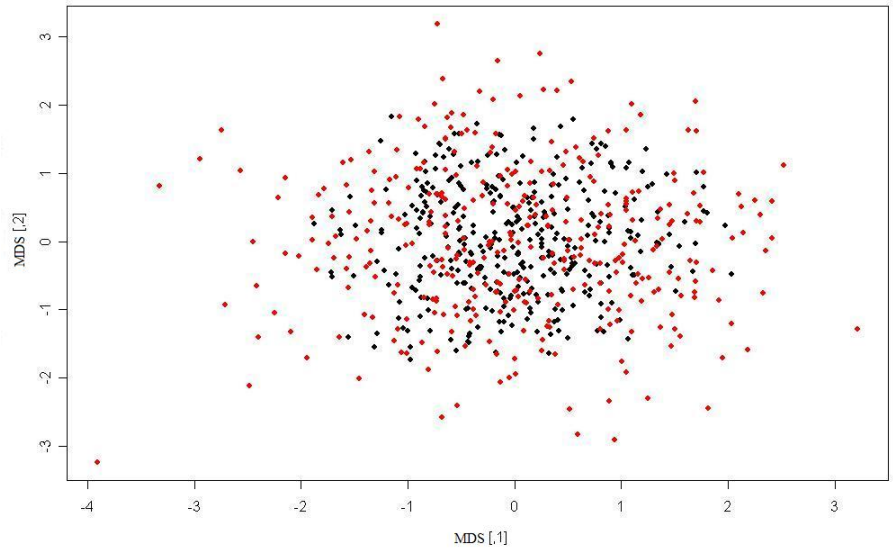
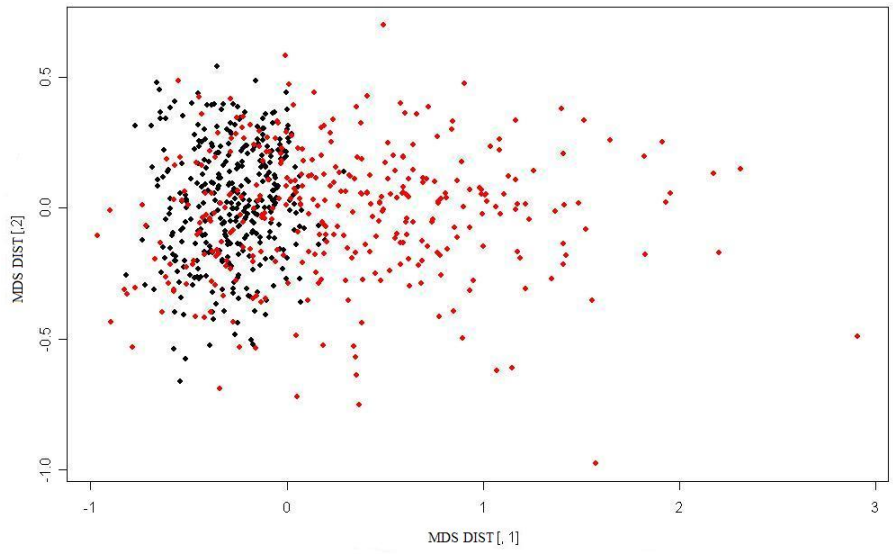


Figure 5.73: Scatterplot of multidimensional scaling representation of ten dimensional Gaussian sphere data in distance space



distance space further demonstrates this. There is an outlying observation at (-4.5,-1.25) that may be causing this structural dilemma and influencing the measurements to nearest neighbor distance profile comparisons. (See Figure 5.76).

Figure 5.74: Boxplots of error rates for four multivariate normal classes

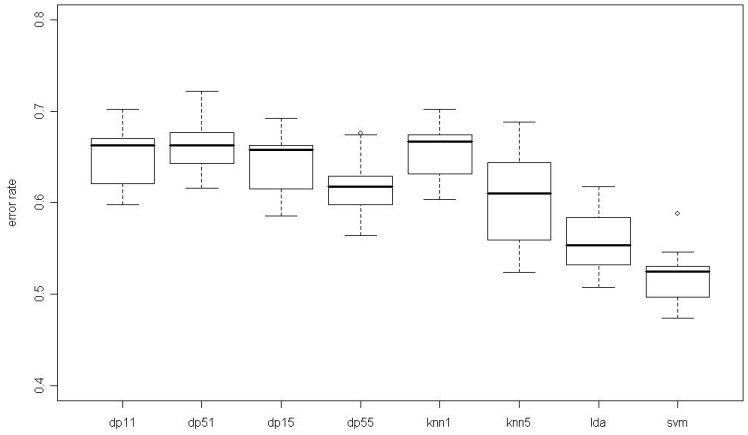


Figure 5.75: Scatterplot of multidimensional scaling representation of four multivariate normal classes data in feature space

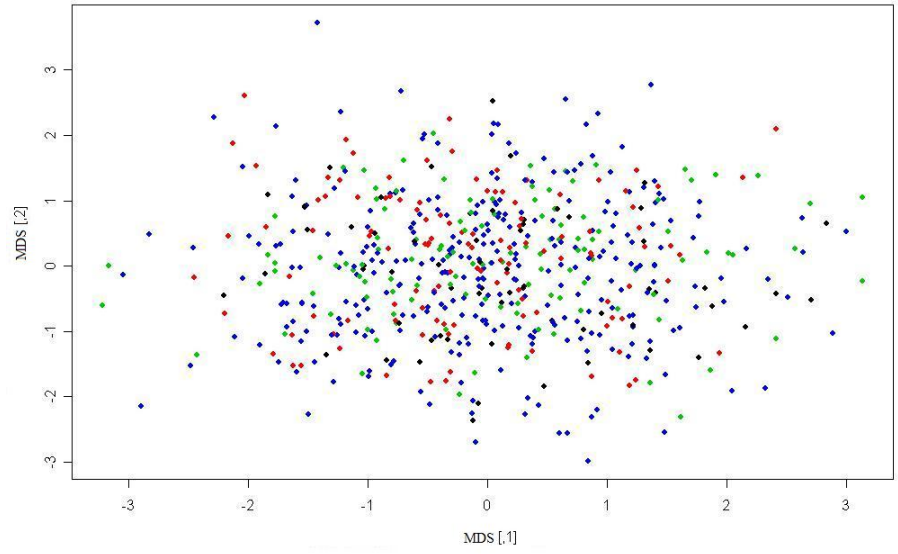
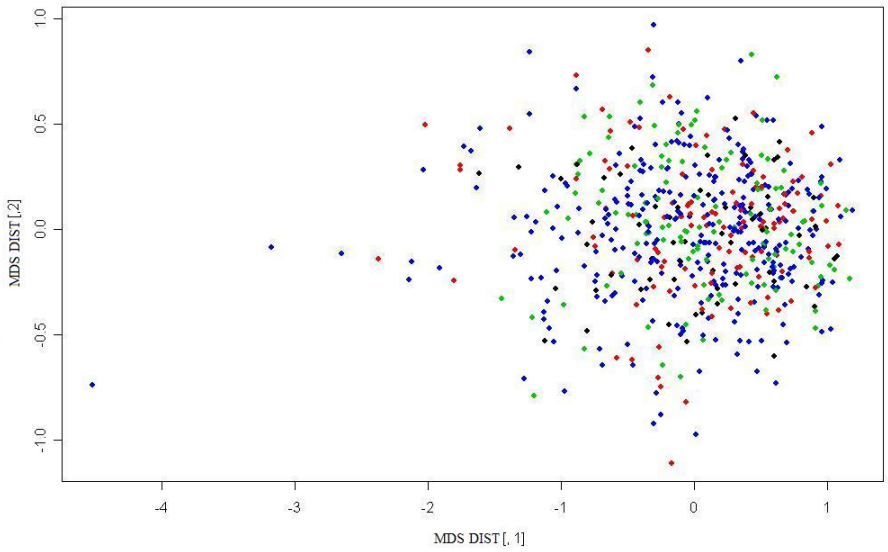


Figure 5.76: Scatterplot of multidimensional scaling representation of four multivariate normal classes data in distance space



5.2.8 Two completely separable classes

All error rates are very small in Simulation 4.8 because the two classes are very far from each other with no overlap. In comparison with Hastie's result in [Hastie & Tibshirani, 1996], performance was in relatively the same region. (See Figures 5.77, 5.78 and 5.79).

5.3 Simulation study: Gaussian and uniform hyperspheres

For further study, more simulations were conducted based on the observations made from the previous analyses. The following results were obtained using the different methods previously discussed. Error rates were computed based on the misclassifications for each method so comparisons can be drawn as to the accuracy of each method. For each dataset, training and testing samples were selected 20 times randomly then standardized. Because boxplot representations of all these error rates

Figure 5.77: Boxplots of error rates for two completely separable classes

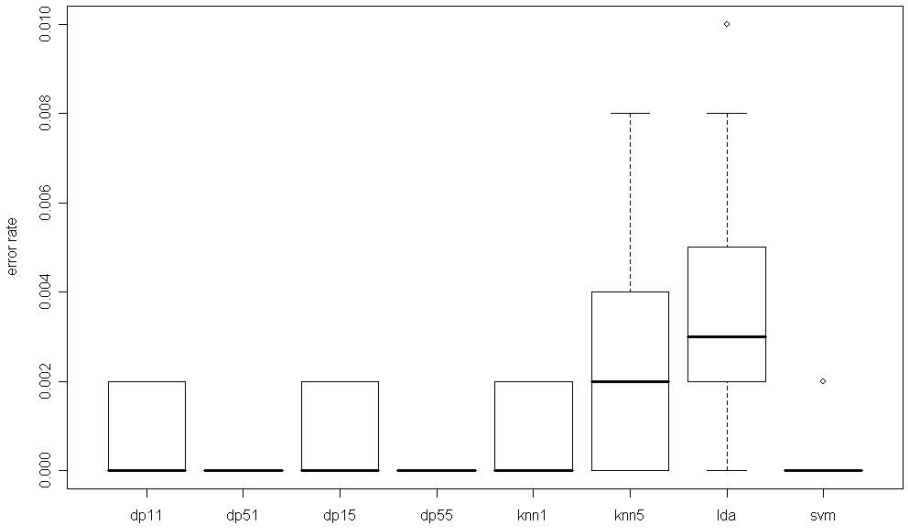


Figure 5.78: Scatterplot of multidimensional scaling representation of two completely separable classes data in feature space

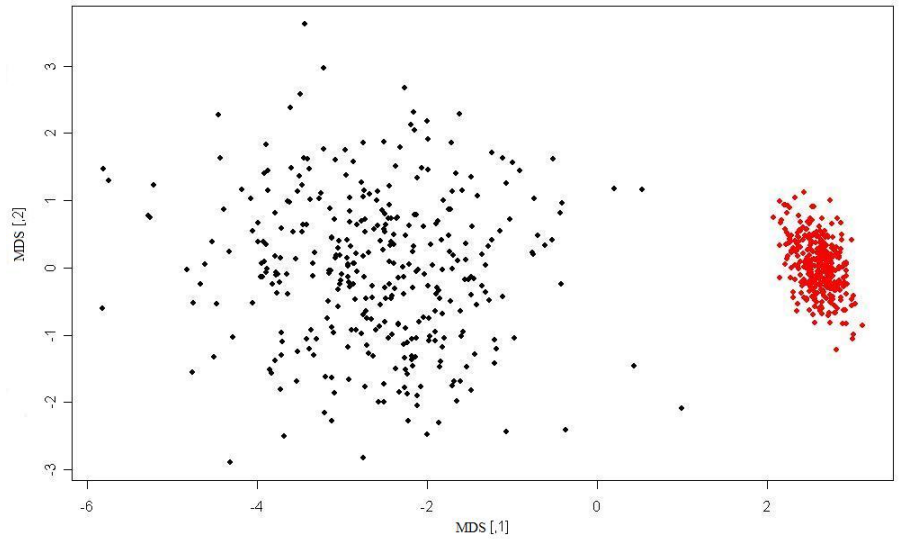
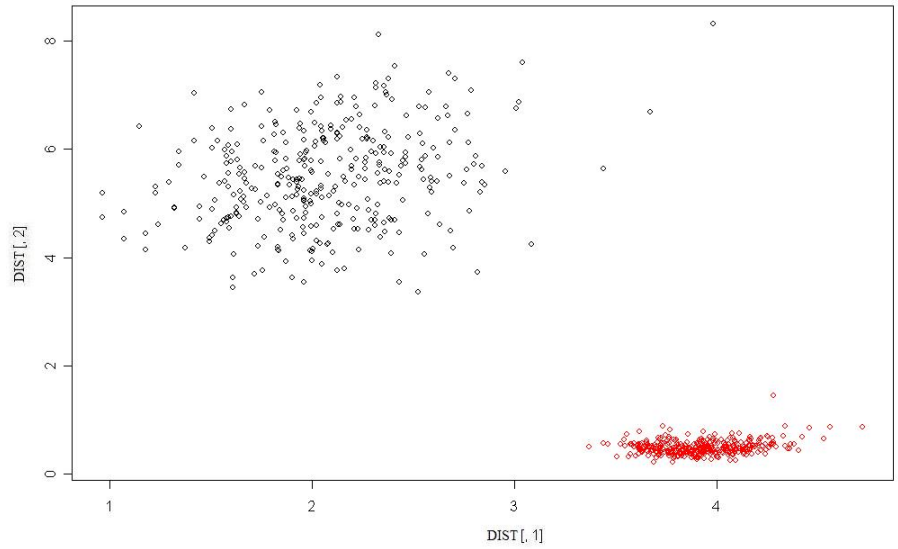


Figure 5.79: Scatterplot of two completely separable classes data in distance space



would not be beneficial, the medians for each of the 20 simulations was used instead. Under both Normal distributional assumptions and Uniform distributional assumptions, different values for p, the number of predictors for each class and δ, representing the influence of distance between classes where distance between classes is $\sqrt{\delta/p}$ were used and compared for different values of k, where k1 represents the $k_{1^{st}}$ nearest neighbor used to measure point to group distance and k2 represents the $k_{2^{nd}}$ nearest neighbor used to measure group to group distance.

5.3.1 Two Gaussian hyperspheres

5.3.2 Two uniform hyperspheres

For the Normal as p increases, the error increases. As δ increases, the error decreases. As k_1 increases, the error fluctuates as it does when k_2 increases. This is always true for the Uniform. As p increases, the error increases. As δ increases, the error decreases. As k_1 increases, the error fluctuates as it does when k_2 increases.

Table 5.1: Median error rates for one dimensional Gaussian classes

		δ								
k1	k2	0	1	2	3	4	5	6	7	10
1	1	0.4850	0.4500	0.3975	0.3525	0.2825	0.2425	0.2275	0.1950	0.1050
	2	0.5025	0.4600	0.3875	0.3350	0.2800	0.2325	0.2150	0.1900	0.1050
	3	0.4850	0.4225	0.3500	0.3075	0.2625	0.2250	0.1950	0.1650	0.0875
	4	0.5050	0.4350	0.3650	0.3150	0.2600	0.2300	0.1900	0.1600	0.0875
	5	0.4900	0.4225	0.3525	0.3050	0.2550	0.2150	0.1825	0.1625	0.0800
2	1	0.5025	0.4300	0.3600	0.2975	0.2400	0.2225	0.1850	0.1475	0.0825
	5	0.5000	0.4150	0.3350	0.2600	0.1975	0.1900	0.1550	0.1375	0.0700
3	1	0.5050	0.4175	0.3525	0.2950	0.2350	0.2125	0.1700	0.1500	0.0825
	3	0.5125	0.4125	0.3125	0.2700	0.2150	0.1875	0.1625	0.1275	0.0725
	5	0.5025	0.3925	0.2950	0.2675	0.2000	0.1725	0.1525	0.1225	0.0725
4	1	0.5050	0.4125	0.3275	0.2675	0.2100	0.1850	0.1625	0.1325	0.0775
	5	0.5000	0.3800	0.2800	0.4275	0.1850	0.1575	0.1400	0.1150	0.0575
5	1	0.5050	0.4150	0.3225	0.2950	0.2175	0.1875	0.1675	0.1675	0.0850
	2	0.5075	0.4125	0.3150	0.2800	0.2175	0.1900	0.1725	0.1475	0.0825
	3	0.4950	0.3950	0.3025	0.2625	0.1950	0.1700	0.1475	0.1250	0.0750
	4	0.4950	0.4000	0.3050	0.2625	0.2025	0.1650	0.1450	0.1300	0.0675
	5	0.5075	0.3800	0.2875	0.2475	0.1900	0.1525	0.1375	0.1175	0.0650

Table 5.2: Median error rates for two dimensional Gaussian discs

		δ								
k1	k2	0	1	2	3	4	5	6	7	10
1	1	0.5175	0.4575	0.4125	0.3500	0.3100	0.2600	0.2225	0.1725	0.1100
	2	0.4975	0.4600	0.3850	0.3450	0.2975	0.2525	0.2200	0.1725	0.1225
	3	0.5100	0.4650	0.3650	0.3250	0.2800	0.2275	0.1975	0.1625	0.0925
	4	0.5225	0.4650	0.3775	0.3250	0.2625	0.2225	0.1975	0.1700	0.1025
	5	0.5175	0.4375	0.3650	0.3075	0.2650	0.2150	0.1925	0.1550	0.0875
2	1	0.4900	0.4350	0.3750	0.3075	0.2650	0.2375	0.1950	0.1550	0.1000
	5	0.5025	0.4150	0.3425	0.2600	0.2175	0.1875	0.1475	0.1250	0.0775
3	1	0.4900	0.4375	0.3700	0.2825	0.2550	0.2100	0.1750	0.1425	0.1000
	3	0.5175	0.4050	0.3250	0.2575	0.2225	0.1800	0.1450	0.1275	0.0775
	5	0.5075	0.3900	0.3225	0.2500	0.2100	0.1750	0.1400	0.1150	0.0700
4	1	0.4975	0.4400	0.3500	0.2950	0.2525	0.2150	0.1650	0.1425	0.0925
	5	0.4925	0.3900	0.3125	0.2400	0.2050	0.1750	0.1350	0.1100	0.0650
5	1	0.5025	0.4200	0.3375	0.2825	0.2525	0.2075	0.1650	0.1450	0.0900
	2	0.5000	0.4350	0.3500	0.2825	0.2550	0.2175	0.1775	0.1500	0.0975
	3	0.5050	0.3925	0.3075	0.2425	0.2300	0.1650	0.1400	0.1225	0.0750
	4	0.5050	0.4125	0.3075	0.2375	0.2275	0.1675	0.1375	0.1225	0.0600
	5	0.5050	0.3825	0.2900	0.2225	0.2050	0.1650	0.1250	0.1125	0.0675

Table 5.3: Median error rates for three dimensional Gaussian spheres

		δ								
k1	k2	0	1	2	3	4	5	6	7	10
1	1	0.5125	0.4725	0.4175	0.3525	0.3225	0.2750	0.2275	0.2100	0.1400
	2	0.4950	0.4675	0.4100	0.3575	0.3125	0.2775	0.2250	0.2000	0.1325
	3	0.5200	0.4700	0.4050	0.3325	0.2750	0.2475	0.2025	0.1700	0.1125
	4	0.5125	0.4625	0.3900	0.3275	0.2825	0.2350	0.2025	0.1550	0.1025
	5	0.5000	0.4575	0.3850	0.3125	0.2750	0.2250	0.1925	0.1650	0.1025
2	1	0.5050	0.4600	0.3825	0.3100	0.2825	0.2450	0.1875	0.1775	0.1050
	5	0.5125	0.4300	0.3350	0.2700	0.2375	0.2000	0.1500	0.1400	0.0725
3	1	0.5100	0.4500	0.3600	0.3075	0.2650	0.2325	0.1850	0.1650	0.1075
	3	0.5150	0.3925	0.3025	0.2350	0.2050	0.1775	0.1400	0.1125	0.0650
	5	0.5050	0.4350	0.3250	0.2675	0.2200	0.1900	0.1500	0.1250	0.0650
4	1	0.5050	0.4275	0.3775	0.3100	0.2475	0.2250	0.1775	0.1650	0.0975
	5	0.5000	0.3950	0.3125	0.2625	0.2075	0.1850	0.1400	0.1250	0.0700
5	1	0.4975	0.4325	0.3650	0.3075	0.2475	0.2300	0.1950	0.1475	0.0900
	2	0.4975	0.4325	0.3625	0.2925	0.2550	0.2200	0.1825	0.1625	0.0900
	3	0.4950	0.4075	0.3175	0.2700	0.2075	0.1900	0.1550	0.1225	0.0700
	4	0.4925	0.4150	0.3250	0.2725	0.2225	0.1825	0.1550	0.1250	0.0750
	5	0.4900	0.4050	0.3000	0.2550	0.2025	0.1750	0.1375	0.1150	0.0650

Table 5.4: Median error rates for four dimensional Gaussian hyperspheres

		δ								
k1	k2	0	1	2	3	4	5	6	7	10
1	1	0.5000	0.4750	0.4250	0.3750	0.3150	0.2675	0.2300	0.2000	0.1300
	2	0.5075	0.4850	0.4275	0.3725	0.3100	0.2875	0.2400	0.1950	0.1250
	3	0.4975	0.4775	0.4050	0.3425	0.2800	0.2400	0.1975	0.1725	0.1100
	4	0.5100	0.4625	0.4025	0.3300	0.2825	0.2275	0.2075	0.1725	0.1050
	5	0.5150	0.4550	0.4025	0.3325	0.2625	0.2300	0.1850	0.1725	0.1025
2	1	0.4975	0.4400	0.4000	0.3275	0.2675	0.2300	0.2025	0.1775	0.1100
	5	0.4875	0.4125	0.3475	0.2900	0.2275	0.1900	0.1600	0.1350	0.0850
3	1	0.4925	0.4400	0.3850	0.3150	0.2875	0.2225	0.1875	0.1550	0.1075
	3	0.4925	0.4000	0.3100	0.2525	0.2000	0.1550	0.1350	0.1125	0.0700
	5	0.4975	0.4200	0.3400	0.2750	0.2125	0.1825	0.1450	0.1250	0.0850
4	1	0.5000	0.4475	0.3725	0.3050	0.2675	0.2200	0.1775	0.1475	0.0975
	5	0.5025	0.4150	0.3275	0.2600	0.2050	0.1750	0.1400	0.1200	0.0775
5	1	0.5075	0.4400	0.3850	0.2850	0.2625	0.2075	0.1800	0.1500	0.1050
	2	0.4950	0.4450	0.3875	0.3100	0.2550	0.1975	0.1875	0.1500	0.1050
	3	0.5025	0.4100	0.3425	0.2625	0.2250	0.1725	0.1475	0.1275	0.0775
	4	0.5000	0.4075	0.3450	0.2475	0.2225	0.1925	0.1475	0.1250	0.0825
	5	0.4975	0.3975	0.3225	0.2500	0.2050	0.1675	0.1400	0.1100	0.0750

Table 5.5: Median error rates for ten dimensional Gaussian hyperspheres

		δ								
k1	k2	0	1	2	3	4	5	6	7	10
1	1	0.4975	0.4900	0.4375	0.3850	0.3675	0.3100	0.3050	0.2525	0.1450
	2	0.5000	0.4925	0.4425	0.3975	0.3700	0.3000	0.2950	0.2475	0.1650
	3	0.4900	0.4850	0.4300	0.3700	0.3425	0.2800	0.2650	0.2150	0.1375
	4	0.4975	0.4950	0.4250	0.3925	0.3500	0.2750	0.2525	0.2175	0.1225
	5	0.4875	0.4850	0.4150	0.3700	0.3325	0.2625	0.2325	0.2050	0.1225
2	1	0.5000	0.4775	0.4100	0.3875	0.3450	0.2675	0.2525	0.2025	0.1150
	5	0.4900	0.4600	0.3950	0.3350	0.2800	0.2200	0.1975	0.1625	0.0875
3	1	0.5050	0.4825	0.4250	0.3775	0.3050	0.2450	0.2250	0.1950	0.1175
	3	0.4800	0.4575	0.3850	0.3275	0.2650	0.2225	0.1825	0.1550	0.0950
	5	0.5100	0.4450	0.3750	0.3125	0.2675	0.1975	0.1725	0.1500	0.0825
4	1	0.5025	0.4500	0.3950	0.3575	0.2750	0.2350	0.2100	0.1725	0.1025
	5	0.5025	0.4150	0.3650	0.2950	0.2300	0.1925	0.1650	0.1450	0.0800
5	1	0.4975	0.4750	0.3900	0.3525	0.2900	0.2350	0.2050	0.1750	0.1025
	2	0.5050	0.4600	0.3825	0.3550	0.2875	0.2350	0.2025	0.1750	0.0975
	3	0.5050	0.4475	0.3550	0.3100	0.2700	0.1875	0.1725	0.1375	0.0800
	4	0.5050	0.4250	0.3575	0.3175	0.2525	0.1950	0.1700	0.1375	0.0850
	5	0.5075	0.4275	0.3450	0.3025	0.2500	0.1750	0.1625	0.1375	0.0725

Table 5.6: Median error rates for one dimensional uniform classes

		δ				
k1	k2	0	0.5	1	1.5	2
1	1	0.4925	0.4050	0.3675	0.3150	0.2650
	2	0.4825	0.4100	0.3625	0.3000	0.2500
	3	0.4950	0.3975	0.3525	0.2800	0.2175
	4	0.4850	0.3950	0.3350	0.2750	0.2050
	5	0.4825	0.4050	0.3325	0.2525	0.2025
2	1	0.5000	0.4100	0.3600	0.3075	0.2425
	5	0.5050	0.3825	0.3175	0.2675	0.1975
3	1	0.5000	0.4150	0.3625	0.3000	0.2375
	3	0.5100	0.3900	0.3425	0.2750	0.2275
	5	0.5125	0.3725	0.3325	0.2575	0.1950
4	1	0.5050	0.4075	0.3450	0.3025	0.2300
	5	0.5000	0.3725	0.3125	0.2475	0.1900
5	1	0.5050	0.3950	0.3450	0.2825	0.2300
	2	0.5050	0.4175	0.3475	0.2950	0.2450
	3	0.5025	0.3825	0.3450	0.2600	0.2125
	4	0.5250	0.3950	0.3200	0.2775	0.2225
	5	0.5075	0.3800	0.3200	0.2675	0.2000

Table 5.7: Median error rates for two dimensional uniform discs

		δ				
k1	k2	0	0.5	1	1.5	2
1	1	0.4925	0.34	0.2475	0.175	0.1025
	2	0.505	0.3325	0.2225	0.16	0.115
	3	0.4925	0.32	0.23	0.165	0.11
	4	0.4775	0.3175	0.225	0.17	0.1025
	5	0.4925	0.3075	0.2275	0.16	0.1025
2	1	0.505	0.335	0.235	0.15	0.1125
	5	0.5175	0.305	0.225	0.15	0.1125
3	1	0.495	0.325	0.2225	0.1525	0.115
	3	0.4925	0.3075	0.22	0.1525	0.11
	5	0.4875	0.305	0.22	0.1525	0.1075
4	1	0.4875	0.3075	0.2325	0.16	0.115
	5	0.485	0.3125	0.215	0.15	0.11
5	1	0.5	0.3	0.22	0.16	0.115
	2	0.4975	0.295	0.205	0.1575	0.11
	3	0.5	0.3	0.215	0.1475	0.11
	4	0.5075	0.2875	0.22	0.145	0.11
	5	0.4975	0.295	0.2125	0.1375	0.1

Table 5.8: Median error rates for three dimensional uniform spheres

		δ				
k1	k2	0	0.5	1	1.5	2
1	1	0.4975	0.2950	0.1950	0.1475	0.0775
	2	0.5075	0.2975	0.1975	0.1500	0.0850
	3	0.4950	0.2800	0.1900	0.1325	0.0650
	4	0.4975	0.2775	0.1950	0.1375	0.0700
	5	0.5100	0.2750	0.1850	0.1175	0.0650
2	1	0.4950	0.2925	0.1825	0.1225	0.0750
	5	0.4875	0.2750	0.1750	0.1075	0.0650
3	1	0.4900	0.2950	0.1825	0.1275	0.0700
	3	0.5150	0.2850	0.1725	0.1050	0.0700
	5	0.5100	0.2675	0.1725	0.1025	0.0650
4	1	0.5000	0.2900	0.1775	0.1225	0.0675
	5	0.4775	0.2500	0.1450	0.0975	0.0575
5	1	0.5075	0.2950	0.1725	0.1175	0.0650
	2	0.4875	0.2875	0.1700	0.1225	0.0700
	3	0.4875	0.2550	0.1525	0.1050	0.0650
	4	0.4975	0.2550	0.1525	0.1025	0.0600
	5	0.4950	0.2550	0.1650	0.0950	0.0600

Table 5.9: Median error rates for four dimensional uniform hyperspheres

		δ				
k1	k2	0	0.5	1	1.5	2
1	1	0.5125	0.2650	0.1400	0.0850	0.0650
	2	0.5050	0.2575	0.1350	0.0875	0.0550
	3	0.4975	0.2225	0.1150	0.0725	0.0450
	4	0.4900	0.2150	0.1125	0.0750	0.0450
	5	0.4850	0.2075	0.1125	0.0650	0.0450
2	1	0.4925	0.2300	0.1325	0.0825	0.0500
	5	0.4975	0.1825	0.1000	0.0600	0.0450
3	1	0.4975	0.2275	0.1250	0.0750	0.0450
	3	0.5025	0.1825	0.1025	0.0625	0.0400
	5	0.5025	0.1775	0.0950	0.0625	0.0375
4	1	0.5075	0.2225	0.1150	0.0700	0.0475
	5	0.5025	0.1750	0.0950	0.0575	0.0375
5	1	0.5050	0.2150	0.1175	0.0650	0.0425
	2	0.5100	0.2125	0.1175	0.0675	0.0450
	3	0.5150	0.1700	0.0950	0.0550	0.0350
	4	0.4975	0.1775	0.0975	0.0600	0.0350
	5	0.4975	0.1650	0.0900	0.0550	0.0350

Table 5.10: Median error rates for ten dimensional uniform hyperspheres

		δ				
k1	k2	0	0.5	1	1.5	2
1	1	0.5000	0.0050	0.0000	0.0000	0.0000
	2	0.5000	0.0100	0.0000	0.0000	0.0000
	3	0.4900	0.0050	0.0000	0.0000	0.0000
	4	0.5025	0.0050	0.0000	0.0000	0.0000
	5	0.4975	0.0050	0.0000	0.0000	0.0000
2	1	0.5025	0.0075	0.0000	0.0000	0.0000
	5	0.5100	0.0025	0.0000	0.0000	0.0000
3	1	0.5200	0.0050	0.0000	0.0000	0.0000
	3	0.5050	0.0050	0.0000	0.0000	0.0000
	5	0.5175	0.0050	0.0000	0.0000	0.0000
4	1	0.5025	0.0050	0.0000	0.0000	0.0000
	5	0.5100	0.0050	0.0000	0.0000	0.0000
5	1	0.5250	0.0050	0.0000	0.0000	0.0000
	2	0.5125	0.0050	0.0000	0.0000	0.0000
	3	0.5150	0.0000	0.0000	0.0000	0.0000
	4	0.5175	0.0025	0.0000	0.0000	0.0000
	5	0.5100	0.0000	0.0000	0.0000	0.0000

When comparing the Normal simulations to the Uniform simulations, it can be noticed that for the same p and same δ, the Normal simulations perform better. This may be caused by the fact that in the overlapping regions, this would occur for the Normal in the tails where there is less probability of observations, where as for the Uniform the probability of observations appearing in that region is just as likely as any other. As a result, in the overlapping regions when distance profile nearest neighbor has more difficulty, there are less problematic observations to handle for the Normal than with the Uniform.

5.4 Conclusions and conjectures

The UCI datasets as well as the simulations offer evidence that distance profiles provide an effective means of classification in certain instances. Heavily influential outlying observations as well as certain types of categorical or binary data hinder the effectiveness of this method; however, in many instances, distance profile nearest neighbor did offer a slight improvement over other techniques explored. In the instances when data was more or less continuous, irregardless of the number of predictors, distance profile nearest neighbor performed reasonably well. It provided more reasonably good improvement over other methods in cases where data appear to be generated from multimodal patterns. Given more time, an investigation into the abnormal cases may provide insight into the effectiveness of this technique.

Appendix A

Code for classification methods

All of this code should be implemented in R [R Development Core Team, 2006].

A.1 Nearest neighbor

```
# this defines the function that compares group to group distances
knn_pt <- function(train,test,cl,k2=1,votes=FALSE,mydist=dist)
{
# Initialization same as knn(class) with a few different variable names

train <- as.matrix(train)           # reads in train set as a matrix
if(is.null(dim(test)))
   dim(test) <- c(1,length(test))   # if test data is only a vector,
                                    # then ensures the test data is a row
test <- as.matrix(test)             # reads in test set as a matrix
if(any(is.na(train)) || any(is.na(test)) || any(is.na(cl)))
   stop("no missing values allowed")  # if any dataset contains missing values,
                                       # prints out an error message
nctr <- ncol(train)   # specifies the number of columns for the train dataset
nrtr <- nrow(train)   # specifies the number of rows for the train dataset
ncts <- ncol(test)    # specifies the number of columns for the test dataset
nrts <- nrow(test)    # specifies the number of rows for the test dataset

if(length(cl)!=nrtr)
   stop("‘train’ and ‘class’ have different lengths")
                     # if the number of classes does not
                     # equal the number of observations of the
                     # train dataset, then prints out an error message
```

```
if(nrtr<k2)
    {
      warning(gettextf("k2= %d exceeds number  %d of patterns",k2,nrtr),domain=NA)
      k2 <- nrtr
    }
                    # if the specified k2 is larger than the number of entries
                    # in the train dataset, then prints out an error message

if(k2<1)
   stop(gettextf("k2= %d must be at least 1",k2),domain=NA)
                    # if k2 is not a valid positive entry,
                    # then prints out an error message

if(ncts!=nctr)
   stop("dims of 'test' and 'train' differ")
                    # if the number of predictors is not the same
                    # for both the train dataset and the test dataset,
                    # then prints out an error message

cl <- as.factor(cl)                  # ensures the classes are factors
classes <- as.factor(levels(cl))     # computes the levels of the classes
ncl <- length(classes)               # computes the number of different classes

M <- NULL
counts <- NULL

for(i in 1:nrts)  # performs operations for each object in test dataset
     {
       count <- NULL
       class_counts <- NULL
       y <- test[i,]                     # takes the ith row of the test dataset
       diff_y <- mydist(c(y,train))[1:nrtr]     # computes distance between ith
                                                # object in test dataset to every
                                                # object in train dataset
       df <- data.frame(diff_y,cl)       # adds the class component so that
                                         # distances are associated with a group
       df_ordered <- df[order(df[,1]),]  # sorts the distances from smallest to
                                         # largest
       dford <- df_ordered[1:k2,]     # takes the first k2 smallest distances
```

```
    for (cc in 1:ncl)          # performs operations for each class
          {
               # counts how many elements belong to the class from the k2
               # smallest distances computed
            count <- length(dford[dford[,2]==classes[cc],2])
            class_counts <- cbind(class_counts, count)
          }

      # if there are any ties,
      # then randomly chooses a class to assign the object to
      class <- seq(along = class_counts)[class_counts == max(class_counts)]
      if (length(class) > 1)
          class <- sample(class, 1)

      # Adds the predicted class for the current test observation
      # to the vector of results
      M <- c(M,as.vector(classes[class]))
      counts <- rbind(counts,class_counts)   # specifies how many objects
                                  # from the test dataset are in each class
    }

# Output
dimnames(counts)[[2]] <- levels(cl)   # assigns the class labels to the columns
results <- list(class=M,votes=counts) # specifies each of the classes as well as
                                      # how many test objects belong to each class
if(votes) results      # outputs the number of objects in each class if they exist
else results $cl
}
```

A.2 Distance profile nearest neighbor

```
# this defines the function that computes point to group distances
knn_profCV <- function(train,test,cl,k2=1,k1=1,mydistpt=dist,mydistprof=dist)
 {

train <- as.matrix(train)           # reads in train set as a matrix
if(is.null(dim(test)))
   dim(test) <- c(1,length(test))   # if test data is only a vector,
                                    # then ensures the test data is a row
test <- as.matrix(test)             # reads in test set as a matrix
if(any(is.na(train)) || any(is.na(test)) || any(is.na(cl)))
   stop("no missing values allowed")   # if any dataset contains missing values,
                                       # prints out an error message
nctr <- ncol(train)   # specifies the number of columns for the train dataset
nrtr <- nrow(train)   # specifies the number of rows for the train dataset
ncts <- ncol(test)    # specifies the number of columns for the test dataset
nrts <- nrow(test)    # specifies the number of rows for the test dataset

if(length(cl)!=nrtr)
   stop("'train' and 'class' have different lengths")
                     # if the number of classes does not
                     # equal the number of observations of the
                     # train dataset, then prints out an error message

if(nrtr<k1)
    {
      warning(gettextf("k1= %d exceeds number  %d of patterns",k1,nrtr),domain=NA)
      k1 <- nrtr
    }
                     # if the specified k1 is larger than the number of entries
                     # in the train dataset, then prints out an error message
```

```
if(k1<1)
   stop(gettextf("k1= %d must be at least 1",k1),domain=NA)
                    # if k1 is not a valid positive entry,
                    # then prints out an error message
if(ncts!=nctr)
   stop("dims of 'test' and 'train' differ")
                    # if the number of predictors is not the same
                    # for both the train dataset and the test dataset,
                    # then prints out an error message

cl <- as.factor(cl)                  # ensures the classes are factors
classes <- as.factor(levels(cl))     # computes the levels of the classes
ncl <- length(classes)               # computes the number of different classes

P <- NULL

for(i in 1:nrtr)     # performs operations for each object in train dataset
     {
       group <- NULL
       group_dist <- NULL
       group_dist_final <- NULL
       dist <- NULL
       y <- train[i,]     # takes the ith row of the train dataset
       train_CV <- train[-i,]      # creates new train dataset that doesn't
                                   # contain object to find distance profile for
       cl_CV <- cl[-i]    # removes the class for the object under consideration
       diff_y <- mydistprof(c(y,train_CV))[1:nrow(train)-1]
                    # computes distance between ith object in original train
                    # dataset to every object in new train dataset
       df <- data.frame(diff_y,cl_CV)   # adds the class component so that
                                        # distances are associated with a group
       df <- as.matrix(df)
       df_final <- df[order(df[,1]),]   # sorts the distances from smallest to
                                        # largest
```

```
     for (g in 1:ncl)                  # performs operations for each class
           {
             group <- (df_final[df_final[,2]==classes[g],])
                     # finds all of the distances for a specified group
             group_dist <- group[order(group[,1]),]
                     # ensures the distances are sorted
                     # from smallest to largest within the group
             group_dist_final <- group_dist[k1,-2]
                     # takes the first k1 smallest distances
             dist <- cbind(dist,group_dist_final)
                     # puts all the reduced distance profiles together for each
                     # of the classes
           }
       # puts all the reduced distance profiles together for each of the objects
        P <- rbind(P,dist)
      }

P <- as.matrix(P)
a <- levels(cl)
b <- paste(a,"_train_dist")
dimnames(P)[[2]] <- b   # assigns headings to columns specifying train distances
                        # for each class

N <- NULL

for(j in 1:nrts)
     {
       group <- NULL
       group_dist <- NULL
       group_dist_final <- NULL
       dist <- NULL
       x <- NULL
       x <- test[j,]  # takes the jth row of the test dataset
       diff_x <- mydistprof(c(x,train))[1:nrtr]
                     # computes distance between jth object in test dataset
                     # to every object in original train dataset
       df <- data.frame(diff_x,cl)   # adds the class component so that distances
                                     # are associated with a group
       df <- as.matrix(df)
       df_final <- df[order(df[,1]),]   # sorts the distances from smallest to
                                        # largest
```

```
      for (g in 1:ncl)   # performs operations for each class
            {
              group <- (df_final[df_final[,2]==classes[g],])
                      # finds all of the distances for a specified group
              group_dist <- group[order(group[,1]),]
                      # ensures the distances are sorted
                      # from smallest to largest within the group
              group_dist_final <- group_dist[k1,-2]
                      # takes the first k1 smallest distances
              dist <- cbind(dist,group_dist_final)
                      # puts all the reduced distance profiles together for each
                      # of the classes
            }
        # puts all the reduced distance profiles together for each of the objects
         N <- rbind(N,dist)
       }

N <- as.matrix(N)
a <- levels(cl)
b <- paste(a,"_test_dist")
dimnames(P)[[2]] <- b # assigns headings to columns specifying test
                        # distances for each class

# calls nearest neighbor function
# performs nn to find "closest" distance profile
knn_pt(P,N,cl,k2,mydist=mydistpt)
 }
```

A.3 Anchored distance profile nearest neighbor

```
# this defines the function that computes point to group distances
knn_profCV <- function(train,test,cl,k2=1,k1=1,mydistpt=dist,mydistprof=dist)
 {

train <- as.matrix(train)           # reads in train set as a matrix
if(is.null(dim(test)))
   dim(test) <- c(1,length(test))   # if test data is only a vector,
                                    # then ensures the test data is a row
test <- as.matrix(test)             # reads in test set as a matrix
if(any(is.na(train)) || any(is.na(test)) || any(is.na(cl)))
   stop("no missing values allowed")  # if any dataset contains missing values,
                                       # prints out an error message
nctr <- ncol(train)   # specifies the number of columns for the train dataset
nrtr <- nrow(train)   # specifies the number of rows for the train dataset
ncts <- ncol(test)    # specifies the number of columns for the test dataset
nrts <- nrow(test)    # specifies the number of rows for the test dataset

if(length(cl)!=nrtr)
   stop("'train' and 'class' have different lengths")
                     # if the number of classes does not
                     # equal the number of observations of the
                     # train dataset, then prints out an error message

if(nrtr<k1)
    {
      warning(gettextf("k1= %d exceeds number  %d of patterns",k1,nrtr),domain=NA)
      k1 <- nrtr
    }
                     # if the specified k1 is larger than the number of entries
                     # in the train dataset, then prints out an error message
```

```
if(k1<1)
   stop(gettextf("k1= %d must be at least 1",k1),domain=NA)
                    # if k1 is not a valid positive entry,
                    # then prints out an error message

if(ncts!=nctr)
   stop("dims of 'test' and 'train' differ")
                    # if the number of predictors is not the same
                    # for both the train dataset and the test dataset,
                    # then prints out an error message

cl <- as.factor(cl)                  # ensures the classes are factors
classes <- as.factor(levels(cl))     # computes the levels of the classes
ncl <- length(classes)               # computes the number of different classes

P <- NULL

for(i in 1:nrtr)     # performs operations for each object in train dataset
     {
       group <- NULL
       group_dist <- NULL
       group_dist_final <- NULL
       dist <- NULL
       y <- train[i,]     # takes the ith row of the train dataset
       diff_y <- mydistprof(c(y,train))[1:nrtr]
                    # computes distance between ith object in original train
                    # dataset to every object in train dataset (including itself)
       df <- data.frame(diff_y,cl)      # adds the class component so that
                                        # distances are associated with a group
       df <- as.matrix(df)
       df_final <- df[order(df[,1]),]   # sorts the distances from smallest to
                                        # largest
```

```
     for (g in 1:ncl)                  # performs operations for each class
           {
             group <- (df_final[df_final[,2]==classes[g],])
                     # finds all of the distances for a specified group
             group_dist <- group[order(group[,1]),]
                     # ensures the distances are sorted
                     # from smallest to largest within the group
             group_dist_final <- group_dist[k1,-2]
                     # takes the first k1 smallest distances
             dist <- cbind(dist,group_dist_final)
                     # puts all the reduced distance profiles together for each
                     # of the classes
           }
       # puts all the reduced distance profiles together for each of the objects
        P <- rbind(P,dist)
      }

P <- as.matrix(P)
a <- levels(cl)
b <- paste(a,"_train_dist")
dimnames(P)[[2]] <- b  # assigns headings to columns specifying train distances
                       # for each class

N <- NULL

for(j in 1:nrts)
     {
       group <- NULL
       group_dist <- NULL
       group_dist_final <- NULL
       dist <- NULL
       x <- NULL
       x <- test[j,]  # takes the jth row of the test dataset
       diff_x <- mydistprof(c(x,train))[1:nrtr]
                     # computes distance between jth object in test dataset
                     # to every object in original train dataset
       df <- data.frame(diff_x,cl)   # adds the class component so that distances
                                     # are associated with a group
       df <- as.matrix(df)
       df_final <- df[order(df[,1]),]   # sorts the distances from smallest to
                                        # largest
```

```
      for (g in 1:ncl)   # performs operations for each class
            {
              group <- (df_final[df_final[,2]==classes[g],])
                      # finds all of the distances for a specified group
              group_dist <- group[order(group[,1]),]
                      # ensures the distances are sorted
                      # from smallest to largest within the group
              group_dist_final <- group_dist[k1,-2]
                      # takes the first k1 smallest distances
              dist <- cbind(dist,group_dist_final)
                      # puts all the reduced distance profiles together for each
                      # of the classes
            }
        # puts all the reduced distance profiles together for each of the objects
         N <- rbind(N,dist)
       }

N <- as.matrix(N)
a <- levels(cl)
b <- paste(a,"_test_dist")
dimnames(P)[[2]] <- b  # assigns headings to columns specifying test
                       # distances for each class

# calls nearest neighbor function
# performs nn to find "closest" distance profile
knn_pt(P,N,cl,k2,mydist=mydistpt)
 }
```

Appendix B

Code for generation of simulated data

All of this code should be implemented in R [R Development Core Team, 2006].

B.1 Two bivariate normal classes

```
#generate train datasets

#train class 1
mu1<-c(0,0)
sigma1<-matrix(c(1,(3*sqrt(2)/4),(3*sqrt(2)/4),2),2,2)
f<-mvrnorm(100,mu1,sigma1)
c1<-matrix("cl1",100,1)
class1<-cbind(c1,f)

#train class 2
mu2<-c(2,0)
sigma2<-matrix(c(1,(3*sqrt(2)/4),(3*sqrt(2)/4),2),2,2)
f<-mvrnorm(100,mu2,sigma2)
c2<-matrix("cl2",100,1)
class2<-cbind(c2,f)

sim1datatrain<-rbind(class1,class2)
```

```
#generate test datasets

#test class 1
f<-mvrnorm(250,mu1,sigma1)
c1<-matrix("cl1",250,1)
class1<-cbind(c1,f)

#test class 2
f<-mvrnorm(250,mu2,sigma2)
c2<-matrix("cl2",250,1)
class2<-cbind(c2,f)

sim1datatest<-rbind(class1,class2)

#standardization

sim1train<-matrix(as.numeric(sim1datatrain[,2:ncol(sim1datatrain)]),
                  nrow(sim1datatrain),ncol(sim1datatrain)-1)

sim1trainstd<-matrix(NA,nrow=nrow(sim1train),ncol=ncol(sim1train))

sim1test<-matrix(as.numeric(sim1datatest[,2:ncol(sim1datatest)]),
                 nrow(sim1datatest),ncol(sim1datatest)-1)
sim1teststd<-matrix(NA,nrow=nrow(sim1test),ncol=ncol(sim1test))

meantrain<-apply(sim1train,2,FUN=mean)
sdtrain<-apply(sim1train,2,FUN=sd)

for (i in 1:nrow(sim1train))
      {
        for (j in 1:ncol(sim1train))
              sim1trainstd[i,j]<-(sim1train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim1test))
      {
        for (j in 1:ncol(sim1test))
              sim1teststd[i,j]<-(sim1test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim1trainclass<-as.factor(sim1datatrain[,1])
sim1testclass<-as.factor(sim1datatest[,1])
```

B.2 Two bivariate normal classes with noise

```
#generate train datasets

#train class 1
mu1<-c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
sigma1<-diag(16)
f<-mvrnorm(100,mu1,sigma1)
c1<-matrix("c1",100,1)
class1<-cbind(c1,f)

#train class 2
mu2<-c(2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
sigma2<-diag(16)
f<-mvrnorm(100,mu2,sigma2)
c2<-matrix("c2",100,1)
class2<-cbind(c2,f)

sim2datatrain<-rbind(class1,class2)

#generate test datasets

#test class 1
f<-mvrnorm(250,mu1,sigma1)
c1<-matrix("c1",250,1)
class1<-cbind(c1,f)

#test class 2
f<-mvrnorm(250,mu2,sigma2)
2<-matrix("c2",250,1)
class2<-cbind(c2,f)

sim2datatest<-rbind(class1,class2)
```

```
#standardization

sim2train<-matrix(as.numeric(sim2datatrain[,2:ncol(sim2datatrain)]),
                  nrow(sim2datatrain),ncol(sim2datatrain)-1)

sim2trainstd<-matrix(NA,nrow=nrow(sim2train),ncol=ncol(sim2train))

sim2test<-matrix(as.numeric(sim2datatest[,2:ncol(sim2datatest)]),
                 nrow(sim2datatest),ncol(sim2datatest)-1)

sim2teststd<-matrix(NA,nrow=nrow(sim2test),ncol=ncol(sim2test))

meantrain<-apply(sim2train,2,FUN=mean)
sdtrain<-apply(sim2train,2,FUN=sd)

for (i in 1:nrow(sim2train))
      {
        for (j in 1:ncol(sim2train))
              sim2trainstd[i,j]<-(sim2train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim2test))
      {
        for (j in 1:ncol(sim2test))
              sim2teststd[i,j]<-(sim2test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim2trainclass<-as.factor(sim2datatrain[,1])
sim2testclass<-as.factor(sim2datatest[,1])
```

B.3 Four multimodal classes

```
set<-cbind(expand.grid(1:5,1:5)[sample(1:25,12),],rep(1:4,each=3))

#generate train datasets

#train class 1
mu11<-c(set$Var1[1],set$Var2[1])
sigma11<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu11,sigma11)
mu12<-c(set$Var1[2],set$Var2[2])
sigma12<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu12,sigma12)
mu13<-c(set$Var1[3],set$Var2[3])
sigma13<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu13,sigma13)
f<-rbind(f1,f2,f3)
c1<-matrix("cl1",60,1)
class1<-cbind(c1,f)

#train class 2
mu21<-c(set$Var1[4],set$Var2[4])
sigma21<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu21,sigma21)
mu22<-c(set$Var1[5],set$Var2[5])
sigma22<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu22,sigma22)
mu23<-c(set$Var1[6],set$Var2[6])
sigma23<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu23,sigma23)
f<-rbind(f1,f2,f3)
c2<-matrix("cl2",60,1)
class2<-cbind(c2,f)
```

```
#train class 3
mu31<-c(set$Var1[7],set$Var2[7])
sigma31<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu31,sigma31)
mu32<-c(set$Var1[8],set$Var2[8])
sigma32<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu32,sigma32)
mu33<-c(set$Var1[9],set$Var2[9])
sigma33<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu33,sigma33)
f<-rbind(f1,f2,f3)
c3<-matrix("cl3",60,1)
class3<-cbind(c3,f)

#train class 4
mu41<-c(set$Var1[10],set$Var2[10])
sigma41<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu41,sigma41)
mu42<-c(set$Var1[11],set$Var2[11])
sigma42<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu42,sigma42)
mu43<-c(set$Var1[12],set$Var2[12])
sigma43<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu43,sigma43)
f<-rbind(f1,f2,f3)
c4<-matrix("cl4",60,1)
class4<-cbind(c4,f)

sim3datatrain<-rbind(class1,class2,class3,class4)
```

```
#generate test datasets

#test class 1
f1<-mvrnorm(40,mu11,sigma11)
f2<-mvrnorm(40,mu12,sigma12)
f3<-mvrnorm(40,mu13,sigma13)
f<-rbind(f1,f2,f3)
c1<-matrix("cl1",120,1)
class1<-cbind(c1,f)

#test class 2
f1<-mvrnorm(40,mu21,sigma21)
f2<-mvrnorm(40,mu22,sigma22)
f3<-mvrnorm(40,mu23,sigma23)
f<-rbind(f1,f2,f3)
c2<-matrix("cl2",120,1)
class2<-cbind(c2,f)

#test class 3
f1<-mvrnorm(40,mu31,sigma31)
f2<-mvrnorm(40,mu32,sigma32)
f3<-mvrnorm(40,mu33,sigma33)
f<-rbind(f1,f2,f3)
c3<-matrix("cl3",120,1)
class3<-cbind(c3,f)

#test class 4
f1<-mvrnorm(40,mu41,sigma41)
f2<-mvrnorm(40,mu42,sigma42)
f3<-mvrnorm(40,mu43,sigma43)
f<-rbind(f1,f2,f3)
c4<-matrix("cl4",120,1)
class4<-cbind(c4,f)

sim3datatest<-rbind(class1,class2,class3,class4)
```

```
#standardization

sim3train<-matrix(as.numeric(sim3datatrain[,2:ncol(sim3datatrain)]),
                  nrow(sim3datatrain),ncol(sim3datatrain)-1)
sim3trainstd<-matrix(NA,nrow=nrow(sim3train),ncol=ncol(sim3train))

sim3test<-matrix(as.numeric(sim3datatest[,2:ncol(sim3datatest)]),
                 nrow(sim3datatest),ncol(sim3datatest)-1)
sim3teststd<-matrix(NA,nrow=nrow(sim3test),ncol=ncol(sim3test))

meantrain<-apply(sim3train,2,FUN=mean)
sdtrain<-apply(sim3train,2,FUN=sd)

for (i in 1:nrow(sim3train))
      {
        for (j in 1:ncol(sim3train))
              sim3trainstd[i,j]<-(sim3train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim3test))
      {
        for (j in 1:ncol(sim3test))
              sim3teststd[i,j]<-(sim3test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim3trainclass<-as.factor(sim3datatrain[,1])
sim3testclass<-as.factor(sim3datatest[,1])
```

B.4 Four multimodal classes with noise

```
set<-cbind(expand.grid(1:5,1:5)[sample(1:25,12),],rep(1:4,each=3))

#generate train datasets

#train class 1
mu11<-c(set$Var1[1],set$Var2[1])
sigma11<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu11,sigma11)
mu12<-c(set$Var1[2],set$Var2[2])
sigma12<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu12,sigma12)
mu13<-c(set$Var1[3],set$Var2[3])
sigma13<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu13,sigma13)
f<-rbind(f1,f2,f3)
c1<-matrix("cl1",60,1)
class1<-cbind(c1,f)

#train class 2
mu21<-c(set$Var1[4],set$Var2[4])
sigma21<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu21,sigma21)
mu22<-c(set$Var1[5],set$Var2[5])
sigma22<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu22,sigma22)
mu23<-c(set$Var1[6],set$Var2[6])
sigma23<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu23,sigma23)
f<-rbind(f1,f2,f3)
c2<-matrix("cl2",60,1)
class2<-cbind(c2,f)
```

```
#train class 3
mu31<-c(set$Var1[7],set$Var2[7])
sigma31<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu31,sigma31)
mu32<-c(set$Var1[8],set$Var2[8])
sigma32<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu32,sigma32)
mu33<-c(set$Var1[9],set$Var2[9])
sigma33<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu33,sigma33)
f<-rbind(f1,f2,f3)
c3<-matrix("cl3",60,1)
class3<-cbind(c3,f)

#train class 4
mu41<-c(set$Var1[10],set$Var2[10])
sigma41<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f1<-mvrnorm(20,mu41,sigma41)
mu42<-c(set$Var1[11],set$Var2[11])
sigma42<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f2<-mvrnorm(20,mu42,sigma42)
mu43<-c(set$Var1[12],set$Var2[12])
sigma43<-matrix(c((0.25)^2,0,0,(0.25)^2),2,2)
f3<-mvrnorm(20,mu43,sigma43)
f<-rbind(f1,f2,f3)
c4<-matrix("cl4",60,1)
class4<-cbind(c4,f)

mu<-c(0,0,0,0,0,0,0,0)
sigma<-diag(8)
f<-mvrnorm(240,mu,sigma)

sim4datatrain<-rbind(class1,class2,class3,class4)
sim4datatrain<-cbind(sim4datatrain,f)
```

```
#generate test datasets

#test class 1
f1<-mvrnorm(40,mu11,sigma11)
f2<-mvrnorm(40,mu12,sigma12)
f3<-mvrnorm(40,mu13,sigma13)
f<-rbind(f1,f2,f3)
c1<-matrix("cl1",120,1)
class1<-cbind(c1,f)

#test class 2
f1<-mvrnorm(40,mu21,sigma21)
f2<-mvrnorm(40,mu22,sigma22)
f3<-mvrnorm(40,mu23,sigma23)
f<-rbind(f1,f2,f3)
c2<-matrix("cl2",120,1)
class2<-cbind(c2,f)

#test class 3
f1<-mvrnorm(40,mu31,sigma31)
f2<-mvrnorm(40,mu32,sigma32)
f3<-mvrnorm(40,mu33,sigma33)
f<-rbind(f1,f2,f3)
c3<-matrix("cl3",120,1)
class3<-cbind(c3,f)

#test class 4
f1<-mvrnorm(40,mu41,sigma41)
f2<-mvrnorm(40,mu42,sigma42)
f3<-mvrnorm(40,mu43,sigma43)
f<-rbind(f1,f2,f3)
c4<-matrix("cl4",120,1)
class4<-cbind(c4,f)

mu<-c(0,0,0,0,0,0,0,0)
sigma<-diag(8)
f<-mvrnorm(480,mu,sigma)

sim4datatest<-rbind(class1,class2,class3,class4)
sim4datatest<-cbind(sim4datatest,f)
```

```
#standardization

sim4train<-matrix(as.numeric(sim4datatrain[,2:ncol(sim4datatrain)]),
                  nrow(sim4datatrain),ncol(sim4datatrain)-1)
sim4trainstd<-matrix(NA,nrow=nrow(sim4train),ncol=ncol(sim4train))

sim4test<-matrix(as.numeric(sim4datatest[,2:ncol(sim4datatest)]),
                 nrow(sim4datatest),ncol(sim4datatest)-1)
sim4teststd<-matrix(NA,nrow=nrow(sim4test),ncol=ncol(sim4test))

meantrain<-apply(sim4train,2,FUN=mean)
sdtrain<-apply(sim4train,2,FUN=sd)

for (i in 1:nrow(sim4train))
      {
        for (j in 1:ncol(sim4train))
              sim4trainstd[i,j]<-(sim4train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim4test))
      {
        for (j in 1:ncol(sim4test))
              sim4teststd[i,j]<-(sim4test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim4trainclass<-as.factor(sim4datatrain[,1])
sim4testclass<-as.factor(sim4datatest[,1])
```

B.5 Four dimensional sphere with noise

```
#generate train datasets

#train class 1
mu1<-c(0,0,0,0)
sigma1<-diag(4)
x1t4<-mvrnorm(100,mu1,sigma1)
rad<-sqrt(diag(x1t4\%*\%t(x1t4)))
for (i in 1:length(rad))
      {
        while (rad[i]$<$3)
                {
                  for (j in 1:ncol(x1t4))
                        {
                          x1t4[i,j]<-rnorm(1,0,1)
                          rad<-sqrt(diag(x1t4\%*\%t(x1t4)))
                        }
                }
      }
mu<-c(0,0,0,0,0,0)
sigma<-diag(6)
f<-mvrnorm(100,mu,sigma)
c1<-matrix("cl1",100,1)
class1<-cbind(c1,x1t4,f)

#train class 2
mu2<-c(0,0,0,0,0,0,0,0,0,0)
sigma2<-diag(10)
f<-mvrnorm(100,mu2,sigma2)
c2<-matrix("cl2",100,1)
class2<-cbind(c2,f)

sim5datatrain<-rbind(class1,class2)
```

```
#generate test datasets

#test class 1
x1t4<-mvrnorm(250,mu1,sigma1)
rad<-sqrt(diag(x1t4\%*\%t(x1t4)))
for (i in 1:length(rad))
      {
        while (rad[i]<3)
                {
                  for (j in 1:ncol(x1t4))
                        {
                          x1t4[i,j]<-rnorm(1,0,1)
                          rad<-sqrt(diag(x1t4\%*\%t(x1t4)))
                        }
                }
      }
f<-mvrnorm(250,mu,sigma)
c1<-matrix("cl1",250,1)
class1<-cbind(c1,x1t4,f)

#test class 2
f<-mvrnorm(250,mu2,sigma2)
c2<-matrix("cl2",250,1)
class2<-cbind(c2,f)

sim5datatest<-rbind(class1,class2)
```

```
#standardization

sim5train<-matrix(as.numeric(sim5datatrain[,2:ncol(sim5datatrain)]),
nrow(sim5datatrain),ncol(sim5datatrain)-1)
sim5trainstd<-matrix(NA,nrow=nrow(sim5train),ncol=ncol(sim5train))

sim5test<-matrix(as.numeric(sim5datatest[,2:ncol(sim5datatest)]),
nrow(sim5datatest),ncol(sim5datatest)-1)
sim5teststd<-matrix(NA,nrow=nrow(sim5test),ncol=ncol(sim5test))

meantrain<-apply(sim5train,2,FUN=mean)
sdtrain<-apply(sim5train,2,FUN=sd)

for (i in 1:nrow(sim5train))
      {
        for (j in 1:ncol(sim5train))
        sim5trainstd[i,j]<-(sim5train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim5test))
      {
        for (j in 1:ncol(sim5test))
        sim5teststd[i,j]<-(sim5test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim5trainclass<-as.factor(sim5datatrain[,1])
sim5testclass<-as.factor(sim5datatest[,1])
```

B.6 Ten dimensional sphere

```
#generate train datasets

#train class 1
mu1<-c(0,0,0,0,0,0,0,0,0,0)
sigma1<-diag(10)
x1t10<-mvrnorm(100,mu1,sigma1)
rad<-(diag(x1t10\%*\%t(x1t10)))
for (i in 1:length(rad))
      {
        while (rad[i]$>$sqrt(40) $|$ rad[i]$<$sqrt(22.4))
                {
                  for (j in 1:ncol(x1t10))
                        {
                          x1t10[i,j]<-rnorm(1,0,1)
                          rad<-(diag(x1t10\%*\%t(x1t10)))
                        }
                }
      }
c1<-matrix("cl1",100,1)
class1<-cbind(c1,x1t10)

#train class 2
mu2<-c(0,0,0,0,0,0,0,0,0,0)
sigma2<-diag(10)
f<-mvrnorm(100,mu2,sigma2)
c2<-matrix("cl2",100,1)
class2<-cbind(c2,f)

sim6datatrain<-rbind(class1,class2)
```

```
#generate test datasets

#test class 1
x1t10<-mvrnorm(250,mu1,sigma1)
rad<-(diag(x1t10\%*\%t(x1t10)))
for (i in 1:length(rad))
      {
        while (rad[i]$>$sqrt(40) $|$ rad[i]$<$sqrt(22.4))
                {
                  for (j in 1:ncol(x1t10))
                        {
                          x1t10[i,j]<-rnorm(1,0,1)
                          rad<-(diag(x1t10\%*\%t(x1t10)))
                        }
                }
       }
c1<-matrix("cl1",250,1)
class1<-cbind(c1,x1t10)

#test class 2
f<-mvrnorm(250,mu2,sigma2)
c2<-matrix("cl2",250,1)
class2<-cbind(c2,f)

sim6datatest<-rbind(class1,class2)
```

```
#standardization

sim6train<-matrix(as.numeric(sim6datatrain[,2:ncol(sim6datatrain)]),
                  nrow(sim6datatrain),ncol(sim6datatrain)-1)
sim6trainstd<-matrix(NA,nrow=nrow(sim6train),ncol=ncol(sim6train))

sim6test<-matrix(as.numeric(sim6datatest[,2:ncol(sim6datatest)]),
                 nrow(sim6datatest),ncol(sim6datatest)-1)
sim6teststd<-matrix(NA,nrow=nrow(sim6test),ncol=ncol(sim6test))

meantrain<-apply(sim6train,2,FUN=mean)
sdtrain<-apply(sim6train,2,FUN=sd)

for (i in 1:nrow(sim6train))
      {
        for (j in 1:ncol(sim6train))
        sim6trainstd[i,j]<-(sim6train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim6test))
      {
        for (j in 1:ncol(sim6test))
        sim6teststd[i,j]<-(sim6test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim6trainclass<-as.factor(sim6datatrain[,1])
sim6testclass<-as.factor(sim6datatest[,1])
```

B.7 Four multivariate normal classes

```
#generate train datasets

x1<-rnorm(100,0,1)
x2<-rnorm(100,0,1)
x3<-rnorm(100,0,1)
x4<-rnorm(100,0,1)
x5<-rnorm(100,0,1)
x6<-rnorm(100,0,1)
x<-cbind(x1,x2,x3,x4,x5,x6)
class<-runif(100,0,1)
cl<-NULL
for (i in 1:length(class))
      {
        if (class[i]$<$=0.1)
           cl[i]<-"cl1"
        else if (class[i]$$>$$0.1 \& class[i]$<$=0.3)
                cl[i]<-"cl2"
        else if (class[i]$$>$$0.3 \& class[i]$<$=0.5)
                cl[i]<-"cl3"
        else cl[i]<-"cl4"
      }

sim7datatrain<-cbind(cl,x)
```

```
#generate test datasets

x1<-rnorm(500,0,1)
x2<-rnorm(500,0,1)
x3<-rnorm(500,0,1)
x4<-rnorm(500,0,1)
x5<-rnorm(500,0,1)
x6<-rnorm(500,0,1)
x<-cbind(x1,x2,x3,x4,x5,x6)
class<-runif(500,0,1)
cl<-NULL
for (i in 1:length(class))
      {
        if (class[i]$<$=0.1)
           cl[i]<-"cl1"
        else if (class[i]$$>$$0.1 \& class[i]$<$=0.3)
                cl[i]<-"cl2"
        else if (class[i]$$>$$0.3 \& class[i]$<$=0.5)
                cl[i]<-"cl3"
        else cl[i]<-"cl4"
      }

sim7datatest<-cbind(cl,x)
```

```
#standardization

sim7train<-matrix(as.numeric(sim7datatrain[,2:ncol(sim7datatrain)]),
nrow(sim7datatrain),ncol(sim7datatrain)-1)
sim7trainstd<-matrix(NA,nrow=nrow(sim7train),ncol=ncol(sim7train))

sim7test<-matrix(as.numeric(sim7datatest[,2:ncol(sim7datatest)]),
nrow(sim7datatest),ncol(sim7datatest)-1)
sim7teststd<-matrix(NA,nrow=nrow(sim7test),ncol=ncol(sim7test))

meantrain<-apply(sim7train,2,FUN=mean)
sdtrain<-apply(sim7train,2,FUN=sd)

for (i in 1:nrow(sim7train))
      {
        for (j in 1:ncol(sim7train))
              sim7trainstd[i,j]<-(sim7train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim7test))
      {
        for (j in 1:ncol(sim7test))
             sim7teststd[i,j]<-(sim7test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim7trainclass<-as.factor(sim7datatrain[,1])
sim7testclass<-as.factor(sim7datatest[,1])
```

B.8 Two completely separable classes

```
#generate train datasets

#train class 1
x1<-rnorm(100,0,1)
x2<-rnorm(100,0,1)
x3<-rnorm(100,0,1)
x4<-rnorm(100,0,1)
x5<-rnorm(100,0,1)
x6<-rnorm(100,0,1)
x7<-rnorm(100,0,1)
x8<-rnorm(100,0,1)
x9<-rnorm(100,0,1)
x10<-rnorm(100,0,1)
x<-cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
c1<-matrix("cl1",100,1)
class1<-cbind(c1,x)

#train class 2
x1<-rnorm(100,sqrt(1/2),1/1)
x2<-rnorm(100,sqrt(2/2),1/2)
x3<-rnorm(100,sqrt(3/2),1/3)
x4<-rnorm(100,sqrt(4/2),1/4)
x5<-rnorm(100,sqrt(5/2),1/5)
x6<-rnorm(100,sqrt(6/2),1/6)
x7<-rnorm(100,sqrt(7/2),1/7)
x8<-rnorm(100,sqrt(8/2),1/8)
x9<-rnorm(100,sqrt(9/2),1/9)
x10<-rnorm(100,sqrt(10/2),1/10)
x<-cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
c2<-matrix("cl2",100,1)
class2<-cbind(c2,x)

sim8datatrain<-rbind(class1,class2)
```

```
#generate test datasets

#test class 1
x1<-rnorm(250,0,1)
x2<-rnorm(250,0,1)
x3<-rnorm(250,0,1)
x4<-rnorm(250,0,1)
x5<-rnorm(250,0,1)
x6<-rnorm(250,0,1)
x7<-rnorm(250,0,1)
x8<-rnorm(250,0,1)
x9<-rnorm(250,0,1)
x10<-rnorm(250,0,1)
x<-cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
c1<-matrix("cl1",250,1)
class1<-cbind(c1,x)

#test class 2
x1<-rnorm(250,sqrt(1/2),1/1)
x2<-rnorm(250,sqrt(2/2),1/2)
x3<-rnorm(250,sqrt(3/2),1/3)
x4<-rnorm(250,sqrt(4/2),1/4)
x5<-rnorm(250,sqrt(5/2),1/5)
x6<-rnorm(250,sqrt(6/2),1/6)
x7<-rnorm(250,sqrt(7/2),1/7)
x8<-rnorm(250,sqrt(8/2),1/8)
x9<-rnorm(250,sqrt(9/2),1/9)
x10<-rnorm(250,sqrt(10/2),1/10)
x<-cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
c2<-matrix("cl2",250,1)
class2<-cbind(c2,x)

sim8datatest<-rbind(class1,class2)
```

```
#standardization

sim8train<-matrix(as.numeric(sim8datatrain[,2:ncol(sim8datatrain)]),
nrow(sim8datatrain),ncol(sim8datatrain)-1)
sim8trainstd<-matrix(NA,nrow=nrow(sim8train),ncol=ncol(sim8train))

sim8test<-matrix(as.numeric(sim8datatest[,2:ncol(sim8datatest)]),
nrow(sim8datatest),ncol(sim8datatest)-1)
sim8teststd<-matrix(NA,nrow=nrow(sim8test),ncol=ncol(sim8test))

meantrain<-apply(sim8train,2,FUN=mean)
sdtrain<-apply(sim8train,2,FUN=sd)

for (i in 1:nrow(sim8train))
      {
        for (j in 1:ncol(sim8train))
              sim8trainstd[i,j]<-(sim8train[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(sim8test))
      {
        for (j in 1:ncol(sim8test))
              sim8teststd[i,j]<-(sim8test[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
sim8trainclass<-as.factor(sim8datatrain[,1])
sim8testclass<-as.factor(sim8datatest[,1])
```

B.9 Two Gaussian hyperspheres

```
#generate train datasets

n<-100
p<-1
delta<-0
x1<-mvrnorm(n,rep(0,p),diag(p))
x2<-mvrnorm(n,rep(sqrt(delta/p),p),diag(p))
cl<-rep(c(1,2),each=n)
Ndatatrain<-rbind(x1,x2)
Ndatatrain<-cbind(cl,Ndatatrain)

#generate test datasets

n<-100
p<-1
delta<-0
x1<-mvrnorm(n,rep(0,p),diag(p))
x2<-mvrnorm(n,rep(sqrt(delta/p),p),diag(p))
cl<-rep(c(1,2),each=n)
Ndatatest<-rbind(x1,x2)
Ndatatest<-cbind(cl,Ndatatest)
```

```
#standardization
Ntrain<-matrix(as.numeric(Ndatatrain[,2:ncol(Ndatatrain)]),
               nrow(Ndatatrain),ncol(Ndatatrain)-1)
Ntrainstd<-matrix(NA,nrow=nrow(Ntrain),ncol=ncol(Ntrain))

Ntest<-matrix(as.numeric(Ndatatest[,2:ncol(Ndatatest)]),
              nrow(Ndatatest),ncol(Ndatatest)-1)
Nteststd<-matrix(NA,nrow=nrow(Ntest),ncol=ncol(Ntest))

meantrain<-apply(Ntrain,2,FUN=mean)
sdtrain<-apply(Ntrain,2,FUN=sd)

for (i in 1:nrow(Ntrain))
      {
        for (j in 1:ncol(Ntrain))
              Ntrainstd[i,j]<-(Ntrain[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(Ntest))
      {
        for (j in 1:ncol(Ntest))
              Nteststd[i,j]<-(Ntest[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
Ntrainclass<-as.factor(Ndatatrain[,1])
Ntestclass<-as.factor(Ndatatest[,1])
```

B.10 Two uniform hyperspheres

```
# For p=1

#create train data sets

p<-1
radius<-1
volume<-pi^(p/2)*radius^p/gamma(p/2+1)
density<-50 # number of points per unit of volume
#n<-round(density*volume,0) # maintains density of points per unit volume
n<-100
delta<-2
x<-matrix(runif(2*n,-1,1)*radius,nrow=2*n)
x[-(1:n),]<-x[-(1:n),]+rep(sqrt(delta/p),p)
cl<-rep(c(1,2),each=n)

UDdatatrain<-cbind(cl,x)

#create test data sets

p<-1
radius<-1
volume<-pi^(p/2)*radius^p/gamma(p/2+1)
density<-50 # number of points per unit of volume
#n<-round(density*volume,0) # maintains density of points per unit volume
n<-100
delta<-2
x<-matrix(runif(2*n,-1,1)*radius,nrow=2*n)
x[-(1:n),]<-x[-(1:n),]+rep(sqrt(delta/p),p)
cl<-rep(c(1,2),each=n)

UDdatatest<-cbind(cl,x)
```

```
#standardization
UDtrain<-matrix(as.numeric(UDdatatrain[,2:ncol(UDdatatrain)]),
                nrow(UDdatatrain),ncol(UDdatatrain)-1)
UDtrainstd<-matrix(NA,nrow=nrow(UDtrain),ncol=ncol(UDtrain))

UDtest<-matrix(as.numeric(UDdatatest[,2:ncol(UDdatatest)]),
               nrow(UDdatatest),ncol(UDdatatest)-1)
UDteststd<-matrix(NA,nrow=nrow(UDtest),ncol=ncol(UDtest))

meantrain<-apply(UDtrain,2,FUN=mean)
sdtrain<-apply(UDtrain,2,FUN=sd)

for (i in 1:nrow(UDtrain))
      {
        for (j in 1:ncol(UDtrain))
              UDtrainstd[i,j]<-(UDtrain[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(UDtest))
      {
        for (j in 1:ncol(UDtest))
              UDteststd[i,j]<-(UDtest[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
UDtrainclass<-as.factor(UDdatatrain[,1])
UDtestclass<-as.factor(UDdatatest[,1])
```

```
# For p=2

#create train data sets

p<-2
radius<-1
volume<-pi^(p/2)*radius^p/gamma(p/2+1)
density<-100 # number of points per unit of volume
#n<-round(density*volume,0) # maintains density of points per unit volume
n<-100
delta<-1
r<-(runif(2*n,0,1))^(1/p)*radius
phi<-matrix(runif(2*n*(p-1),0,2*pi),nrow=(2*n))
xfirst<-cbind(rep(1,2*n),sin(phi))
x<-r*cbind(xfirst[,-p]*cos(phi),xfirst[,p])
x[-(1:n),]<-x[-(1:n),]+rep(sqrt(delta/p),p)
cl<-rep(c(1,2),each=n)

UDdatatrain<-cbind(cl,x)

#create test data sets

p<-2
radius<-1
volume<-pi^(p/2)*radius^p/gamma(p/2+1)
density<-100 # number of points per unit of volume
#n<-round(density*volume,0) # maintains density of points per unit volume
n<-100
delta<-1
r<-(runif(2*n,0,1))^(1/p)*radius
phi<-matrix(runif(2*n*(p-1),0,2*pi),nrow=(2*n))
xfirst<-cbind(rep(1,2*n),sin(phi))
x<-r*cbind(xfirst[,-p]*cos(phi),xfirst[,p])
x[-(1:n),]<-x[-(1:n),]+rep(sqrt(delta/p),p)
cl<-rep(c(1,2),each=n)

UDdatatest<-cbind(cl,x)
```

```
#standardization
UDtrain<-matrix(as.numeric(UDdatatrain[,2:ncol(UDdatatrain)]),
                nrow(UDdatatrain),ncol(UDdatatrain)-1)
UDtrainstd<-matrix(NA,nrow=nrow(UDtrain),ncol=ncol(UDtrain))

UDtest<-matrix(as.numeric(UDdatatest[,2:ncol(UDdatatest)]),
               nrow(UDdatatest),ncol(UDdatatest)-1)
UDteststd<-matrix(NA,nrow=nrow(UDtest),ncol=ncol(UDtest))

meantrain<-apply(UDtrain,2,FUN=mean)
sdtrain<-apply(UDtrain,2,FUN=sd)

for (i in 1:nrow(UDtrain))
      {
        for (j in 1:ncol(UDtrain))
              UDtrainstd[i,j]<-(UDtrain[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(UDtest))
      {
        for (j in 1:ncol(UDtest))
              UDteststd[i,j]<-(UDtest[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
UDtrainclass<-as.factor(UDdatatrain[,1])
UDtestclass<-as.factor(UDdatatest[,1])
```

```
# For p>=3

#create train data sets

p<-3
radius<-1
volume<-pi^(p/2)*radius^p/gamma(p/2+1)
#n<-round(100*volume,0) # maintains density of 100 points per unit of volume
n<-100
delta<-1
r<-(runif(2*n,0,1))^(1/p)*radius
phi<-matrix(runif(2*n*(p-1),0,2*pi),nrow=(2*n))
xfirst<-cbind(rep(1,2*n),t(apply(sin(phi),1,cumprod)))
x<-r*cbind(xfirst[,-p]*cos(phi),xfirst[,p])
x[-(1:n),]<-x[-(1:n),]+rep(sqrt(delta/p),p)
cl<-rep(c(1,2),each=n)

UDdatatrain<-cbind(cl,x)

#create test data sets

p<-3
radius<-1
volume<-pi^(p/2)*radius^p/gamma(p/2+1)
#n<-round(100*volume,0) # maintains density of 100 points per unit of volume
n<-100
delta<-1
r<-(runif(2*n,0,1))^(1/p)*radius
phi<-matrix(runif(2*n*(p-1),0,2*pi),nrow=(2*n))
xfirst<-cbind(rep(1,2*n),t(apply(sin(phi),1,cumprod)))
x<-r*cbind(xfirst[,-p]*cos(phi),xfirst[,p])
x[-(1:n),]<-x[-(1:n),]+rep(sqrt(delta/p),p)
cl<-rep(c(1,2),each=n)

UDdatatest<-cbind(cl,x)
```

```
#standardization
UDtrain<-matrix(as.numeric(UDdatatrain[,2:ncol(UDdatatrain)]),
                nrow(UDdatatrain),ncol(UDdatatrain)-1)
UDtrainstd<-matrix(NA,nrow=nrow(UDtrain),ncol=ncol(UDtrain))

UDtest<-matrix(as.numeric(UDdatatest[,2:ncol(UDdatatest)]),
               nrow(UDdatatest),ncol(UDdatatest)-1)
UDteststd<-matrix(NA,nrow=nrow(UDtest),ncol=ncol(UDtest))

meantrain<-apply(UDtrain,2,FUN=mean)
sdtrain<-apply(UDtrain,2,FUN=sd)

for (i in 1:nrow(UDtrain))
      {
        for (j in 1:ncol(UDtrain))
              UDtrainstd[i,j]<-(UDtrain[i,j]-meantrain[j])/(sdtrain[j])
      }

for (i in 1:nrow(UDtest))
      {
        for (j in 1:ncol(UDtest))
              UDteststd[i,j]<-(UDtest[i,j]-meantrain[j])/(sdtrain[j])
      }

#classes
UDtrainclass<-as.factor(UDdatatrain[,1])
UDtestclass<-as.factor(UDdatatest[,1])
```

Bibliography

[Banks et al., 2004] Banks, D., House, L., McMorris, F. R., Arabie, P. & Gaul, W., eds (2004). Classification, clustering, and data mining applications Studies in Classification, Data Analysis, and Knowledge Organization Berlin. Springer-Verlag.

[Cover & Hart, 1967] Cover, T. M. & Hart, P. E. (1967). IEEE Trans. on Information Theory IT-13, 21–27.

[Dasarathy, 1991] Dasarathy, B. V. (1991). Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, California.

[Diaconis & Efron, 1983] Diaconis, P. & Efron, B. (1983). Scientific American 248.

[D.J. Newman & Merz, 1998] D.J. Newman, S. Hettich, C. B. & Merz, C. (1998). UCI Repository of machine learning databases.

[Dudani, 1976] Dudani, S. A. (1976). IEEE Trans. Systems Man Cybernet. SMC-6, 325–327.

[Fisher, 1936] Fisher, R. A. (1936). Annual Eugenics 7-II, 179 – 188.

[Fix & Hodges, 1951] Fix, E. & Hodges, J. L. (1951). USAF School of Aviation Medicine 21-49-004.

[Fix & Hodges, 1952] Fix, E. & Hodges, J. L. (1952). USAF School of Aviation Medicine 21-49-004.

[Fukunaga & Flick, 1984] Fukunaga, K. & Flick, T. E. (1984). IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-6, 779–788.

[Fukunaga & Hummels, 1987] Fukunaga, K. & Hummels, D. M. (1987). IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-9, 103–112.

[G. Cestnik & Bratko, 1987] G. Cestnik, I. K. & Bratko, I. (1987). Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. Sigma Press.

[Gnanadesikan et al., 1989] Gnanadesikan, R., Blashfield, R. K., Breiman, L., Dunn, O. J., Friedman, J. H., Fu, K. S., Hartigan, J. A., Kettenring, J. R., Lachenbruch, P. A., Olshen, R. A. & Rohlf, F. J. (1989). Statist. Sci. 4, 34–69.

[Gokcen & Peng, 2002] Gokcen, I. & Peng, J. (Jan 2002). Lecture Notes in Computer Science 2457, 104–113.

[Gorman & Sejnowski, 1988] Gorman, R. P. & Sejnowski, T. J. (1988). Neural Networks 1, 75 – 89.

[Hand, 1997] Hand, D. (1997). Construction and Assessment of Classification Rules. John Wiley & Sons Inc., New York.

[Hastie & Tibshirani, 1996] Hastie, T. & Tibshirani, R. (1996). IEEE Trans. on Pattern Analysis and Machine Intelligence 18, 607–616.

[Hastie et al., 2001] Hastie, T., Tibshirani, R. & Friedman, J. (2001). The elements of statistical learning. Springer Series in Statistics, Springer-Verlag, New York. Data mining, inference, and prediction.

[Johnson & Wichern, 2002] Johnson, R. A. & Wichern, D. W. (2002). Applied multivariate statistical analysis. Fifth edition, Prentice Hall Inc., Upper Saddle River, NJ.

[Lim & Shih, 1999] Lim, T. S., L. W. Y. & Shih, Y. S. (1999). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms. Machine Learning.

[McLachlan, 1992] McLachlan, G. J. (1992). Discriminant analysis and statistical pattern recognition. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics, John Wiley & Sons Inc., New York. A Wiley-Interscience Publication.

[Michalski & Lavrac, 1986] Michalski, R. S., M. I. H. J. & Lavrac, N. (1986). The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains.

[Patrick & Fischer, 1970] Patrick, E. A. & Fischer, III, F. P. (1970). Information and Control 16, 128–152.

[Peterson, 1970] Peterson, D. W. (1970). IEEE Trans. Information Theory IT-16, 26–31.

[Pettis & Dubes, 1979] Pettis, K. W., B. T. A. J. A. K. & Dubes, R. C. (1979). IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-1, 25–37.

[R Development Core Team, 2006] R Development Core Team (2006). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing Vienna, Austria. ISBN 3-900051-07-0.

[Rabiner & Schafer, 1978] Rabiner, L. R. & Schafer, R. W. (1978). Digital Processing of Speech Signals. Prentice Hall.

[Schlimmer, 1987] Schlimmer, J. C. (1987). Concept acquisition through representational adjustment.

[Sigillito & Baker, 1989] Sigillito, V. G., W. S. P. H. L. V. & Baker, K. B. (1989). Johns Hopkins APL Technical Digest 10, 262 – 266.

[Smith & Johannes, 1988] Smith, J. W., E. J. E. D. W. C. K. W. C. & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. IEEE Computer Society Press.

[Tomek, 1976] Tomek, I. (1976). IEEE Trans. Systems Man Cybernet. SMC-6.