SOME CONTRIBUTIONS TO HIGH DIMENSIONAL STATISTICAL LEARNING

Hanwen Huang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Statistics and Operations Research (Statistics).

Chapel Hill
2011

Approved by

Advisor: J. S. Marron

Advisor: Yufeng Liu

Reader: Andrew B. Nobel

Reader: Jan Hannig

Reader: D. Neil Hayes

ABSTRACT

HANWEN HUANG: Some Contributions to High Dimensional Statistical Learning
(Under the direction of Professor J. S. Marron and Professor Yufeng Liu)

This dissertation consists of two major contributions to high dimensional statistical learning. The focus is on classification which is one of the central research topics in the field of statistical learning. This research is on both binary and multiclass learning.

For binary classification, we propose the Bi-Directional Discrimination (BDD) method which generalizes linear classifiers from one hyperplane to two or more hyperplanes. BDD combines the strengths of linear and general nonlinear methods. Linear classifiers are very popular, but can suffer some serious limitations when the classes have distinct subpopulations. General nonlinear classifiers can give improved classification error rates, but do not give clear interpretation of the results and present great challenges in terms of overfitting in high dimensions. BDD gives much of the flexibility of a general nonlinear classifier while maintaining the interpretability, and less tendency towards overfitting, of linear classifiers. While the idea is generally applicable, we focus our discussion on the generalization of the Support Vector Machine (SVM) and Distance Weighted Discrimination (DWD) methods. The performance and usefulness of the proposed method are assessed using asymptotics, and demonstrated through analysis of simulated and real data.

For multiclass learning, the DWD method is generalized from the binary case to the multiclass case. DWD is a powerful tool for solving binary classification problems which has been shown to improve upon SVM in high dimensional situations. We extend the binary DWD to the multiclass DWD. In addition to some well known extensions which simply combine several binary DWD classifiers, we propose a global multiclass DWD (MDWD) which finds a single classifier that simultaneously considers all classes. Our theoretical re-

sults show that MDWD is Fisher consistent, even in the particularly challenging case when there is no dominating class (i.e., maximal class conditional probability is less than 1/2). The performances of different multiclass DWD methods are assessed through simulation and real data studies.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Statistical learning plays a key role in many areas of science, finance and industry. The major focus of statistical learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data (Duda et al. (2000); Hastie et al. (2009)). This learning process falls into two main categories: supervised learning, and unsupervised learning. In supervised learning, the training sample data comprise input vectors along with the corresponding target values (output objects). The task of the supervised learner is to find a model (hypothesis) using the given data and to predict the target values for any new data. Supervised learning is called classification if the target values can be categorized into discrete classes. Unsupervised learning is a class of problems in which one seeks to summarize and explain key features of the data. It is distinguished from supervised learning in that the given data consists of input vectors without any corresponding target values. Our work focuses on classifications.

1.1 Statistical Classification Problem

Statistical classification is a supervised learning procedure in which each element in the sample is labeled as belonging to some class. Denote by $\mathbf{x}_i = (x_{i1}, \cdots, x_{id}) \in R^d$ the

input for the ith training case, and let y_i be the corresponding class label which can only take values in a discrete set. Classification is the problem of building a classification rule $\hat{y} = \hat{G}(\mathbf{x})$ based on the training sample $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$ of labeled cases, where the joint values of all of the variables are known. This rule will enable us to predict the class label $\hat{y}$ for any new object with input $\mathbf{x}$.

The simplest and most widely studied case is *two-class learning* where there are only two classes or categories. The two classes are often coded as -1 and $+1$. The more complicated case is called *multi-class learning* when there are more than two classes. The most commonly used coding for K-classes is an element of $\mathcal{G} = \{1, \cdots, K\}$.

Suppose that $(\mathbf{x}, y)$ are random variables governed by some joint probability distribution $\mathrm{P}(\mathbf{x}, y)$, and the examples are independently and identically generated from $\mathrm{P}(\mathbf{x}, y)$. The classification can be formally characterized as a density estimation problem where one is concerned with determining properties of the conditional probability $\mathrm{P}(y|\mathbf{x})$. Once the conditional (discrete) distribution $\mathrm{P}(y|\mathbf{x})$ is given, the *Bayes classifier* classifies the object to the most probable class, i.e., $\hat{G}(\mathbf{x}) = g_k$ if $\mathrm{P}(g_k|\mathbf{x}) = \max_{g \in \mathcal{G}} \mathrm{P}(g|\mathbf{x})$.

For classification problems, the feature space can be optimally divided into a collection of regions labeled according to the Bayes classification. The machine learning view of classification ignores probability distributions, but instead focuses on decision boundaries. Optimal decision boundaries result in regions which yield minimal classification errors. Points on each region will be classified as belonging to the corresponding class. The decision boundaries are also called separating hyperplanes if they are based on linear combinations of the input features.

1.2 Summary of Existing Classification Methods

There are many existing classification methods in the literature. Examples include K-Nearest Neighbors (kNN) (Cover and Hart (1967)), Neural Networks (see Anderson and Rosenfeld (1988) for a good discussion), Fisher Linear Discrimination Analysis (LDA) (Fisher (1936)), Logistic Regression (see Section 4.4 in Hastie et al. (2009) for a good discussion), Support Vector Machine (SVM) (proposed by Vapnik (1995), see Cristianini and Taylor (2000) for a good introduction) and Distance Weighted Discrimination (DWD) (Marron et al. (2007)). In the following, we give a brief description of each method for the binary classification problem.

K-nearest Neighbors

The k-nearest neighbors (kNN) algorithm is amongst the simplest of all machine learning methods. In this method, one first finds in the d-dimensional feature space the k closest objects from the training set to the new object being classified. The object is simply assigned to the majority class amongst these k neighbors, where k is a positive integer, typically small. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor. Since the neighbor is nearby, it is likely to be similar to the object being classified and so is likely to belong to the same class as that object.

Nearest neighbor methods are easy to implement and can also give quite good results if the features contain sufficient information (and if they are weighted carefully in the computation of the distance). However, as noted in Hastie et al. (2009), there are several serious disadvantages of the nearest-neighbor methods. First, they do not represent the distribution of objects in a low dimensional parameter space but rather retain the entire training set as a description of the object distribution. Therefore, the method is slow if the training set has many examples. Second, the kNN methods are very sensitive to the

presence of irrelevant variables. Adding variables that have random values for all objects (so they do not separate the classes) can cause these methods to fail.

Neural Networks

As noted by Hastie et al. (2009), neural networks are computational models that try to simulate the structure and functional aspects of biological neural networks. They are multi-stage classification methods. First derive features $Z_m,\ m = 1, \cdots, M$, from linear combinations of the inputs X via the activation function $\sigma()$, and then model the target Y as a function of linear combinations of the Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \boldsymbol{\alpha}_m^T X), \\ T &= \beta_0 + \boldsymbol{\beta}^T \mathbf{Z}, \quad f(X) = g(T), \end{aligned} \tag{1.1}$$

where $\mathbf{Z} = (Z_1, \cdots, Z_M)$. The Z_m are called hidden units because the values Z_m are not directly observed. The activation function $\sigma(v)$ is usually chosen to be the *sigmoid* $\sigma(v) = 1/(1 + e^{-v})$. The output function $g(T)$ is typically chosen to be identity function $g(T) = T$ or the $softmax$ function $g(T) = e^T/(1 + e^T)$.

The neutral network model has unknown parameters $\{\alpha_{0m}, \boldsymbol{\alpha}_m;\ m = 1, \cdots, M\}$ and $\{\beta_0, \boldsymbol{\beta}\}$, often called weights, and we seek values for them that make the model fit the training data well. Usually, sum-of-squared error or cross-entropy (deviance) defined as $-\sum_{i=1}^n y_i \log f(x_i)$ are used as the measure of fit, and the corresponding classifier is $\hat{G}(x) = \text{sign}(\hat{f}(x))$.

As noted by Duda et al. (2000), neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. The significant disadvantage of neural networks is that it is very

difficult to intuitively understand how the net is making its decision. In practice, it is hard to determine which of the features being used are important and useful for classification and which are worthless.

Fisher Linear Discrimination Analysis

Fisher's linear discriminant method seeks to find the linear combination of features which best separate classes of objects. LDA methods can be approached nonparametrically using the mean difference between the classes. The LDA methods adjust for common covariance structure by first transforming the data space of each class using their pooled within class covariance, i.e.,

$$\Sigma_w = \frac{n_{-1}\Sigma_{-1} + n_{+1}\Sigma_{+1}}{n}, \qquad \tilde{X}_k = \Sigma_w^{-1/2} X_k, \text{ for } k = -1, +1, \tag{1.2}$$

where n_k denotes the number of the samples in the kth class, and $n = n_{-1} + n_{+1}$. Then the LDA separating hyperplane is the perpendicular bisector of the line segment between the two class means in the transformed space. LDA can also be viewed as the likelihood ratio discrimination, e.g, as noted by Hastie et al. (2009). In particular, assume that the conditional probability density functions $\mathrm{P}(\mathbf{x}|y = -1)$ and $\mathrm{P}(\mathbf{x}|y = +1)$ are both normally distributed and the class covariances are identical $\Sigma_{-1} = \Sigma_{+1} = \Sigma$. Under these assumptions, the likelihood ratio discrimination reduces to LDA.

In principle, the LDA decision criterion predicts points as being from the negative class when $\mathbf{w}^T\mathbf{x} < c$ for some threshold constant c, where $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})$, and where $\boldsymbol{\mu}_{-1}$, $\boldsymbol{\mu}_{+1}$ are mean vectors of the negative class and the positive class respectively. In practice we do not know the parameters of the Gaussian distributions, and will need to estimate

them using training data:

$$\hat{\boldsymbol{\mu}}_k = \sum_{g_i=k} \frac{\mathbf{x}_i}{n_k}, \quad k = -1, +1; \tag{1.3}$$

$$\hat{\Sigma} = \frac{1}{n-2}\left(\sum_{g_i=-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{-1})(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{-1})^T + \sum_{g_i=1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_1)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_1)^T\right). \tag{1.4}$$

Note that this classifier is *linear*, in the sense that it is based on a linear function of $\mathbf{x}$.

The basic assumption of LDA is that the data originates from two classes, where the data in each class is distributed in the feature space according to a normal distribution. Despite its simplicity, as mentioned in Friedman (1989), the main weakness of LDA is that it assumes more structure in the data than is usually necessary (namely a certain normal distribution per class), and sometimes, more than what can be satisfactorily learned from the data.

Logistic Regression

The logistic regression model arises from the desire to model the posterior probabilities of the two classes via linear functions in $\mathbf{x}$. The model has the form

$$P(G = +1|X = \mathbf{x}) = \frac{\exp(\beta_0 + \boldsymbol{\beta}^T\mathbf{x})}{1 + \exp(\beta_0 + \boldsymbol{\beta}^T\mathbf{x})}. \tag{1.5}$$

The parameters to be estimated are $\theta = \{\beta_0, \boldsymbol{\beta}\}$. Given the conditional distribution $P(G = +1|X = \mathbf{x})$, y follows the binomial distribution, thus the logistic regression models can be fitted by maximum likelihood. The log-likelihood for n observations is

$$l(\theta) = \sum_{i=1}^{n}\Big[I(y_i = +1)\log p_{+1}(\mathbf{x}_i;\theta) + I(y_1 = -1)\log(1 - p_{+1}(\mathbf{x}_i;\theta))\Big], \tag{1.6}$$

where $p_{+1}(\mathbf{x};\theta) = \mathrm{P}(G = +1|X = \mathbf{x})$. The label for the new input $\mathbf{x}$ is predicted to be $\hat{G}(\mathbf{x}) = \mathrm{sign}(p_{+1}(\mathbf{x};\hat{\theta}) - 1/2)$. The logistic regression model can be considered as linear in the sense that the log-odds-ratio between the posterior probabilities of two classes is modeled as a linear function of $\mathbf{x}$.

Logistic regression is robust in the sense that it does not assume a linear relationship between the input variables and output variables, also the normal distribution is not required. However, the disadvantages of logistic regression is that it requires much more data to achieve stable, meaningful results.

Support Vector Machine

SVM performs classification by constructing a d-dimensional hyperplane that seeks to optimally separate the data into 2 categories based on certain criteria. For the separable cases, as shown in Figure 1.1, there are infinite number of lines that we can draw to separate the two classes. The SVM hyperplane (green dashed line) is oriented in such a way that the minimum distance between the separating hyperplane and the data points from each class is maximized. The minimum distance is equivalent to the distance from the green dashed line to each of the two black thin dashed lines parallel to it. This distance is also called the *geometric margin*. The three data points covered by black boxes on the two thin dashed lines are called the *support vectors*.

If we choose $\mathbf{w} \in R^d$ as the normal vector for our hyperplane and $\beta \in R$ to determine its position, in general cases (not necessarily separable), the SVM analysis involves the following minimization

$$\min_{\mathbf{w},\beta,\xi_i} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i\right] \tag{1.7}$$

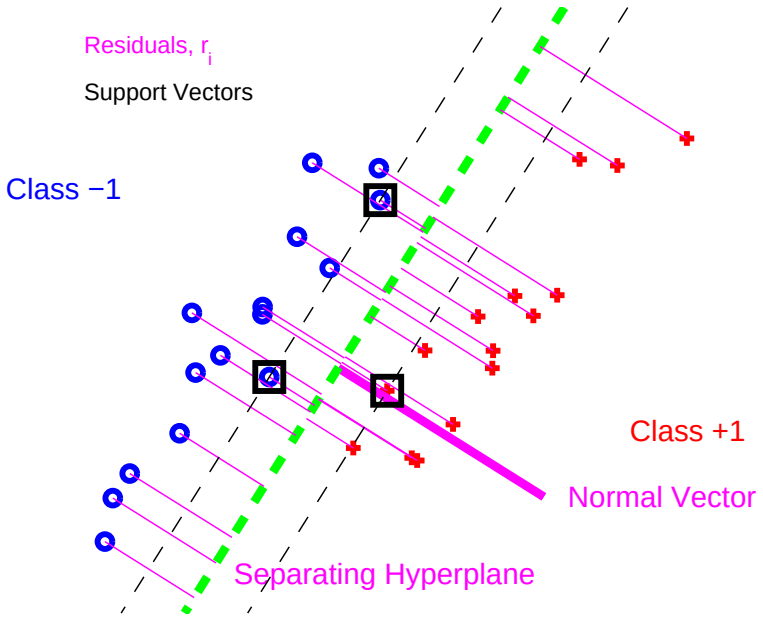


Figure 1.1: Illustration of SVM using a toy example. The red plus signs represent the positive class and the blue circle signs represent the negative class. The black boxes highlight the support vectors. The black dashed lines show where the functional margin is 1.

subject to:

$$y_i(\mathbf{x}_i^T\mathbf{w} + \beta) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \text{for} \quad i = 1, 2, \cdots, n. \tag{1.8}$$

The *functional margin* is defined to be $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + \beta$. In (1.7), C is a tuning parameter, and the $\xi_i,\ i = 1, \cdots, n$ are slack variables for handling nonseparable data. Intuitively, the sign of $f(\mathbf{x})$ is used to classify a new unseen example $\mathbf{x}$. The larger the C, the higher the penalty for violation of separability. Thus, C should be chosen with care to avoid overfitting.

One important feature of SVM is that only the support vectors, i.e. the points falling exactly on the hyperplanes which satisfy $f(\mathbf{x}) = 1$ and the violated points with $\xi_i > 0$, have a direct impact on determining the coefficients of the SVM.

Distance Weighted Discrimination

Recently, Marron et al. (2007) proposed a new binary classification method, Distance Weighted Discrimination (DWD) which is specifically designed for High Dimension Low Sample Size (HDLSS) situations. DWD has similar performance to SVM when the number of samples is larger than the number of dimensions, but performs better than SVM in HDLSS cases. Like SVM, DWD is also a large margin classifier method and performs classification tasks by constructing a hyperplane in a multidimensional space that separates the two classes. The DWD hyperplane is constructed by minimizing the sum of the inverses of perpendicular distances from a candidate for the hyperplane to the data points. Suppose the separating hyperplane is expressed as $\mathbf{x}^T\mathbf{w}+\beta = 0$, then $(\mathbf{w}, \beta)$ can be found by solving the optimization problem,

$$\min_{\mathbf{w},\beta,\xi} \sum_{i=1}^{n} \Big(\frac{1}{r_i} + C\xi_i\Big), \tag{1.9}$$

subject to:

$$r_i = y_i(\mathbf{x}_i^T\mathbf{w} + \beta) + \xi_i \quad \text{for} \quad i = 1, \cdots, n, \quad \|\mathbf{w}\|^2 \leq 1, \tag{1.10}$$

$$r_i \geq 0, \quad \xi_i \geq 0 \quad \text{for} \quad i = 1, \cdots, n. \tag{1.11}$$

DWD is different from SVM in that it seeks to maximize a notion of average distance instead of only the minimum distance between the two classes. Thus, DWD allows all data points ($\xi_i \geq 0$) rather than just those support vectors to have a direct impact on the separating hyperplane. It gives high significance to those points that are close to the hyperplane, with little impact from points that are farther away. The computation of the DWD is based on Second Order Cone Programming (SOCP), a modern computationally intensive optimization method (see http://www.math.nus.edu.sg/ mattohkc/sdpt3.html for an update software for doing this).

1.3 Kernel Space

Among the set of all classification methods, the linear methods are an important and widely studied family. The linear classification rule can be obtained as $\hat{G}(\mathbf{x}) = \text{sign} f(\mathbf{x})$ based on the function $f(\mathbf{x})$ which is a linear combination of the input features $\mathbf{x}$. The linear classification methods are convenient because they have simple functional forms and the relative contribution of each covariate is easy to interpret.

However, in practice, the true decision boundary will frequently be quite nonlinear in $\mathbf{x}$ as shown in Figure 1.2. The appealing *kernel* approach to going beyond linearity is to enlarge the feature space with additional variables, which are transformations of $\mathbf{x}$, and then use linear methods in this new space. This idea is illustrated by Figure 1.2 where FDA is applied to a simple two-dimensional toy example. Note that the results based on x_1, x_2 only (upper-left panel) are not able to effectively capture the class difference in this case. The performances will be improved as more variables are added to the model as shown in the other three panels in Figure 1.2. Adding x_1^2 (upper-right panel) or x_2^2 (lower-left panel) alone offers significant improvement over linear method while adding both x_1^2 and x_2^2 leads a much more improved decision boundary which almost makes a perfect separation between the two classes.

Let $h_m(\mathbf{x}) : R^d \to R$ denote the mth transformation also called the $basis\ transformation$ of $\mathbf{x}$, $m = 1, \cdots, M$. Once the basis functions h_m have been determined, linear classification can be performed on the h_m. We fit the classifier using input features $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \cdots, h_M(\mathbf{x}))$, and produce the (nonlinear) function $\hat{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \hat{\boldsymbol{\beta}} + \hat{\beta}_0$. The classifier is $\hat{G}(\mathbf{x}) = \text{sign}(\hat{f}(\mathbf{x}))$ as before. Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space.

Once the dimension of the enlarged space gets very large, the computations will become

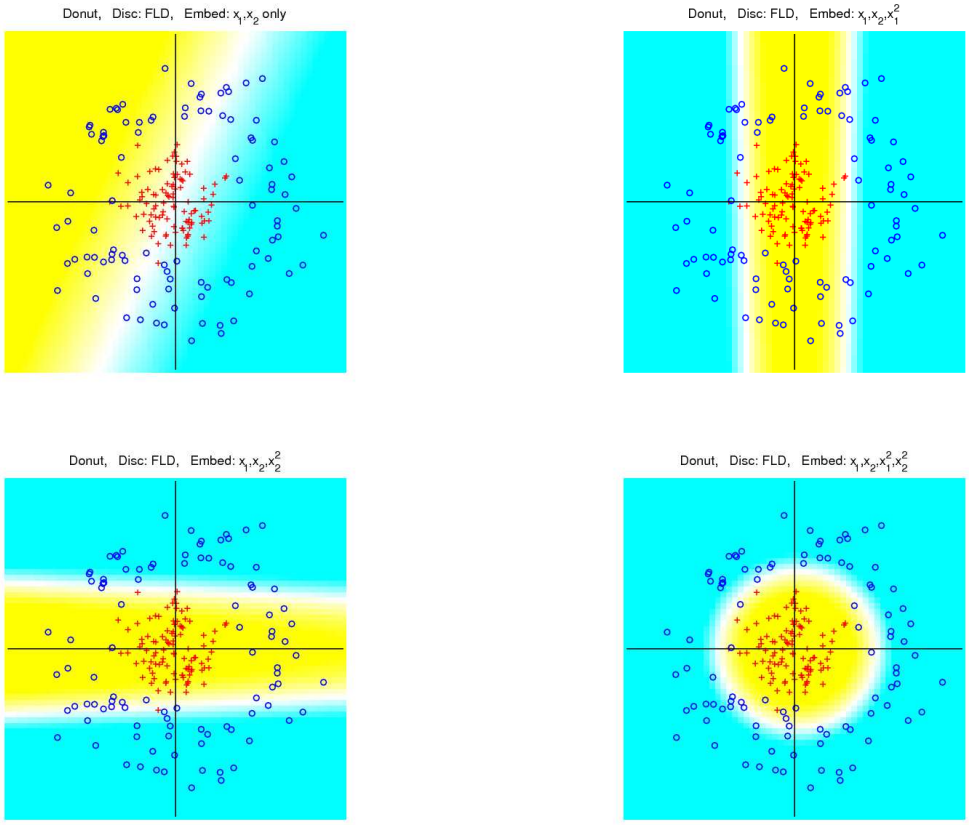


Figure 1.2: Illustration of the kernel embedding idea using a two-dimensional toy example. The four panels use different polynomial embedding. The white band represents the decision boundary. The two classes are represented by red plus and blue circle symbols. Results shown in the four panels are obtained by using variables x_1, x_2 (upper-left), x_1, x_2, x_1^2 (upper-right), x_1, x_2, x_2^2 (lower-left), and x_1, x_2, x_1^2, x_2^2 (lower-right), respectively.

a challenge. Also with sufficient basis functions, the data will nearly always be separable, and there will be large potential for overfitting. Many classification methods attempt to address this overfitting problem using some form of regularization.

We first use SVM as an example to show how to implement this basis transformation using the *kernel trick* and then cast it into the larger context of regularization methods to deal with overfitting. The SVM optimization problem (1.7) can be presented in such a way that the input feature space only appears in terms of inner products. We describe this using the transformed feature vectors $\mathbf{h}(\mathbf{x})$. The Lagrange dual function of (1.7) with $\mathbf{x}$ replaced by $\mathbf{h}(\mathbf{x})$ has the form

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \langle \mathbf{h}(\mathbf{x}_i), \mathbf{h}(\mathbf{x}_{i'}) \rangle. \tag{1.12}$$

The solution function can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}_i) \rangle + \beta_0. \tag{1.13}$$

So both (1.12) and (1.13) involve $\mathbf{h}(\mathbf{x})$ only through inner products. Thus we don't need to specify the transformation $\mathbf{h}(\mathbf{x})$, but only need to know the kernel function $K(\mathbf{x}, \mathbf{x}') = \langle h(\mathbf{x}), h(\mathbf{x}') \rangle$ that computes inner products in the transformed space. Here K should be a symmetric positive (semi-) definite function.

It is well known that many important classification methods can be fit in a general class of regularization problem of the form written as solutions to

$$\min_{f \in \mathcal{H}} \Big[\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f) \Big], \tag{1.14}$$

where $L(y, f(\mathbf{x}))$ is a loss function, $J(f)$ is a penalty functional, and $\mathcal{H}$ is a space of functions on which $J(f)$ is defined.

Suppose that the f in (1.14) lives in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ generated by a positive definite kernel $K(\mathbf{x}, \mathbf{x}')$. Further define the penalty functional for the space $\mathcal{H}_K$ to be the squared norm $J(f) = \|f\|^2_{\mathcal{H}_K}$. Then (1.14) can be written as

$$\min_{f \in \mathcal{H}_K} \Big[\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_K} \Big]. \tag{1.15}$$

It can be shown using the representer theorem (Wahba (1990)) that the solution to (1.15) is finite-dimensional, and has the form $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$. This approach is called the kernel trick.

Using the kernel trick, a linear algorithm can easily be transformed into a non-linear algorithm by mapping the data into a high dimensional feature space. This non-linear algorithm is equivalent to the linear algorithm operating in that space. The nice feature of the kernel trick is that it enables us to operate in the new space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products of the base functions between all pairs of data in the original feature space. This operation is often computationally cheaper than the explicit computation of the coordinates.

If the kernel function is chosen to be $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$, the corresponding kernel space is equivalent to the original feature space. Some commonly used kernel functions include:

- lth Degree polynomial: $K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^l$,
- Radial basis: $K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/c)$,
- Neutral network: $K(\mathbf{x}, \mathbf{x}') = \tanh(\kappa_1 \langle \mathbf{x}, \mathbf{x}' \rangle + \kappa_2)$.

The kernel space corresponding to the first choice is finite-dimensional and the kernel spaces corresponding to the second and third choices are infinite-dimensional. Algorithms capable of operating with kernel tricks include LDA, SVM, DWD and many others.

1.4 Bi-Directional Discrimination

Linear classifiers are simple and easy to interpret, but can suffer some serious limitations in the complicated situations. Kernel learning enables us to easily generalize the linear classifiers to nonlinear classifiers and improve the classification error rates. A potential trade off is that nonlinear classifiers may not give clear interpretation of the results in terms of the original features. Motivated by these concerns, we propose the Bi-Directional Discrimination (BDD) classification method in Chapter 2 which generalizes the classification boundary from using only one hyperplane to using two hyperplanes. The BDD method is anticipated to be more effective in the cases where the classes have distinct sub-populations.

In Section 2.2, we use SVM and DWD to illustrate how to generalize one-direction methods to the proposed BDD method. It is important to note that the generalization can apply to any other linear classification methods as well. The optimization problems for the BDD method involve replacing the linear function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + \beta$ (in the linear classification methods) by the product of two linear functions $f_1(\mathbf{x})f_2(\mathbf{x})$, where $f_1(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_1 + \beta_1$ and $f_2(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_2 + \beta_2$. As a consequence, we have two separating hyperplanes instead of one. The classification rule can be stated as $\hat{G}(\mathbf{x}) = \text{sign}(\hat{f}_1(\mathbf{x})\hat{f}_2(\mathbf{x}))$, i.e., points in the regions $\hat{f}_1(\mathbf{x}) > 0$, $\hat{f}_2(\mathbf{x}) > 0$ or $\hat{f}_1(\mathbf{x}) < 0$, $\hat{f}_2(\mathbf{x}) < 0$ are labeled as belonging to the positive class while points in the regions $\hat{f}_1(\mathbf{x}) > 0$, $\hat{f}_2(\mathbf{x}) < 0$ or $\hat{f}_1(\mathbf{x}) < 0$, $\hat{f}_2(\mathbf{x}) > 0$ are labeled as belonging to the negative class. The two directions introduced in the BDD method can also provide a visualization tool for HDLSS data.

It is difficult to solve the optimization problems which involve the form $f_1(\mathbf{x})f_2(\mathbf{x})$ for $(\mathbf{w}_1, \beta_1)$ and $(\mathbf{w}_2, \beta_2)$ simultaneously. In Section 2.2.2, we propose an iterative algorithm to obtain the BDD optimization solution in such a way that at each iteration we first fix one hyperplane and transform the problem into the form of the usual one-direction linear

classification problem. Then the other hyperplane can be solved from this transformed problem. This procedure is repeated until convergence is achieved.

Like many iterative algorithms, local minima are also a serious concern here especially in the HDLSS situations. Thus how to choose proper initial values will become an important issue. In Section 2.2.3, we propose four methods for choosing initial values based on two different considerations. We call the four methods Cluster1-1, Cluster2-2, Cluster1-2 and FullQuadProj respectively. The first three methods choose the initial values by considering the different subcluster situations within each class. The last method chooses the initial values by finding the two hyperplanes which best approximate an appropriate full quadratic kernel method. For each method, there are situations in which it performs better than the others.

The proposed BDD method is studied in Section 2.3 through several simulations and two real data examples. The performances of various initial value methods are evaluated using data visualization and careful studies of the test errors (for simulated data) and cross-validation errors (for real data). Comparison with the usual one-direction linear classification methods is also included. The numerical results show that in contrast to the one-direction methods, the BDD method is competitive for different data settings and gives major improvement in the case when there are distinct subclusters within each class.

In Section 2.3.2, we study the asymptotic properties of the BDD method in the limit as $d \to \infty$ with the sample size n fixed. This is different from the classic asymptotics which is in the limit as $n \to \infty$. We give the asymptotic geometric representations of the data set which include subclusters. We also study when the BDD method performs better than the usual one-direction classification methods.

1.5 Multiclass Classification

Summary of Existing Multiclass Classification Methods

Now turn our attention to the multicategory classification problem. Binary classification is a well studied special case. In practice, multicategory problems are important as well. Binary classification methods can be generalized in many ways to handle multiple classes. Some multicategory classification methods are straightforward extension of binary ideas such as kNN, neural network, LDA, and logistic regression discussed in Section 1.2. However, the extension from binary to multicategory case is more challenging for others.

The generalization of the kNN method is straightforward. In the multicategory case, one first finds the k closest objects from the training sample to a new object being classified, then assign this object to the class which appears most frequently among these k neighbors. For the neural network method, the generalization needs to introduce K functions f_k, for $k = 1, \cdots, K$, which are defined as

$$\begin{aligned} T_k &= \beta_{0k} + \boldsymbol{\beta}_k^T \mathbf{Z}, \qquad k = 1, \cdots, K, \\ f_k(X) &= g_k(\mathbf{T}), \qquad k = 1, \cdots, K. \end{aligned}$$

The unknown parameters can be solved in the same way as for the binary case. The corresponding classifier is $\hat{G}(x) = \text{argmax}_k \hat{f}_k(x)$. The extension of the LDA classifier can be implemented using the following steps. First compute the pooled within class covariance

$$\Sigma_w = \sum_{k=1}^{K} n_k \Sigma_k / n, , \tag{1.16}$$

and use it to transform the data

$$\tilde{X}_k = \Sigma_w^{-1/2} X_k, \text{ for } k = 1, \cdots, K. \tag{1.17}$$

Then label a new object according to the closest class centroid of the training data in the transformed space. The generalization of logistic regression can be carried out by modeling the posterior probabilities of K classes as

$$P(G = k|X = \mathbf{x}) = \frac{\exp(\beta_{k0} + \boldsymbol{\beta}_k^T \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \boldsymbol{\beta}_l^T \mathbf{x})}, \ k = 1, \cdots, K-1, \tag{1.18}$$

$$P(G = K|X = \mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \boldsymbol{\beta}_l^T \mathbf{x})}. \tag{1.19}$$

The classifier is $\hat{G}(\mathbf{x}) = \text{argmax}_k p_k(\mathbf{x}; \hat{\theta})$ with $p_k(\mathbf{x}; \theta) = \text{P}(G = k|X = \mathbf{x})$.

Multiclass SVM and DWD

The generalization from the binary case to the multicategory case for large margin classification methods like SVM and DWD requires careful consideration. There are a number of different multicategory extensions of SVM in the literature. However, the extension of the DWD method has not been studied previously. We have developed several DWD extension methods in Chapter 3 and studied some statistical issues associated with them.

Two general strategies are commonly used to tackle multicategory SVM problem. One strategy is to solve the multicategory problem by solving a series of binary problems. The second one treats the population in a simultaneous fashion and considers all classes at once in a single optimization problem. Various aggregation of all pairwise classifiers and one-versus-the-rest approaches are the first strategy (Duda et al. (2000); Hastie et al. (2009)). Various extension methods along the line of the second strategy include Lee et al. (2004); Weston and Watkins (1999); Crammer and Singer (2000); Liu and Shen (2006). Following

the SVM results, our work involves the study of the extension of DWD from the binary case to the multicategory case using both strategies. We make comparisons among various methods and settings by extensive simulated data and real data applications.

Many statistical properties of binary classifiers, such as Fisher consistency, have been well investigated in a variety of settings. However, it turns out that one can lose consistency in the generalization from the binary to the multicategory case. Fisher consistency is a desired condition for a classification method although a consistent method may not always give better classification accuracy. Liu (2007) reviewed Fisher consistency of several commonly used extensions and proposed some modifications to make the inconsistent extensions consistent. Fisher consistency for the binary DWD method has been proved by Qiao et al. (2010). We have investigated the Fisher consistency properties of multiclass DWD methods in different settings in Section 3.4.

CHAPTER 2

Bi-Directional Discrimination with Application to Data Visualization

2.1 Introduction

As noted in Section 1.4, while linear classifiers have been very widely used, there is an important collection of problems where they can be dramatically improved upon. This is illustrated in Figure 2.1. In this case, each class contains diverse sub-populations. For example, in microarray analysis, within each class of interest (e.g., disease versus control) immaterial differences such as male versus female can lead to diverse sub-populations. A toy example illustrating this is given in Figure 2.1 which shows a scatter plot of two dimensional data. The positive and negative classes are represented as red and blue respectively. Each class is further divided into two sub-clusters which are distinguished using different symbols in the scatter plot. The linear SVM model is fit to these data and its decision boundary is denoted by the solid line in the left panel of Figure 2.1. Note that linear methods for classification are not able to effectively capture the class difference in this case which motivates us to find a more general hypersurface that can divide the two classes of samples.

One of the nice features of the DWD and SVM methods is that their extension from the

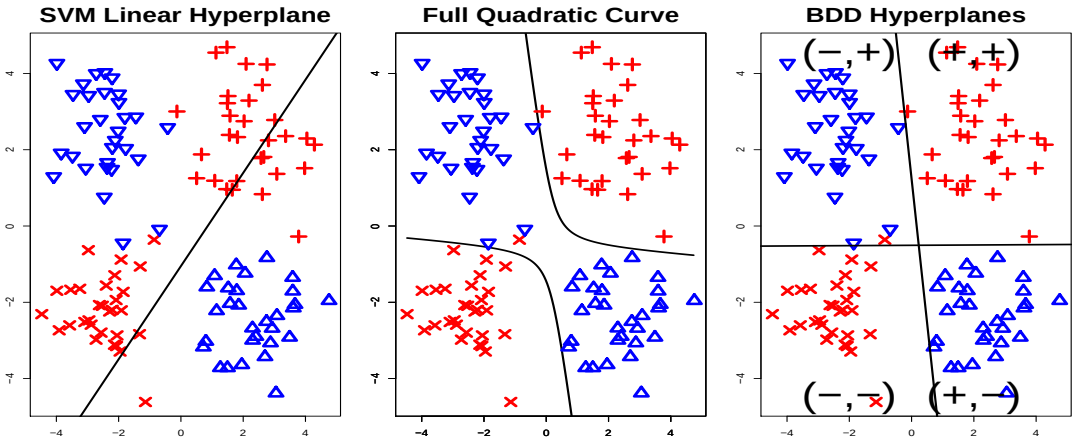


Figure 2.1: Toy data example in two dimensions with three different discrimination curves shown using a solid line-type. Red color (plus and "x" symbols') indicates the positive class and blue color (up and down triangles) indicates the negative class. Different symbols in the same class represent different sub-clusters. Note the two non-linear methods give (middle and right panels) major improvements.

linear case to the non-linear case is allowed and quite straightforward using the kernel trick (Aizerman et al. (1964); Boser et al. (1992)). This is accomplished by mapping the data to a high-dimensional space where the classification is achieved via a linear classifier, and then by mapping the results back to the original feature space. This results in a non-planar hypersurface that can be more adapted to the complexity of the interface between the two classes, and thus is more effective. The solid curves in the middle panel of Figure 2.1 are the non-linear decision boundary implemented using the full quadratic kernel method (Vapnik (1995); Burges (1998)). Its performance is clearly much better than that of the linear classifier.

Although non-linear classification methods can be very effective in resolving subtle and complex class differences, they do not easily provide intuitive interpretation of the result, as compared to the linear ones. Especially in the HDLSS settings, the complex form of general non-linear classifiers makes it difficult to apply them to data visualization. Moreover, general non-linear methods may deal with a space whose dimensionality is much

higher than that of the original feature space and thus tend to be far more prone to serious overfitting in high dimensions. In order to get around these problems and be able to display class differences in a way that is not only effective, but also suitable for visual interpretation of the data, we develop a new classification method in this dissertation, called Bi-Directional Discrimination (BDD). The basic idea of BDD is to find two (or more) linear hyperplanes instead of one to separate the two classes. The BDD decision boundary is shown in the right panel of Figure 2.1 which does a more intuitively appealing job of separating the two classes since it not only provides good between-class separation but also clearly divides each class into two sub-clusters.

The BDD method has big advantages over general non-linear methods especially for HDLSS data. HDLSS data are becoming increasingly common in various fields including genetic microarrays, medical imaging and chemometrics, etc. If the dimension is d, the number of parameters included in the BDD method will be $2d$, much less than that included in the quadratic kernel method which is at least $\begin{pmatrix} d \\ 2 \end{pmatrix}$. As a consequence of its simplicity, the overfitting problem for the full quadratic kernel illustrated in Section 2.3 is greatly reduced by BDD. Another important feature of our BDD method is that its two hyperplanes can automatically provide a visualization tool for HDLSS data. For many tasks of HDLSS data analysis, visualization plays an important role. This is key for efficient integration of human expertise - not only to include background knowledge, intuition and creativity, but also the powerful human pattern recognition and processing capability (Walter (2004)). Therefore, studying the projections of the data points onto the two directions solved by our BDD approaches can help us obtain more insights from the data, and thus reveals a whole new family of methods between the simple one-direction linear methods and the full general non-linear methods.

Other approaches in the literature also have the potential to address the subcluster

problem that is tackled by BDD. For example, Gaussian mixture models (see e.g. Hastie and Tibshirani (1996)) have the flexibility to associate Gaussian mixture components to each subclass to facilitate effective classification when the classes have subpopulation. But they are not suitable for high dimensional analysis, which is the main motivation of BDD. Classification and Regression Trees (CART) and more advanced tree methods (Breiman et al. (1984)) can tackle very high dimensions, but are much less flexible than BDD because they only allow splits in coordinate directions.

In this chapter, we initially focus on the two-directional method. We also generalize BDD to multiple directions. In particular, we discuss the three-directional method as well as its implementation. Moreover, although our BDD method is motivated by the SVM and DWD methods, we note that the fundamental concept is more general and can be applied to the extension of any other linear classifier as well. In this dissertation, we only focus on the discussion of the SVM and DWD methods and use them as examples to illustrate how the BDD method works.

The rest of the chapter is organized as follows. In Section 2.2 we briefly review the one-direction SVM and DWD methodologies, from now on labeled 1SVM and 1DWD respectively, and introduce their extension to BDD. We develop iterative algorithms to solve the optimization problem of BDD in Section 2.2.2. In particular, the challenging issue of initial values is discussed in Section 2.2.3. Extensions of BDD to more than two directions are discussed in Section 2.2.4. In Section 2.3 we present numerical results on both simulated and real data to demonstrate the effectiveness of our method. Some asymptotic properties which demonstrate the value of BDD in the presence of subclusters, in the limit as the dimension tends to infinity, are explored in Section 2.4. We provide the proofs of the theorems in Section 2.5 and some conclusions in Section 2.6.

2.2 Bi-Directional Discrimination Framework

This section gives the details of how to generalize the 1SVM and 1DWD methods from the usual one-direction case to the two-direction case. Let us first set the notation to be used. Suppose that the training data set consists of n d-vectors x_i together with corresponding class indicators $y_i \in \{+1, -1\}$, which are distributed according to some unknown probability distribution function $P(\mathbf{x}, \mathbf{y})$.

2.2.1 Review of Uni-Directional Methods

The main idea behind the classical one-direction classification problem in the separable case is to find the separating hyperplane with maximum separation between the two classes. More specifically, the 1SVM hyperplane maximizes the distance between the hyperplane and the closest data point of each class, while the 1DWD hyperplane minimizes the sum of the reciprocals of the distances from every data point to the separating hyperplane. One important goal is to do prediction, i.e., if we choose $\mathbf{w} \in \mathcal{R}^d$ as the normal vector for our hyperplane and $\beta \in \mathcal{R}$ to determine its position, the sign of $f = \mathbf{x}^T\mathbf{w} + \beta$ can be used for prediction of class labels for new inputs $\mathbf{x}$. The optimization problems, for both the 1SVM and 1DWD approaches, depend on the signed distance from each data point to the decision boundary, which is defined as

$$r_i^0 \;=\; y_i(\mathbf{x}_i^T\mathbf{w} + \beta),\ i = 1 \cdots, n. \tag{2.1}$$

If separation between the two classes is not feasible, we need to add perturbation terms to make sure that all residuals are positive (Cortes and Vapnik (1995)). We obtain

$$r_i \;=\; y_i(\mathbf{x}_i^T\mathbf{w} + \beta) + \xi_i, \tag{2.2}$$

where the slack variable $\xi_i \geq 0$, and the equality holds when the data vector $\mathbf{x}_i$ lies on the correct side of the separating hyperplane. The hyperplane parameters $(\mathbf{w}, \beta)$ can be determined to encourage all r_i to be positive and large. The 1SVM classifier solves the regularization problem

$$min_{\{\mathbf{w},\beta\}} \left(\frac{1}{2}||\mathbf{w}||^2 + C_{1SVM} \sum_{i=1}^{n} \xi_i \right), \tag{2.3}$$

subject to $y_i(\mathbf{x}_i^T\mathbf{w} + \beta) + \xi_i \geq 1$ and $\xi_i \geq 0$, where $C_{1SVM} > 0$ is the penalty parameter, which balances the separation and the amount of violation of the constraints. Here $||\mathbf{w}||$ refers to the Euclidean norm of $\mathbf{w}$.

The optimization formula (2.3) is the primal problem of the 1SVM. Using Lagrange multipliers, it can be converted to an equivalent dual problem as follows

$$min_{\alpha} \left(\frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{n} \alpha_i \right), \tag{2.4}$$

subject to $\sum_{i=1}^{n} y_i \alpha_i = 0$; $0 \leq \alpha_i \leq C_{1SVM}$, $\forall i$. This convex optimization problem has quadratic objective function and linear constraints and can be easily solved. Once the solution of (2.4) is obtained, $\mathbf{w}$ can be calculated as $\sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i$, and β can be computed using the Karash-Kuhn-Tucker (KKT) conditions of the optimization theory (Fletcher (1987)).

The optimization task of 1DWD is to find a separating hyperplane which solves

$$min_{\{\mathbf{w},\beta\}} \sum_{i} \left(\frac{1}{r_i} + C_{1DWD} \xi_i \right), \tag{2.5}$$

subject to $r_i = y_i(\mathbf{x}_i^T\mathbf{w} + \beta) + \xi_i \geq 0$, $||\mathbf{w}||^2 = 1$ and $\xi_i \geq 0$, where $C_{1DWD} > 0$ is the 1DWD penalty parameter. The optimization formula (2.5) can be reparametrized as a second order cone programming (SOCP) problem, which has a linear objective function

and is subject to linear constraints with the requirement that various sub-vectors of the decision vector must lie in second-order cones. SOCP problems have also been extensively studied and there exist well established algorithms for solving them, see Alizadeh et al. (2001). The dual problem of 1DWD can also be described in terms of the SOCP settings. Both primal and dual problems of 1DWD have optimal solutions. For detailed description of 1DWD formulation and optimization, we refer to the original 1DWD paper (Marron et al. (2007)).

2.2.2 Bi-Directional Discrimination

In the two-direction case, we have two hyperplanes represented by parameters $(\mathbf{w}_1, \beta_1)$ and $(\mathbf{w}_2, \beta_2)$ respectively. Let $f_1 = \mathbf{x}^T\mathbf{w}_1+\beta_1$ and $f_2 = \mathbf{x}^T\mathbf{w}_2+\beta_2$ be classification functions representing each of the two separating hyperplanes (as $f_1 = 0$ and $f_2 = 0$). As shown in the right panel of Figure 2.1, we denote by $(+,+)$ the region which satisfies $f_1 > 0$ and $f_2 > 0$. The other three regions can be denoted in a similar way. It turns out that the data from the positive class tend to be located on the upper-right and lower-left regions with labels $(+,+)$ and $(-,-)$ while the data from the negative class tend to lie in the upper-left and lower-right regions with labels $(-,+)$ and $(+,-)$. Thus $sign(f_1 f_2)$ is used as the predicted rule in the two-direction setting. Therefore, a natural way of generalizing linear classifiers is to replace the signed distance r_i of the ith data point (2.2) with

$$s_i \;\; = \;\; y_i f_1 f_2 + \xi_i. \tag{2.6}$$

Once the s_i are given, the optimization problem solved by the two-direction SVM can be stated as

$$min_{\mathbf{w},\beta,\xi} \frac{1}{2}(||\mathbf{w}_1||^2 + ||\mathbf{w}_2||^2) + C_{SVM} \sum_{i=1}^{n} \xi_i \tag{2.7}$$

subject to

$$s_i = y_i(\mathbf{x}_i^T\mathbf{w}_1 + \beta_1)(\mathbf{x}_i^T\mathbf{w}_2 + \beta_2) + \xi_i \geq 1, \xi_i \geq 0. \tag{2.8}$$

The illustration plots for one-direction SVM and two-direction SVM are shown in the left panel and the right panel of Figure 2.2 respectively. The decision boundary of two-direction SVM consists of two lines. The curves defined by $yf_1f_2 = 1$ are four hyperbolas which correspond to the two lines defined by $yf = 1$ in the one-direction plot. Thus two-direction SVM seeks to choose two hyperplanes to maximize the distances between the four hyperbolas that are as far apart as possible. The data points which lie on the four hyperbolas are the support vectors for two-direction SVM.

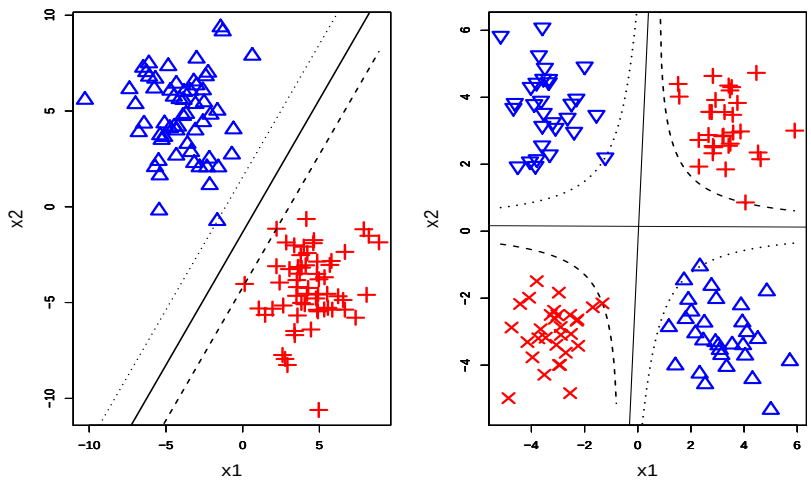


Figure 2.2: Illustration plots for both one-direction SVM (left panel) and two-direction SVM (right panel). Solid lines represent decision boundaries. Dashed and dotted lines in the left panel are defined by $f = 1$ and $f = -1$ respectively. Dashed curves and Dotted curves in the right panel are defined by $f_1f_2 = 1$ and $f_1f_2 = -1$ respectively.

Similarly, the optimization problem solved by the two-direction DWD can be stated as

$$min_{\mathbf{w},\beta,\xi} \sum_i \left(\frac{1}{s_i} + C_{DWD}\xi_i \right), \tag{2.9}$$

subject to

$$s_i = y_i(\mathbf{x}_i^T\mathbf{w}_1 + \beta_1)(\mathbf{x}_i^T\mathbf{w}_2 + \beta_2) + \xi_i \geq 0, \xi_i \geq 0,$$
$$||\mathbf{w}_1||^2 + \beta_1^2 = 1, ||\mathbf{w}_2||^2 + \beta_2^2 = 1. \tag{2.10}$$

To meet the uniqueness requirement, here we use the constraints $||\mathbf{w}_j||^2 + \beta_j^2 = 1$ instead of $||\mathbf{w}_j||^2 = 1$, $j = 1, 2$, as used in the original 1DWD method. We choose this type of constraint to ensure that the optimization problem can be described in SOCP terms.

The multiplicative form of the s_i in (2.6) poses significantly greater optimization challenges and makes it difficult to simultaneously solve for $(\mathbf{w}_1, \beta_1)$ and $(\mathbf{w}_2, \beta_2)$ both in (2.7) and in (2.9). However, we note that as long as one of the two hyperplanes is given, (2.7) and (2.9) can be solved for the other hyperplane using methods similar to the ordinary 1SVM and 1DWD. This property suggests that iterative algorithms can be used here. Therefore we propose to solve the two-direction minimization problem by minimizing a sequence of one-direction sub-problems. We can proceed as follows. First propose initial values for $\{\mathbf{w}_1^{(0)}, \beta_1^{(0)}\}$. Then obtain $\{\mathbf{w}_2^{(0)}, \beta_2^{(0)}\}$ by solving the revised 1SVM and 1DWD problems with y_i replaced by $\hat{y}_i = y_i(\mathbf{x}_i^T\mathbf{w}_1^{(0)} + \beta_1^{(0)})$. Then based on $\{\mathbf{w}_2^{(0)}, \beta_2^{(0)}\}$ we can obtain $\{\mathbf{w}_1^{(1)}, \beta_1^{(1)}\}$ and repeat this process until convergence of both parameters. Thus, a solution can be achieved by alternately updating each hyperplane based on each fixed value of the other one. In each iteration, we only need to solve the modified 1SVM or 1DWD problems whose response values are continuous $(\hat{y}_i)$ instead of binary (y_i). In all cases we considered, this algorithm converges in at most 10 steps.

2.2.3 Starting Points

Local minima can be a serious concern for iterative optimization methods. The solution based on the iterative algorithm described in Section 2.2.2 strongly depends on the choice of

the initial values $\{\mathbf{w}_1^{(0)}, \beta_1^{(0)}\}$. Different initial values may end up with different solutions. Especially for high dimensional situations, the objective functions can have many local minima due to the complexity of their special multiplicative form. Figure 2.3 shows the distribution of the final (after convergence of the iterative algorithm) objective function values based on a single realization of a simulated data set with d=1000. Details of this simulation are discussed in Section 2.3.2. The blue kernel density estimation (KDE) plot is derived from 1000 samples of objective function values, each of which is calculated based on one randomly selected starting point. The vertical lines with different colors represent the results derived from some special initial points. Here MIN-RAND represents the minimum values among the 1000 random simulations and the other notations will be discussed in detail in this section. Figure 2.3 illustrates how crucial the starting points are to our optimization algorithm. Our next goal is to propose some appropriate ways to choose good initial values.

These ideas are effectively illustrated using a set of 3 two-dimensional toy examples described in Section 2.2.3.1. An approach to starting values based on the full quadratic kernel embedding is developed in Section 2.2.3.2. An alternative approach, based on clustering, is given in Section 2.2.3.3.

2.2.3.1 Toy Examples

Since one of the motivations of our two-direction classification method comes from the fact that there might be further sub-clusters within each class, we can illustrate our methods using examples as shown in Figure 2.4 where three types of 2-dimensional data sets are generated from normal distributions with different means.

Example 1 (4-Cluster-Twisted):

This example is shown in the plot (a) of Figure 2.4 which includes four clusters, two

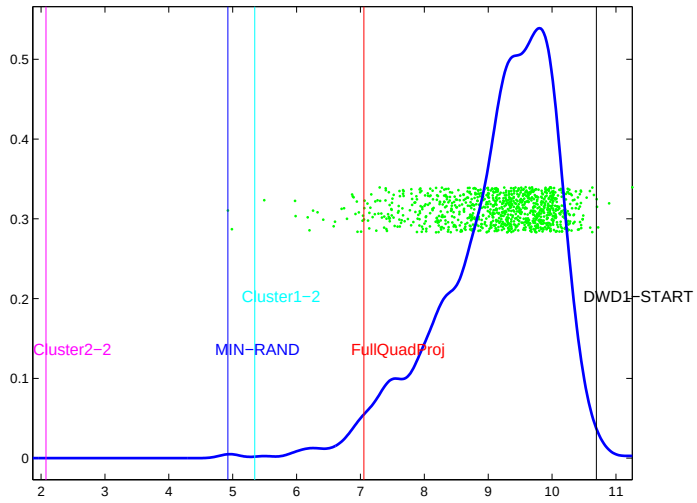


Figure 2.3: KDE plot of objective function values for different starting points.

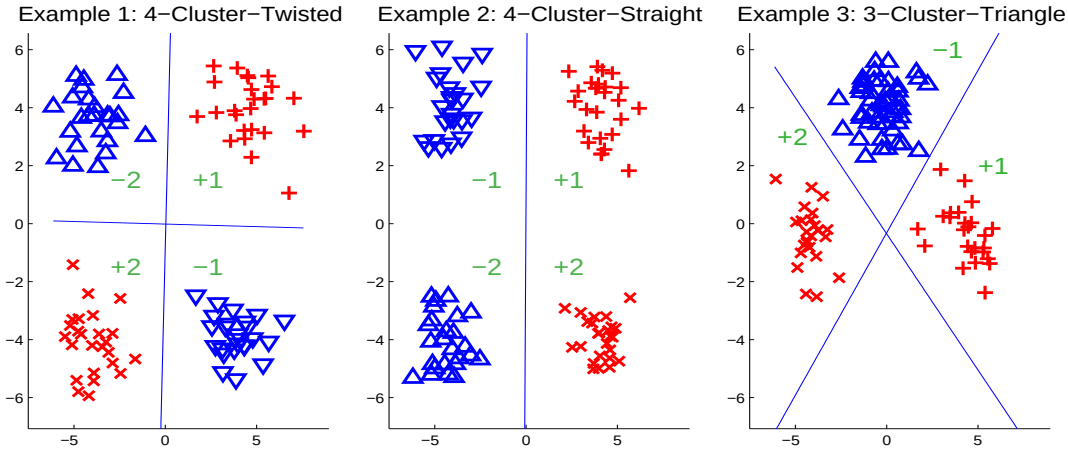


Figure 2.4: Illustration of some different sub-cluster situations for binary classification problems. Red color (plus and "x" symbols') indicates the positive class and blue color (up and down triangles) indicates the negative class. Different symbols in the same class represent different sub-clusters.

for each class. The four clusters are sampled from four shifted standard bi-variate normal distributions whose means are (μ, μ), $(-\mu, \mu)$, $(-\mu, -\mu)$, and $(-\mu, \mu)$ with $\mu = \sqrt{5}$. We label the four distinct clusters as $+1$ (red plus sign), $+2$ (red "x" sign), -1 (blue down-triangle), -2 (blue up-triangle) as shown in the plot. Clusters $+1$ and $+2$ (centered in the upper right and lower left quadrants) belong to the positive class and clusters -1 and -2 (centered in the upper left and lower right quadrants) belong to the negative class. The

numbers of individuals in each cluster are 25.

Example 2 (4-Cluster-Straight):

This example is shown in the plot (b) of Figure 2.4 which also includes four clusters whose means are (μ, μ), $(-\mu, \mu)$, $(-\mu, -\mu)$, and $(-\mu, \mu)$ with $\mu = \sqrt{5}$ and identity covariance. In this example, clusters $+1$ and $+2$ (centered in the upper right and lower right quadrants) belong to the positive class and clusters -1 and -2 (centered in the upper left and lower left quadrants) belong to the negative class. The numbers of individuals in each cluster are 25.

Example 3 (3-Cluster-Triangle):

This example is shown in the plot (c) of Figure 2.4 where only the positive class includes two sub-clusters. Thus we have three shifted standard Gaussian clusters whose means are $(\mu, 0)$, $(-\mu, 0)$, and $(0, \mu)$ with $\mu = \sqrt{5}$. We label the three distinct clusters as $+1$ (red plus sign), $+2$ (red "x" sign) and -1 (blue up-triangle). Clusters $+1$ and $+2$ belong to the positive class and cluster -1 belongs to the negative class. Here $n_{+1} = n_{+2} = n_{-1}/2 = 25$, which means that the total number of individuals in the positive class is equal to that in the negative class.

2.2.3.2 Full Quadratic Kernel Approach

Note that the multiplication of two linear expressions in the optimization problems (2.7) and (2.9) results in a special second order polynomial. We can solve this approximately by comparing with the corresponding non-linear problem. Thus our first type of approach (abbreviated as FullQuadProj) is to finding two initial hyperplanes so that their multiplication is the closest one to the hypersurface solved using the full quadratic kernel method. Here closeness is measured by the sum of square distances. Then use one of these two hyperplanes as initial values for the iterative algorithms. Using the kernel trick, the

extension from the linear case to the non-linear case can be obtained by simply replacing the vector $\mathbf{x}_i$ by $\Phi(\mathbf{x}_i)$, where the non-linear mapping Φ is obtained from the symmetric kernel function K by performing Cholesky factorization $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$. This is equivalent to solving the linear problem in the feature space induced by the kernel K to achieve the nonlinear solution in the original space. We choose the second order polynomial kernel function which is of the form

$$K(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^2. \tag{2.11}$$

Once we get the non-linear classification function evaluated at each data point

$$\bar{y}_i \quad = \quad \Phi(\mathbf{x}_i)^T \bar{\mathbf{w}} + \bar{\beta}, \tag{2.12}$$

we can find the approximate solutions of (2.7) and (2.9) by minimizing the following residual sum-of-squares

$$\sum_{i}^{n} (\bar{y}_i - (\mathbf{x}_i^T \mathbf{w}_1 + \beta_1)(\mathbf{x}_i^T \mathbf{w}_2 + \beta_2))^2. \tag{2.13}$$

Using these solutions as initial values, we can proceed with the iterative method to get the final solution. It is also difficult to find a simple closed form solution to the optimization problem (2.13). However, since it has nice properties in the sense that both function values and derivatives can be analytically evaluated, the solution can be obtained using standard numerical optimization algorithms. We use conjugate gradient methods from Fletcher (1987) to solve this.

2.2.3.3 Clustering Approach

Our second type of initialization approach is proposed on the basis of the sub-cluster structure of the data set. We use the three examples given in Section 2.2.3.1 as a simple illustration for this.

For Example 1 (4-Cluster-Twisted), the ideal choice for the initial hyperplane will be the one-direction hyperplane that separates groups $(+1, -1)$ and $(+2, -2)$ or else the one that separates groups $(+1, -2)$ and $(+2, -1)$. Therefore, our Cluster2-2 method first uses 2-means clustering algorithm to divide the positive class into two clusters labeled as c_{+1} and c_{+2} and similarly divides the negative class into two clusters labeled as c_{-1} and c_{-2}. Then we choose the initial hyperplane as the usual one-direction hyperplane that either separates between groups (c_{+1}, c_{-1}) and (c_{+2}, c_{-2}) or separates between groups (c_{+1}, c_{-2}) and (c_{+2}, c_{-1}).

For Example 2 (4-Cluster-Straight), it is better to choose the usual one-direction linear classifier as an initial value. Thus, our Cluster1-1 method chooses the usual one-direction hyperplane between the positive and the negative classes as the initial value.

For Example 3 (3-Cluster-Triangle), a good choice of the initial value is the one-direction hyperplane that separates groups $(+1)$ and $(+2, -1)$ or the one that separates groups $(+2)$ and $(+1, -1)$. Our Cluster1-2 method using the 2-means clustering algorithm divides the positive class into two clusters labeled as c_{+1}, c_{+2}. The initial hyperplane is chosen to be the one that separates either between groups (c_{+1}) and (c_{+2}, c_{-1}) or between groups (c_{+2}) and (c_{+1}, c_{-1}), where c_{-1} denotes the entire negative class.

We will see that each method for finding initial values has a situation for which it works the best. Typically, there is no prior knowledge as to the sub-cluster structure of the data set. Therefore, we propose to implement all of these proposed initial values and take our solution to be the one that gives the minimum value of the objective function.

2.2.4 More Than Two Directions

Although our focus is this dissertation is on the two-directional method, it can be extended to multiple directions. To generalize BDD to the K-direction case, discrimination is based on K hyperplanes represented by parameters $(\mathbf{w}_1, \beta_1)$, $\cdots$, $(\mathbf{w}_K, \beta_K)$ respectively. Let $f_i = \mathbf{x}^T\mathbf{w}_i + \beta_i$, for $i = 1, \cdots, K$ be classification functions representing each of the K separating hyperplanes (i.e. $f_i = 0$, $i = 1, \cdots, K$). The class label for a new input $\boldsymbol{x}$ is predicted to be $\text{sign}(f_1(\boldsymbol{x}) \cdots f_K(\boldsymbol{x}))$. The optimization problem in (2.7) and (2.8) can be written as

$$min_{\mathbf{w},\beta,\xi} \frac{1}{2} \sum_{j=1}^{K} ||\mathbf{w}_j||^2 + C_{SVM} \sum_{i=1}^{n} \xi_i \tag{2.14}$$

subject to

$$s_i = y_i \prod_{j=1}^{K} (\mathbf{x}_j^T \mathbf{w}_j + \beta_j) + \xi_i \geq 1, \xi_i \geq 0.$$

The DWD problem can be described in a similar way.

The optimization problem (2.14) can be solved using an iterative algorithm similar to the BDD case. Now the choice of initial values is more challenging because we need to choose $K - 1$ initial directions. We only briefly explain the case with $K = 3$ here. For the Tri-Direction Discrimination (TDD) problem, we need to choose two initial hyperplanes. For each class ("+" or "−") we consider three cases in terms of subclusters:

- All data lie in a single, well defined cluster (labeled as $(+)$ or $(-)$).
- The data in the class lie in exactly two distinct subclusters (labeled as $(+I, +II)$ or $(-I, -II)$).
- The data in the class lie in three or more clusters (labeled as $(+1, +2, +3)$ or $(-1, -2, -3)$), where clusters are appropriately combined when there are more than 3).

We recommend choosing the initial two hyperplanes based on the consideration of the following clustering:

1. Cluster1-1: BDD output for $(+)$ versus $(-)$ using the Cluster1-2 method.
2. Cluster1-2: BDD output for $(+I, +II)$ versus $(-)$ or for $(+)$ versus $(-I, -II)$ using Cluster1-1 method.
3. Cluster1-3: get BDD output for all pairwise classification problems of the form $(\pm)$ versus $(\mp i, \mp j)$ for $i, j = 1, 2, 3$ using the Cluster1-2 method and choose the one which gives the lowest TDD objective function value.
4. Cluster2-2a: BDD output for $(+I, +II)$ versus $(-I, -II)$ using the Cluster2-2 method.
5. Cluster2-2b: get the BDD output for all pairwise classification problems of the form $(\pm I)$ versus $(\mp I, \mp II)$ using the Cluster1-2 method and choose the one which gives the lowest TDD objective function value.
6. Cluster2-3: get the BDD output for all such classification problems as $(\pm I, \pm II)$ versus $(\mp i, \mp j)$ for $i, j = 1, 2, 3$ using the Cluster2-2 method and choose the one which gives the lowest TDD objective function value.
7. Cluster3-3: get the BDD output for all pairwise classification problems of the form $(\pm i, \pm j)$ versus $(\mp i', \mp j')$ for $i, j, i', j' = 1, 2, 3$ using the Cluster2-2 method and choose the one which gives the lowest TDD objective function value.

As in Section 2.2.3, the finally selected method of TDD is that which minimizes the objective value. In Section 2.3.3, we will demonstrate TDD using two simulated examples.

2.3 Visualization, Simulation and Data Analysis

In this section, we investigate the performance of the proposed method using both simulated and real data. We apply our BDD method to simulated low dimensional data

sets in Section 2.3.1 and then to simulated high dimensional data sets in Section 2.3.2. We apply our TDD method to simulated data sets in Section 2.3.3. The application of the BDD method to two real data sets is discussed in Section 2.3.4. In Sections 2.3.1, 2.3.2 and 2.3.3, we set the sample sizes of training and test data as 100 and 1000, respectively. We generated the test data from the same distributions as the training data. Both the DWD and SVM methods were used in the numerical calculations and their results were quite similar. Due to space limitations, only DWD results are reported here.

A simple recommendation for the choice of the 1DWD tuning parameter was made in Marron et al. (2007) as $C_{1DWD} = 100/d_t^2$, where d_t is the median of the pairwise between class Euclidean distances. This simple default value of the tuning parameter is implemented by most users and has been shown to work well (Qiao et al. (2010)). For BDD we found this simple default approach to tuning was not as reliable as in the case of 1DWD. It was adequate in our simulated example, so we used it to reduce the computational burden in Sections 2.3.1 and 2.3.2. However, it gave an inferior result for the real data sets in Section 2.3.4, so we use cross-validation (CV) there, and recommend this in general.

2.3.1 Simulated Low Dimensional Examples

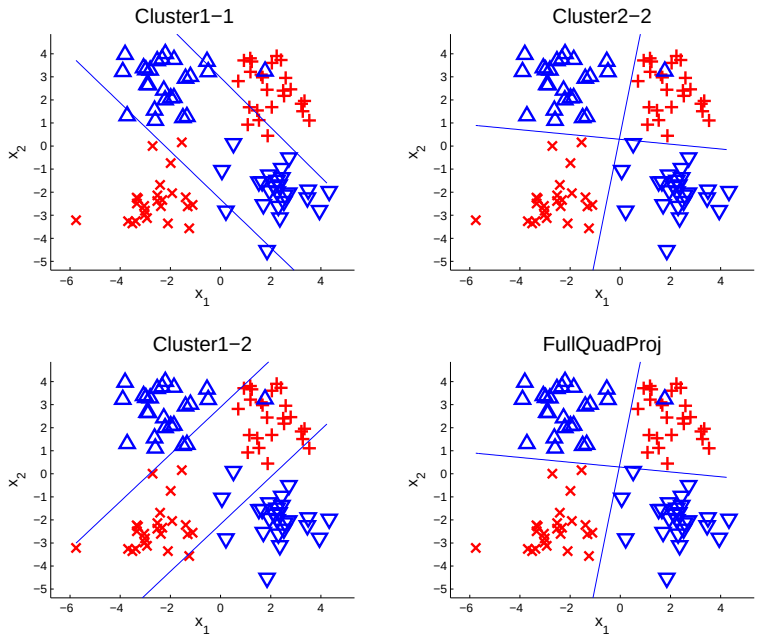
We consider three two-dimensional simulated examples. The simulation setting here is identical to that of Section 2.2.3.1 . We apply the iterative two-direction DWD algorithms described in Section 2.2.3 to each data set. The four different initial hyperplane options (Cluster1-1, Cluster2-2, Cluster1-2 and FullQuadProj) considered in Sections 2.2.3.2 and 2.2.3.3 are used. The combined BDD solution is determined from the one that gives the minimum objective function value among the four options. Let COMBO represent this combined BDD approach which is defined as

$$\text{COMBO} \quad = \quad \text{argmin}\{\text{OBJ(Cluster1-1)},\text{OBJ(Cluster2-2)},$$

OBJ(Cluster1-2),OBJ(FullQuadProj)}.

The data and the resulting two-direction hyperplanes using the four initializations are plotted in the left and middle panels of Figure 2.5 for the 4-Cluster-Twisted example. For comparison, we also randomly simulated 1000 initial values and compute the objective function values based on each of them. The calculated objective function takes only 3 values, with frequencies shown in the table in the right panel of Figure 2.5. These 3 values correspond to three local optimal solutions which correspond to the three distinct decision boundaries shown in the left and middle panels of Figure 2.5. The Cluster2-2 and FullQuadProj methods give the same solution which corresponds to the objective value 59.3. The solutions from the Cluster1-1 and Cluster1-2 methods correspond to the other objective values 157.9 and 162.0. These two solutions are both driven by combining pairs of subgroups into a single group as their corresponding objective values are similar. The third possible way of combining subgroups (chosen by Cluster2-2 and FullQuadProj) is better, as indicated by the much smaller objective value of 59.3. It is important to note that for most of the simulated realizations, the four starting options all choose the global optimal solution. This is consistent with the high frequency of globally optimal solution shown in the table in the right panel of Figure 2.5. The realization shown in Figure 2.5 was carefully culled from the whole collection to display all three types of local optima.

The visualization results for the other two examples are quite similar and will not be shown here. To further investigate the initial value dependence of our BDD method in low dimensional situations, we replicate each example 100 times. The average performance over 100 replications of the proposed BDD methods applied to the three toy examples of Section 2.2.3.1 are exhibited in Table 2.1. The three main blocks summarize results for the three underlying distributions. For each example, a simulated approximation, based on 1000 test data points, of the Bayes Error indicates the difficulties of the corresponding



OBJ values	REL FREQ
59.3	75.3%
157.9	10.7%
162.0	14.0%

Figure 2.5: Application to 4-Cluster-Twisted type of two-dimensional simulated data set. This realization was carefully chosen to show both types of local optima (left panel) and the global optimum (central panel). Observed objective values and their relative frequencies based on 1000 random starts are shown in the table (right panel).

classification problem. For each example, the classification methods are assessed in terms of Training Error, Test Error (based on sets of 1000 test points as above) and the value of the objective functions. For comparison, we include in Table 2.1 the results calculated for 1DWD. Table 2.1 also includes the COMBO results using the initial values which give the minimal objective function values among the four proposed options.

Note that not all data sets are separable so that the training errors are not zero for all methods. For the 4-Cluster-Twisted example, as expected, the Cluster2-2 method performs the best among the three clustering based initialization methods. The FullQuadProj method gives the same performance. The Cluster1-2 method is substantially worse but it is still much better than the conventional 1DWD method. For the 4-Cluster-Straight example, the Cluster1-1 method performs the best among the three clustering based methods. The performances of the Cluster2-2 and FullQuadProj methods are slightly worse. The worst one is the cluster1-2 method. The 1DWD method, as expected, works very

Table 2.1: Performance summary, average error rates over 100 simulations, of the application of the one-direction and the two-direction classification methods to three two-dimensional simulation examples. The numbers in the parentheses show standard errors.

		4-Cluster-Twisted		Bayes Error = 0.025(0.0005)		
	1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
Training	0.350	0.017	0.065	0.026	0.017	0.017
	(0.008)	(0.0015)	(0.0023)	(0.0027)	(0.0015)	(0.0015)
Test	0.36	0.030	0.085	0.039	0.030	0.030
	(0.008)	(0.0006)	(0.0009)	(0.002)	(0.0006)	(0.0006)
Objective		51.6	160	69.9	51.6	51.6
		(1.43)	(1.14)	(4.43)	(1.43)	(1.43)
		4-Cluster-Straight		Bayes Error = 0.013(0.0004)		
	1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
Training	0.011	0.015	0.050	0.011	0.017	0.011
	(0.001)	(0.002)	(0.0036)	(0.001)	(0.0022)	(0.001)
Test	0.014	0.018	0.065	0.014	0.022	0.014
	(0.0004)	(0.0018)	(0.004)	(0.0004)	(0.0025)	(0.0004)
Objective		66.56	108.1	62.9	70.1	62.9
		(1.74)	(3.67)	(0.71)	(2.31)	(0.71)
		3-Cluster-Triangle		Bayes Error = 0.077(0.0008)		
	1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
Training	0.133	0.073	0.073	0.127	0.074	0.073
	(0.0031)	(0.0029)	(0.0029)	(0.0032)	(0.0031)	(0.0029)
Test	0.137	0.092	0.092	0.134	0.093	0.0092
	(0.0012)	(0.001)	(0.001)	(0.0014)	(0.0012)	(0.001)
Objective		157.5	157.5	246.6	159.6	157.5
		(4.09)	(4.09)	(4.06)	(4.34)	(4.09)

well in this example, but Cluster1-1 and COMBO give the same strong performance. For the 3-Cluster-Triangle example, the Cluster1-2 method performs the best among the three clustering based methods. The Cluster2-2 method is the same and the FullQuadProj method is slightly worse. The worst one is Cluster1-1 which is still better than the 1DWD method. Here the performances are measured in terms of the criteria of small objective function value and small test error. For all three examples, the COMBO method, based only on objective values, always chooses the best one among the four initialization methods. The FullQuadProj method gives performances which are fairly comparable to the best among the three clustering based methods which reflects the fact that in low dimensional situations, nonlinear classifiers easily adapt to complex data structure.

2.3.2 Simulated High Dimensional Examples

Consider a typical HDLSS context. Let $d = 1000$. We simulated three types of examples. The first two dimensions are generated using distributions similar to the three $2d$ examples described in Section 2.3.1. We maintain an appropriate signal to noise ratio by taking $\mu = \sqrt{d}/8$ here instead of the constant $\mu = \sqrt{5}$. The rest of the $d - 2$ dimensions are pure noise, i.e., all sampled from the standard normal distribution.

The visualization results of the simulated training data are shown in the plots in Figure 2.6 and Figure 2.7 for the 4-Cluster-Twisted and 3-Cluster-Triangle high dimensional examples, respectively. The visualization results for the 4-Cluster-Straight example is similar to that for the 4-Cluster-Twisted example and thus is not shown here. The plots in the upper left panels show the data in the original co-ordinates in the first two directions. The plots in the lower left panel visualize the projections of the data points onto the 1DWD and orthogonal PC1 directions. The plots in the middle and right panels visualize the projections of the data points onto the two direction vectors based on the 4 initializations from our BDD method with the x-axis representing $f_{1i} = \mathbf{x}_i^T \mathbf{w}_1 + \beta_1$ and the y-axis representing

$f_{2i} = \mathbf{x}_i^T \mathbf{w}_2 + \beta_2$. Note that all of the training data sets are separable and the training errors are equal to zero under both the 1DWD method and the BDD methods for all four initial value options. This is a common property of HDLSS data where there is potential for *overfitting*, since one can almost always find separating hyperplanes to correctly classify all training data points. This is consistent with the idea of Hall et al. (2005), which was explicitly stated in Ahn and Marron (2005) that if the underlying distribution of the data is continuous in the data space and the dimension is larger than the sample size, the data are separable with probability one. A central issue to the statistical analysis of HDLSS data is whether observed separations represent important and reproducible underlying structure, or are spurious artifacts of the sampling variation, i.e., the result of overfitting. This will be studied in detail in Table 2.2

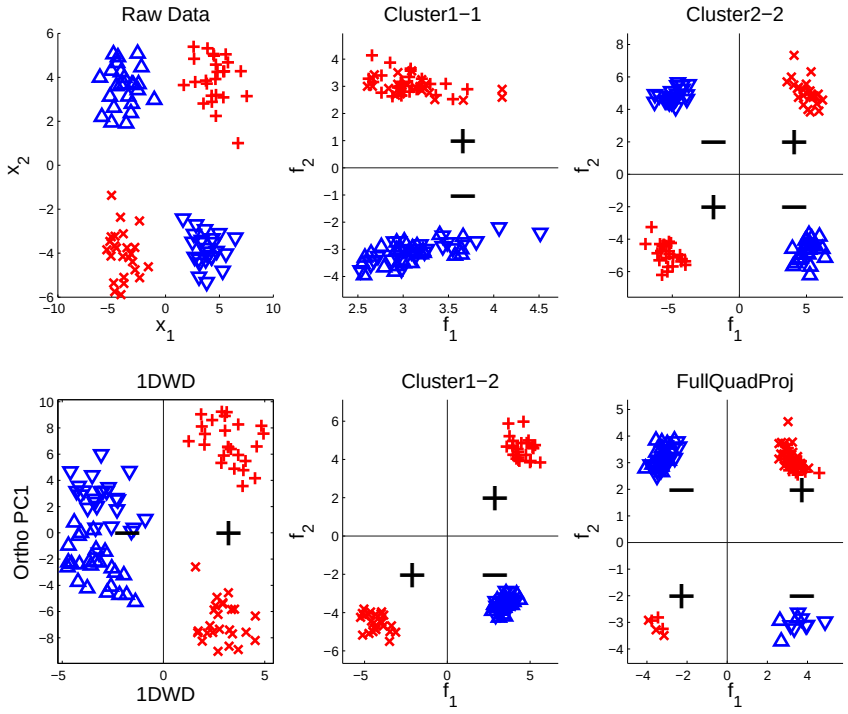


Figure 2.6: Application to 4-Cluster-Twisted type of high dimensional simulated data set. Upper left panel shows the raw data projected onto the first two directions. Projections onto 1DWD and orthogonal PC1 directions are shown in the lower left panel. Projections onto f_1, f_2 directions are shown in the middle and right panels.

For the 4-Cluster-Twisted example (Figure 2.6), Cluster2-2 seems to find the right

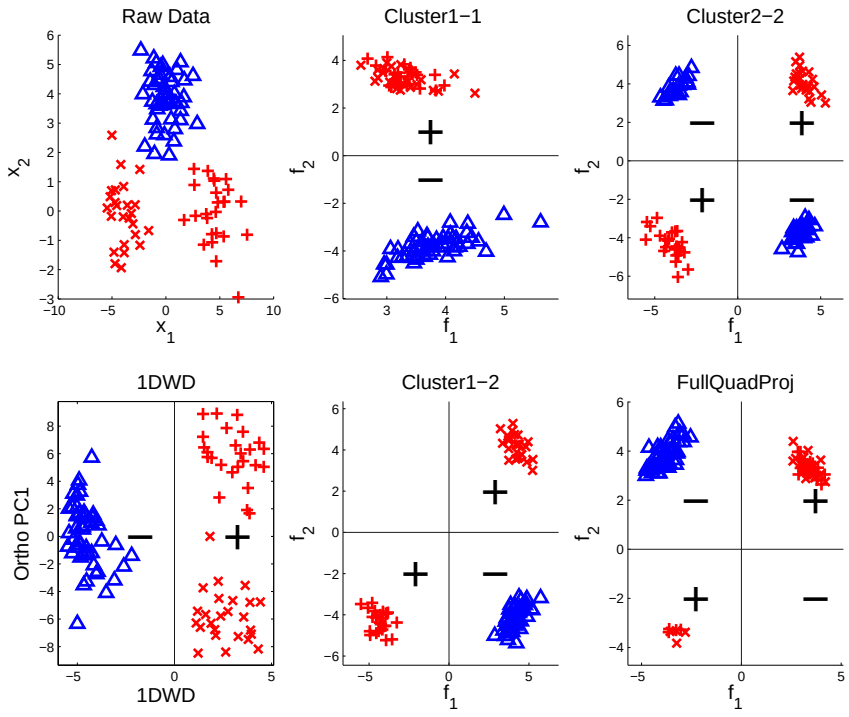


Figure 2.7: Application to 3-Cluster-Triangle type of high dimensional simulated data set. Upper left panel shows the raw data projected onto the first two directions. Projections onto 1DWD and orthogonal PC1 directions are shown in the lower left panel. Projections onto f_1, f_2 directions are shown in the middle and right panels.

structure. The combination of 1DWD and orthogonal PC1 directions can separate the 4 clusters but the structure is twisted in contrast to the original one as shown in the upper left plots. No structure of this type is part of the underlying signal in the data, so we conclude that it is a noise artifact. All the other three BDD methods exhibit artifacts that suggest overfitting. Cluster1-2 attempts to divide the data into three clusters and gives apparently reasonable separation of the negative class into two clusters. Cluster1-1 attempts to divide the data into only two clusters. Even Cluster2-2 may be affected by overfitting, as the clusters are better separated than in the raw data.

The visualization for the 3-Cluster-Triangle example (Figure 2.7) suggests that the Cluster1-2 method works the best for this example because it correctly divides the data into three clusters although the clusters are better separated than in the raw data. All the other three BDD methods fail to correctly find the cluster identification. Cluster2-2 divides the data into four clusters which seems appropriate, but are likely to lead to some loss of generalization ability.

The visualization results in Figures 2.6-2.7 can only provide some partial ideas about the performances of different methods in each example. To analyze the pivotal question of which methods have found reproducible structure in the data, we repeat the simulation 100 times. The performance summaries over 100 replications are listed in Table 2.2. As mentioned before, for HDLSS data, the training errors are frequently zero for all methods and thus are not included in this table. For the 4-Cluster-Twisted example, not surprisingly, the Cluster2-2 method works the best. The Cluster1-1 method has no power. The performances of the Cluster1-2 and the FullQuadProj methods are in-between. For the 4-Cluster-Straight example, the Cluster1-1 method works the best as expected and the Cluster2-2 method has no power. For the 3-Cluster-Triangle example, the Cluster1-2 method works the best as expected. For all three examples, the COMBO method always chooses the best one among the four initialization methods. On the other hand, the normal

1DWD method can achieve the best performance only in the 4-Cluster-Straight example. It has no power in the 4-Cluster-Twisted example and gives moderate performance in the 3-Cluster-Triangle example.

Note that the FullQuadProj method typically did not achieve the best performance due to a tendency towards overfitting as shown in Figures 2.6 and 2.7. As mentioned before, the FullQuadProj method works in a space with much higher dimension than the original one and thus is very prone to overfitting. The consequence of overfitting is that small changes in training data can have significant influence on the outcome of the prediction for test data and models will have high classification error. Comparing the objective values for the 4-Cluster-Twisted example shown in Table 2.2 with the corresponding KDE plot in Figure 2.3, we note that random choice of the initial values can hardly achieve good optimization for HDLSS data. The reason is that the optimization problems considered here have many more local solutions in the high-dimensional situation than in the low-dimensional situation.

There is one subtle point we want to mention here. For the Cluster1-1 method, as shown in the upper left panel of Figures 2.6 and 2.7, only one hyperplane of the BDD solution has an impact on the classification, because all the data points are located on the same side of the other hyperplane. Thus in the case when one-direction methods work well, the inclusion of the second hyperplane can make the performance worse. This is illustrated in Figure 2.8 where the Cluster1-1 method is applied to the 4-Cluster-Straight example for both training data (left panel) and test data (right panel). The left panel shows that all training data points are on one side of the second hyperplane (the vertical line) of our BDD solution. However, this is no longer true for the test data as shown (on the same scale of axis) in the right panel, where a quite large test error is seen. Figure 2.8 suggests that the inclusion of the second hyperplane can greatly increase the test error although it does not affect the training points. We address this issue as follows. When the Cluster1-1

Table 2.2: Performance summary, average error rates over 100 simulations, of the application of the one-direction and the two-direction classification methods to three high-dimensional simulation examples. The numbers in the parentheses show standard error.

	4-Cluster-Twisted		Bayes Error = 5×10^{-5} (2×10^{-5})			
	1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
Test	0.502	0.0024	0.219	0.501	0.397	0.0024
	(0.0016)	(1.7×10^{-4})	(0.0012)	(0.0015)	(0.0089)	(1.7×10^{-4})
Objective		4.06	6.59	10.01	9.25	4.06
		(0.01)	(0.015)	(0.025)	(0.067)	(0.01)
	4-Cluster-Straight		Bayes Error = $1 \times 10^{-5}(1 \times 10^{-5})$			
	1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
Test	0.0012	0.503	0.439	0.0012	0.394	0.0012
	(1.1×10^{-4})	(0.0016)	(0.0016)	(1.1×10^{-4})	(0.0028)	(1.1×10^{-4})
Objective		6.54	6.29	6.23	6.79	6.23
		(0.019)	(0.014)	(0.015)	(0.063)	(0.015)
	3-Cluster-Triangle		Bayes Error = 3.9×10^{-3} (2×10^{-4})			
	1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
Test	0.149	0.26	0.061	0.264	0.304	0.061
	(0.0014)	(0.0029)	(0.001)	(0.0013)	(0.001)	
Objective		6.85	5.80	7.73	8.18	5.80
		(0.018)	(0.016)	(0.022)	(0.081)	(0.016)

method gives the lowest objective value for a data set (e.g., 4-Cluster-Straight example) and all training data are on the same side of one of the two hyperplanes, we choose the first hyperplane to be the usual one-direction separating hyperplane and the second one to be $\mathbf{w}_2 = \mathbf{0}$ and $\beta_2 = 1$. This solution is equivalent to the normal one-direction solution and the second hyperplane does not play any role in the classification.

2.3.3 Simulated Tri-Directional Examples

Two examples are shown here to further demonstrate the usefulness of our TDD method. The first one is called the Linear 4-Cluster Gaussian mixture toy example in which our two-direction BDD may not give major improvements over linear methods. As shown in Figure 2.9 below, linear methods tend to miss the 4 cluster structure of the data. While BDD looks better visually, in fact the test error rate (misclassification error rate on independently generated test data) is no better (both are around 25%). However, our

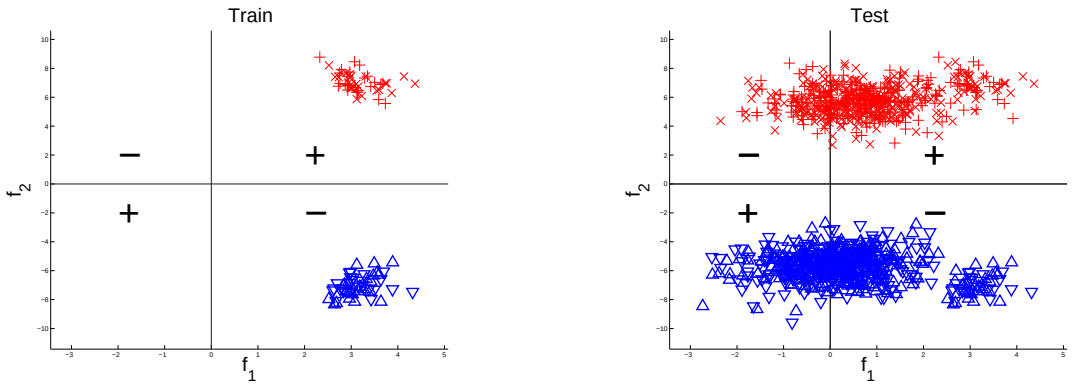


Figure 2.8: Visualization of a 4-Cluster-Straight example using Cluster1-1 initialization for both training (left) and test (right) data.

TDD works much better in this case, by correctly accounting for the 4 cluster nature of the data. Note that the Cluster2-2b initialization method was chosen by the objective function minimization for the TDD solution.

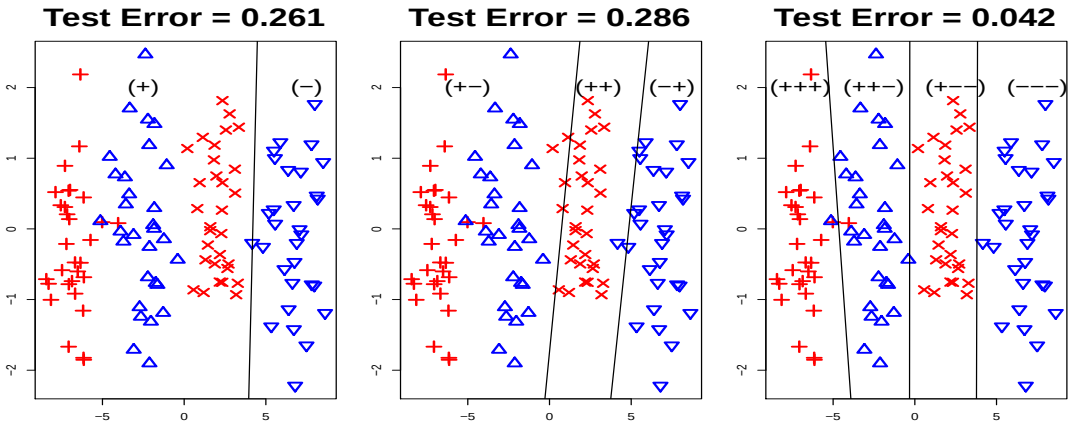


Figure 2.9: Classification results for the Linear 4-Cluster Gaussian mixture example: The positive class is a mixture of $N(-7.5, 1)$ and $N(2.5, 1)$ denoted by "+" and "x" symbols respectively and the negative class is a mixture of $N(-2.5, 1)$ and $N(7.5, 1)$ denoted by triangles. The left panel is the classification boundary obtained by 1SVM; the middle panel shows the classification boundary obtained by BDD; the right panel shows the classification boundary obtained by TDD. The error rates show that the one-directional method and BDD deliver similar performance while TDD works the best for this example.

The second one is the Donut example, shown in Figure 2.10, which is a typical example to demonstrate the use of full kernel methods in low dimensional problems. As we can

see from Figure 2.10, BDD offers significant improvement over linear methods and TDD performs even better. Interestingly, the TDD can yield similar performance as a full nonparametric kernel method by using only three directions. Note that the Cluster1-3 initialization method was chosen by the objective function minimization for the TDD solution.

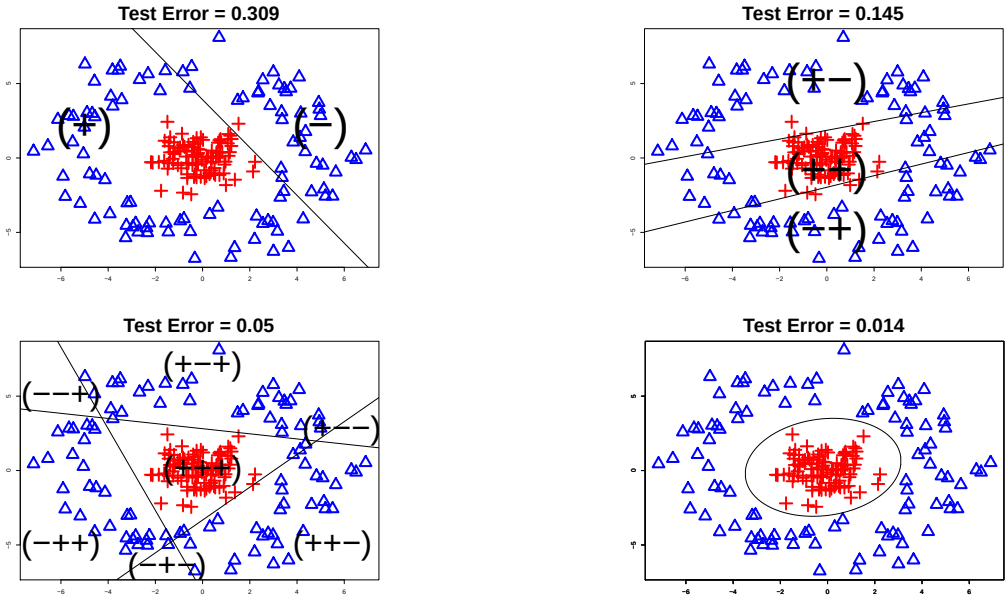


Figure 2.10: Classification results for the donut example. The positive class, denoted by "+" symbol, lies within a small center, the negative class, denoted by triangle, surrounds this entirely. The top left, top right, bottom left, bottom right display the classification boundaries by 1SVM, BDD, TDD, and the full quadratic-kernel SVM, respectively. Note that BDD offers improvement over the one-directional method (the error rate changes from 31% to 15%), and TDD further improves BDD(error rate changes from 15% to 5%). Interestingly, TDD gives performance that is not far from that of the full quadratic-kernel SVM although it only uses three directions.

2.3.4 Real Data

In this section, we apply our BDD method to two real data sets. We analyze a Lung Carcinoma data set in Section 2.3.4.1 and a Glioblastoma Multiforme data set in Section 2.3.4.2. The first example includes four clusters, two for each class. The subcluster labels

for each class are known and our method can identify them correctly. In the second example we apply our method to the two class problem without knowing whether there are subpopulations within each class or not, our method finds two distinct subclusters within the class Neural which are worth further biological investigation.

2.3.4.1 Lung Carcinoma Data

In this section we show the performance of the BDD method on the Human Lung Carcinoma Microarray Data set (available from http://www.broad.mit.edu/lung/). Here, we focus the analysis on four unambiguous histological types for which there is little diagnostic confusion: normal lung, pulmonary carcinoid tumors, colon cancer metastases, and small cell carcinoma. These samples have been described in detail previously (Bhattacharjee et al. (2001); Meyerson and Hayes (2005)) and are analyzed by Liu et al. (2008). The original data contain 12,625 genes. After filtering the genes using the ratio of the sample standard deviation and sample mean of each gene, 2,530 genes with large ratios are kept in the data set, as in Liu et al. (2008). The data set contains 51 patients with 2,530 genes. Among the 51 samples, there are 20 pulmonary carcinoid samples (Carcinoid), 8 colon cancer metastases (Colon), 17 normal lung samples (Normal) and 6 small cell carcinoma samples (SmallCell).

We considered several combinations of the data, and the most interesting was the classification problem which treats Normal & SmallCell as the positive class and Carcinoid & Colon as the negative class. The four methods of choosing initial values considered in this dissertation are applied to this data set for the calculation of the BDD decision boundaries. Similar visualization results to those of Section 2.3.2, based on the default tuning parameter, are shown in Figure 2.11.

From the visualization results, the Cluster2-2 method performs much better than the

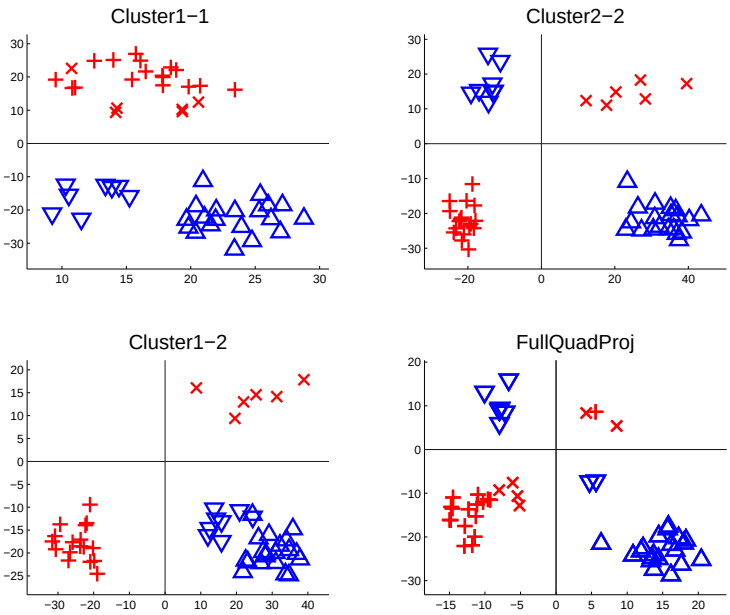


Figure 2.11: Application to the human lung carcinoma microarray data set: Normal (red "+") + SmallCell (red "x") versus Carcinoid (blue up-triangle) + Colon (blue down-triangle). Note Cluster2-2 method correctly subdivide the classes.

other three methods, in the sense that it correctly split the four distinct types of data (Normal, SmallCell, Carcinoid and Colon) into four clusters.

The visualization results have suggested some interesting structures for the data set but it is also important to study the generalizability of each method. Due to the limited sample size, we study the generalization properties of our method using CV. The data set is randomly split into a training set (80%) and a test set (20%). We further split the training set (80% and 20%) to give 5-fold CV for tuning parameter selection. This division of the training data is randomly repeated 100 times and the tuning parameter is chosen to be the one which gives the lowest average CV error. The test error is calculated based on this parameter.

The CV errors for the classification problem are listed in Table 2.3. The CV errors are computed on the basis of 100 random splits of the data set, and the means are summarized in the table. The Monte Carlo standard errors over 100 splits are included in the paren-

Table 2.3: Cross validation errors over 100 replications for the human lung carcinoma microarray data set. The numbers in the parentheses show standard errors.

1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
1.2(0.33)	0(0)	1.4(0.38)	2.8(0.53)	1.7(0.4)	0.3(0.22)

theses in the table. All methods give relatively low CV errors due to the nature of this problem. The Cluster2-2 method gives the best performance among the four initialization methods. The CV error from the usual 1DWD method is higher than that from our BDD method. This is consistent with the visualization results shown in Figure 2.11.

2.3.4.2 Glioblastoma Multiforme Data

Glioblastoma Multiforme (GBM) is the most common form of malignant brain cancer in adults. For the purposes of the current analysis, we selected a cohort of patients with GBM cancer whose brain samples were assayed on three gene expression platforms (Affymetrix HuEx array, Affymetrix U133A array, and Agilent 244K array) into a single unified data set. Several clinical relevant subtypes were identified using integrated genomic analysis in Verhaak et al. (2010). For our analysis example, we focused on Mesenchymal and Neural subtypes because there was some feeling that the Neural might really have two subclasses. After filtering the genes using the ratio of the sample standard deviation and sample mean of each gene, the data set contains 186 patients with 2,727 genes. Among the 186 samples, there are 117 Mesenchymal samples (MES) and 69 Neural samples (NL).

We consider the classification problem which treats MES as the positive class and NL as the negative class. Similar to the first example, the visualization plots and the CV errors are shown in Figure 2.12 and Table 2.4 respectively. As shown in the visualization plots, Cluster2-2 method tends to split both classes into two subclusters and Cluster1-2 method tends to split the NL class into two subclusters. CV errors show that Cluster1-2 method gives the best performance although the difference between Cluster1-2 method and

Table 2.4: Cross validation errors for GBM data MES versus NL

1DWD	Cluster2-2	Cluster1-2	Cluster1-1	FullQuadProj	COMBO
3.00(0.28)	8.97(0.59)	2.92(0.26)	3.00(0.28)	4.19(0.48)	3.05(0.28)

1DWD method is not so significant. Note that the CV error from the COMBO method is slightly bigger than the ones from the Cluster1-1 method and the Cluster1-2 method. This is because a lower objective value does not always correspond to a lower error rate when we compare two methods with slightly different objective values. Therefore, our CV analysis for every individual method is very important here and confirms that the NL class has two distinct subclusters. To verify whether or not these clusters represent potentially important new cancer subclasses, the SigClust method (Liu et al. (2008)) was performed to evaluate the significance of this cluster splitting of the NL class, i.e., as given by our Cluster1-2 method as shown in the bottom left panel of Figure 2.12. The resulting p-value is very significant at 1.12×10^{-17}. Therefore, we conclude that there are further subclusters within the NL samples that are worth deeper biological investigation.

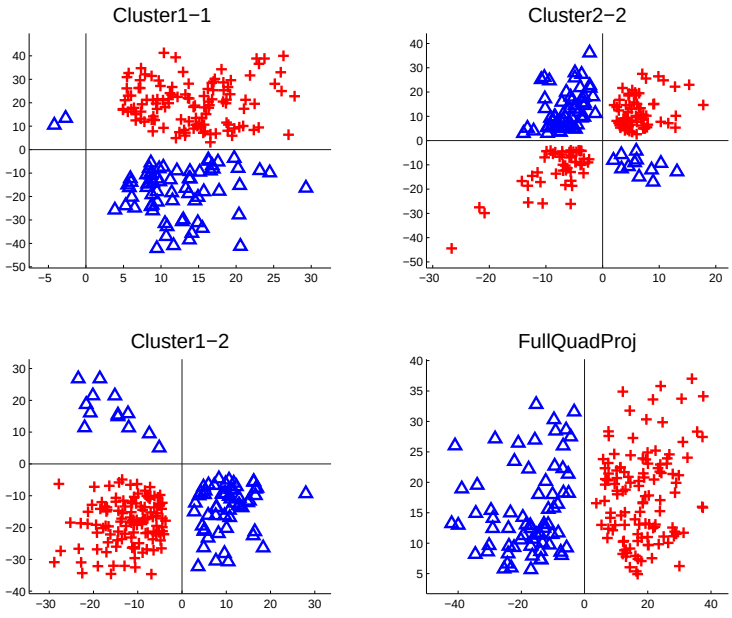


Figure 2.12: Application to GBM data set: MES (red "+" sign) versus NL (blue triangle).

Two subsets of genes were selected based on the 200 biggest absolute coefficient values of the normal direction vectors solved from 1DWD and BDD Cluster1-2 methods. We found 51 common genes in both groups. The heatmaps of the two subsets are shown in Figure 2.13. The left panel is for the genes selected from 1DWD method and shows two distinct clusters, one for each class. The right panel is for the genes selected from BDD Cluster1-2 method and shows three distinct clusters, one for MES and two for NL. This is consistent with the visualization result shown in the lower left panel in Figure 2.12.

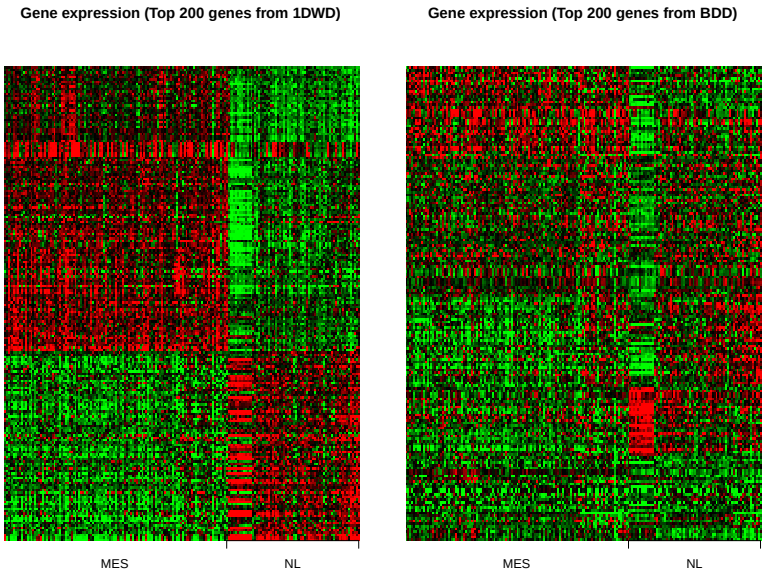


Figure 2.13: Heatmap of GBM data by using top 200 genes selected from 1DWD methods (left panel) and BDD Cluster1-2 methods (right panel). Genes are in the rows and samples are in the columns

2.4 Mathematical Statistics

To gain further insight into BDD, in this section we study some of its theoretical properties. A parallel theory can be developed for TDD, but the main ideas are seen most directly through examples of the computationally simpler BDD.

We could investigate classical asymptotics in the limit as $n \to \infty$. But because the main

value of BDD is in the HDLSS case, we consider asymptotics of the method for $d \to \infty$ with the sample size n fixed. Hall et al. (2005) first demonstrated the mathematical statistical insight available from this type of asymptotics. They showed that, under some conditions, each data point in a sample of size n tends to lie near a vertex of a regular n-simplex and all the randomness in the data appears in the form of a random rotation of this simplex, i.e., they found a geometric representation for HDLSS data. This data structure yield new insight into the binary classification problem. In practice, data points from the positive class (size n^+) and those from the negative class (size n^-) can be viewed as an n^+-simplex and an n^--simplex respectively. This gave direct results on completely perfect and completely imperfect classifications.

The regularity conditions for the geometric representation in Hall et al. (2005) requires that the entries of the data vector satisfy a ρ-mixing condition. Ahn et al. (2007) gave a milder condition using asymptotic properties of the sample covariance. A more general and even milder set of conditions for the result of Hall et al. (2005) is given in Jung and Marron (2009); Qiao et al. (2010).

Here we obtain new statistical insight as to the superior properties of BDD, using this HDLSS geometry. To illustrate the important principles that underlie BDD, in the binary classification problem, we consider two examples. The first example is discussed in Section 2.4.1 which includes four clusters, two for each class. The second example is discussed in Section 2.4.2 which includes three clusters, two for the positive class and one for the negative class.

2.4.1 Four Clusters Case

The first example includes four clusters labeled as $+1$, $+2$, -1, and -2 respectively. Assume that data points from clusters $+1$ and $+2$ belong to the positive class and those

from clusters -1 and -2 belong to the negative class.

We use the regularity conditions of Qiao et al. (2010). Consider the $+1$ cluster consisting of data vectors $\mathbf{x}_1^{+1}(d), \cdots, \mathbf{x}_{n_{+1}}^{+1}(d)$ with d variables, where $\mathbf{x}_j^{+1}(d) = (x_{1j}^{+1}, \cdots, x_{dj}^{+1})^T \in R^d$, $j = 1, \cdots, n_{+1}$. Assume these vectors are independent and identically distributed from a d-dimensional multivariate distribution. Concatenate these into a $d \times n_{+1}$ data matrix $X_d^{+1} = [\mathbf{x}_1^{+1}(d), \cdots, \mathbf{x}_{n_{+1}}^{+1}(d))]$ with $d > n_{+1}$.

For a fixed n_{+1}, consider a sequence of random data matrices $X_1^{+1}, \cdots, X_d^{+1}, \cdots$, indexed by the number of rows d. Assume each X_d^{+1} comes from a d-dimensional multivariate distribution with covariance matrix Σ_d^{+1}. Let $\lambda_{1,d}^{+1} \geq \cdots \geq \lambda_{d,d}^{+1}$ be the eigenvalues of Σ_d^{+1}. Assume the following:

(1) The fourth moments of each entry of each column of X_d^{+1} are uniformly bounded.

(2) The entries of $Z_d^{+1} = (\Sigma_d^{+1})^{-\frac{1}{2}} X_d^{+1}$ are independent.

(3) The eigenvalues of Σ_d^{+1} are sufficiently diffused, in the sense that

$$\epsilon_d^{+1} = \frac{\sum_{j=1}^d (\lambda_{j,d}^{+1})^2}{(\sum_{j=1}^d \lambda_{j,d}^{+1})^2} \to 0 \; as \; d \to \infty. \tag{2.15}$$

(4) The sum of the eigenvalues of $\Sigma_{+1,d}$ is of the same order as d, in the sense that $\sum_{j=1}^d \lambda_{j,d}^{+1} = O(d)$ and $1/\sum_{j=1}^d \lambda_{j,d}^{+1} = O(1/d)$.

Define the scaled variance $(\sigma_d^{+1})^2 = \frac{1}{d}\sum_{j=1}^d \lambda_{j,d}^{+1}$. Under Assumptions (1)-(4), as $d \to \infty$, these n_{+1} data vectors tend to form a regular n_{+1}-simplex in R^d with side length $\sqrt{2}d\sigma_d^{+1}$. Assume that the other three independent data samples $X^{+2}(d)$, $X^{-1}(d)$ and $X^{-2}(d)$ also satisfy Assumptions (1)-(4). Define σ_d^{+2}, σ_d^{-1} and σ_d^{-2} similarly to σ_d^{+1}. Then the four clusters can be viewed asymptotically as four simplices within the d-dimensional space having side lengths $\sqrt{2}d\sigma_d^{+1}$, $\sqrt{2}d\sigma_d^{+2}$, $\sqrt{2}d\sigma_d^{-1}$, and $\sqrt{2}d\sigma_d^{-2}$ respectively. Based on this geometric representation, our next goal is to develop conditions under which the two-

direction SVM (DWD) method is better than the usual 1SVM (1DWD) method.

In general, the population mean positions of the four clusters lie in a 3-dimensional hyperplane in R^d. They are located at the vertices of a tetrahedron. In order to illustrate the basic idea of when the two-direction method is preferred, we consider a simple setting as shown in Figure 2.14. Given two sequences of between-class distances $l_{+,d} \geq 0, l_{-,d} \geq 0$ and a sequence of within-class distances $l_{0,d} \geq 0$, assume that

(5) The mean positions of the four clusters $+1$, $+2$, -1, and -2 in the 3-dimensional space are $C_{+1,d} = (l_{+,d}/2, 0, l_{0,d}/2)$,$C_{+2,d} = (-l_{+,d}/2, 0, l_{0,d}/2)$, $C_{-1,d} = (0, l_{-,d}/2, -l_{0,d}/2)$, $C_{-2,d} = (0, -l_{-,d}/2, -l_{0,d}/2)$ respectively.

These mean positions can also be parametrized in terms of variance shifts and rotations but this form we used to make the main idea most clear. Note that the geometries of this setting are fully characterized by three sequences $l_{+,d}, l_{-,d}$ and $l_{0,d}$. Here $l_{+,d}$ is equal to the length of the line segment $C_{+1,d}C_{+2,d}$ and $l_{-,d}$ is equal to the length of line segment $C_{-1,d}C_{-2,d}$, and thus can be viewed as notions of within-class distances. Similarly, $l_{0,d}$ is equal to the distance between the line $C_{+1,d}C_{+2,d}$ and the line $C_{+1,d}C_{+2,d}$, and thus can be viewed as a notion of between-class distance, i.e., the distance between the positive class and the negative class. Here for simple understanding of the main ideas, we assume that the sample sizes and variances of the two clusters within each class are the same, i.e.

(6) $n_{+1} = n_{+2} = n_+/2$, $n_{-1} = n_{-2} = n_-/2$.

(7) For given constants $\sigma_+, \sigma_- > 0$, $\sigma_{+1,d} = \sigma_{+2,d} = \sigma_{+,d} \to \sigma_+$, $\sigma_{-1,d} = \sigma_{-2,d} = \sigma_{-,d} \to \sigma_-$, as $d \to \infty$.

For some distance orders $\alpha_\pm \geq 0$ and $\alpha_0 \geq 0$, we study the asymptotic behaviors of the one-direction and the two-direction classifiers as the within-class distances $l_{\pm,d}$ grow at the rate of $d^{\alpha_\pm}$ and the between-class distance $l_{0,d}$ grows at the rate of d^{α_0}, in the sense

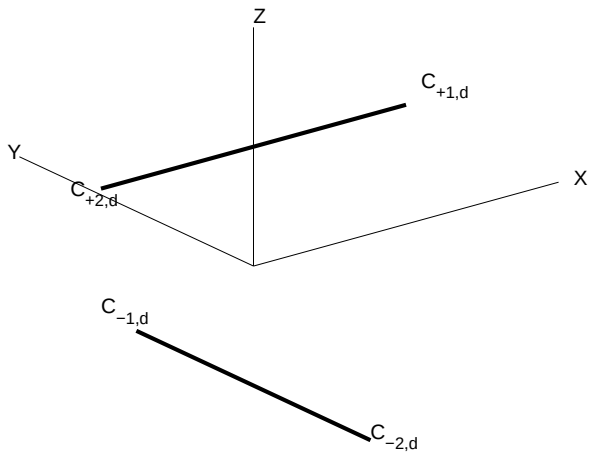


Figure 2.14: Illustration of the mean positions ($C_{+1,d}$,$C_{+2,d}$,$C_{-1,d}$,$C_{-2,d}$) of the four clusters, where ($C_{+1,d}$,$C_{+2,d}$) belong to the positive class and ($C_{-1,d}$,$C_{-2,d}$) belong to the negative class.

that $\frac{l_{\pm,d}}{d^{\alpha_\pm}} \to \mu_\pm$ and $\frac{l_{0,d}}{d^{\alpha_0}} \to \mu_0$ for some $\mu_\pm > 0$ and $\mu_0 > 0$. To test the performance of the classification methods, we need to add a new random point to a d-variate space which is independent of the data in $X_d^{+1} \cup X_d^{+2} \cup X_d^{-1} \cup X_d^{-2}$ and has the distribution of any of the four clusters. The following Theorem characterizes the HDLSS asymptotics of the one-direction and the two-direction methods for the unbalanced case under the above setting.

Theorem 2.1. *Assume that $\sigma_+^2/n_+ > \sigma_-^2/n_-$; if need be, interchange $+$ and $-$ to achieve this. Under Assumptions (1)-(7), we have the following results:*

1. *1SVM gives either completely correct or incorrect classification as:*
   (a) *If $lim_{d\to\infty}(\mu_0^2 d^{2\alpha_0-1}) > \sigma_+^2/n_+ - \sigma_-^2/n_-$, then the probability that the usual 1SVM hyperplane gives correct classification of new points converges to 1 as $d \to \infty$.*
   (b) *If $lim_{d\to\infty}(\mu_0^2 d^{2\alpha_0-1}) < \sigma_+^2/n_+ - \sigma_-^2/n_-$, then with probability converging to 1 as $d \to \infty$ a new datum from either population will be classified by the usual 1SVM hyperplane as belonging to the positive population.*
2. *BDD gives completely correct classification as:*

(a) If either $lim_{d\to\infty}(\mu_0^2 d^{2\alpha_0-1}) > \sigma_+^2/n_+ - \sigma_-^2/n_-$ *or* $lim_{d\to\infty}(\mu_\pm d^{\alpha_\pm-\frac{1}{2}}) > 0$*, the probability that the BDD classifier gives correct classification of new points converges to 1 as* $d \to \infty$*.*

Part 1 of Theorem 2.1 says that the 1SVM method gives an asymptotically correct classification of a new point when the between-class distance is large enough, in the sense that either $\alpha_0 > 1/2$ or $\alpha_0 = 1/2$ and $\mu_0^2 > \sigma_+^2/n_+ - \sigma_-^2/n_-$. When the between class distance is small enough in the sense that either $\alpha_0 < 1/2$, or $\alpha_0 = 1/2$ and $\mu_0^2 < \sigma_+^2/n_+ - \sigma_-^2/n_-$, the one-direction method will fail regardless of the size of the within-class distances. Part 2 of Theorem 2.1 shows that the BDD classification works as well as the one-direction method when either $\alpha_0 > 1/2$ or $\alpha_0 = 1/2$ and $\mu_0^2 > \sigma_+^2/n_+ - \sigma_-^2/n_-$. More importantly, the major improvement available from BDD is demonstrated in the result that it will classify correctly when $\alpha_\pm \geq 1/2$, for any value of between-class distance.

The following Theorem characterizes the HDLSS asymptotics of the one-direction and the two-direction methods for the balanced case under the above setting.

Theorem 2.2. *Under Assumptions (1)-(7),, assume that* $\sigma_+ = \sigma_-$ *and* $n_+ = n_-$*, we have the following results:*

1. *If* $\alpha_0 \geq 1/2$*, the probability that the usual 1SVM hyperplane gives correct classification of new points converges to 1 as* $d \to \infty$*. If* $\alpha_0 < 1/2$*, this probability converges to 1/2.*
2. *If either* $\alpha_0 \geq 1/2$ *or else* $\alpha_\pm \geq 1/2$*, the probability that the BDD classifier gives correct classification of new points converges to 1 as* $d \to \infty$*.*

Remark: Under the assumptions of Theorem 2.2, since all four clusters have the same sample sizes, the DWD hyperplanes asymptotically coincide with the corresponding SVM hyperplanes for both one-direction and two-direction cases. Thus all the above conclusions in Theorem 2.2 for SVM methods hold for DWD methods as well.

A summary of how the classification performance of Theorem 2.2 for the one-direction methods and the two-direction methods is driven by the rates $\alpha_{\pm}$ and α_0 is illustrated in Figure 2.15. We divide the set of possible contexts into a "strong classification" part (right side) and a "weak classification" part (left side) according to whether $\alpha_0 \geq 1/2$ or $\alpha_0 < 1/2$. We also divide the set of contexts into a "strong clustering" part (top side) and a "weak clustering" (bottom side) part according to whether $\alpha_{\pm} \geq 1/2$ or $\alpha_{\pm} < 1/2$. The one-direction classification methods work in the "strong classification" areas, i.e., areas (I) and (IV). Our two-direction methods work not only in the "strong classification" area but also in the "strong clustering" area, i.e., areas (I), (II) and (IV). The one-direction methods will fail to classify in the "weak classification" areas. Our two-direction methods are superior because they continue to work in this area when the clustering is strong.

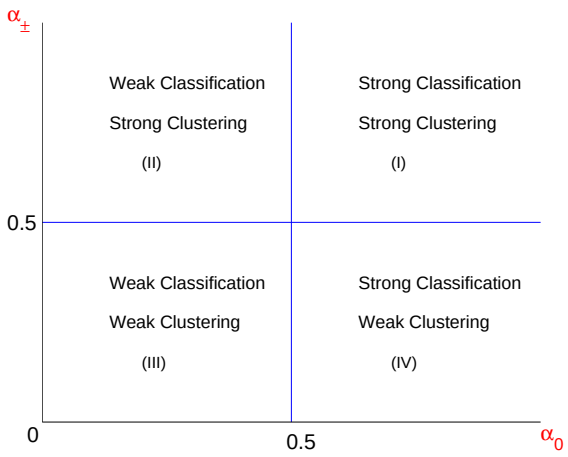


Figure 2.15: Summary of the classification performance given in Theorem 2.2 for the one-direction methods and the two-direction methods.

As a remark, we note that the BDD classification will work well with the Cluster2-2 type of initial value method when $\alpha_{\pm} \geq 1/2$ and will work well with the 1DWD-START type of initial value method when $\alpha_0 \geq 1/2$. Furthermore, our method for choice between starting values will tend to find the correct one.

2.4.2 Three Clusters Case

The second example includes three clusters labeled as $+1$, $+2$, and -1 respectively. Thus only the positive class contains two distinct clusters. Similar to the four clusters case, we consider a simple setting here. Given within-class distances $l_{+,d} \geq 0$ and a between-class distance $l_{0,d} \geq 0$, assume that

(A) The mean positions of the three clusters $+1$, $+2$, -1 in the 2-dimensional space are $C_{+1,d} = (l_{+,d}/2, l_{0,d}/2)$, $C_{+2,d} = (-l_{+,d}/2, l_{0,d}/2)$, $C_{-,d} = (0, -l_{0,d}/2)$ respectively.

(B) $n_{+1} = n_{+2} = n_+/2$.

(C) For a given constant $\sigma_+ > 0$, $\sigma_{+1,d} = \sigma_{+2,d} = \sigma_{+,d} \to \sigma_+$, as $d \to \infty$.

The following Theorem characterizes the HDLSS asymptotics of the one-direction and the two-direction methods under the above setting.

Theorem 2.3. *Assume that $\sigma_+^2/n_+ > \sigma_-^2/n_-$; if need be, interchange $+$ and $-$ to achieve this. Under Assumptions (1)-(4) and assumptions (A)-(C), we have the following results:*

1. *1SVM gives either completely correct or incorrect classification as:*
   (a) *If $lim_{d\to\infty}(\mu_0^2 d^{2\alpha_0 - 1}) > \sigma_+^2/n_+ - \sigma_-^2/n_-$, then the probability that the usual 1SVM hyperplane gives correct classification of new points converges to 1 as $d \to \infty$.*
   (b) *If $lim_{d\to\infty}(\mu_0^2 d^{2\alpha_0 - 1}) < \sigma_+^2/n_+ - \sigma_-^2/n_-$, then with probability converging to 1 as $d \to \infty$ a new datum from either population will be classified by the usual 1SVM hyperplane as belonging to the positive population.*
2. *BDD gives completely correct classification as:*
   (a) *If either $lim_{d\to\infty}(\mu_0^2 d^{2\alpha_0 - 1}) > \sigma_+^2/n_+ - \sigma_-^2/n_-$ or else $lim_{d\to\infty}(\mu_+^2 d^{2\alpha_+ - 1}) > 8\sigma_+^2/n_+$, the probability that the BDD classifier gives correct classification of new points converges to 1 as $d \to \infty$.*

The first part of Theorem 2.3 discusses the performance of 1SVM and is similar to the first part of Theorem 2.1 and Theorem 2.2. The performance of the 1SVM method is uniquely determined by the between-class distance. The second part of Theorem 2.3 provides the conditions under which the BDD methods give correct classification of a new data point. It shows that the BDD works as well as the one-direction method when the between-class distance is large enough, in the sense that either $\alpha_0 > 1/2$ or $\alpha_0 = 1/2$ and $\mu_0^2 > \sigma_+^2/n_+ - \sigma_-^2/n_-$. More importantly, when the two clusters in positive class are separated enough in the sense that either $\alpha_+ > 1/2$ or $\alpha_+ = 1/2$ and $\mu_+^2 > 8\sigma_+^2/n_+$, our BDD method will classify correctly regardless of value of between-class distance.

2.5 Proof

2.5.1 Proof of Theorems 2.1 and 2.2

Recall that we refer to the limiting simplices of the samples X_d^{+1}, X_d^{+2}, X_d^{-1}, and X_d^{-2} as the four simplices. Let $\boldsymbol{O}_{+1}$, $\boldsymbol{O}_{+2}$, $\boldsymbol{O}_{-1}$, and $\boldsymbol{O}_{-2}$ denote the centroids of these four simplices. To analyze the geometries of these centroids, note that, for each d, the union of the data and the class centroids, $X_d^{+1} \cup X_d^{+2} \cup X_d^{-1} \cup X_d^{-2} \cup \{C_{+1,d}, C_{+2,d}, C_{-1,d}, C_{-2,d}\}$ generate a subspace of dimension $n+3$ ($n = n_{+1}+n_{+2}+n_{-1}+n_{-2}$). Using the orthogonality properties developed by Hall et al. (2005); Jung and Marron (2009), for each d, there is a rotation of this space, i.e., a suitable basis so that the elements of X_d^{+1} can be written as

$$\{(l_{+,d}/2, 0, l_{0,d}/2), \sqrt{d}\sigma_{+,d}(\underbrace{1, \cdots, 0}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}}), (\underbrace{0, \cdots, 0}_{n_{-2}})\}, \cdots,$$
$$\{(l_{+,d}/2, 0, l_{0,d}/2), \sqrt{d}\sigma_{+,d}(\underbrace{0, \cdots, 1}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}}), (\underbrace{0, \cdots, 0}_{n_{-2}})\}. \quad (2.16)$$

Note that the centroid O_{+1} has co-ordinates

$$\{(l_{+,d}/2, 0, l_{0,d}/2), \sqrt{d}\sigma_{+,d}(\underbrace{1/n_{+1}, \cdots, 1/n_{+1}}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}}), (\underbrace{0, \cdots, 0}_{n_{-2}})\}. \quad (2.17)$$

Similarly, the other three centroids $\boldsymbol{O}_{+2}$, $\boldsymbol{O}_{-1}$, and $\boldsymbol{O}_{-2}$, can be represented in the same $(3+n)$-dimensional space as the vectors

$$\{(-l_{+,d}/2, 0, l_{0,d}/2), (\underbrace{0, \cdots, 0}_{n_{+1}}), \sqrt{d}\sigma_{+,d}(\underbrace{1/n_{+2}, \cdots, 1/n_{+2}}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}}), (\underbrace{0, \cdots, 0}_{n_{-2}})\}, \quad (2.18)$$

$$\{(0, l_{-,d}/2, -l_{0,d}/2), (\underbrace{0, \cdots, 0}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), \sqrt{d}\sigma_{-,d}(\underbrace{1/n_{-1}, \cdots, 1/n_{-1}}_{n_{-1}}), (\underbrace{0, \cdots, 0}_{n_{-2}})\}, \quad (2.19)$$

$$\{(0, -l_{-,d}/2, -l_{0,d}/2), (\underbrace{0, \cdots, 0}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}}), \sqrt{d}\sigma_{-,d}(\underbrace{1/n_{-2}, \cdots, 1/n_{-2}}_{n_{-2}})\}, \quad (2.20)$$

respectively.

Denote L_{ij} be the line that connects $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ and $[\boldsymbol{O}_i\boldsymbol{O}_j]$ be the line segment between $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, where $\boldsymbol{O}_i$ is the centroid of ith cluster. A straightforward extension of Theorem 1 of Hall et al. (2005) is

Lemma 2.1. *Assume that the two closest points in lines L_{12} and L_{34} lie inside the line segments $[\boldsymbol{O}_1\boldsymbol{O}_2]$ and $[\boldsymbol{O}_3\boldsymbol{O}_4]$ respectively. Then the 1SVM hyperplane that separates between group (1,2) and group (3,4) perpendicularly bisects the line segment between these two closest points.*

Proof of Lemma 2.1: The line L_{ij} can be represented as $L_{ij} = \{(\boldsymbol{O}_i + (\boldsymbol{O}_j - \boldsymbol{O}_i)t) : t \in R\}$. The line segment $[\boldsymbol{O}_i\boldsymbol{O}_j]$ can be represented as $[\boldsymbol{O}_i\boldsymbol{O}_j] = \{(\boldsymbol{O}_i + (\boldsymbol{O}_j - \boldsymbol{O}_i)t) : t \in [0,1]\}$. The squared distance between two lines L_{12} and L_{34} can be expressed as

$$\begin{aligned} D &= min_{t_1,t_2}|(\boldsymbol{O}_1 + (\boldsymbol{O}_2 - \boldsymbol{O}_1)t_1) - (\boldsymbol{O}_3 + (\boldsymbol{O}_4 - \boldsymbol{O}_3)t_2)|^2 \\ &= min_{t_1,t_2}|\boldsymbol{O}_{13} + \boldsymbol{O}_{21}t_1 - \boldsymbol{O}_{43}t_2|^2, \end{aligned} \quad (2.21)$$

where t_1, t_2 are two free parameters and $\boldsymbol{O}_{ij} = \boldsymbol{O}_i - \boldsymbol{O}_j$. We try to find $\hat{t}_1$ and $\hat{t}_2$ which minimize the square distance D. Taking derivative over t_1 and t_2, we get two equations

$$\begin{aligned} |\boldsymbol{O}_{21}|^2\hat{t}_1 - \boldsymbol{O}_{21}^T\boldsymbol{O}_{43}\hat{t}_2 + \boldsymbol{O}_{21}^T\boldsymbol{O}_{13} &= 0, \\ \boldsymbol{O}_{21}^T\boldsymbol{O}_{43}\hat{t}_1 - |\boldsymbol{O}_{43}|^2\hat{t}_2 + \boldsymbol{O}_{43}^T\boldsymbol{O}_{13} &= 0. \end{aligned} \tag{2.22}$$

The solution $\hat{t}_1$ and $\hat{t}_2$ are

$$\begin{aligned} \hat{t}_1 &= \frac{-\boldsymbol{O}_{21}^T\boldsymbol{O}_{13}|\boldsymbol{O}_{43}|^2 + \boldsymbol{O}_{43}^T\boldsymbol{O}_{13}\boldsymbol{O}_{21}^T\boldsymbol{O}_{43}}{|\boldsymbol{O}_{21}|^2|\boldsymbol{O}_{43}|^2 - (\boldsymbol{O}_{21}^T\boldsymbol{O}_{43})^2}, \\ \hat{t}_2 &= \frac{|\boldsymbol{O}_{21}|^2\boldsymbol{O}_{43}^T\boldsymbol{O}_{13} - \boldsymbol{O}_{21}^T\boldsymbol{O}_{43}\boldsymbol{O}_{21}^T\boldsymbol{O}_{13}}{|\boldsymbol{O}_{21}|^2|\boldsymbol{O}_{43}|^2 - (\boldsymbol{O}_{21}^T\boldsymbol{O}_{43})^2}. \end{aligned} \tag{2.23}$$

If the two points $\boldsymbol{O}_I^* = \boldsymbol{O}_1 + \boldsymbol{O}_{21}\hat{t}_1$ and $\boldsymbol{O}_{II}^* = \boldsymbol{O}_3 + \boldsymbol{O}_{43}\hat{t}_2$ lie inside the line segments $[\boldsymbol{O}_1\boldsymbol{O}_2]$ and $[\boldsymbol{O}_3\boldsymbol{O}_4]$ respectively, then $\hat{t}_1 \in [0,1]$ and $\hat{t}_2 \in [0,1]$, and the distance between these two points is the closest distance between the convex hull of the positive class points and the convex hull of the negative class points. The hyperplane which is perpendicularly bisects the line segment between these two points is $f(x) = x^T\hat{\beta} + \hat{b} = 0$, where

$$\begin{aligned} \hat{\beta} &= \frac{2(\boldsymbol{O}_{13} + \boldsymbol{O}_{21}\hat{t}_1 - \boldsymbol{O}_{43}\hat{t}_2)}{\hat{D}}, \\ \hat{b} &= -\frac{(\boldsymbol{O}_{13} + \boldsymbol{O}_{21}\hat{t}_1 - \boldsymbol{O}_{43}\hat{t}_2)^T(\boldsymbol{O}_1 + \boldsymbol{O}_3)}{\hat{D}}, \end{aligned} \tag{2.24}$$

where $\hat{D} = |\boldsymbol{O}_{13} + \boldsymbol{O}_{21}\hat{t}_1 - \boldsymbol{O}_{43}\hat{t}_2|^2$. We can show that all points from the positive group (1,2) satisfy $x^T\hat{\beta}+\hat{b} = 1$ while all points from the negative group (3,4) satisfy $x^T\hat{\beta}+\hat{b} = -1$. Assume that x_1 is from cluster 1, we have

$$x_1^T\hat{\beta} + \hat{b} = (x_1 - \boldsymbol{O}_1)^T\hat{\beta} + \boldsymbol{O}_1^T\hat{\beta} + \hat{b} = 1, \tag{2.25}$$

since $(x_1 - \boldsymbol{O}_1)^T\hat{\beta} = 0$ and $\boldsymbol{O}_1^T\hat{\beta} + \hat{b} = 1$. Similar results can be obtained for data points

from other 3 clusters. Thus, we conclude that an asymptotic representation of the SVM classifier is $f(x) = x^T\hat{\beta} + \hat{b}$. □

Using the SVM hyperplane derived above, it is straightforward to show that the new datum x_i^* from the ith cluster, for $i = 1, 2, 3, 4$, will give the classification function values

$$f(x_1^*) = x_1^{*T}\hat{\beta} + \hat{b} = 1 - \frac{2(1-\hat{t}_1)\sigma_1^2}{n_1\hat{D}}, \tag{2.26}$$

$$f(x_2^*) = x_2^{*T}\hat{\beta} + \hat{b} = 1 - \frac{2\hat{t}_1\sigma_2^2}{n_2\hat{D}}, \tag{2.27}$$

$$f(x_3^*) = x_3^{*T}\hat{\beta} + \hat{b} = -1 + \frac{2(1-\hat{t}_2)\sigma_3^2}{n_3\hat{D}}, \tag{2.28}$$

$$f(x_4^*) = x_4^{*T}\hat{\beta} + \hat{b} = -1 + \frac{2\hat{t}_2\sigma_4^2}{n_4\hat{D}}, \tag{2.29}$$

respectively. Here n_i and σ_i^2 denote the sample size and the scaled variance of the ith cluster respectively.

1. Let us first derive the 1SVM hyperplane that separates the positive class (X_d^{+1}, X_d^{+2}) from the negative class (X_d^{-1}, X_d^{-2}). Plug the formulas (2.17)–(2.20) into (2.26)–(2.29) with $\boldsymbol{O}_1 = \boldsymbol{O}_{+1}$, $\boldsymbol{O}_2 = \boldsymbol{O}_{+2}$, $\boldsymbol{O}_3 = \boldsymbol{O}_{-1}$ and $\boldsymbol{O}_4 = \boldsymbol{O}_{-2}$, we have that

$$f(x_+^*) = \frac{l_{0,d}^2/d - \sigma_{+,d}^2/n_+ + \sigma_{-,d}^2/n_-}{l_{0,d}^2/d + \sigma_{+,d}^2/n_+ + \sigma_{-,d}^2/n_-} \longrightarrow \frac{\mu_0 d^{2\alpha_+ - 1} - \sigma_+^2/n_+ + \sigma_-^2/n_-}{\mu_0 d^{2\alpha_+ - 1} + \sigma_+^2/n_+ + \sigma_-^2/n_-} \tag{2.30}$$

for a new data point $\mathbf{x}_+^*$ from the positive class, and

$$f(x_-^*) = -\frac{l_{0,d}^2/d + \sigma_{+,d}^2/n_+ - \sigma_{-,d}^2/n_-}{l_{0,d}^2/d + \sigma_{+,d}^2/n_+ + \sigma_{-,d}^2/n_-} \longrightarrow -\frac{\mu_0 d^{2\alpha_+ - 1} + \sigma_+^2/n_+ - \sigma_-^2/n_-}{\mu_0 d^{2\alpha_+ - 1} + \sigma_+^2/n_+ + \sigma_-^2/n_-} \tag{2.31}$$

for a new data point $\mathbf{x}_-^*$ from the negative class. Hence the part one of Theorem 1 can be easily concluded.

2. For BDD method, if $lim_{d\to\infty}\mu_0^2 d^{2\alpha_0 - 1} > \sigma_+^2/n_+ - \sigma_-^2/n_-$, we choose the usual 1SVM hyperplane as initial values for the BDD method, the two hyperplanes of the BDD method

can be represented as $f_1(\mathbf{x}^*) = f(\mathbf{x}^*)$ and $f_2(\mathbf{x}^*) = 1$. Thus the same conclusion as in 1 follows immediately.

If we choose the Cluster2-2 type of initial values, the BDD classifier includes the following two hyperplanes. The first one is the 1SVM hyperplane that separates between groups (X_d^{+1}, X_d^{-1}) and (X_d^{+2}, X_d^{-2}), which provides the classifier f_1. In the same manner by the replacement of $\boldsymbol{O}_1 = \boldsymbol{O}_{+1}$, $\boldsymbol{O}_2 = \boldsymbol{O}_{-1}$, $\boldsymbol{O}_3 = \boldsymbol{O}_{+2}$ and $\boldsymbol{O}_4 = \boldsymbol{O}_{-2}$ in (2.26)–(2.29), it can be shown that, $f_1(\mathbf{x}^*)$ equals to $\frac{l_{+,d}^2/d}{l_{+,d}^2/d+(\sigma_d^+)^2/n_+}$, $-\frac{\mu_+^2 d^{2\alpha_+-1}}{\mu_+^2 d^{2\alpha_+-1}+\frac{(\sigma_d^+)^2}{n_+}}$, $\frac{\mu_-^2 d^{2\alpha_--1}}{\mu_-^2 d^{2\alpha_--1}+\frac{(\sigma_d^-)^2}{n_-}}$, and $-\frac{\mu_-^2 d^{2\alpha_--1}}{\mu_-^2 d^{2\alpha_--1}+\frac{(\sigma_d^-)^2}{n_-}}$ for a new data point which is of type X_d^{+1}, X_d^{+2}, X_d^{-1} and X_d^{-2} respectively.

The second hyperplane is the 1SVM hyperplane that separates between groups (X_d^{+1}, X_d^{-2}) and (X_d^{+2}, X_d^{-1}), which provides the classifier f_2. Then, by the replacement of $\boldsymbol{O}_1 = \boldsymbol{O}_{+1}$, $\boldsymbol{O}_2 = \boldsymbol{O}_{-2}$, $\boldsymbol{O}_3 = \boldsymbol{O}_{+2}$ and $\boldsymbol{O}_4 = \boldsymbol{O}_{-1}$ in (2.26)–(2.29), $f_2(x^*)$ equals to $\frac{\mu_+^2 d^{2\alpha_+-1}}{\mu_+^2 d^{2\alpha_+-1}+\frac{(\sigma_d^+)^2}{n_+}}$, $-\frac{\mu_+^2 d^{2\alpha_+-1}}{\mu_+^2 d^{2\alpha_+-1}+\frac{(\sigma_d^+)^2}{n_+}}$, $-\frac{\mu_-^2 d^{2\alpha_--1}}{\mu_-^2 d^{2\alpha_--1}+\frac{(\sigma_d^-)^2}{n_-}}$, and $\frac{\mu_-^2 d^{2\alpha_--1}}{\mu_-^2 d^{2\alpha_--1}+\frac{(\sigma_d^-)^2}{n_-}}$ for a new data point which is of type X_d^{+1}, X_d^{+2}, X_d^{-1} and X_d^{-2} respectively.

Thus we can show that, as $d \to \infty$, the probability that the two-direction classifier $f_1 f_2$ provides correct classification tends to 1 if both $lim_{d\to\infty}\mu_+^2 d^{2\alpha_+-1} > 0$ and $lim_{d\to\infty}\mu_-^2 d^{2\alpha_--1} > 0$.

Combine the above two cases, we immediately get part 2 of Theorem 1. □

Proof of Theorem 2 follow from the proof of Theorem 1 immediately. □

2.5.2 Proof of Theorem 2.3

Note that the centroid of $\boldsymbol{O}_{+1}$ has co-ordinates

$$\{(l_{+,d}/2, l_{0,d}/2), \sqrt{d}\sigma_{+,d}(\underbrace{1/n_{+1}, \cdots, 1/n_{+1}}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}})\}. \tag{2.32}$$

Similarly, the other two centroids $\boldsymbol{O}_{+2}$, $\boldsymbol{O}_{-1}$, can be represented in the same $(2+n)$-dimensional space as the vectors

$$\{(-l_{+,d}/2, l_{0,d}/2), (\underbrace{0, \cdots, 0}_{n_{+1}}), \sqrt{d}\sigma_{+,d}(\underbrace{1/n_{+2}, \cdots, 1/n_{+2}}_{n_{+2}}), (\underbrace{0, \cdots, 0}_{n_{-1}})\}, \tag{2.33}$$

$$\{(0, -l_{0,d}/2), (\underbrace{0, \cdots, 0}_{n_{+1}}), (\underbrace{0, \cdots, 0}_{n_{+2}}), \sqrt{d}\sigma_{-,d}(\underbrace{1/n_{-1}, \cdots, 1/n_{-1}}_{n_{-1}})\}, \tag{2.34}$$

respectively.

Lemma 2.2. *The 1SVM hyperplane that separates group (1,2) and group 3 perpendicularly bisects the line segment between $\boldsymbol{O}_3$ and the closest point in the line segment $[\boldsymbol{O}_1\boldsymbol{O}_2]$.*

Proof of Lemma 2.2: The square distance between $\boldsymbol{O}_3$ and $[\boldsymbol{O}_1\boldsymbol{O}_2]$ can be expressed as

$$\begin{aligned} D &= \min_{0\le t\le 1}|(\boldsymbol{O}_1 + (\boldsymbol{O}_2 - \boldsymbol{O}_1)t - \boldsymbol{O}_3|^2 \\ &= \min_{0\le t\le 1}|\boldsymbol{O}_{13} - \boldsymbol{O}_{12}t|^2. \end{aligned} \tag{2.35}$$

The solution is

$$\hat{t} = \begin{cases} 0 & \text{if } \frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} < 0 \\ \frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} & \text{if } 0 \le \frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} \le 1 \\ 1 & \text{if } \frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} > 1 \end{cases}$$

It is easy to check that the distance between $\boldsymbol{O}_3$ and the closest point in line segment $[\boldsymbol{O}_1\boldsymbol{O}_2]$ is the closest distance between the convex hull of the positive class points and the convex hull of the negative class points. Thus the SVM hyperplane is the same as the hyperplane which is perpendicularly bisects this line segment and can be represented as $x^T\hat{\beta} + \hat{b} = 0$, where

$$\hat{\beta} = \frac{2(\boldsymbol{O}_{13} - \boldsymbol{O}_{12}\hat{t})}{\hat{D}},$$

$$\hat{b} \;=\; -\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T(\boldsymbol{O}_1+\boldsymbol{O}_3-\boldsymbol{O}_{12}\hat{t})}{\hat{D}}, \tag{2.36}$$

where $\hat{D}=|\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t}|^2$. We can show that all points from the cluster 1 satisfy $x^T\hat{\beta}+\hat{b}=1+2\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T\boldsymbol{O}_{12}\hat{t}}{\hat{D}}\geq 1$, all points from the cluster 2 satisfy $x^T\hat{\beta}+\hat{b}=1+2\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T\boldsymbol{O}_{12}(\hat{t}-1)}{\hat{D}}\geq 1$ and all points from the cluster 3 satisfy $x^T\hat{\beta}+\hat{b}=-1$. Assume that x_1 is from cluster 1, we have

$$x_1^T\hat{\beta}+\hat{b}=(x_1-\boldsymbol{O}_1)^T\hat{\beta}+\boldsymbol{O}_1^T\hat{\beta}+\hat{b}=1+2\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T\boldsymbol{O}_{12}\hat{t}}{\hat{D}}, \tag{2.37}$$

since $(x_1-\boldsymbol{O}_1)^T\hat{\beta}=0$ and $\boldsymbol{O}_1^T\hat{\beta}+\hat{b}=1+2\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T\boldsymbol{O}_{12}\hat{t}}{\hat{D}}$. Similar results can be obtained for data points from the other 2 clusters. Thus, we conclude that an asymptotic representation of the SVM classifier is $x^T\hat{\beta}+\hat{b}$. □

Using the SVM hyperplane derived above, it is straightforward to show that the new datum x_i^* from the ith cluster, for $i=1,2,3$, will give the classification function values

$$f(x_1^*) \;=\; x_1^{*T}\hat{\beta}+\hat{b}=1+2\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T\boldsymbol{O}_{12}\hat{t}}{\hat{D}}-2\frac{(1-\hat{t})\sigma_1^2}{n_1\hat{D}}, \tag{2.38}$$

$$f(x_2^*) \;=\; x_2^{*T}\hat{\beta}+\hat{b}=1+2\frac{(\boldsymbol{O}_{13}-\boldsymbol{O}_{12}\hat{t})^T\boldsymbol{O}_{12}(\hat{t}-1)}{\hat{D}}-2\frac{\hat{t}\sigma_2^2}{n_2\hat{D}}, \tag{2.39}$$

$$f(x_3^*) \;=\; x_3^{*T}\hat{\beta}+\hat{b}=-1+2\frac{\sigma_3^2}{n_3\hat{D}}, \tag{2.40}$$

respectively.

1. Let us first derive the 1SVM hyperplane that separates the positive class (X_d^{+1},X_d^{+2}) from the negative class (X_d^{-1}). For convenience, we define $\tilde{\mu}_i=\lim_{d\to\infty}\mu_i d^{\alpha_i-1/2}$ for $i=\{+,0\}$. Plug the formulas (2.32)–(2.34) into (2.38)–(2.40) with $\boldsymbol{O}_1=\boldsymbol{O}_{+1}$, $\boldsymbol{O}_2=\boldsymbol{O}_{+2}$, $\boldsymbol{O}_3=\boldsymbol{O}_{-1}$, we have that

$$\frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2}=\frac{1}{2}. \tag{2.41}$$

Therefore $\hat{t} = 1/2$ and

$$f(x_+^*) = \frac{\tilde{\mu}_0^2 - \sigma_+^2/n_+ + \sigma_-^2/n_-}{\tilde{\mu}_0^2 + \sigma_+^2/n_+ + \sigma_-^2/n_-} \tag{2.42}$$

for a new data point $\mathbf{x}_+^*$ from the positive class, and

$$f(x_-^*) = -\frac{\tilde{\mu}_0^2 + \sigma_+^2/n_+ - \sigma_-^2/n_-}{\tilde{\mu}_0^2 + \sigma_+^2/n_+ + \sigma_-^2/n_-} \tag{2.43}$$

for a new data point $\mathbf{x}_-^*$ from the negative class. Hence the part one of Theorem 3.

2. For BDD method, if $\tilde{\mu}_0^2 > \sigma_+^2/n_+ - \sigma_-^2/n_-$, we choose the usual 1SVM hyperplane as initial values for the BDD method, the two hyperplanes of the BDD method can be represented as $f_1(\mathbf{x}^*) = f(\mathbf{x}^*)$ and $f_2(\mathbf{x}^*) = 1$. Thus the same conclusion as in 1 follows immediately.

If we choose the Cluster1-2 type of initial values, the BDD classifier includes the following two hyperplanes. The first one is the 1SVM hyperplane that separates between groups (X_d^{+1}, X_d^{-1}) and (X_d^{+2}), which provides the classifier f_1. In the same manner by the replacement of $\boldsymbol{O}_1 = \boldsymbol{O}_{+1}$, $\boldsymbol{O}_2 = \boldsymbol{O}_{-1}$, and $\boldsymbol{O}_3 = \boldsymbol{O}_{+2}$, it can be shown that,

$$\frac{\boldsymbol{O}_{13}^T \boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} = \frac{\frac{\tilde{\mu}_+^2}{2} + 2\frac{\sigma_+^2}{n_+}}{\frac{\tilde{\mu}_+^2}{4} + \tilde{\mu}_0^2 + 2\frac{\sigma_+^2}{n_+} + \frac{\sigma_-^2}{n_-}}. \tag{2.44}$$

Therefore if $\tilde{\mu}_+^2 \leq 4(\tilde{\mu}_0^2 + \frac{\sigma_-^2}{n_-})$, we have $0 \leq \frac{\boldsymbol{O}_{13}^T \boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} \leq 1$ and

$$\hat{t} = \frac{2(\tilde{\mu}_+^2 + 4\sigma_+^2/n_+)}{\tilde{\mu}_+^2 + 4(\tilde{\mu}_0^2 + 2\sigma_+^2/n_+ + \sigma_-^2/n_-)}, \tag{2.45}$$

and

$$f_1(\mathbf{x}_{+1}^*) = \frac{\sigma_+^4 + \tilde{\mu}_+^2(\tilde{\mu}_0^2 + 2\sigma_+^2/n_+ + \sigma_-^2/n_-)}{(\tilde{\mu}_+^2 + 4\sigma_+^2/n_+)(\tilde{\mu}_0^2 + \sigma_+^2/n_+ + \sigma_-^2/n_-)}, \tag{2.46}$$

$$f_1(\mathbf{x}^*_{+2}) = -\frac{\tilde{\mu}_+^2(\tilde{\mu}_0^2+\sigma_-^2/n_-)-4\sigma_+^4/n_+^2}{(\tilde{\mu}_+^2+4\sigma_+^2/n_+)(\tilde{\mu}_0^2+\sigma_+^2/n_++\sigma_-^2/n_-)}, \quad (2.47)$$

$$f_1(\mathbf{x}^*_{-1}) = \frac{\tilde{\mu}_0^2+\sigma_+^2/n_+}{\tilde{\mu}_0^2+\sigma_+^2/n_++\sigma_-^2/n_-} \quad (2.48)$$

If $\tilde{\mu}_+^2 > 8\sigma_+^2/n_+$, we have $f_1(\mathbf{x}^*_{+1}) > 0$, $f_1(\mathbf{x}^*_{+2}) < 0$ and $f_1(\mathbf{x}^*_{-1}) > 0$. The first and the third ones are quite straightforward. To show the second one, we use $8\sigma_+^2/n_+ < \tilde{\mu}_+^2 \leq 4(\tilde{\mu}_0^2+\frac{\sigma_-^2}{n_-})$ to get $(\tilde{\mu}_0^2+\frac{\sigma_-^2}{n_-}) > 2\sigma_+^2/n_+$, thus $\tilde{\mu}_+^2 > \frac{16\sigma_+^4/n_+^2}{\tilde{\mu}_0^2+\sigma_-^2/n_-} > \frac{4\sigma_+^4/n_+^2}{\tilde{\mu}_0^2+\sigma_-^2/n_-}$.

If $\tilde{\mu}_+^2 > 4(\tilde{\mu}_0^2+\frac{\sigma_-^2}{n_-})$, we have $\frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} > 1$, thus $\hat{t}=1$ and

$$f_1(\mathbf{x}^*_{+1}) = \frac{3\tilde{\mu}_+^2+8\sigma_+^2/n_+-4(\tilde{\mu}_0^2+\sigma_-^2/n_-)}{\tilde{\mu}_+^2+8\sigma_+^2/n_++4(\tilde{\mu}_0^2+\sigma_-^2/n_-)}, \quad (2.49)$$

$$f_1(\mathbf{x}^*_{+2}) = -\frac{\tilde{\mu}_+^2-8\sigma_+^2/n_++4(\tilde{\mu}_0^2+\sigma_-^2/n_-)}{\tilde{\mu}_+^2+8\sigma_+^2/n_++4(\tilde{\mu}_0^2+\sigma_-^2/n_-)}, \quad (2.50)$$

$$f_1(\mathbf{x}^*_{-1}) = \frac{\tilde{\mu}_+^2+8\sigma_+^2/n_++4(\tilde{\mu}_0^2-\sigma_-^2/n_-)}{\tilde{\mu}_+^2+8\sigma_+^2/n_++4(\tilde{\mu}_0^2+\sigma_-^2/n_-)}. \quad (2.51)$$

If $\tilde{\mu}_+^2 > 8\sigma_+^2/n_+$, we have $f_1(\mathbf{x}^*_{+1}) > 0$, $f_1(\mathbf{x}^*_{+2}) < 0$ and $f_1(\mathbf{x}^*_{-1}) > 0$.

The second hyperplane is the 1SVM hyperplane that separates between groups (X_d^{+1}) and (X_d^{+2}, X_d^{-1}), which provides the classifier f_2. In the same manner by the replacement of $\boldsymbol{O}_1 = \boldsymbol{O}_{+2}$, $\boldsymbol{O}_2 = \boldsymbol{O}_{-1}$, and $\boldsymbol{O}_3 = \boldsymbol{O}_{+1}$, it can be shown that,

$$\frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} = \frac{\frac{\tilde{\mu}_+^2}{2}+2\frac{\sigma_+^2}{n_+}}{\frac{\tilde{\mu}_+^2}{4}+\tilde{\mu}_0^2+2\frac{\sigma_+^2}{n_+}+\frac{\sigma_-^2}{n_-}}. \quad (2.52)$$

Therefore if $\tilde{\mu}_+^2 \leq 4(\tilde{\mu}_0^2+\frac{\sigma_-^2}{n_-})$, we have $0 \leq \frac{\boldsymbol{O}_{13}^T\boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} \leq 1$ and

$$\hat{t} = \frac{2(\tilde{\mu}_+^2+4\sigma_+^2/n_+)}{\tilde{\mu}_+^2+4(\tilde{\mu}_0^2+2\sigma_+^2/n_++\sigma_-^2/n_-)}, \quad (2.53)$$

and

$$f_2(\mathbf{x}^*_{+1}) = \frac{\tilde{\mu}_+^2(\tilde{\mu}_0^2 + \sigma_-^2/n_-) - 4\sigma_+^4/n_+^2}{(\tilde{\mu}_+^2 + 4\sigma_+^2/n_+)(\tilde{\mu}_0^2 + \sigma_+^2/n_+ + \sigma_-^2/n_-)}, \tag{2.54}$$

$$f_2(\mathbf{x}^*_{+2}) = -\frac{\sigma_+^4 + \tilde{\mu}_+^2(\tilde{\mu}_0^2 + 2\sigma_+^2/n_+ + \sigma_-^2/n_-)}{(\tilde{\mu}_+^2 + 4\sigma_+^2/n_+)(\tilde{\mu}_0^2 + \sigma_+^2/n_+ + \sigma_-^2/n_-)}, \tag{2.55}$$

$$f_2(\mathbf{x}^*_{-1}) = -\frac{\tilde{\mu}_0^2 + \sigma_+^2/n_+}{\tilde{\mu}_0^2 + \sigma_+^2/n_+ + \sigma_-^2/n_-} \tag{2.56}$$

If $\tilde{\mu}_+^2 > 4(\tilde{\mu}_0^2 + \frac{\sigma_-^2}{n_-})$, we have $\frac{\boldsymbol{O}_{13}^T \boldsymbol{O}_{12}}{|\boldsymbol{O}_{12}|^2} > 1$, $\hat{t} = 1$ and

$$f_2(\mathbf{x}^*_{+1}) = \frac{\tilde{\mu}_+^2 - 8\sigma_+^2/n_+ + 4(\tilde{\mu}_0^2 + \sigma_-^2/n_-)}{\tilde{\mu}_+^2 + 8\sigma_+^2/n_+ + 4(\tilde{\mu}_0^2 + \sigma_-^2/n_-)}, \tag{2.57}$$

$$f_2(\mathbf{x}^*_{+2}) = -\frac{3\tilde{\mu}_+^2 + 8\sigma_+^2/n_+ - 4(\tilde{\mu}_0^2 + \sigma_-^2/n_-)}{\tilde{\mu}_+^2 + 8\sigma_+^2/n_+ + 4(\tilde{\mu}_0^2 + \sigma_-^2/n_-)}, \tag{2.58}$$

$$f_2(\mathbf{x}^*_{-1}) = -\frac{\tilde{\mu}_+^2 + 8\sigma_+^2/n_+ + 4(\tilde{\mu}_0^2 - \sigma_-^2/n_-)}{\tilde{\mu}_+^2 + 8\sigma_+^2/n_+ + 4(\tilde{\mu}_0^2 + \sigma_-^2/n_-)}. \tag{2.59}$$

For both cases, we can show that if $\tilde{\mu_+}^2 > 8\sigma_+^2/n_+$, we have $f_2(\mathbf{x}^*_{+1}) > 0$, $f_2(\mathbf{x}^*_{+2}) < 0$ and $f_2(\mathbf{x}^*_{-1}) < 0$.

Thus we can show that, as $d \to \infty$, the probability that the two-direction classifier $f_1 f_2$ provides correct classification tends to 1 if $lim_{d\to\infty}\mu_+^2 d^{2\alpha_+ - 1} > 8\sigma_+^2/n_+$.

Combine the above two cases, we immediately get part 2 of Theorem 3. □

2.6 Discussion

In this chapter, we have proposed a new set of classification methods which use multiple hyperplanes to do classification and provide more information about the data structure. Our methods open up an important area that lies between standard linear approaches (linear SVM, DWD, etc., which lack needed flexibility in many cases) and full on kernel approaches (which are very flexible, but suffer greatly from overfitting in HDLSS situa-

tions). Although BDD can be used for low dimensional problems as well, it is specifically designed for handling HDLSS data which are becoming increasingly common in various fields such as genetics, drug discovery and image analysis. Our method not only builds a useful visualization tool for high dimensional data but also can be used to find important sub-clusters within each class. Although our focus is on the SVM and DWD, the basic idea can be applied to many other linear classification methods as well.

We note that many real data sets are unbalanced, i.e., the sample sizes from the two classes are very different. As mentioned in Qiao and Liu (2009) and Qiao et al. (2010), the SVM and DWD decision boundaries heavily depend on the ratio of sample sizes from the two classes considered. Our next step is to generalize our BDD method to incorporate weights such that our method can have even broader application. We will allow different weights on the two classes, with greater weight on the minority class and smaller weight on the majority class.

In this dissertation, our simulation and real data studies mainly focus on two direction cases. As discussed in Section 2.2.4, the extension of our method to multi-directional cases is quite straightforward and already implemented. Our next research direction is to extend our method to more complicated data sets which include multiple subclasses.

CHAPTER 3

Multiclass Distance Weighted Discrimination

3.1 Introduction

As noted in Section 1.5, binary classification is a well studied special case. In many applications, *multiclass* (or *multicategory*) problems are important as well. Binary classification methods can be generalized in many ways to handle multiple classes. Generalizations from binary SVM to multiclass SVM have been well studied in the literature. Two general strategies are commonly used to tackle the multiclass SVM problem. One strategy is to solve the multiclass problem by solving a series of binary problems. Examples include One-Versus-One (OVO) and One-Versus-The-Rest (OVR) approaches (Duda et al. (2000); Hastie et al. (2009)). The second strategy treats the population in a simultaneous fashion and considers all classes at once. Various methods along the line of the second strategy include Weston and Watkins (1999); Crammer and Singer (2000); Lee et al. (2004); Liu and Shen (2006); Liu and Yuan (2010). However, to our knowledge, generalization from binary DWD to multiclass DWD has not been studied. This chapter involves the study of the extension of DWD from the binary case to the multiclass case using both strategies.

For multiclass classification methods, one needs either to construct several binary classifiers or to solve a larger optimization problem which involves all classes at the same time.

The OVO and OVR methods are computationally simple, and the global method is computationally more complex. However, the OVO method has the disadvantage of potential variance increase, because a smaller number of observations are used to learn each classifier. The OVR method may fail under the circumstance where there is no dominating class, see Friedman (1996) and Lee et al. (2004). This leads to an interesting question of whether a more sophisticated method can achieve stronger results than the combination of several simple binary methods.

For multiclass SVM problems, comparisons of these three methods have been studied. Hsu and Lin (2002) conducted large-scale experiments and claimed that the OVO method is more suitable for practical use than the other methods. Lee et al. (2004) and Liu and Yuan (2010) demonstrated the superiority of their global method over the simple OVR method through some numerical studies. Rifkin and Klautau (2004) claimed that a simple OVR method is as accurate as any other approach. They supported their position by a critical review of the existing literatures and some experimental work. It is interesting to consider whether similar results can be obtained using the DWD method. We will carry out some simulation studies in this paper for all three methods and indicate the situations under which each specific method is preferred.

Microarray analysis has become a powerful tool in biological science. Microarray technologies allow for the measurement of thousands of gene expression levels simultaneously. The primary goal of a microarray study is to extract useful information from differential expression and provide insight into biological effects. However, nonbiological experimental variation such as batch effects are commonly observed in microarray experiments due to different experimental conditions. Large batch effects can make it difficult to obtain meaningful and accurate biological results and also make it difficult to integrate data from several sources or from multiple independent studies. Disregarding batch effects could result in misleading conclusions. Therefore, it is important and necessary to identify

and adjust batch effects prior to microarray data analysis. Common approaches include mean/median centering, Singular Value Decomposition (SVD Alter et al. (2000)) and ANOVA-like modeling (Wolfinger et al. (2001)) to balance the expression measurement across experiments. More sophisticated procedures have also been developed including an empirical Bayes method (Tibshirani et al. (2002); Johnson et al. (2007)), DWD (Benito et al. (2004); Liu et al. (2009)), and XPN (Shabalin et al. (2008)). See Scherer (2009) for a good review of this area.

The DWD classification method has been shown to provide effective batch adjustment for microarray data by Benito et al. (2004), and Liu et al. (2009). They also demonstrated that DWD can work better than SVM and SVD for the adjustment of systematic microarray effects. Benito et al. (2004) implement batch adjustment by first projecting the data onto the DWD normal direction and then moving the means of the two classes to a common point along that direction. When there are more than two batches, they take a stepwise approach. For example, for data including three batches, they first made a batch adjustment between Batches 1 and 2 (combined) and Batch 3. Next, they applied the same method to the adjusted data, to separate Batch 1 from Batch 2. This stepwise method creates an additional level of complexity especially when the number of batches considered is large because we need to decide which pair should be chosen in each step. For a K class problem, our proposed global multiclass DWD (MDWD) method will simultaneously produce K direction vectors which provide the basis of our new batch adjustment method. The K normal direction vectors determine a subspace which contains each class mean. We move each class in such a way that the class means move to a common point in this subspace. In Section 3.2 we will show how our new multiclass batch adjustment method gives better performance than any combination of binary methods.

The rest of the chapter is organized as follows. In Section 3.2 we present the batch adjustment results for a real data set using our MDWD method. Different types of mul-

ticlass DWD methods including OVO, OVR and MDWD are introduced in Section 3.3. Some theoretical properties of multiclass DWD are explored in Section 3.4. In Section 3.5 we present numerical results on simulated data to compare the performances of different methods. We collect proofs of the theoretical results in Section 3.6 and provide some discussions in Section 3.7.

3.2 Illustration of Batch Adjustment

Here we study data from a cohort of patients with Glioblastoma Multiforme (GBM) brain cancer whose brain samples were assayed using mircroRNA expression arrays (Agilent Human miRNA Microarray Rel12.0), as well as by multiple other genomic technologies, see TCGA (2010). Tumors analyzed by TCGA were procured from multiple institutions over the course of several years and accordingly the genomic analyses, including microarrays, were performed in batches of several dozen individuals at a time. Such a production process is well known to generate batch effects, although in many cases such as TCGA, no obvious alternative has emerged to allow large scale profiling projects to proceed Leek et al. (2010). Therefore, investigators generally expect and correct for batch effects rigorously. In the current example, we tackle a particularly challenging dataset generated from the miRNA arrays. These arrays are challenging because only a minority of miRNA's measured by the array are expressed in any given sample. Most probes on the array are therefore measuring only background or nonspecific hybridization, a measurement which is highly influenced by batch and completely uninfluenced by biology. The data for the example used in the current work is a set of 168 arrays which have been quantile normalized for overall image array intensity and for which replicate measurements for a single probe have been averaged. On average, there are 3 probes (a probe set) designed to assess each miRNA on the array, and no attempt was made in the current analysis to collapse probe sets to

miRNA's. Each probe was median centered across all arrays and all measurements for that probe were divided by the standard deviation for that probe across all arrays. No probe filtering was performed. Since the goal of normalization is to correct for batch effects while retaining biologic differences, we investigated our ability to detect clinical phenotypes in the data before and after batch correction. The phenotype we chose to detect is a tumor classification based on gene expression analysis (as opposed to miRNA analysis which are the data studied in the current work). Four clinically relevant subtypes were identified using mRNA analysis in Verhaak et al. (2010), they are Proneural, Neural, Classical, and Mesenchymal. Among the 168 samples, there are 52 Proneural, 24 Neural, 36 Classical, and 56 Mesenchymal samples.

Figure 3.1 studies the raw GBM data using a scatter plot matrix visualization based on the first four Principal Component (PC) axes. Observations from different batches are distinguished by different colors. The symbol types indicate the biological classes. The plots on the diagonal show the one-dimensional projections of the data onto each PC direction vector. A different height is added to each symbol just for convenient visual separation. In each diagonal plot we also include several smooth histograms, colored according to the batch label. The off-diagonal plots are projections of the data onto 2-d planes, determined by the various pairs of the PC directions. Note that Batch 5 (red color) is clearly separated from the rest of the batches in the PC1 direction. Figure 3.1 gives some suggestion of biological classes; for example in the PC 4 direction, Proneural (circle) seems to separate itself from the rest. However, this class is not very distinct in the sense that the distances between batches are large relative to the distances between biological classes. Therefore, it will be very useful to remove the batch effects before doing data analysis.

The steps of the proposed MDWD batch adjustment are as follows: (1) The MDWD direction vectors generate a subspace. (2) The subpopulations (e.g. respective batch subsets) are all projected onto that subspace. (3) The coordinates of the subpopulation

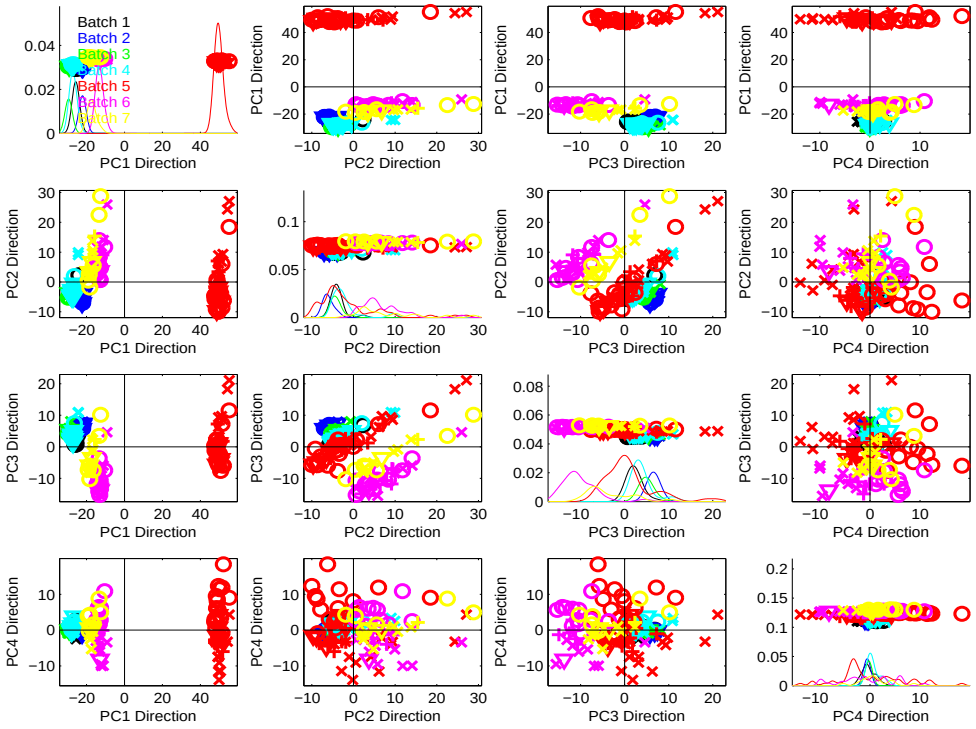


Figure 3.1: PCA projection scatter plot view of raw GBM data, showing 1D (diagonal) and 2D projections of raw data onto PC directions. Groupings of colors indicate batch biases. Samples from Classical, Mesenchymal, Proneural, and Neural are indicated by "+", "x", circle and triangle symbols respectively. This shows a very strong batch effect, so that adjustment is essential before combining data sets.

projected means are computed. (4) Each subpopulation is shifted in such a way that its projected mean is moved in the subspace to a fixed point which is common to all subpopulations. An important advantage of the MDWD adjustment over PCA adjustment is that it preserves the variation that is not due to batch effects, because the MDWD directions maximize the separations between the batches and ignore the variation in the data.

Figure 3.2 shows (using the same view) the same data after the MDWD adjustment. Now in all of the PC directions, the huge differences among batches visible in Figure 3.1 have disappeared, because the colors, representing the seven batches, are very well mixed. This shows that the systematic sample batch effects in the data have been effectively removed. Note that in the first row, second column subplot of Figure 3.2, the Proneural

(circles on the right) are clearly distinguished from the Mesenchymal (x symbol on the left). Thus the batch differences are much smaller in magnitude than the biological features in this data set.

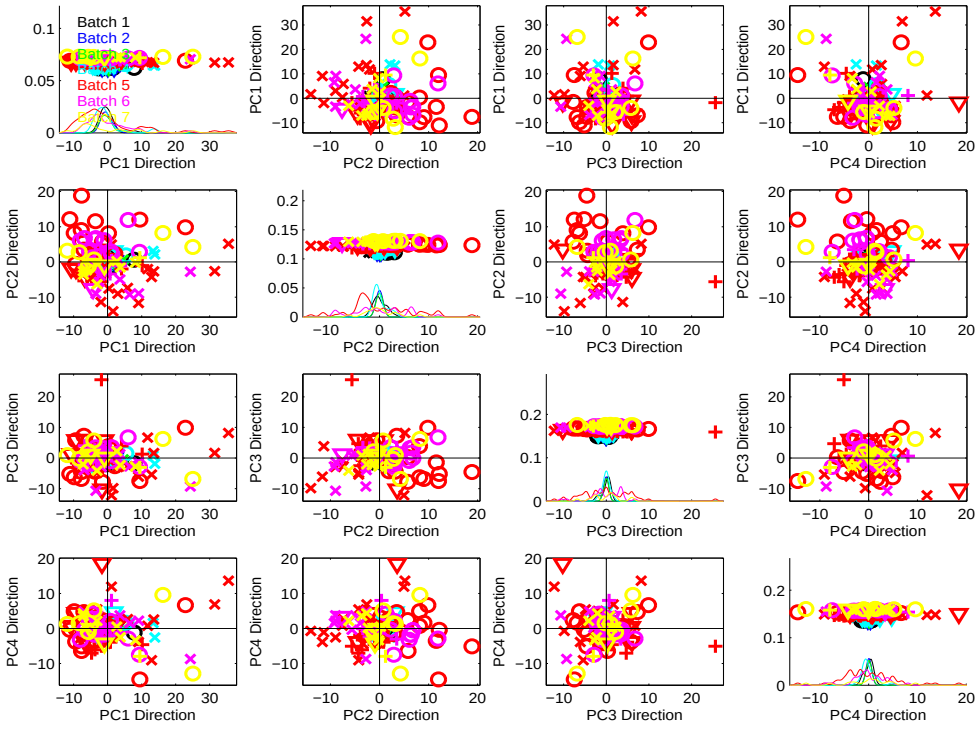


Figure 3.2: PCA scatter plot view of MDWD adjusted GBM data (labels are the same as in Figure 3.1), showing effective removal of batch biases. Biological class differences are now much more clear.

Adjusting batch effects in microarray data sets with more than two batches using the OVR and OVO methods can be implemented by the combination of a series of binary adjustments. The stepwise approach described in Benito et al. (2004) is based on the OVR DWD method. The batch adjustment using the OVO method also takes a stepwise approach as follows. In each step, a pair of classes are combined together through a binary adjustment. So the number of unadjusted classes is reduced by one after each step. This process is repeated until all classes have been combined together.

The main drawback of the OVR and OVO adjustment methods is that their results depend on the order, i.e, which pair of classes are used in each binary problem. In each

step, the number of options in constructing the binary problem increases with the number of total classes. Therefore, in the case where the number of classes considered is big, this can be a complicated problem because it is hard to find the optimal order among so many options. Moreover, the class size can be quite unbalanced which will further complicate the problem as shown in Qiao et al. (2010). A significant advantage of the MDWD method over the OVR and OVO methods is that it provides a convenient way to do batch adjustment for data sets with more than two batches. The MDWD method considers all batches at once and makes adjustment simultaneously for all batches.

3.3 Methodology

In the classification problem, we are given a training dataset consisting of n observations $(\mathbf{x}_i, y_i)$ for $i = 1, \cdots, n$. Here $\mathbf{x}_i \in R^d$ represents an input vector, and $y_i \in \{1, \cdots, K\}$ denotes the corresponding output class label. We assume that each $(\mathbf{x}_i, y_i)$ is independent random vectors distributed according to some unknown distribution function $P(\mathbf{x}, y)$. The task is to build a classification rule $\phi(\mathbf{x}) : R^d \to \{1, \cdots, K\}$ which can be used to predict the class label for a new input $\mathbf{x}$. In this section, we generalize binary DWD to the multiclass case. We first define OVR and OVO DWD which are based on solving several binary DWD classifications. Then we introduce MDWD which considers all classes in a single optimization.

3.3.1 Simple Pair-Wise Extension

The OVR constructs K binary classifiers, each one trained to distinguish the examples in the single class from the examples in all remaining classes. When it is desired to classify a new example, the K classifiers are run, and the classifier which outputs the largest value is chosen.

In contrast to SVM, which seeks to maximize the smallest residual distance to the separating hyperplane, DWD aims to minimize the sum of inverse residuals. In particular, for the ith DWD classifier which is trained with all of the examples in the ith class with positive labels and all other examples with negative labels, we solve the following problem

$$\min_{\mathbf{w}^i,\beta^i,\xi^i} \sum_k \Big(\frac{1}{r_k} + C\xi_k^i\Big), \tag{3.1}$$

$$\begin{aligned} \text{subject to} \quad & r_k = (\mathbf{x}_k^T\mathbf{w}^i + \beta^i) + \xi_k^i, \text{ for } k : y_k = i, \\ & r_k = -(\mathbf{x}_k^T\mathbf{w}^i + \beta^i) + \xi_k^i, \text{ for } k : y_k \neq i, \\ & \mathbf{w}^{iT}\mathbf{w}^i \leq 1,\ r_k \geq 0,\ \xi_k^i \geq 0. \end{aligned} \tag{3.2}$$

After solving (3.1), there are K decision functions and we say $\mathbf{x}$ is in the class which has the largest value of the decision function, i.e. class of $\mathbf{x} = \text{argmax}_i(\mathbf{x}^T\mathbf{w}^i + \beta^i)$.

The OVO approach constructs $K(K-1)/2$ classifiers where each one is trained on data from two classes. For the classifier i, j which is trained on data from the ith class and the jth class, we solve the similar binary classification problem

$$\min_{\mathbf{w}^{ij},\beta^{ij},\xi^{ij}} \sum_{k:y_k=i \text{ or } y_k=j} \Big(\frac{1}{r_k} + C\xi_k^{ij}\Big), \tag{3.3}$$

$$\begin{aligned} \text{subject to} \quad & r_k = (\mathbf{x}_k^T\mathbf{w}^{ij} + \beta^{ij}) + \xi_k^{ij}, \text{ for } k : y_k = i, \\ & r_k = -(\mathbf{x}_k^T\mathbf{w}^{ij} + \beta^{ij}) + \xi_k^{ij}, \text{ for } k : y_k = j, \\ & \mathbf{w}^{ijT}\mathbf{w}^{ij} \leq 1,\ r_k \geq 0,\ \xi_k^{ij} \geq 0. \end{aligned} \tag{3.4}$$

There are different methods for combining the results of all $K(K-1)/2$ classifiers. The most commonly used method is called Friedman's "Max-wins" voting strategy: if $\text{sign}(\mathbf{x}_k^T\mathbf{w}^{ij} + \beta^{ij})$ says $\mathbf{x}$ is in the ith class, then the vote total for the ith class is increased by one; otherwise the vote total for the jth class is increased by one. Then we predict $\mathbf{x}$ is in the class with the largest vote total.

3.3.2 Full Multiclass Version

Here we propose an approach for multiclass DWD problems by considering all classes at once and solving one single optimization problem simultaneously. We will show that the generalized formulation encompasses that of the two category DWD, regaining the desirable properties of the binary DWD. Consider a K-class classification problem. There are many different ways to represent classifiers. One of the most natural ways is to introduce a vector of discriminant functions $\mathbf{f} = (f_1, \cdots, f_K)$, where each component represents one class. For any new input $\mathbf{x}$, its label is estimated via a decision rule $\hat{y} = \text{argmax}_k f_k(\mathbf{x})$, where $f_k(\mathbf{x}) = \mathbf{x}^T \mathbf{w}_k + \boldsymbol{\beta}_k$. We also write $\mathbf{w}$ for $(\mathbf{w}_1, \cdots, \mathbf{w}_K)$, with $||\mathbf{w}||^2 = \sum_{k=1}^K ||\mathbf{w}_k||^2$.

For extension of DWD from the binary to the multiclass case, the objective function can be naturally constructed in such a way that it encourages f_y to be the largest among K functions. Here we formulate multiclass DWD in terms of the following optimization problem

$$\min_{\mathbf{w}, \boldsymbol{\beta}, \xi} \sum_{i=1}^{n} \sum_{k \neq j} \left(\frac{1}{r_{jk}^i} + C\xi_{jk}^i \right), \tag{3.5}$$

$$\text{subject to } r_{jk}^i = f_j(x_i) - f_k(x_i) + \xi_{jk}^i, \text{ for } y_i = j, \ k \neq j$$

$$r_{jk}^i \geq 0, \ \xi_{jk}^i \geq 0, \ \sum_{k=1}^{K} \mathbf{w}_k = \mathbf{0}, \ \sum_{k=1}^{K} \beta_k = 0, \ ||\mathbf{w}||^2 \leq 1. \tag{3.6}$$

Note that the i-th individual's contribution to the first term in the objective function (3.5) is the sum of the inverse of the differences between $f_{y_i}(\mathbf{x}_i)$ and all the other functions. This represents a natural generalization of the term $y_i f(\mathbf{x}_i)$ appearing in the binary DWD loss function. The parameter C in the second term in (3.5) controls the penalty on the variable ξ, the amount of violation of classification. It also plays the role of tuning parameter. Similar to the binary case, using additional variables and constraints, the optimization problem (3.5) can be reformulated as a second-order cone programming problem. If $K = 2$,

it is easy to show that the problem (3.5) reduces to the original binary DWD.

3.4 Theoretical Properties

In this section we study some of the statistical properties of multiclass DWD. We will focus on Fisher consistency. In statistics, Fisher consistency is a desirable property of an estimator asserting that if the estimator were calculated using the entire population rather than a sample, the true value of the estimated parameter would be obtained (Fisher (1922)). For example, suppose an estimator of a parameter θ based on the sample can be represented as a functional of the empirical distribution F_n, $\theta = T(F_n)$. Then the estimator is said to be Fisher consistent if its population analog, $T(F)$, is the same as the parameter θ.

Adapting the idea to the procedures of loss function minimization for classification, consider a procedure of finding f from all measurable functions $\mathcal{F}$ that minimizes a loss with respect to $\hat{L} = (1/n)\sum_{i=1}^{n} L(f(x_i), y_i)$. We say that a loss function L is Fisher consistent if the population minimizer of the loss $EL(f(X), Y)$ leads to the Bayes optimal decision rule (Bartlett et al. (2006)). The Fisher-consistent condition basically says that with infinite samples, one can exactly recover the Bayes rule by minimizing the loss. Let $p_i(\mathbf{x}) = P(y = i|\mathbf{x})$ denote the conditional probability of the ith class ($i = 1, \cdots, K$), then the Bayes decision rule is $\text{argmax}_{k\in\{1,\cdots,K\}} P(y = k|\mathbf{x})$. For notational simplicity, we denote $p_i(\mathbf{x})$ by p_i for $i = 1, \cdots, K$ in the following.

Fisher consistency has been well investigated for binary classification methods. Many commonly used methods, such as SVM and DWD, are Fisher consistent. However, it turns out that one loses consistency in the generalization from the binary SVM to some multiclass SVM methods. In this section we study whether or not the consistency can be kept for the generalization from binary DWD to multiclass DWD. We first study the Fisher consistency

of OVO and OVR DWD in Section 3.4.1 and then study the Fisher consistency of MDWD in Section 3.4.2.

3.4.1 Fisher Consistency of Pair-Wise Version

It is easy to study the consistency property of the OVO type of approach to the multi-class classification problem, since the properties of the corresponding binary classifiers are well studied. Friedman (1996) pointed out that the "Max-wins" rule is equivalent to the Bayes rule when the class posterior probabilities p_i are known:

$$\text{argmax}_i(p_i) = \text{argmax}_i\Big[\sum_{j\neq i} I\Big(\frac{p_i}{p_i+p_j} > \frac{p_j}{p_i+p_j}\Big)\Big]. \tag{3.7}$$

Equation (3.7) suggests that the OVO method will be Fisher consistent as long as the consistency of its underlying binary classifiers is satisfied. This allows us to conclude that the OVO DWD is Fisher consistent since the Fisher consistency of binary DWD has been proved in Qiao et al. (2010).

For the OVR SVM, Lee et al. (2004) argued that Fisher consistency holds only in the case when there exists a dominating class, i.e., a class j with $p_j > 1/2$, because only the support vectors appear in each optimization, resulting in a flat region of the loss. More specifically, the minimizer of the OVR SVM classifier is $\text{sign}(p_i - \frac{1}{2})$ for $i = 1, \cdots, K$. If there is a class j with $p_j > \frac{1}{2}$, then we can easily pick the majority class j because f_j would be near 1 and all of the other f_i would be close to -1. However, if there is no dominating class, then all f_i's would be close to -1, making the classifier inconsistent.

In sharp contrast, since DWD uses all data points, the resulting loss is smoothly decreasing, so Fisher consistency holds much more broadly in the sense that the solution satisfies $f_i^* > f_j^*$ if $p_i > p_j$ regardless of whether p_i is bigger than $\frac{1}{2}$ or not. The following

theorem establishes Fisher consistency of the OVR DWD method:

Theorem 3.1. *Let f_i^* be the minimizer of the ith binary DWD classifier defined in the OVR DWD method (5). Assume that the unique maximum of p_i for $i = 1, \cdots, K$ exists. Then argmax$_i(f_i^*)$ =argmax$_i(p_i)$.*

Proof of this Theorem and other proofs are given in Section 3.6.

3.4.2 Fisher Consistency of Full Multiclass Version

Qiao et al. (2010) proved the Fisher consistency of binary DWD by using an equivalent formulation of the DWD optimization. We will show the Fisher consistency of multiclass DWD based on the extension of the equivalent formulation from the binary case to the multiclass case.

For each $i = 1, \cdots, n$ and $k \in \{1, \cdots, K\}/\{y_i\}$, we define $f_{y_i k}^i = f_{y_i}^i - f_k^i = (\mathbf{x}_i^T \mathbf{w}_{y_i} + \beta_{y_i}) - (\mathbf{x}_i^T \mathbf{w}_k + \beta_k)$. The multiclass DWD optimization problem (3.5) can be shown to be equivalent to the following problem

$$\min_{\mathbf{w},\boldsymbol{\beta}:||\mathbf{w}||\leq 1} \sum_{i=1}^{n} \sum_{k\neq j} \min_{\xi_{y_i k}^i \geq 0} \left(\frac{1}{f_{y_i k}^i + \xi_{y_i k}^i} + C\xi_{y_i k}^i \right). \tag{3.8}$$

It can be shown that the optimal solution for the inside optimization part of (3.8) is given by $(\xi_{y_i k}^i)^* = \frac{1}{\sqrt{C}} - f_{y_i k}^i$ if $f_{y_i k}^i \leq \frac{1}{\sqrt{C}}$; $(\xi_{y_i k}^i)^* = 0$ otherwise. Then the multiclass DWD problem amounts to

$$\min_{\mathbf{w},\boldsymbol{\beta}} \sum_{i=1}^{n} \sum_{k\neq y_i} \left([2\sqrt{C} - C f_{y_i k}^i] I\left[f_{y_i k}^i \leq \frac{1}{\sqrt{C}} \right] + \frac{1}{f_{y_i k}^i} I\left[f_{y_i k}^i \geq \frac{1}{\sqrt{C}} \right] \right) \tag{3.9}$$

$$\text{subject to } ||\mathbf{w}||^2 \leq 1. \tag{3.10}$$

If we define the multiclass DWD loss function as

$$V(\mathbf{f}, y) = \sum_{j \neq y} l(f_{yj}), \tag{3.11}$$

where

$$l(f_{yj}) = \begin{cases} 2\sqrt{C} - C f_{yj} & \text{if} f_{yj} \leq \frac{1}{\sqrt{C}} \\ \frac{1}{f_{yj}} & \text{otherwise,} \end{cases}$$

then the multiclass DWD optimization is $\min_{\mathbf{w},\boldsymbol{\beta}} \sum_{i=1}^{n} V(\mathbf{f}(\mathbf{w}, \boldsymbol{\beta}), y_i),\ s.t.\ ||\mathbf{w}|| \leq 1$.

Consider $y \in \{1, \cdots, K\}$. For any classification function $\mathbf{f} = (f_1, \cdots, f_K)$, the expected multiclass DWD loss, that is, the risk, is $R(\mathbf{f}) = E(E(V(\mathbf{f}(\mathbf{x}), y)|\mathbf{x}))$. Fisher consistency requires that $\text{argmax}_k f_k^* = \text{argmax}_k p_k$, where $\mathbf{f}^*(\mathbf{x}) = (f_1^*(\mathbf{x}), \cdots, f_K^*(\mathbf{x}))$ denotes the minimizer of $R(\mathbf{f})$. Theorem 3.2 shows the Fisher consistency of multiclass DWD.

Theorem 3.2. *Let $\mathbf{f}^*$ be the global minimizer of $R(\mathbf{f}) = E(E(V(\mathbf{f}(\mathbf{x}), y)|\mathbf{x}))$, where $V(\cdot)$ is the multiclass DWD loss given in (3.11). Assume that the unique maximum of p_k for $k = 1, \cdots, K$ exists. Then $argmax_k(f_k^*) = argmax_k(p_k)$.*

There are previous studies on Fisher consistency of multiclass SVM methods such as Zhang (2004); Lee et al. (2004); Tewari and Bartlett (2005); Liu (2007); Zou et al. (2008). Liu (2007) summarized the Fisher consistency properties of four commonly used SVM loss functions:

(a) (Zou et al. (2008)) $[1 - f_y(\mathbf{x})]_+$;

(b) (Lee et al. (2004)) $\sum_{j \neq y} [1 + f_j(\mathbf{x})]_+$;

(c) (Vapnik (1998); Weston and Watkins (1999); Bredensteiner and Bennett (1999)) $\sum_{j \neq y} [1 - (f_y(x) - f_j(x))]_+$;

(d) (Crammer and Singer (2000); Liu and Shen (2006)) $[1 - \min_j(f_y(\mathbf{x}) - f_j(\mathbf{x}))]_+$.

It was shown in Liu (2007) that, under the sum-to-zero constraint, except for loss (b), classifiers based on these losses are not always Fisher consistent. Two approaches were proposed in Liu (2007) to modify inconsistent classifiers to be consistent. It is interesting to see that the DWD loss function we used in (3.11) is related to the SVM counterpart (c). But the DWD loss function yields a Fisher consistent classifier without modification. The reason is that the loss function (3.11) is continuously differentiable as opposed to the SVM loss function which is not differentiable. This appealing property of DWD is due to the fact that all data points have a direct influence, instead of only the support vectors.

3.5 Simulations

In this section, simulations are conducted to investigate the performance of the proposed OVR, OVO and MDWD methods. For comparison, the results from the Bayes classifiers, which are derived based on the true underlying distributions, are also included.

The simulated data sets include training, tuning and test sets. We generated the tuning and testing data from the same distributions as the training data. For the reason noted in Shao (1993), we set the sample sizes of tuning sets equal to that of the training sets. The sizes of the test sets are taken to be 10 times bigger than that of the training sets. Each experiment was replicated 100 times. Tuning sets are used to choose the tuning parameter C through a grid search, and the testing errors, evaluated on independent testing data, are used to measure the accuracy of various classifiers.

We have tried many different settings, including both low- and high-dimensional. To save space, we only report the results from the high-dimensional settings since the focus of MDWD is on high-dimensional situations. We consider HDLSS settings with $d = 1000$

in all simulations.

Our simulation results show that in situations where each class can be well separated from the rest, the performance of all three multiclass DWD methods are quite similar and are close to the optimal Bayes rule. We do not explicitly show these results here. The first example we show belongs to the situations where not all individual classes can be well separated from the rest. The data include three classes with the sample size of each training class being 50. The three classes are generated using three different Gaussian distributions with unit covariance and the first two components of the mean vectors as $(-5, 0)$, $(5, 0)$ and $(0, 1)$. The rest of the $d - 2$ dimensions are pure noise, i.e., all sampled from the standard normal distribution. If it is known that one should look in the direction of the first two coordinates, then the three classes are easy to separate, as shown by the tiny test error of the Bayes rule. However, in high dimensions, it can be quite challenging to find those directions. To investigate the generalizability property of the different methods, we exhibit the average performance over 100 replications in the first row of Table 3.1. The table summarizes the mean and standard error (over the 100 replications) of the proportion (out of 1500 members of each test data set) of incorrect classifications. Note that none of the three methods can achieve results close to optimal. But both OVO and MDWD are quite comparable, and much better than OVR, which is consistent with the ideas of Friedman (1996).

Example 2 is a case where MDWD is the best of these three methods. The data include three classes with the same sample size as Example 1. The first two components of the distributions for the three classes are Gaussians with means $(-10, 0)$, $(10, 0)$ and $(0, 2)$ and variances $(5, 1)$, $(5, 1)$ and $(1, 2)$. Figure 3.3 shows the projections of the data points and the decision boundaries onto the first two directions. The first, second and third classes have $n_1 = n_2 = n_3 = 50$ data vectors denoted by red plus, blue square, and white circle signs respectively. The optimal Bayes decision boundary is quadratic for this case due to

Table 3.1: Test errors (in percentage) over 100 replications

	OVR	OVO	MDWD	Bayes
Example 1	16.18	7.59	7.59	0.72
	(0.11)	(0.08)	(0.08)	(0.02)
Example 2	9.45	8.64	6.96	4.67
	(0.15)	(0.12)	(0.14)	(0.05)
Example 3	19.36	19.81	18.02	15.23
	(0.06)	(0.05)	(0.13)	(0.07)
Example 4	29.00	24.63	15.28	0.70
	(0.17)	(0.19)	(0.20)	(0.02)
Example 5	30.96	28.75	19.08	3.39
	(0.17)	(0.18)	(0.27)	(0.06)

the fact that the variances are different among three classes. In this example, classes 1 and 2 can be easily separated and it is challenging to separate class 3 from the other two. The MDWD classifier results in a decision area for class 3 which is close to the one provided by the Bayes rule. In contrast, the OVR and the OVO decision areas for class 3 are either too thin or too wide near the bottom of plot where most data lie. The small distances and different covariances among classes make it difficult to do separation using the OVR and OVO methods. The MDWD method can provide improvement in this situation as shown by both the test error rate in the second row of Table 3.1 and the illustration in Figure 3.3.

Example 3 also includes three classes with the same sample size as the previous two examples. The distributions of the first two classes are single Gaussians with the first two components of mean vectors as $(-10, 0)$ and $(10, 0)$. However, the third class is a mixture of two Gaussian distributions with the first two components of mean vectors as $(10, 1)$ and $(10, 20)$. We have 60% of the data from the first Gaussian component and the remaining from the second component. The small distance between the first component of class 3 and class 2 makes it difficult to separate these two classes using binary DWD. The MDWD method can provide improvement in this situation. It considers all data points from the

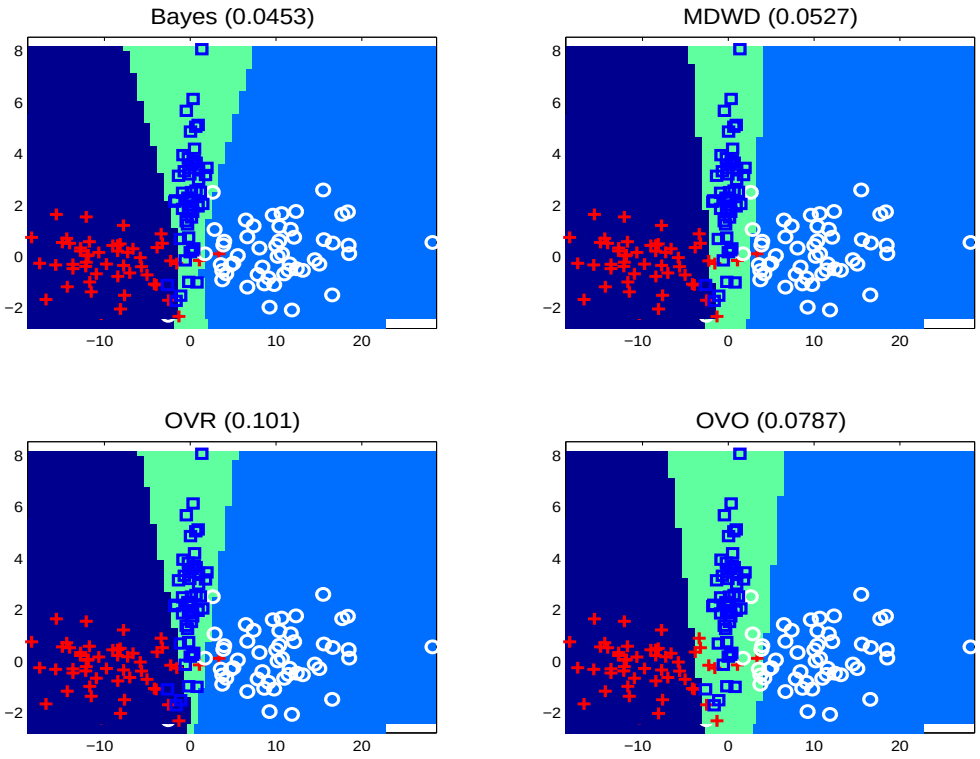


Figure 3.3: Plots of data points and decision boundaries in the first two coordinate axis directions for one training set of Example 2. Upper left panel for Bayes boundary, upper right for MDWD, lower left for OVR, lower right for OVO. The numbers in the parentheses show the test errors for this set.

three classes simultaneously and the impact of the second component in class 3 can help the separation between classes 2 and 3. Thus it improves the test error rate over the OVO method as shown in the third row of Table 1.

All three of the above examples are balanced designs, i.e., the sample size of each class is the same. Examples 4 and 5 are unbalanced cases where different classes have different sample sizes. Example 4 includes three classes with training sample sizes being 50, 20, and 30 respectively. The distributions of the three classes are the same as those in Example 1. The test errors for this example (the fourth row of Table 1) show clear improvement of the MDWD method over the other two. Example 5 includes four classes with training sample sizes being 50, 20, 30, and 10 respectively. The distributions of the four classes are Gaussian with unit covariance and the first three components of the mean vectors being (-

5,0,0), (5,0,0), (0,2,0) and (0,0,2) respectively. The outperformance of the MDWD method over the other methods for this example can be shown in the fifth row of Table 1. Examples 4 and 5 show that the MDWD method can give a big improvement in classification error rate over the OVO and OVR methods in unbalanced situations. This is a quite appealing property of MDWD because real data are often unbalanced.

3.6 Proof

3.6.1 Proof of Theorem 3.1

From Qiao et al. (2010), we get that the minimizer for the ith binary classifier in the OVR DWD method is

$$f_i^* = \frac{1}{\sqrt{C}} \begin{cases} \sqrt{\frac{p_i}{1-p_i}} & \text{if } p_i > \frac{1}{2} \\ 0 & \text{if } p_i = \frac{1}{2} \\ -\sqrt{\frac{1-p_i}{p_i}} & \text{if } p_i < \frac{1}{2}. \end{cases}$$

Thus, we can easily show that $f_i^* > f_j^*$ if $p_i > p_j$ regardless whether p_i is bigger than $\frac{1}{2}$ or not. Hence the Theorem immediately follows. □

3.6.2 Proof of Theorem 3.2

Note that $R(\mathbf{f}) = E(E(V(\mathbf{f}(\mathbf{x}), y)|\mathbf{x}))$, We can minimize $R(\mathbf{f})$ by minimizing $E(V(\mathbf{f}(\mathbf{x}), y)|\mathbf{x})$ for every $\mathbf{x}$. For any fixed $\mathbf{x}$, $E(V(\mathbf{f}(\mathbf{x}), y)|\mathbf{x})$ can be written as $\sum_{j=1}^{K} p_j(\mathbf{x})[\sum_{k\neq j} l(f_{jk}(\mathbf{x}))]$. For any given $\mathbf{X} = \mathbf{x}$, assume that $p_j(\mathbf{x}) > p_k(\mathbf{x})$. Then we can conclude that the solution of $f_j^*(\mathbf{x}) \geq f_k^*(\mathbf{x})$. To show this, suppose that $f_j^*(\mathbf{x}) < f_k^*(\mathbf{x})$; then it is easy to see that switching $f_j^*(\mathbf{x})$ and $f_k^*(\mathbf{x})$ will yield a smaller objective value due to the decreasing

property of l. Without loss of generality, assume that $p_1(\mathbf{x}) > p_2(\mathbf{x}) \geq p_3(\mathbf{x}) \cdots \geq p_K(\mathbf{x})$, which implies that the minimizer must satisfy $f_1^*(\mathbf{x}) \geq f_2^*(\mathbf{x}) \geq \cdots \geq f_K^*(\mathbf{x})$. We need to show that $f_1^*(\mathbf{x}) > f_2^*(\mathbf{x})$. For notational convenience, let $f_k = f_k(\mathbf{x})$ and $p_k = p_k(\mathbf{x})$ for $k = 1, \cdots, K$. Consider $f_1 - f_2 = s_1$, $f_2 - f_3 = s_2$, $\cdots$, $f_{K-1} - f_K = s_{K-1}$. Then the problem reduces to

$$\min_{\mathbf{s}} L(\mathbf{s}) \tag{3.12}$$

$$\text{subject to } s_j \geq 0,\ j = 1, \cdots, K-1, \tag{3.13}$$

where

$$L(\mathbf{s}) = \sum_{k=1}^{K} p_k(l(-s_1 - \cdots - s_{k-1}) + \cdots + l(-s_{k-1}) + l(s_k) + \cdots + l(s_k + \cdots + s_{K-1})).$$

Since the objective function is continuously differentiable and the constraints are linear, the optimal solution $\mathbf{s}^*$ of (3.12) must satisfy Karush-Kuhn-Tucker (KKT) condition, i.e.

$$\begin{aligned}
& \frac{\partial L(\mathbf{s})}{\partial s_i} - \alpha_i \Big|_{\mathbf{s}=\mathbf{s}^*} \\
= & \sum_{k=1}^{i} p_k(l'(s_k^* + \cdots + s_i^*) + \cdots + l'(s_k^* + \cdots + s_{K-1}^*)) \\
& + \sum_{k=i+1}^{K} p_k(-l'(-s_1^* - ... - s_{k-1}^*) - \cdots - l'(-s_i^* - \cdots - s_{k-1}^*)) - \alpha_i \\
= & \sum_{k=1}^{i} p_k(l'(s_k^* + \cdots + s_i^*) + \cdots + l'(s_k^* + \cdots + s_{K-1}^*)) + iC \sum_{k=i+1}^{K} p_k - \alpha_i \\
= & \ 0,
\end{aligned} \tag{3.14}$$

where

$$\alpha_i \geq 0 \text{ and } \alpha_i s_i^* = 0, \text{ for all } i = 1, \cdots K-1.$$

(3.15)

It is sufficient to show that $s_1^* = 0$ is not a minimizer. Toward this end, suppose that $s_1^* = 0$, we have

$$\begin{aligned}\left.\frac{\partial L}{\partial s_1}\right|_{\mathbf{s}=\mathbf{s}^*} &= p_1(l'(s_1^*) + \cdots + l'(s_1^* + \cdots + s_{K-1}^*)) + \sum_{k=2}^{K} Cp_k \\ &= p_1(l'(0) + l'(s_2^*) \cdots + l'(s_2^* \cdots + s_{K-1}^*)) + \sum_{k=2}^{K} Cp_k = \alpha_1, \end{aligned} \tag{3.16}$$

and

$$\begin{aligned}\left.\frac{\partial L}{\partial s_2}\right|_{\mathbf{s}=\mathbf{s}^*} &= p_1(l'(s_1^* + s_2^*) + \cdots + l'(s_1^* + \cdots + s_{K-1}^*)) \\ &\quad + p_2(l'(s_2^*) + \cdots + l'(s_2^* + \cdots + s_{K-1}^*)) + 2\sum_{k=3}^{K} Cp_k \\ &= (p_1 + p_2)(l'(s_2^*) + \cdots + l'(s_2^* + \cdots + s_{K-1}^*)) + 2\sum_{k=3}^{K} Cp_k = \alpha_2. \end{aligned} \tag{3.17}$$

From (3.16) we have

$$l'(s_2^*) + \cdots + l'(s_2^* + \cdots + s_{K-1}^*) = \frac{\alpha_1 - \sum_{k=2}^{K} Cp_k + Cp_1}{p_1} = \frac{\alpha_1 - C + 2Cp_1}{p_1}.$$

Substitute into (3.17), we have

$$\begin{aligned}\alpha_2 &= (p_1 + p_2)\frac{\alpha_1 - C + 2Cp_1}{p_1} + 2\sum_{k=3}^{K} Cp_k \\ &= \frac{(p_1 + p_2)\alpha_1 + C(p_1 - p_2)}{p_1} > 0, \end{aligned} \tag{3.18}$$

which implies $s_2^* = 0$ from the fact that $\alpha_2 s_2^* = 0$.

Suppose that $\alpha_j = 0$ for all $j = 1, \cdots, i-1$. From (3.16), we have

$$l'(s_i^*) + \cdots + l'(s_i^* + \cdots + s_{K-1}^*) = \frac{\alpha_1 - \sum_{k=2}^{K} Cp_k + (i-1)Cp_1}{p_1} = \frac{\alpha_1 - C + iCp_1}{p_1}.$$

Then substitute into the ith formulae, we have

$$\begin{aligned} \alpha_i &= (p_1 + \cdots + p_i)(l'(s_i^*) + \cdots + l'(s_i^* + \cdots + s_{K-1}^*)) + i \sum_{k=i+1}^{K} Cp_k \\ &= (p_1 + \cdots + p_i)\frac{\alpha_1 - C + iCp_1}{p_1} + iC(1 - (p_1 + \cdots + p_i)) \\ &= \frac{(p_1 + \cdots + p_i)\alpha_1 + iCp_1 - C(p_1 + \cdots + p_i)}{p_1} > 0, \end{aligned} \tag{3.19}$$

thus we have $s_i^* = 0$. We conclude that $s_j^* = 0$ for all $j = 1, \cdots, K-1$. But from (3.16), we have that

$$\alpha_1 = (K-1)p_1 l'(0) + \sum_{k=2}^{K} Cp_k = \sum_{k=2}^{K} Cp_k - C(K-1)p_1 < 0,$$

which is contradict to the KKT requirement that $\alpha_1 \geq 0$. Thus $s_1^* = 0$ can not be the minimizer which implies f_1^* is the unique maximum. □

3.7 Discussion

In this chapter we have extended the DWD classification method to the multiclass case. In addition to the OVR and OVO approaches which solve the multiclass problem via a sequence of binary DWD, we have proposed a new MDWD approach which generalizes the binary DWD to a simultaneous multiclass formulation. Our theoretical results show that MDWD is Fisher consistent even in the absence of a dominating class for multiclass problems. The simulation studies show that our MDWD method can always work as well as, and frequently better than, the existing OVR and OVO methods in multiclass problems.

An important direct application of our MDWD is to provide a powerful method for the adjustment of various types of systematic biases such as source and batch effects in microarray experiments. We have demonstrated the usefulness of this method through application to a microarray data set. We recommend MDWD as a general approach for removing systematic bias effects from microarray data and for merging different data sets.

Although our focus in this chapter is on the application of batch adjustment, the proposed MDWD method can also be applied to general multiclass classification problems, as indicated by our simulation studies. An important future research issue is the HDLSS asymptotics. Hall et al. (2005) showed that under certain conditions, there exists a geometric representation of data in the high dimensional case. This representation has been successfully applied to study the asymptotic properties of binary classifiers such as SVM, DWD, and Bi-Drectional Discrimination (BDD) (Hall et al. (2005); Qiao et al. (2010); Huang et al. (2010)). However, no HDLSS asymptotic studies have yet been carried out for any multiclass classifier. In future research, we will use this geometric representation to study the asymptotic behaviors of the proposed multiclass DWD classifier in HDLSS settings.

Bibliography

J. Ahn and J. S. Marron. The direction of maximal data piling in high dimensional space. Submitted, 2005.

Jeongyoun Ahn, J. S. Marron, Keith M. Muller, and Yueh-Yun Chi. The high-dimension, low-sample-size geometric representation holds under mild conditions. *Biometrika*, pages 760–766, August 2007.

A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25: 821–837, 1964.

F. Alizadeh, F. Alizadeh, D. Goldfarb, and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2001.

Orly Alter, Patrick O. Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 97(18):10101–10106, 2000.

J. Anderson and E. Rosenfeld. *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, MA, 1988.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006. (Was Department of Statistics, U.C. Berkeley Technical Report number 638, 2003).

Monica Benito, Joel Parker, Quan Du, Lambert Skoog, Annika Lindblom, Charles M. Perou, and J. S. Marron. Adjustment of systematic microarray data biases. *Bioinformatics*, 20:105–144, 2004.

A Bhattacharjee, W G Richards, J Staunton, C Li, S Monti, P Vasa, C Ladd, J Beheshti, R Bueno, M Gillette, M Loda, G Weber, E J Mark, E S Lander, W Wong, B E Johnson, T R Golub, D J Sugarbaker, and M Meyerson. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences of the United States of America*, 98:13790 – 5, 2001/11/20/ 2001.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory (COLT)*, pages 144–152. ACM Press, 1992.

Erin J. Bredensteiner and Kristin P. Bennett. Multicategory classification by support vector machines. *Computational Optimizations and Applications*, 12:53–79, 1999.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, NY, 1984.

Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

Corinna Cortes and Vladimir Vapnik. Support vector networks. In *Machine Learning*, volume 20, pages 273–297, 1995.

T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27, 1967.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46, 2000.

N. Cristianini and Shawe J. Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, 2000.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

Ronald A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society, A*, 222:309–368, 1922.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

R. Fletcher. *Practical Methods of Optimization*, chapter 8.7 : Polynomial time algorithms, pages 183–188. John Wiley & Sons, New York, second edition, 1987.

Jerome H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, pages 175–165, 1989.

Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

Peter Hall, J. Marron, and Amnon Neeman. Geometric representation of high dimension, low sample size data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3):427–444, 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

Trevor Hastie and Robert Tibshirani. Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society, Series B*, 58:155–176, 1996.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

H. Huang, Y. Liu, and J. S. Marron. Bi-directional discrimination with application to data visualization. Submitted, 2010.

W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, 8(1):118–127, 2007.

S. K. Jung and J. S. Marron. Pca consistency in high dimension, low sample size context. *The Annals of Statistics*, 37:4104–4130, 2009.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–82, 2004.

Jeffrey T Leek, Robert B Scharpf, Heacutector Corrada Bravo, David Simcha, Benjamin Langmead, W Evan Johnson, Donald Geman, Keith Baggerly, and Rafael A Irizarry. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature reviews. Genetics*, 11(10):733–9, 2010. ISSN 1471-0064.

X. Liu, J. Parker, C. Fan, C. M. Perou, and J. S. Marron. Visualization of cross-platform microarray normalization. In A. Scherer, editor, *Batch Effects and Noise in Microarray Experiments: Sources and Solutions*, pages 167–181. Wiley, New York, 2009.

Yufeng Liu. Fisher consistency of multicategory support vector machines. In *Eleventh International Conference on Artificial Intelligence and Statistics*, pages 289–296, 2007.

Yufeng Liu and Xiaotong Shen. Multicategory ψ-learning. *Journal of the American Statistical Association*, 101:500–509, 2006.

Yufeng Liu and Ming Yuan. Reinforced multicategory support vector machines. *Journal of Computational and Graphical Statistics*, page to appear, 2010.

Yufeng Liu, David Neil Hayes, Andrew Nobel, and J. S Marron. Statistical significance of clustering for high-dimension, low-sample size data. *Journal of the American Statistical Association*, 103(483):1281–1293, 2008.

J. S. Marron, M. Todd, and J. Ahn. Distance-weighted discrimination. *Journal of the American Statistical Association*, 102:1267–1271, 2007.

M. Meyerson and D. N. Hayes. Microarray approaches to gene expression analysis. In *Molecular Diagnostics: For the Clinical Laboratorian. 2nd ed. Tsongalis GJ, Coleman WB, eds.*, pages 121–148. Totowa, NJ: Humana Press, 2005.

X. Qiao, H. H. Zhang, Y. Liu, M. J. Todd, and J. S. Marron. Asymptotic properties of distance-weighted discrimination. *Journal of the American Statistical Association*, 105 (489):401–414, 2010.

Xingye Qiao and Yufeng Liu. Adaptive weighted learning for unbalanced multicategory classification. *Biometrics*, 65:159–168, 2009.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004. ISSN 1532-4435.

A. Scherer. *Batch Effects and Noise in Microarray Experiments: Sources and Solutions*. Wiley,New York, 2009.

Andrey A. Shabalin, Hkon Tjelmeland, Cheng Fan, Charles M. Perou, and Andrew B. Nobel. Merging two gene-expression studies via cross-platform normalization. *Bioinformatics*, 24(9):1154–1160, 2008.

Jun Shao. Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422):pp. 486–494, 1993.

TCGA. The cancer genome atlas research network. *http://cancergenome.nih.gov/wwd/pilot_program/research_network/cgcc.asp*, 2010.

Ambuj Tewari and Peter L. Bartlett. On the consistency of multiclass classification methods. In Peter Auer and Ron Meir, editors, *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 2005.

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 99(10):6567–6572, 2002.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, 1995.

V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.

Roel G. Verhaak, Katherine A. Hoadley, Elizabeth Purdom, Victoria Wang, Yuan Qi, Matthew D. Wilkerson, C. Ryan Miller, Li Ding, Todd Golub, Jill P. Mesirov, Gabriele Alexe, Michael Lawrence, Michael O'Kelly, Pablo Tamayo, Barbara A. Weir, Stacey Gabriel, Wendy Winckler, Supriya Gupta, Lakshmi Jakkula, Heidi S. Feiler, J. Graeme Hodgson, C. David James, Jann N. Sarkaria, Cameron Brennan, Ari Kahn, Paul T. Spellman, Richard K. Wilson, Terence P. Speed, Joe W. Gray, Matthew Meyerson, Gad Getz, Charles M. Perou, D. Neil Hayes, and Cancer Genome Atlas Research Network. Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in pdgfra, idh1, egfr, and nf1. *Cancer cell*, 17(1):98–110, 2010.

G. Wahba. *Spline Models for Observing Data*. SIAM, Philadelphia, 1990.

Jörg A. Walter. H–MDS: A new approach for interactive visualization with multidimensional scaling in the hyperbolic space. *Information Systems, Elsevier*, 29(4):274–292, 2004.

Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, pages 219–224, 1999.

Russell D. Wolfinger, Greg Gibson, Elizabeth D. Wolfinger, Lee Bennett, Hisham Hamadeh, Pierre Bushel, Cynthia Afshari, and Richard S. Paules. Assessing gene significance from cdna microarray expression data via mixed models. *Journal of Computational Biology*, 8:625–637, 2001.

Tong Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004.

Hui Zou, Ji Zhu, and Trevor Hastie. New multicategory boosting algorithms based on multicategory fisher-consistent losses. *The Annals of Applied Statistics*, pages 1290–1306, 2008.