Application of Statistical Computing to Statistical Learning

Jude Chukwura Obi

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

The University of Leeds

School of Mathematics

November 2016
The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

© 2016 The University of Leeds and Jude Chukwura Obi

The right of Jude Chukwura Obi to be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.
Acknowledgements

My profound gratitude goes to my supervisors, in the first place to Dr. Peter Thwaites for his understanding, encouragement and patience with me, in my pursuit of a PhD. I appreciate all his indispensable objective criticisms of my work, and sincerely thank God for making him one of my supervisors. May God truly bless him.

In a like manner, I immensely appreciate the crucial contributions of Prof. John Kent to this work. Working with him as a student under his supervision, amply helped me to rediscover my potential; notably my strengths and weaknesses alike. I equally thank God for him, and may God’s blessings be upon him.

My wife, Mrs. Chioma Obi, and Children, the three Chukwus; Chukwudum, Chukwujimaka and Chukwudalu are wonderful companions. Their combined prayers and moral support are immensely appreciated. May God continue to bless them for me.

Above all, I give all thanks and praise to God the Father, for His divine grace, and presence in my life. His tremendous influence in me has made it possible that I am today where I am. To God be all the glory, honour, thanksgiving and praise for ever and ever, Amen.
This study focuses on supervised learning, an aspect of statistical learning. The supervised learning is concerned with prediction, and prediction problems are distinguished by the output predicted. The output of prediction is either a categorical or continuous variable. If the output is a categorical variable, we have classification otherwise what obtains is regression. We therefore identify classification and regression as two prediction tools.

We further identify many features commonly shared by these prediction tools, and as a result, opine that it may be possible to use a regression function in classification or vice versa. Thus, we direct our research towards classification, and intend to:

(i) Compare the differences and similarities between two main classifiers namely, Fisher’s Discriminant Analysis (FDA) and Support Vector Machine (SVM).

(ii) Introduce a regression based classification function, with acronym RDA (Regression Discriminant Analysis).

(iii) Provide proof that RDA and FDA are identical.

(iv) Introduce other classification functions based on multiple regression variants
(ridge regression and Lasso) namely, Lasso Discriminant Analysis (LaDA) and Ridge Regression Discriminant Analysis (RRDA).

We further conduct experiments using real world datasets to verify if the error rates of RDA and FDA on the same datasets are identical or not. We also conduct similar experiments to verify if differences arising from the error rates of using LaDA, RRDA, FDA and Regularized Fisher’s Discriminant Analysis (RFDA) on the same datasets are statistically different from each other or not. In the end, we explore benefits that may derive from the use of LaDA as a classifier, particularly in connection with variable selection.
Contents

Acknowledgements ii

Abstract iv

Contents v

List of Figures x

List of Tables xiii

Definition of Terms xv

Abbreviations xviii

Organization of Work xix

1 Introduction 1

1.1 Regression and Classification Data 5

1.2 Multicollinearity 7

1.2.1 Detection of Multicollinearity 8

1.3 Regularization 12
CONTENTS

1.3.1 Ridge Regression .................................................. 12
  1.3.1.1 Ridge Regression Loss Function and Parameter Estimation ................................................. 14
  1.3.1.2 Expected Value of Ridge Estimator ................. 16
  1.3.1.3 Variance of the Ridge Estimator .................. 16
1.3.2 Least Absolute Shrinkage and Selection Operator (Lasso) .... 18
  1.3.2.1 Lasso Loss Function and Parameter Estimation .... 19
1.3.3 Cross Validation (CV) ............................................. 20
  1.3.3.1 Leave One Out (LOO) Cross Validation ......... 22
1.3.4 Benefits of Regularization ..................................... 22
1.4 Logistic Regression ................................................ 26

2 Classification ......................................................... 30
  2.1 Fisher’s Discriminant Analysis ................................. 31
    2.1.1 Intuitive Approach ........................................ 31
    2.1.2 Fisher’s Procedure ....................................... 32
    2.1.3 Alternative Method ....................................... 35
  2.2 Support Vector Machine (SVM) ................................. 39
    2.2.1 The Idea behind the Support Vector Machine .......... 39
      2.2.1.1 Hyperplane ........................................ 39
      2.2.1.2 Separating Hyperplane ............................. 41
      2.2.1.3 Support Vectors, Margin Lines and Margin ...... 41
      2.2.1.4 Distance from any Arbitrary Point to the Hyperplane .... 42
      2.2.1.5 Determination of SVM Margin ................. 44
    2.2.2 SVM Optimization ........................................ 45
## CONTENTS

2.2.2.1 Linearly Separable or Hard Margin SVM ................................ 46
2.2.2.2 Linearly Non-separable or Soft Margin SVM .................. 50
2.2.2.3 Non-Linear SVM ........................................... 57

2.3 Evaluation of Classifiers .................................................. 63
   2.3.1 The Error Rate ................................................... 64
   2.3.2 Accuracy Rate .................................................... 65
   2.3.3 Receiver Operating Characteristic (ROC) ..................... 66
   2.3.4 Sensitivity and Specificity ..................................... 68

2.4 Class Predictions in Logistic Regression ................................. 70

2.5 Cost-Sensitive Classification ............................................. 72

3 Investigative and Analytical Study of FDA and SVM .................... 75
   3.1 Introduction .......................................................... 75
   3.2 Differences and Similarities between Fisher’s Discriminant Analysis
       and the Support Vector Machine .................................... 78
       3.2.1 Differences based on Theoretical Background ............ 78
       3.2.2 Regularization .................................................. 80
       3.2.3 Linear Transformation ......................................... 82
       3.2.3.1 Effect of Change in Coordinates ...................... 85
       3.2.4 Behaviour of FDA and SVM on Data ....................... 89
       3.2.4.1 Responses of FDA and SVM to Class’s Overlap ..... 89
       3.2.4.2 Effect of Outliers on FDA and SVM ................. 92
       3.2.4.3 Non-Linearily Separable Datasets, and Usefulness
               of FDA and SVM in High Dimensions ............. 94
   3.3 Breaking the SVM ...................................................... 97
CONTENTS

3.3.1 Isotropic FDA ................................................. 98
3.3.2 Regularized FDA .............................................. 99
3.3.3 Equicovariance FDA ......................................... 99
3.3.4 Investigation .................................................. 100

4 Regression Discriminant Analysis (RDA) 103

4.1 Introduction ..................................................... 103
4.2 Mathematical Examination ................................. 104
  4.2.1 Graphical Illustration .................................... 109
4.3 Regression Discriminant Analysis Variants .......... 111
4.4 Estimation of Vector of Coefficients using RRDA and LaDA .. 115
4.5 RRDA and LaDA Trace ........................................ 117
  4.5.1 Some Illustrations ......................................... 118
  4.5.2 Benefits of Using LaDA as a Classifier ............ 121

5 Statistical Investigations 123

5.1 Introduction ..................................................... 123
5.2 Datasets .......................................................... 124
5.3 Comparison of the Error Rates of FDA, RRDA and LaDA ..... 131
  5.3.1 Compliance with the Underlying Assumptions ........ 134
5.4 Comparison of the Error Rates of RFDA, RRDA and LaDA ..... 136
  5.4.1 Compliance with the Underlying Assumptions ........ 140
5.5 Verification of the Effectiveness of Variable Selection Using LaDA 145

6 Summary and Conclusions 149

6.1 Future Work ..................................................... 153
## CONTENTS

**Appendices**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Program in Connection with Evaluation of Classifiers</td>
<td>155</td>
</tr>
<tr>
<td>A.1 Class Predictions in Logistic Regression (Section 2.4)</td>
<td>155</td>
</tr>
<tr>
<td>B Non-linear Classification Program</td>
<td>157</td>
</tr>
<tr>
<td>B.1 Non-Linear Transformation using Radial Basis Function or Gaussian Kernel (Section 3.2.4.3)</td>
<td>157</td>
</tr>
<tr>
<td>B.2 Non-Linear Transformation using a Quadratic Non-Linear Map $\phi(x)$ (Section 3.2.4.3)</td>
<td>159</td>
</tr>
<tr>
<td>C R Codes in Connection with Breaking the SVM (Section 3.3)</td>
<td>161</td>
</tr>
<tr>
<td>C.1 Isotropic FDA</td>
<td>161</td>
</tr>
<tr>
<td>C.2 Equicovariance FDA</td>
<td>163</td>
</tr>
<tr>
<td>C.3 Regularized FDA</td>
<td>166</td>
</tr>
<tr>
<td>C.4 FDA</td>
<td>168</td>
</tr>
<tr>
<td>C.5 SVM</td>
<td>171</td>
</tr>
<tr>
<td>D R Codes used in Chapters 4 and 5</td>
<td>173</td>
</tr>
<tr>
<td>D.1 Regression Discriminant Analysis (RDA) ROC Curve</td>
<td>173</td>
</tr>
<tr>
<td>D.2 Lasso Discriminant Analysis</td>
<td>176</td>
</tr>
<tr>
<td>D.3 Ridge Regression Discriminant Analysis</td>
<td>178</td>
</tr>
<tr>
<td>D.4 Regularized FDA</td>
<td>181</td>
</tr>
<tr>
<td>D.5 Repeated Measures ANOVA</td>
<td>183</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Bias variance trade-off.</td>
<td>24</td>
</tr>
<tr>
<td>2.1</td>
<td>Hyperplanes dividing the red and blue classes into two.</td>
<td>40</td>
</tr>
<tr>
<td>2.2</td>
<td>Graphical illustration of Support Vectors, Margin Lines and Margin.</td>
<td>42</td>
</tr>
<tr>
<td>2.3</td>
<td>Illustration of the distance from any arbitrary point $x_{\alpha}$ say, to $x_h$ on the Hyperplane. The hyperplane is represented by a solid black line separating the red and blue classes.</td>
<td>43</td>
</tr>
<tr>
<td>2.4</td>
<td>A linearly non-separable classification problem, because no hyperplane can separate the two classes without the possibility of misclassifying at least one data point.</td>
<td>51</td>
</tr>
<tr>
<td>2.5</td>
<td>Graphical illustration of how slack variables behave.</td>
<td>52</td>
</tr>
<tr>
<td>2.6</td>
<td>Changes in the margin over different values of $\lambda$, for non-linearly separable datasets.</td>
<td>54</td>
</tr>
<tr>
<td>2.7</td>
<td>A graphical display of relationships among $\alpha_i$, $\xi_i$ and $\lambda$.</td>
<td>55</td>
</tr>
<tr>
<td>2.8</td>
<td>Non-linearly separable datasets.</td>
<td>58</td>
</tr>
<tr>
<td>2.9</td>
<td>A linear classifier in feature space defined by $X_1$ and $X_2$ features, for the dataset of Figure 2.8(a).</td>
<td>59</td>
</tr>
<tr>
<td>2.10</td>
<td>A sample ROC Curve.</td>
<td>68</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

2.11 ROC Curve for simulated dataset. The optimal point on the curve is $\tau_1$, where the vertical line intersects the curve. A corresponding value on the x-axis is 0.6410. At this point, sensitivity plus specificity is at the maximum. .................................................. 71

3.1 Linear transformations with respect to rotation and shearing. The solid green and black lines are respectively the decision boundary of FDA and SVM separating hyperplane. The two lines are superimposed on each other in both (a) and (b). The dotted lines are the SVM margin lines, and data points on the lines are the support vectors. ............................ 83

3.2 A change in behaviour of SVM following a dataset transformation. In (a), we have two data points as support vectors and following a dataset transformation, we obtained three data points support vectors as shown in (b). This shows that SVM is not invariant given some linear transformations. .......................................................... 85

3.3 Illustration of various degrees of class overlap. The dotted black lines are the SVM margin lines and the solid black lines are the separating hyperplanes of SVM. The green solid lines are the decision boundaries of FDA. The support vectors are the data points overlaid by small triangles. In (a), we have a linearly separable dataset but as we go from (b) to (f), we obtain different forms of class overlap. ................................. 90

3.4 Responses of the FDA and SVM to different types of outliers. The outliers include: (a) Y-outliers on the correct side of the plane, (b) X-outliers on the correct side of the plane, and (c) X-outliers on the wrong side of the plane. .................................................. 93
3.5 The behaviour of FDA and SVM on the non-linearly separable datasets in (a). In (b), we observe that the classifiers performed poorly as many data points are misclassified by both classifiers. 95

4.1 Graphical illustrations showing that the classification results of FDA and RDA are the same. 110

4.2 Plots for $L_1$ and $L_2$ functions in a one parameter model. 114

4.3 Graphical illustration of estimation of coefficients using RRDA and LaDA. 116

4.4 RRDA and LaDA traces using a classification based dataset. 117

5.1 Box plots of error rates of FDA, LaDA and RRDA 133

5.2 Normal Q-Q plots in respect of FDA, LaDA and RRDA. 134

5.3 Box plots in respect of the error rates and system time for LaDA, RRDA and RFDA. 139

5.4 Normal Q-Q plots in respect of RFDA, LaDA and RRDA on 25 datasets. 140

5.5 Histograms in respect of system time for RFDA, LaDA and RRDA on 25 datasets. 143
List of Tables

1.1 A 3-fold Cross Validation table in which Round 1 gives a fraction of the dataset used for training the model, and the other for testing the model’s performances. In Round 2, a different fraction of the same dataset was used for model training and the other for testing model’s performances. The same procedure is repeated in Round 3, with entirely different fraction of the dataset for model training, and the other for testing model’s performances. ........................................... 21

2.1 Different types of kernel functions. ........................................... 61

2.2 A sample confusion matrix. ........................................... 65

3.1 Average error rates for each classifier, given different sizes of the training sets. A linear kernel was considered for SVM, which in effect, is as good as using the datasets in their untransformed form. This way, we eliminate the introduction of bias in the error rates of SVM, in comparison with the error rates of the other classifiers. ........................................... 101
4.1 Error rates of RRDA and LaDA on four different datasets. The error rates were obtained using a two fold cross validation, consisting of the training and test sets. For both Australia and Ionosphere datasets, we used 70% of each dataset to train the classifiers and the remaining 30% used to test their performances. Regarding the simulated dataset, 60% of the dataset was used for training the classifiers and the remaining 40% used to test their performances. The data source for HVWON supplied both training and test samples.

5.1 Error rates of FDA, LaDA and RRDA on different datasets

5.2 Error rates of RFDA, LaDA and RRDA, and the system time it took each classifier to finish execution given different datasets.

5.3 Error rates using all the explanatory variables (Full Var), and variables selected with LaDA (LaDA Var) for both FDA and RRDA.
Definition of Terms

$\beta$ is used differently in Chapters 1 and 2. In Chapter 1, it is used to denote the parameters of a regression model, and we estimate them using $\hat{\beta}$. In Chapter 2, it denotes the SVM vector of coefficients. Here, their values $(\beta_0, \beta_1, \cdots, \beta_p)$ are directly obtained from a given set of data. 39

$x_i$ represents the $i^{th}$ row of a data matrix, say $X$, written as a column vector. 42

design matrix is a matrix consisting of different values of the explanatory variables, with provision for the intercept term. The dimensions of this matrix are $n \times p$, where $n$ is the number of the training instances and $p$ is the number of explanatory variables. Often, this matrix is denoted with $X$. 7

explanatory variables is a preferred alternative term for independent variables, because we often observe some form of dependencies among independent variables. Explanatory variables are used to determine or predict the dependent variables, also called output or response variables. They are either supplied by the experimenter, or are output of a different experiment. 4

feature space is a term used to define the expanded features of a dataset, from
the original features in the input space. A feature space is $p$-dimensional, where $p$ is any real number greater than 1. For instance, $X_1$ and $X_2$ can be features of a dataset in the input space; a feature space in $\mathbb{R}^4$ can be defined as $X_1, X_1^2, X_2, X_2^2$. It means that a dataset that was formally in two dimensional input space, is now in four dimensional feature space. 58

**input space** concerns all the features that define a given set of data, and we also refer to these features as explanatory variables. A dataset consists of at least one feature in the input space. 58

**matrix of input data** refers to the data matrix for the training and test sets. 93

**plane** refers to a flat two dimensional surface divided into two, with each half containing either a $+ve$ or $-ve$ class. 43

**predictor variable** is the same as explanatory variable, and it is used interchangeably with explanatory variable. 18

**Statistical tools** are concepts, procedures and methodologies in statistics that help us make sense from data. We can also refer to them as statistical instruments for data analysis. 1

**system time** is the time it takes a programme to run on a computer system. It is often measured in seconds, but can be converted to other modes of measurement as deemed necessary. 132
training instances refer to all the $i^{th}$ rows of the training set. The number of training instances is $n$, where $n$ is also the same as the sample size of the training set. 71

vector of output is the same as the response variables. 93
Abbreviations

**FDA**  Fisher’s Discriminant Analysis.

**SVM**  Support Vector Machines.

**RDA**  Regression Discriminant Analysis.

**LaDA**  Lasso Discriminant Analysis.

**RRDA**  Ridge Regression Discriminant Analysis.

**RFDA**  Regularized Fisher’s Discriminant Analysis.

**IFDA**  Isotropic Fisher’s Discriminant Analysis.

**EFDA**  Equicovariance Fisher’s Discriminant Analysis.

**OLS**  Ordinary Least Squares.
Organization of Work

The thesis consists of five substantive Chapters. Chapters 1 and 2 include introduction and review of related literature. New work is contained in Chapters 3, 4 and 5.

In Chapter 3, we discussed the differences and similarities between FDA and SVM. We also compared the performances of FDA variants with SVM.

Chapter 4 contains discussion on RDA which is also new, and we provided a proof that it is identical to FDA. We also discussed other classification functions based on multiple regression variants. They include RRDA and LaDA.

The new work in Chapter 5 concerns statistical investigation to compare the error rates of FDA, RRDA and LaDA. We also compared the error rates of RFDA, RRDA and LaDA. Another new work here involves the verification of effectiveness of variables selected using LaDA.

In the course of investigation carried out in Chapter 5, we discovered that using the glmnet package in R, we could obtain the best lambda for RFDA. Until now, the procedure that is in use takes a lot of time.
Chapter 1

Introduction

Statistical learning is concerned with knowledge discovery with a connection to statistics. It refers to a vast set of tools for understanding data (James et al., 2013). For instance, we can claim that a new system or procedure for carrying out a set of instructions is better than the existing one, but eventually some statistical tools are needed to assess such a claim. Assessment of the claim may involve analysis of data generated from the system, in order to take an informed decision on how the system now performs. In this respect, learning refers to the statistical tools that help to provide deeper learning or understanding on performances of the system, thereby helping us to understand the system more than before.

Statistical tools can be classified into unsupervised or supervised learning (James et al., 2013). Unsupervised learning lacks the input-output pair interactions (Kyan et al., 2014), which means that input data has no class labels, and no variables to predict, but rather tries to find data structures by their relationship (Pacheco, 2015). The most common unsupervised learning method is cluster analysis (Math-
CHAPTER 1. INTRODUCTION

Works, 2016; Kaufman et al., 2009), which is used for exploratory data analysis to find hidden patterns or grouping in data, and another is density estimation (Scott, 2015). Our interest, however, is on supervised learning, and further discussions will centre around it.

Supervised learning refers to studies in statistics, or the use of statistical tools in which input and output are involved (Bishop, 2007). Such studies cut across parametric and non-parametric settings, and a case in point is prediction. Prediction can be referred to as the use of input in order to generate output (J. Friedman et al., 2001). The outputs are called predictions or predicted values, and usually depend on a set of inputs. The input is either a single explanatory variable or a set of explanatory variables, whereas the output is either a class or continuous variable. In prediction, a model or algorithm is involved, and the procedure for carrying out prediction often requires that a model is trained given a set of inputs (T. J. Hastie et al., 2011). Training here is about making informed choice of a function, that best describes a relation between input and output. Training of models involves the use of training and test sets. A training set is a set of data used to discover potentially predictive relationships, and a test set is also a set of data used to assess the strength of the relationship (Wikipedia, 2016e).

The processes of training, and assessing the performances of a model are often repeated several times, until we access a function that optimally performs. A function is optimised if in comparison with all possible similar functions, that describe a relationship between input and output, given a dataset, it gives the smallest prediction error. Prediction error is the error associated with incorrect prediction of the output. As a statistic, it measures how well or badly a function
has performed given the magnitude of the error, often expressed as a percentage. A function with zero or smallest prediction error in comparison with other similar functions is usually preferred.

The output of prediction is either continuous or categorical variables. With continuous variables as output, we refer to regression (ordinary least squares regression), otherwise what obtains is classification. Regression and classification are therefore two valid tools for prediction. It is important to mention that many characteristics are shared in common by these prediction tools. For instance in OLS regression, as well as classification, a matrix of input and vector of output are required for training and testing of models. Again, since the input in most cases is at least two dimensional, there can be concern for numerical stability of data. Also, in instances where the input is very high dimensional, a procedure that is more robust is used in either case, to obtain a function that optimally performs.

In addition, assessment of prediction tools is via prediction error. The method of calculating this error depends on the prediction tool in question. For instance in OLS regression, we use the mean square error (MSE) whereas error rate is often used in the case of classification. The \( MSE \) is defined as:

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{1.1}
\]

where \( \hat{y} \) refers to the fitted values.

One thing that is of particular interest to us is the fact that both OLS regression and classification functions can be similarly expressed. By this, we mean that the equations describing both functions can be stated similarly. For instance (1.2) and
(1.3) refer to OLS regression and classification functions respectively. In (1.2), we have a vector of coefficients given as 
\[
\begin{pmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_p
\end{pmatrix}^T, 
\]
and a constant term \(b_0\). In (1.3), we similarly have 
\[
\begin{pmatrix}
  w_1 \\
  w_2 \\
  \vdots \\
  w_p
\end{pmatrix}^T 
\]
as a vector of coefficients, with a constant term \(w^T \mu\). Note that \(x_1, x_2, \ldots, x_p\) stand for observations in both cases.

\[
y = f(x) \\
= b_1 x_1 + \cdots + b_p x_p + b_0 
\]

(1.2)

\[
z = g(x) \\
= w^T (x + \mu) \\
= w_1 x_1 + \cdots + w_p x_p + w^T \mu 
\]

(1.3)

Further examination of both (1.2) and (1.3), shows that it may be possible to find a situation where the two coefficient vectors are at least proportional, and the explanatory variables exactly the same. In such a situation, one wonders if it could be possible that any informed alteration of say \(f\), can be useful for predicting \(z\), or informed alteration of \(g\), useful for predicting \(y\).

In line with the foregoing, and particularly considering all the features that both prediction tools share in common, this study is thus motivated. It is our view that a lot can be gained by studying both prediction tools. For instance, it may be possible to discover a way to use OLS regression as a tool for classification or vice versa. For this reason, we aim to sustain the research in discovering any unknown information that may be held in the output of prediction tools, since both tools differ mainly in their output. As a follow-up, we equally target the
following objectives:

- To explore common problems confronting regression and classification.
- To discover how the commonly shared features between regression and classification can help us use each tool in a different environment. For instance, we would like to know if it can be possible to use regression as a tool for classification or vice versa.
- To find out if some regression variants can also function as tools for classification or vice versa.
- To explore differences and similarities between some classification tools.
- To create awareness on why some classification tools work differently, and appear to be very successful.

It is noteworthy that the logistic regression (Cox, 1958) is one study that considered the use of regression in classification. The logistic regression fits a non linear model to a linear combination of explanatory variables. It differs from our main idea because we aim to fit a linear model for classification based on the multiple regression. An in-depth review of the logistic regression is contained in Section 1.4.

1.1 Regression and Classification Data

Let $X$ be data and $x_i$ be the $i^{th}$ row of $X$ written as a column vector. Define a training set

$$D = \{(x_i, y_i)\}_{i=1}^n.$$  \hspace{1cm} (1.4)
where $\mathbf{x}_i \in \mathbb{R}^p$, and $n$ is the size of the training set. Also, $y_i$ is a continuous or categorical response variable. A regression or classification function based on $D$, takes into account two extremes; when $n >> p$ and when $p >> n$. When $n >> p$, strong presence of multicollinearity causes $X$ to be close to singular. Perfect multicollinearity leads to singular $X$, and lastly, for $p >> n$, we have singular $X$ regardless of whether the explanatory variables are linearly independent or not.

When $X$ is close to singular, the estimated values of regression coefficients are very sensitive to slight changes in data, and to the addition or deletion of variables in the regression equation (Chatterjee et al., 2015). The variances associated with the least squares estimate of the regression coefficients are high (Yan, 2009), hence the regression coefficients are unstable, and difficult to interpret (Frost, 2013).

For a classification problem, the outcome of $X$ becoming close to a singular data matrix does not constitute as much of a problem as in regression. However, singular $X$ is a problem to both regression and classification. When $X$ is singular, it is not likely that we obtain the vectors of coefficients defined in both (1.2) and (1.3). As a result, it is unlikely that prediction can take place.

Thus far, it is important to note that multicollinearity can constitute a problem to prediction in different ways. For this reason, in the section that follows, we shall aim to explain what multicollinearity is, and also examine different ways of handling it.
1.2 Multicollinearity

Multicollinearity occurs when there is high correlation between two or more explanatory variables (Mesele, 2016). Any two or more explanatory variables are correlated if there is a linear relationship between the variables. This means that one variable can be linearly predicted from the other variable or variables, with a substantial degree of accuracy (Wikipedia, 2016c). A linear relationship between variables can be either positive or negative. If it is positive, both correlated variables increase or decrease at the same time. If it is negative, as one variable increases, the other decreases or vice versa.

Multicollinearity can also be defined in terms of orthogonality. For instance, when explanatory variables are orthogonal, all eigenvalues of the design matrix are equal to +1 or −1, and the design matrix is of full rank. If at least one eigenvalue is equal to zero, or particularly close to zero, then non-orthogonality exists and this shows that multicollinearity is present (Vinod et al., 1981).

For example, consider the spectral decomposition of a matrix, say $A$:

\[ A = \sum_{j=1}^{p} \lambda_j v_j v_j^T, \quad (1.5) \]

where $\lambda_j$ are the eigenvalues of $A$ and $v_j$ is the eigenvector corresponding to the $j^{th}$ eigenvalue. Then,

\[ A^{-1} = \sum_{j=1}^{p} \lambda_j^{-1} v_j v_j^T. \quad (1.6) \]

If multicollinearity is present, at least one eigenvalue of $A$ will be zero, meaning that $A^{-1}$ does not exist.
As already mentioned, multicollinearity gives rise to unstable regression coefficients. On the other hand, unstable regression coefficients show that the design matrix is ill-conditioned. A matrix is ill-conditioned if the condition number (see Section 1.2.1) is very large. The implication of a high condition number is that small changes in input leads to large changes in response. With a condition number equal to infinity, for instance, the matrix in question is not invertible.

Meanwhile, recall that when there is strong presence of multicollinearity, the least squares regression coefficients have high variances associated with them. However, as observed by (Wikipedia, 2016c; Chatterjee et al., 2015), multicollinearity does not reduce the predictive power or reliability of the model as a whole, at least within the sample data set; it only affects calculations regarding individual predictors. Based on the authors view, a multiple regression model with correlated predictors can indicate how well the entire bundle of predictors predicts the outcome variable, but it may not give valid results about any individual predictor, or about which predictors are redundant with respect to others. For this reason, under condition of severe non-orthogonality, a decision to reject variables on the basis of high p-values may be viewed as misleading.

1.2.1 Detection of Multicollinearity

Different methods for detecting multicollinearity include:

(a) Use of correlation matrix

This involves computation of the correlation matrix of predictors, followed by inspection to discover if pairs of predictors are highly correlated. A correlation ma-
trix consists of correlation coefficients of pairs of predictors. A sample correlation coefficient for any pair of predictors, say, $X_1$ and $X_2$ is given by

$$r_{X_1X_2} = \frac{S_{X_1X_2}}{S_{X_1}S_{X_2}},$$  \hspace{1cm} (1.7)$$

where $S_{X_1X_2}$ is the sample covariance for $X_1$ and $X_2$. Also, $S_{X_1}$ and $S_{X_2}$ are respectively standard deviations. Note that $r_{X_1X_2}$ takes a value between $-1$ and $1$ inclusively. If $r_{X_1X_2}$ is equal to $1$ or $-1$, the two variables are highly positively or negatively correlated. We therefore say that there is a perfect positive or negative linear relationship between $X_1$ and $X_2$. If $r_{X_1X_2} = 0$, then there is no correlation between $X_1$ and $X_2$, and both variables are linearly independent. Although this method may appear easy to use, it has the weakness of not being able to discover the underlying dependency among variables. For this reason, we often make use of more sensitive methods like the variance inflation factor.

(b) **Variance Inflation Factor (VIF)**

VIF quantifies how severe the presence of multicollinearity is. It provides an index for measuring by how much the variance of the least squares regression coefficients is inflated as a result of multicollinearity (Yan, 2009; PennState, 2016).

To calculate VIF, the authors considered a model with $X_k$ as the only predictor, such that

$$y_i = \beta_0 + \beta_k X_{ik} + \varepsilon_i.$$  \hspace{1cm} (1.8)$$

Variance of the estimated coefficient $\hat{\beta}_k$ is
\[ Var(\hat{\beta_k})_{\text{min}} = \frac{\sigma^2}{\sum_{i=1}^{n} (X_{ik} - \bar{X}_k)^2}. \]

The addition of min, the author noted, is to show the smallest variance that can be obtained at this time. Further, a number of correlated predictors are added to (1.8) to obtain a new model,

\[ y_i = \beta_0 + \beta_1 X_{i1} + \beta_k X_{ik} + \cdots + \beta_p X_{ip} + \varepsilon_i. \quad (1.9) \]

Since some of the predictors are correlated with \( X_k \), the variance of \( \hat{\beta_k} \) is now inflated. Hence, it can be shown that

\[ Var(\hat{\beta_k}) = \frac{\sigma^2}{\sum_{i=1}^{n} (X_{ik} - \bar{X}_k)^2} \times \frac{1}{1 - R_k^2}, \]

where \( R_k^2 \) is the value of the coefficient of determination calculated for the regression of the \( k^{th} \) predictor on other predictors. Note that \( R^2 \) is a statistic that measures how close the data are to the fitted regression line.

To determine by how much the variance of \( \hat{\beta_k} \) is now inflated, we use the ratio of the two variances as follows:

\[ \frac{Var(\hat{\beta_k})}{Var(\hat{\beta_k})_{\text{min}}} = \frac{\frac{\sigma^2}{\sum_{i=1}^{n} (X_{ik} - \bar{X}_k)^2} \times \frac{1}{1 - R_k^2}}{\sum_{i=1}^{n} (X_{ik} - \bar{X}_k)^2} = \frac{1}{1 - R_k^2} = VIF. \]

The variance inflation factor is calculated for each individual predictor, and the smallest value it can assume is 1. If \( VIF = 1 \), the \( k^{th} \) predictor is uncorrelated.
with other predictors, hence multicollinearity is not present. If $1 < VIF < 5$, the $k^{th}$ predictor is regarded as moderately correlated with other predictors whereas if $VIF \geq 5$, then we have high correlation which indicates a strong presence of multicollinearity (Chatterjee et al., 2015). In R, packages like car and fmsb can be used to calculate VIF.

(c) The Use of Condition Number

Condition number measures the closeness of a matrix to becoming singular. It can be low, high or infinity. If a condition number is either high or infinity, then there is a strong presence of multicollinearity, and the matrix in question is singular. Different authors use different methods to calculate condition number, but their calculations, especially the ones we have reviewed, are based on the use of eigenvalues. We shall use $K$ with a subscript to denote each individual method. Supposing $\lambda_{max}$ and $\lambda_{min}$ are respectively the maximum and minimum eigenvalues of a correlation matrix, (Vinod et al., 1981) defined a condition number as

$$K_1 = \sqrt{\frac{\lambda_{max}}{\lambda_{min}}}.$$  

(Montgomery et al., 1992) similarly defined it as

$$K_2 = \frac{\lambda_{max}}{\lambda_{min}}.$$  

If $\lambda_{min} = 0$, both $K_1$ and $K_2$ are infinite, and it signifies a very strong presence of multicollinearity, hence the matrix involved is singular. If both $K_1$ and $K_2$ are equal to 1, the predictors are orthogonal and the design matrix is of full rank. If a condition number, $K_2$ for instance, is equal to 5, we have a moderate
multicollinearity but if it is in excess of 10, the presence of multicollinearity is regarded as strong (Pagel et al., 1985).

It is one thing to detect multicollinearity, and another to mitigate its effect. The regularization techniques offer effective means of handling multicollinearity, and, in the section that follows, the techniques shall be discussed.

1.3 Regularization

Regularization is a process of introducing additional information, in order to solve an ill-posed problem or prevent overfitting (Wikipedia, 2016d). A design matrix can be regarded as ill-posed if the inverse does not exist, hence the least squares estimators of the regression parameters cannot be obtained. For instance, consider a data matrix $X$, and define a design matrix $P = X^T X$. If $P^{-1}$ does not exist, then we regard the design matrix as ill-posed.

Some techniques for regularization include the ridge regression and Lasso. We also include cross validation because it helps to obtain optimal ridge and Lasso estimators.

1.3.1 Ridge Regression

Ridge regression doubles as a greedy and shrinkage procedure. As a greedy procedure, all explanatory variables are included in the model, but as also a shrinkage procedure, it ensures that the estimates of the model parameters are shrunk or constrained. It tackles the estimation problem by producing biased estimations, but with small variances (Yan, 2009). Because of this, a model based on ridge
parameter estimation, is more reliable in comparison with the one based on least squares parameter estimation, when multicollinearity is present.

To demonstrate how the procedure works, we consider a singular data matrix $X$, meaning that $X^TX$ is singular, and $(X^TX)^{-1}$ is undefined. Hence, the least squares estimator

$$\hat{\beta}_{ls} = (X^TX)^{-1}X^Ty$$

(1.10)

does not exist. We can obtain a solution to (1.10) by adding $\lambda I$ to $X^TX$, to get

$$\hat{\beta}_{ridge} = (X^TX + \lambda I)^{-1}X^Ty.$$  

(1.11)

We refer to (1.11) as a ridge estimator.

The effect of adding $\lambda I$ to $X^TX$ can be illustrated in the following ways: Let

$$X = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}. \quad (1.12)$$

We can observe that

$$X^TX = \begin{pmatrix} 3 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad (1.13)$$

and the eigenvalues of (1.13) are 5, 1 and 0. Because 0 is also one of the eigenvalues,
(1.13) is singular and the inverse does not exist. Now,

\[
X^T X + I_3 = \begin{pmatrix}
4 & -2 & 1 \\
-2 & 3 & 0 \\
1 & 0 & 2
\end{pmatrix}.
\]  

(1.14)

Eigenvalues of (1.14) are now 6, 2 and 1, meaning that we no longer have a singular matrix. Similarly

\[
X^T X + 2I = \begin{pmatrix}
5 & -2 & 1 \\
-2 & 4 & 0 \\
1 & 0 & 3
\end{pmatrix}.
\]

(1.15)

has eigenvalues 7, 3 and 2.

Based on the foregoing, addition of \(\lambda I\) to \(X^T X\) in (1.11), has the same effect of correcting the inverse problem of \(X^T X\), thereby providing a ridge solution.

### 1.3.1.1 Ridge Regression Loss Function and Parameter Estimation

A loss function refers to a function that is minimized in order to obtain another function that is optimized in some sense. The optimized function itself constitutes a decision function (Wikipedia, 2016b; Vapnik et al., 1998), which is used for prediction. For the ridge regression, the loss function is the same as the standard regression loss function. It is,

\[
l(X) = \|y - X\beta\|^2.
\]

(1.16)
The values of $y$ and $X$ are known, but we do not know $\beta$. To obtain ridge regression estimates for $\beta$, we seek for the estimator that minimizes (1.16), subject to $\|\beta\|^2 \leq s$. In other words, we

$$\text{minimize } \|y - X\beta\|^2 \text{ subject to } \|\beta\|^2 \leq s. \quad \text{(1.17)}$$

Note that $\|\beta\|^2$ refers to an $L_2$ penalty, and $s$ is a small positive constant. The introduction of $s$ is to ensure that the regression coefficients are constrained. Put differently, we are dealing with a constrained optimization. For instance, if $s$ is as large as possible, there is no constraint on the coefficients and what obtains is the least squares solution.

The value of $s$ is unknown and as a result, we introduce the Lagrangian to the optimization problem of (1.17), by writing

$$\text{minimize } \|y - X\beta\|^2 + \lambda \|\beta\|^2.$$

Let

$$\mathcal{L} = \|y - X\beta\|^2 + \lambda \|\beta\|^2$$

$$= (y - X\beta)^T (y - X\beta) + \lambda \|\beta\|^2$$

$$= y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X\beta + \lambda \|\beta\|^2$$

$$= y^T y - 2\beta^T X^T y + \beta^T X^T X\beta + \lambda \|\beta\|^2$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = 0$$
\[-2X^T y + 2X^T X \hat{\beta} + 2\lambda \hat{\beta} = 0\]
\[(X^T X + \lambda I)\hat{\beta} = X^T y\]
\[\hat{\beta}_{\text{ridge}} = (X^T X + \lambda I)^{-1}X^T y.\]

We note that (1.17) is minimized by the ridge regression estimator, \(\hat{\beta}_{\text{ridge}}\) (Hoerl et al., 2000).

### 1.3.1.2 Expected Value of Ridge Estimator

The ridge estimator is a biased estimator because

\[
E\hat{\beta}_{\text{ridge}} = E\left[(X^T X + \lambda I)^{-1}X^T y\right]
= E\left[(I + \lambda (X^T X)^{-1})^{-1} (X^T X)^{-1}X^T y\right]
= E\left[(I + \lambda (X^T X)^{-1})^{-1} \hat{\beta}_{\text{ls}}\right]
= (I + \lambda (X^T X)^{-1})^{-1} E(\hat{\beta}_{\text{ls}})
= (I + \lambda (X^T X)^{-1})^{-1} \beta
\]

(1.18)

Since \(E\hat{\beta}_{\text{ridge}} \neq \beta\), \(\hat{\beta}_{\text{ridge}}\) is biased, and if \(\lambda = 0\), then \(E\hat{\beta}_{\text{ridge}} = E\hat{\beta}_{\text{ls}} = \beta\). It implies that if \(\lambda = 0\), then there is no regularization and we are back to the least squares method.

### 1.3.1.3 Variance of the Ridge Estimator

Observe that the expected value of the ridge estimator (1.18) can be rewritten as

\[
E\hat{\beta}_{\text{ridge}} = (I + \lambda (X^T X)^{-1})^{-1} E\hat{\beta}_{\text{ls}}.
\]
Hence,
\[
\hat{\beta}_{\text{ridge}} = (I + \lambda (X^T X)^{-1})^{-1} \hat{\beta}_{\text{ls}},
\]
(1.19)

Let
\[
W = (I + \lambda (X^T X)^{-1})^{-1},
\]
then,
\[
\hat{\beta}_{\text{ridge}} = W \beta_{\text{ls}}.
\]
(1.20)

\[
\text{Var}(\hat{\beta}_{\text{ridge}}) = \text{Var}(W \hat{\beta}_{\text{ls}}) = W \text{Var}(\hat{\beta}_{\text{ls}}) W^T
\]
\[
= \sigma^2 W (X^T X)^{-1} W^T
\]
\[
= \sigma^2 (I + \lambda (X^T X)^{-1})^{-1} (X^T X)^{-1} \left[(I + \lambda (X^T X)^{-1})^{-1}\right]^T
\]
\[
= \sigma^2 ((X^T X) + \lambda I)^{-1} (X^T X)^{-1} X^T X ((X^T X) + \lambda I)^{-1}
\]
\[
= \sigma^2 ((X^T X) + \lambda I)^{-1} X^T X ((X^T X) + \lambda I)^{-1}.
\]

Next, we show that
\[
\text{Var}(\hat{\beta}_{\text{ridge}}) \leq \text{Var}(\hat{\beta}_{\text{ls}}).
\]
(1.21)

To show that (1.21) holds, we shall first examine the variances of \(\hat{\beta}_{\text{ls}}\) and \(\hat{\beta}_{\text{ridge}}\) when \(X\) is orthogonal.

Recall that under orthogonality,
\[
X^T X = (X^T X)^{-1} = I,
\]
meaning that

\[ \text{Var}(\hat{\beta}_{ls}) = \sigma^2 (X^T X)^{-1} = \sigma^2 I. \]

But,

\[ \text{Var}(\hat{\beta}_{ridge}) = \sigma^2 (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1} \]
\[ = \sigma^2 (I + \lambda I)^{-1} I (I + \lambda I)^{-1} \]
\[ = \sigma^2 I (I + \lambda I)^{-2} \]
\[ = \text{Var}(\hat{\beta}_{ls}) \times (I + \lambda I)^{-2}. \]

The matrix \((I + \lambda I)^{-2}\) has non-negative entries, since only positive values are chosen for \(\lambda\). If \(\lambda\) is 0, a situation where \(\text{Var}(\hat{\beta}_{ridge}) = \text{Var}(\hat{\beta}_{ls})\) obtains, but for non-zero values for \(\lambda\), \(\text{Var}(\hat{\beta}_{ridge}) < \text{Var}(\hat{\beta}_{ls})\). Hence,

\[ \text{Var}(\hat{\beta}_{ridge}) \leq \text{Var}(\hat{\beta}_{ls}). \]

We further argue that since (1.21) holds for orthogonal cases, by induction, it also holds for non-orthogonal cases.

### 1.3.2 Least Absolute Shrinkage and Selection Operator (Lasso)

Lasso, like ridge regression, is a shrinkage procedure (Tibshirani, 1996; T. J. Hastie et al., 2011) and additionally, a variable selector. As a shrinkage procedure, it
places constraints on regression coefficients, thereby keeping them from becoming as large as possible. It achieves this through the use of an \( L_1 \) penalty which also sets some of the coefficients to zero, depending on the predictive relevance of their respective predictor variables, and further shrinks the rest. By setting some coefficients to zero, the predictor variables that are redundant are thus eliminated, thereby enabling variable selection to take place.

1.3.2.1 Lasso Loss Function and Parameter Estimation

Lasso and ridge regression have the same loss functions, but the main difference is in their constraints or penalty terms. While ridge regression uses an \( L_2 \) penalty, Lasso uses an \( L_1 \) penalty given as \( \sum_{j=1}^{p} |\hat{\beta}_j| \). In order to obtain the Lasso coefficients, we seek the estimators that

\[
\minimize \|y - X\beta\|^2 \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s
\]

(1.22)

Here, \( s \) is as previously defined and if it is set to equal \( \sum_{j=1}^{p} |\hat{\beta}_{(ls)}| \), then Lasso estimates are the same as least squares estimate. If it is much smaller, some variables may be set to 0 as a consequence.

As we saw in ridge regression, the value of \( s \) is unknown and by introducing the Lagrangian to (1.22), we can equivalently write:

\[
\minimize \|y - X\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|
\]
Let
\[ J(\beta) = \| y - X\beta \|^2 + \lambda \sum_{j=1}^{p} | \beta_j |, \]
therefore,
\[ \frac{\partial J(\beta)}{\partial \beta} = 0 \quad (1.23) \]
\[ = \frac{\partial}{\partial \beta} ((y - X\beta)^T (y - X\beta)) + \lambda \frac{\partial}{\partial \beta} (| \beta_1 | + | \beta_2 | + \cdots + | \beta_p |) = 0. \quad (1.24) \]

The Lasso coefficients obtained by solving (1.24), but since derivative of the norm of \( \beta_j \) is involved, a number of iterations will be needed for optimization to finally take place. In R, packages like Lasso2 or Lars can solve such an optimization problem.

Ridge regression and Lasso are highly dependent on the regularization parameter \( \lambda \). If \( \lambda = 0 \), we are back to the unconstrained (unregularized) case, but if \( \lambda \to \infty \), the model coefficients shrink towards zero, and the regression curve approaches a horizontal line in \( \mathbb{R}^p \). The predictive accuracy of a model can be improved through optimum choice of \( \lambda \), usually between 0 and \( \infty \). We can obtain this through the use of cross validation.

1.3.3 Cross Validation (CV)

Cross validation is a method used for assessing the performance of a model, and it is based on repetitive training and testing or validation. A cross validation example can be \( K \)-fold, where \( 2 \leq K \leq 10 \) or leave one out CV. For the purpose of illustration, we shall consider the case of \( K = 3 \) fold, and 5 different models.
based on different values of $\lambda$, denoted by $\lambda_m$; $m = 1, 2, \cdots, 5$.

First, we split the dataset into three equally sized parts $F_1, F_2$, and $F_3$ as in Table 1.1. Each model is trained three times, and for each round, $2/3$ of the dataset is used for training, and the remaining data for testing. Prediction error ($PE$) is calculated and noted for each round, using

$$PE = \frac{1}{w} \sum_{i=1}^{w} (y_i - \hat{y}_i)^2,$$

(1.25)

where $\hat{y}_i$ are the fitted values, and $w$ is the size of the test set. The performance of each model is defined by the average prediction error, using

$$\text{ave}(PE) = \frac{1}{3} \sum_{r=1}^{3} PE_r,$$

(1.26)

where $r$ is the number of the round involved in model training and testing. This would be done for each value of $\lambda_m$ and finally, the model with smallest average prediction error is chosen, and the value of $\lambda$ corresponding to it is optimal according to the data-based criterion.

<table>
<thead>
<tr>
<th></th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$PE$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
<td>$PE_1$</td>
</tr>
<tr>
<td>Round 2</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
<td>$PE_2$</td>
</tr>
<tr>
<td>Round 3</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
<td>$PE_3$</td>
</tr>
<tr>
<td>ave(PE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: A 3-fold Cross Validation table in which Round 1 gives a fraction of the dataset used for training the model, and the other for testing the model’s performances. In Round 2, a different fraction of the same dataset was used for model training and the other for testing model’s performances. The same procedure is repeated in Round 3, with entirely different fraction of the dataset for model training, and the other for testing model’s performances.
1.3.3.1 Leave One Out (LOO) Cross Validation

Leave one out cross validation is similar to K-fold CV except that in this case, each data point is considered a fold. It is also similarly constructed. For instance, considering the training set defined in (1.4), since each data point $x_i, y_i$ is a fold, we have a total of $n$ folds. Given any fold, say $x_1, y_1$, a model based on $\lambda_m$ is trained using $\{x_i, y_i\}_{i=1}^{n-1}$ and tested on the fold. The prediction error is similarly calculated as in K-fold CV and noted. We repeat this process using a different fold for testing, and the remaining data points for model training. Again, the prediction error is calculated and noted. The process of training and testing goes on until every data point is used for model testing, and the rest for training the model. The average prediction error is calculated at the end, and it gives the performance of the model based on $\lambda_m$. Over all such models (the number of models depends on $m$), the model with the smallest average prediction error is chosen, and the corresponding $\lambda_m$ is optimal.

1.3.4 Benefits of Regularization

We mentioned that the use of regularization techniques provides effective means of handling multicollinearity. In addition, regularization brings about sparsity, particularly through the use of Lasso. Sparsity holds when there are fewer predictor variables in the prediction model than in the original problem under investigation. It is particularly useful in a high dimensional setting, where it may not be necessary that every variable is included in the model.

Lasso, in contrast to ridge regression gives better interpretation of the model, in
the sense that it helps to know redundant predictor variables, and those that are not. Expressed differently, model construction using Lasso considers only non redundant predictor variables, and, in a situation where we have many redundancies, what obtains is a sparse model.

In trade-off between bias and variance, regularization plays an important role. Recall that when multicollinearity is present, estimates of regression coefficients using least squares have inflated variances associating with them. In reducing the variances, ridge regression introduces some bias. But via optimal choice of regularization parameter ($\lambda$), the variances are reduced to a point where $MSE$ of the biased model is lower than the $MSE$ of the unbiased model. This procedure ensures that we have a trade-off between bias and variance.

To illustrate this procedure, we first explain what a model complexity is. Here, we shall consider a dataset that is fixed, with fixed $n$, and varying $p$. A model complexity therefore is a function of the parameters of the model. As the number of model parameters becomes larger, meaning that we increase $p$, the model increases in complexity as shown in Figure 1.1.

We observe that as a model increases in complexity, variances associated with the parameters increase, followed by a decrease in bias. On the other hand, a simple model is characterized by a small model complexity, with attendant low variance and increase in bias. For this reason, it under-fits because it lacks flexibility in capturing the true input-output relationship. An optimized model therefore can be viewed as the one that considers a trade-off between bias and variance. The use of regularization helps to obtain such a model. It achieves this by utilizing an optimum value of $\lambda$, particularly through the use of cross validation.
Another way to visualize the importance of regularization is to examine the error that associated with prediction. For instance, consider the following regression model:

\[ y = g(X) + \varepsilon, \]  

(1.27)

where \( g \) is an unknown function of \( X = (X_1, X_2, \cdots, X_p) \). Here \( \varepsilon \) is a random error term that is not dependent on the data, with mean 0 and variance \( \sigma^2 \). Let \( \hat{g} \) be a fitted regression function of \( g \), and let

\[ \hat{y} = \hat{g}(X). \]  

(1.28)

The accuracy of \( \hat{y} \) in predicting \( y \) is determined by two errors, namely reducible and irreducible. For instance, assuming that \( \hat{g} \) is not a perfect estimate for \( g \), error introduced is known as reducible error. It is reducible because with optimal choice of \( \hat{g} \), the prediction of \( g \) can be improved.

As noted by (James et al., 2013), there can be instances where a perfect estimate
is secured for $g$, meaning that the response takes the form $\hat{y} = g(X)$, but a kind of error still obtains. The place of this error is understandable, if we recall that by definition $\varepsilon$ is a random error term, and independent of $X$. The variability associated with it is unpredictable, and can affect the accuracy of the model. Such unpredictable error is called irreducible because there is no known technique to reasonably influence it. The authors further remarked that irreducible error is usually larger than zero, because at times it contains variables that are unmeasurable, but useful for the prediction of $y$.

To provide further illustration, we make reference to (1.27) and (1.28), and show that the expected value of the residual sum of square consists of reducible and irreducible errors. In other words, we show that

$$E (y - \hat{y})^2 = (g(X) - \hat{g}(X))^2 + \text{Var} (\varepsilon)$$

(1.29)

Now,

$$E (y - \hat{y})^2 = E (g(X) + \varepsilon - \hat{g}(X))^2$$

$$= E (\varepsilon + (g(X) - \hat{g}(X)))^2$$

$$= E \left[ \varepsilon^2 + (g(X) - \hat{g}(X))^2 + 2\varepsilon (g(X) - \hat{g}(X)) \right]$$

$$= E(\varepsilon^2) + E (g(X) - \hat{g}(X))^2 + 2E(\varepsilon) (g(X) - \hat{g}(X))$$

Recall that $E(\varepsilon) = 0$, therefore $2E(\varepsilon) (g(X) - \hat{g}(X)) = 0$. Also

$$E(\varepsilon^2) = E (y - g(X))^2$$
\[ = E (y - E(y))^2; \text{ since } E(y) = g(X) \]
\[ = Var(\varepsilon) \]

Finally,

\[ E (y - \hat{y})^2 = E(\varepsilon^2) + E(g(X) - \hat{g}(X))^2 \]
\[ = (g(X) - \hat{g}(X))^2 + Var(\varepsilon) \]

Reducible + Irreducible

The statistic \( E(y - \hat{y})^2 \) represents the average or expected value of the squared difference between fitted values \( \hat{y} \) and observed values \( y \). The variance defined by the error term (\( \varepsilon \)), remains unpredictable. To improve prediction accuracy, we aim to minimize the reducible error as much as possible. The best way to achieve this is to make optimal choice of \( \hat{g} \), often via the use regularization techniques, with emphasis on application of cross validation to enable optimum choice of \( \lambda \).

### 1.4 Logistic Regression

Logistic regression is used to model the probability of an event occurring given a set of explanatory variables. The explanatory variables can be discrete, continuous, or both, and the response variables do not necessarily follow a normal distribution. The logistic regression model is useful for predicting the probabilities of an dichotomous event. Instances of such event include approval or non approval, pass or failure, survival or non survival etc. The procedure for prediction of probabilities involve the use of a link function. The function will link or tie
a linear combination of the independent variables to the probability distribution of
the response variables. Such probability distribution follows an exponential fam-
ily, namely normal, binomial or Poisson distribution etc. The link function here is
called the logit, i.e., the natural log of the odds.

In addition to the prediction of probabilities, the logistic regression model helps to
obtain the odds of an event. Since the response variables do not necessarily follow
a normal distribution, if datasets do not comply with the normality assumption,
the logistic regression function can be superior to a classification function that is
based on compliance with the normality assumption.

Assumptions

• The response variables do not necessarily follow a normal distribution, but
  any distribution from an exponential family (e.g. Binomial, Normal, Poisson,
  etc).

• The response variables, \( y_1, y_2, \cdots, y_n \) are independently distributed. This
  means that cases or observations are independent.

• A linear relationship is not assumed between the response variable and the
  explanatory variables, but it is assumed between the logit of the response
  and the explanatory variables (e.g. \( \logit(p) = \beta_0 + \beta^T x \)).

• The homogeneity of variances does not need to be satisfied.

• Errors are assumed to be independent but not normally distributed.

• Parameter estimation is carried out using the maximum likelihood estimation
  (MLE), and not the ordinary least squares (OLS). As a result, estimation
relies on large sample approximations.

Some Terminologies

Some commonly used terminologies in the study of a logistic regression include:

- **Odds**: The odds is defined as

\[
\text{Odds} = \frac{p(\text{occurring})}{p(\text{not occurring})} = \frac{p}{1 - p} \tag{1.30}
\]

- **Odds Ratio**: This is defined as the ratio of two odds. Thus,

\[
\text{Odds Ratio} = \frac{\text{Odds}_1}{\text{Odds}_0} = \frac{\frac{p_1}{1-p_1}}{\frac{p_0}{1-p_0}} = \frac{p_1 (1 - p_0)}{p_0 (1 - p_1)}, \tag{1.31}
\]

where 1 references the current odds whereas 0 refers to the former odds.

In logistic regression, the odds ratio for a variable represents how the odds change following a unit increase in that variable, with all other variables held constant.

- **The logit**: We previously defined the logit as the natural logarithm of the odds. Thus,

\[
\text{logit} (p) = \ln \left( \frac{p}{1 - p} \right) = \beta_0 + \beta^T \mathbf{x}. \tag{1.32}
\]

Note that (1.32) shows that the natural logarithm of the odds gives a linear function of the explanatory variables. If we take the antilog of the logit function, we obtain:

\[
(\text{logit} (p))^{-1} = \frac{p}{1 - p} = e^{\beta_0 + \beta^T \mathbf{x}}
\]
We can refer to (1.33) as the predictions of the logistic regression. Parameter estimation here is via the maximum likelihood method, and not the method of least squares as in the case of standard regression. For this reason, parameter estimates are more reliable when larger datasets are involved.

The logistic regression does not have established procedure like FDA or SVM for finding the classification threshold that is optimal. For this reason, it is not immediately possible to obtain the error rate assuming we aim to do so. To overcome this deficiency, the Receiver Operating Characteristic (ROC) is often used, and it helps to obtain an optimal classification threshold for the logistic regression function. The classification threshold can be useful for obtaining the error rate. In Section 2.3, we shall discuss more about the ROC and class predictions in logistic regression (Section 2.4).
Chapter 2

Classification

Classification refers to the prediction of class outcome variables (Johnson et al., 2002), using a classification function. A classification function is the outcome of discrimination, and discrimination is the use of a set of labelled classes, otherwise called a training set, to construct a classifier (or allocation rule) that separates the predefined classes as much as possible (Izenman, 2008). Put another way, discrimination is concerned with the problem of class separation, whereas classification aims to allocate unlabelled input to a class it belongs. For instance, consider the training set $D$ defined in (1.4), and assuming that we partition it into $K$ labelled classes $c_k$, where $k = 1, 2, \ldots, K$; the goal of classification is to take an input vector $x$, and assign it to one of the $K$ classes (Bishop, 2007). The classes are assumed to be disjoint, meaning that each $x_i$ is assigned to one and only one class.

The problem often encountered in classification concerns making a suitable choice of a classification function. The success of a classification function depends on the dataset, because different classification functions perform differently given different
datasets. A classification function can be constructed using different methods, but the merits and demerits of each method, in most cases, also depend on the dataset.

In this Chapter, we shall review two different methods of constructing classification functions, namely Fisher’s Discriminant Analysis (FDA), and Support Vector Machines (SVM). Our interest in both FDA and SVM is because they have been extensively studied in the fields of statistics and machine learning. For this reason, in Chapter 3, we shall investigate their strengths and weaknesses, as well as their differences and similarities.

2.1 Fisher’s Discriminant Analysis

In relation to (1.3), if we let \( w = d^T \hat{\Sigma}^{-1} \) and \( \mu = -\frac{1}{2} (\bar{x}_1 + \bar{x}_2) \), where \( d = \bar{x}_1 - \bar{x}_2 \), \( \bar{x}_1 \) and \( \bar{x}_2 \) are respectively class 1 and class 2 mean vectors, \( \hat{\Sigma} \) is the pooled covariance matrix for the two classes, then what obtains is Fisher’s discriminant function. We shall consider two different methods of deriving the function.

2.1.1 Intuitive Approach

Let us assume we have two classes of independent \( p \)-dimensional samples, from two multivariate populations with common covariance matrix \( \Sigma \), and unknown population mean vectors \( \mu_1 \) and \( \mu_2 \). Let \( \mathbf{x} \in \mathbb{R}^p \) be a \( p \)-dimensional input vector. Our goal is to classify \( \mathbf{x} \) into one of the classes based on its features. Intuitively, we would like to allocate \( \mathbf{x} \) to the class whose mean is closest to \( \mathbf{x} \). For instance,
\( \mathbf{x} \) is allocated to class +1 if
\[
\| \mathbf{x} - \mu_1 \| < \| \mathbf{x} - \mu_2 \|. 
\]
The same result obtains if we square both sides. Thus,
\[
\| \mathbf{x} - \mu_1 \|^2 < \| \mathbf{x} - \mu_2 \|^2
\]
\[
(\mathbf{x} - \mu_1)^T(\mathbf{x} - \mu_1) < (\mathbf{x} - \mu_2)^T(\mathbf{x} - \mu_2)
\]
\[
\mathbf{x}^T\mathbf{x} - 2\mu_1^T\mathbf{x} + \mu_1^T\mu_1 < \mathbf{x}^T\mathbf{x} - 2\mu_2^T\mathbf{x} + \mu_2^T\mu_2
\]
\[
2(\mu_2 - \mu_1)^T\mathbf{x} < \mu_2^T\mu_2 - \mu_1^T\mu_1
\]
\[
(\mu_2 - \mu_1)^T\mathbf{x} < \frac{1}{2}(\mu_2 - \mu_1)^T(\mu_2 + \mu_1)
\]
(2.1)
\[
(\mu_2 - \mu_1)^T\left[ \mathbf{x} - \frac{1}{2}(\mu_2 + \mu_1) \right] < 0
\]
(2.2)
Note that (2.2) holds if \( \mathbf{x} \) is closer to \( \mu_1 \) than \( \mu_2 \), otherwise the inequality sign changes to less than zero. The decision boundary is the boundary for which \((\mu_1 - \mu_2)^T[\mathbf{x} - \frac{1}{2}(\mu_2 + \mu_1)] = 0\). This boundary passes through the midpoint of the class means \((\frac{1}{2}(\mu_1 + \mu_2))\), and is orthogonal to \( \mu_1 - \mu_2 \).

2.1.2 Fisher’s Procedure

The problem with (2.2) is the failure to handle instances where variables are measured in different units. More exactly, there is no attempt to unify the effect of
possible different units of measurements. Also it is unable to account for high variations with some variables. For instance, some variables may be more highly variable than others, and tend to dominate the Euclidean distance. As a result, the decision boundary may not be optimally determined, hence a poor classification result obtains.

In order to correct these weaknesses, (Fisher, 1936) introduced the idea of standardizing both data and class means. A further assumption that the two classes have common covariance matrix $\Sigma$ is made. Standardization is carried out using a symmetric square root matrix $\Sigma^{-\frac{1}{2}}$, thereafter $x$ is allocated to class $+1$ if

\[
\left\| \Sigma^{-\frac{1}{2}}x - \Sigma^{-\frac{1}{2}}\mu_1 \right\| < \left\| \Sigma^{-\frac{1}{2}}x - \Sigma^{-\frac{1}{2}}\mu_2 \right\|, \\
\left\| \Sigma^{-\frac{1}{2}}x - \Sigma^{-\frac{1}{2}}\mu_1 \right\|^2 < \left\| \Sigma^{-\frac{1}{2}}x - \Sigma^{-\frac{1}{2}}\mu_2 \right\|^2, \\
(x - \mu_1)^T \Sigma^{-1}(x - \mu_1) < (x - \mu_2)^T \Sigma^{-1}(x - \mu_2). \tag{2.3}
\]

The use of $\Sigma^{-\frac{1}{2}}$ to effect standardization is only procedural, because following some algebraic operations, we have $\Sigma^{-1}$ as in (2.3). For this reason, the distance measure changes to Mahalanobis distance as against the Euclidean distance of (2.1). As noted by (McLachlan, 1999), the Mahalanobis distance is able to take care of the different scales on which data may be measured.

Now,

\[
x^T \Sigma^{-1}x - 2\mu_1^T \Sigma^{-1}x + \mu_1^T \Sigma^{-1}\mu_1 < x^T \Sigma^{-1}x - 2\mu_2^T \Sigma^{-1}x + \mu_2^T \Sigma^{-1}\mu_2 \\
2(\mu_2 - \mu_1)^T \Sigma^{-1}x < (\mu_2^T \Sigma^{-1}\mu_2) - (\mu_1^T \Sigma^{-1}\mu_1) \\
(\mu_2 - \mu_1)^T \Sigma^{-1}x < \frac{1}{2}(\mu_2 - \mu_1)^T \Sigma^{-1}(\mu_2 + \mu_1)
\]
\[(\mu_2 - \mu_1)^T \Sigma^{-1} x - \frac{1}{2}(\mu_2 - \mu_1)^T \Sigma^{-1}(\mu_2 + \mu_1) < 0\]
\[(\mu_2 - \mu_1)^T \Sigma^{-1} \left[ x - \frac{1}{2}(\mu_1 + \mu_2) \right] < 0\]
\[(\mu_1 - \mu_2)^T \Sigma^{-1} \left[ x - \frac{1}{2}(\mu_1 + \mu_2) \right] > 0. \quad (2.4)\]

The difference between (2.2) and (2.4) is the outcome of standardization we introduced. Similar to the explanation given in Section 2.1.1, if (2.4) holds, it means that \(x\) is closer to \(\mu_1\) than to \(\mu_2\), and allocation is in favour of class +1. Otherwise, the inequality sign changes to less than zero, and \(x\) is allocated to class -1. Also, the decision boundary between the two classes is similarly given by
\[(\mu_1 - \mu_2)^T \Sigma^{-1} \left[ x - \frac{1}{2}(\mu_1 + \mu_2) \right] = 0; \quad (2.5)\]

it passes through \(\frac{1}{2}(\mu_1 + \mu_2)\), and is orthogonal to \(\Sigma^{-1}(\mu_1 - \mu_2)\).

In practice, values of population parameters are not known, and we estimate them from sample data. The estimate of (2.5) is
\[(\bar{x}_1 - \bar{x}_2)^T \hat{\Sigma}^{-1} \left[ x - \frac{1}{2}(\bar{x}_1 + \bar{x}_2) \right] = 0 \quad (2.6)\]

where
\[\bar{x}_1 = \frac{\sum_{i=1}^{n_1} x_{1i}}{n_1}; \quad \bar{x}_2 = \frac{\sum_{i=1}^{n_2} x_{2i}}{n_2},\]

\(n_1\) and \(n_2\) are sample sizes respectively for classes 1 and 2. Also,
\[\hat{\Sigma} = \left( \frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1)} \right) S_1 + \left( \frac{n_2 - 1}{(n_1 - 1) + (n_2 - 1)} \right) S_2 \]
\[= \frac{(n_1 - 1)S_1 + (n_2 - 1)S_2}{n_1 + n_2 - 2};\]
\[ S_1 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (x_{1i} - \bar{x}_1)(x_{1i} - \bar{x}_1)^T, \]

and

\[ S_2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (x_{2i} - \bar{x}_2)(x_{2i} - \bar{x}_2)^T. \]

Further expansion of (2.6) leads to

\[(\bar{x}_1 - \bar{x}_2)^T \hat{\Sigma}^{-1} x = \frac{1}{2}(\bar{x}_1 - \bar{x}_2)^T \hat{\Sigma}^{-1}(\bar{x}_1 + \bar{x}_2).\]

If we let \( d = (\bar{x}_1 - \bar{x}_2), \) and \( \bar{x} = \frac{\bar{x}_1 + \bar{x}_2}{2}, \) then

\[ d^T \hat{\Sigma}^{-1} x = d^T \hat{\Sigma}^{-1} \bar{x}. \]

Let \( m = d^T \hat{\Sigma}^{-1} \bar{x}, \) the Fishers discriminant function can be written as

\[ g(x) = d^T \hat{\Sigma}^{-1} x \quad (2.7) \]

or

\[ g(x^*) = d^T \hat{\Sigma}^{-1} (x - \bar{x}) \quad (2.8) \]

Given (2.7), \( x \) is allocated to class 1 if \( g(x) \geq m, \) otherwise, it is allocated to class 2. If we consider (2.8), allocation is in favour of class 1 if \( g(x^*) \geq 0, \) or class 2 otherwise.

### 2.1.3 Alternative Method

Another way to understand FDA is to consider a linear combination of features, otherwise called the discriminant score (L), defined on the training set \( D \) as follows:
where, $x_1, x_2, \ldots, x_p$ are values assumed by the predictor variables in the discriminant score, and $b_1, b_2, \ldots, b_p$ are coefficients of the discriminant score. Consider a partitioning of $D$ into $K = 2$ discrete classes, meaning that only two classes are involved; we can define $L$ for each of the two classes as follows:

$L_{1i} = b^T x_{1ij} = b_1 x_{1i1} + b_2 x_{1i2} + \cdots + b_p x_{1ip}; \quad i = 1, 2, \ldots, n_1$

$L_{2i} = b^T x_{2ij} = b_1 x_{2i1} + b_2 x_{2i2} + \cdots + b_p x_{2ip}; \quad i = (n_1 + 1), (n_1 + 2), \ldots, (n_1 + n_2)$.

Note that the training set $D = D_1 + D_2$, where $D_1$ and $D_2$ are respectively training samples for classes +1 and −1. Similarly, $n = n_1 + n_2$, where $n_1$ is the sample size for class +1 and $n_2$ the sample size for class −1. Now,

$\bar{L}_1 = \frac{\sum_{i=1}^{n_1} L_{1i}}{n_1} = b^T \bar{x}_1$, and $\bar{L}_2 = \frac{\sum_{i=(n_1+1)}^{n} L_{2i}}{n_2} = b^T \bar{x}_2$.

Similarly,

$\bar{x}_1 = \frac{\sum_{i=1}^{n_1} x_{1i}}{n_1}$ and $\bar{x}_2 = \frac{\sum_{i=(n_1+1)}^{n} x_{2i}}{n_2}$.

The variance associated with $L_1$ can be written as,

$s_1^2 = \sum_{x_i \in D_1} (b^T x_i - b^T \bar{x}_1)^2$

$= \sum_{x_i \in D_1} (b^T (x_i - \bar{x}_1))^2$

$= b^T \left( \sum_{x_i \in D_1} (x_i - \bar{x}_1) (x_i - \bar{x}_1)^T \right) b$

$= b^T S_1 b.$ \hfill (2.10)

L = b_1 x_1 + b_2 x_2 + \cdots + b_p x_p, \hfill (2.9)
Also the variance associated with \( L_2 \) can be written as,

\[
s_2^2 = \sum_{x_i \in D_2} (b^T x_i - b^T \bar{x}_2)^2
\]

\[
= \sum_{x_i \in D_2} (b^T (x_i - \bar{x}_2))^2
\]

\[
= b^T \left( \sum_{x_i \in D_2} (x_i - \bar{x}_2)(x_i - \bar{x}_2)^T \right) b
\]

\[
= b^T S_2 b. \tag{2.11}
\]

Both \( S_1 \) and \( S_2 \) are scatter matrices respectively for \( D_1 \) and \( D_2 \). Combining (2.10) and (2.11),

\[
s_1^2 + s_2^2 = b^T S_1 b + b^T S_2 b
\]

\[
= b^T (S_1 + S_2) b
\]

\[
= b^T S_W b, \tag{2.12}
\]

where \( S_W \) is called the within group scatter matrix.

Also,

\[
(\bar{L}_1 - \bar{L}_2)^2 = (b^T (\bar{x}_1 - \bar{x}_2))^2
\]

\[
= b^T \left( (\bar{x}_1 - \bar{x}_2)(\bar{x}_1 - \bar{x}_2)^T \right) b
\]

\[
= b^T S_B b, \tag{2.13}
\]

where \( S_B = (\bar{x}_1 - \bar{x}_2)(\bar{x}_1 - \bar{x}_2)^T \), is called the between group scatter matrix. The goal of FDA is to find the vector, say \( b^* \), that maximizes the standardized squared
CHAPTER 2. CLASSIFICATION

distance (Rencher, 2003) between $L_1$ and $L_2$, i.e.,

\[
\frac{(\bar{L}_1 - \bar{L}_2)^2}{s_1^2 + s_2^2} = \frac{b^T S_B b}{b^T S_W b}.
\]

(2.14)

We find,

\[
\max_b L(b) = \frac{b^T S_B b}{b^T S_W b}.
\]

(2.15)

Recall that when $f(x)$ and $g(x)$ are functions, for instance,

\[
\frac{d}{dx} \left( \frac{f(x)}{g(x)} \right) = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}.
\]

Therefore setting the derivative equal to zero,

\[
\frac{d}{db} L(b) = \frac{d}{db} \left( \frac{b^T S_B b}{b^T S_W b} \right) = 0 \quad \text{gives}
\]

\[
\frac{2S_B b b^T S_W b - 2b^T S_B b S_W b}{(b^T S_W b)^2} = 0
\]

\[
S_B b = \frac{b^T S_B b}{b^T S_W b} S_W b
\]

\[
= \lambda S_W b; \quad \lambda = \frac{b^T S_B b}{b^T S_W b}
\]

\[
S_W^{-1} S_B b = \lambda b
\]

(2.16)

Equation (2.16) is a generalized eigenvalue problem, where $\lambda$ is a generalized eigenvalue of $S_W^{-1} S_B$. If $S_W$ is non-singular, the optimum vector that maximizes (2.15) is the largest or dominant eigenvector of $S_W^{-1} S_B$, corresponding to the largest eigenvalue. This dominant eigenvector can be given as

\[
b^* \propto d^T \hat{\Sigma}^{-1},
\]

(2.17)
meaning that, \( L = d^T \hat{\Sigma}^{-1} x \).

## 2.2 Support Vector Machine (SVM)

The support vector machine (Cortes et al., 1995) is a binary classifier from the field of machine learning. It has a strong geometric connection, and is a concept based on the hyperplane. SVM has been successfully applied to numerous classification problems (Section 3.1), hence our interest in studying it alongside FDA. In the first place, we shall examine the idea behind it.

### 2.2.1 The Idea behind the Support Vector Machine

The idea behind SVM centres around understanding the meaning of a hyperplane. We tie discussions on hyperplanes to a dataset, and therefore recall the training set \( D \) defined in (1.4). In accordance,

\[
D = \{ x_i, y_i \}_{i=1}^n, \quad x_i \in \mathbb{R}^p,
\]

where \( y_i \in \{+1, -1\} \) denote the +ve and −ve classes.

#### 2.2.1.1 Hyperplane

A hyperplane is a set of all points \( x \in \mathbb{R}^p \), that satisfy the equation \( h(x) = 0 \) (Zaki et al., 2014), where \( h(x) \) is the function of the hyperplane defined by

\[
h(x) = \beta^T x + \beta_0
\]
\[ = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \beta_0. \quad (2.19) \]

Here, $\beta$ is a $p$-dimensional weight vector and $\beta_0$ is a scalar. In $\mathbb{R}^2$, a hyperplane is a line, in $\mathbb{R}^3$ it is a plane, whereas in $\mathbb{R}^p$, it is a $(p - 1)$ dimensional subspace.

In $\mathbb{R}^2$ for instance, we can use an infinite number of hyperplanes to divide any two classes into two. However, only one hyperplane can be regarded as optimum, in the sense that it maintains the greatest distance (Section 2.2.1.4) between the nearest data points from the $+ve$ and $-ve$ classes. For instance, in Figure 2.1, each of the solid black lines is a hyperplane dividing the red and blue classes into two, but only one is optimum. This optimum hyperplane lies in the middle of the nearest data points from both the $+ve$ and $-ve$ classes, thereby maintaining the greatest distance between the data points.

Figure 2.1: Hyperplanes dividing the red and blue classes into two.
2.2.1.2 Separating Hyperplane

Assuming that \( p = 2 \), the hyperplane defined in (2.19) can be used to divide \( D \) into two. Supposing that linear separability exists given \( D \), meaning that it is possible to use a line to completely separate the two classes in \( D \), then we can be certain that there is a separating hyperplane defined by (2.19), such that for all points labelled +1, \( h(\mathbf{x}) \geq 0 \) otherwise, \( h(\mathbf{x}) < 0 \). Thus given the input \( \mathbf{x}_i \), \( h(\mathbf{x}) \) can be used to determine class membership in the following way;

\[
y_i = \begin{cases} 
+1 & \text{if } h(\mathbf{x}_i) \geq 0 \\
-1 & \text{if } h(\mathbf{x}_i) < 0.
\end{cases}
\] (2.20)

The separating hyperplane is optimum, because it maintains the largest distance between the closest data points from the +ve and −ve classes. These closest data points are critical to understanding how the SVM behaves, and they help to define other important concepts such as the support vectors, margin lines and margin.

2.2.1.3 Support Vectors, Margin Lines and Margin

Support vectors are the data points from the +ve and −ve classes, that are closest to the separating hyperplane. They are unique, given a separating hyperplane. For instance in Figure 2.1, many hyperplanes can divide the red and blue classes into two, but only one hyperplane uniquely identifies with the support vectors. This hyperplane passes midway through the support vectors. In other words, given a classification problem, we can find only one unique set of support vectors, otherwise the ensuing hyperplane is not optimal. A support vector can be called
a $+ve$ support vector, if the vector is from the positive class, otherwise it is called a $-ve$ support vector.

Hyperplanes that pass through the support vectors are called margin lines. A $+ve$ margin line passes through the $+ve$ support vectors, whereas a $-ve$ margin line passes through the $-ve$ support vectors. The distance between the two margin lines is called the margin. The margin is very critical in formulating the SVM algorithm, and in the sections that follow, we shall discuss how to find it. In Figure 2.2, we displayed a two dimensional graphical illustration of support vectors, margin lines and the margin.

2.2.1.4 Distance from any Arbitrary Point to the Hyperplane

Before we determine the SVM margin, we shall first find the distance between a hyperplane and any arbitrary point that is not on the hyperplane. In Figure 2.3, $x_a$ is any arbitrary point that is not on the hyperplane, and $x_h$ is the orthogonal
projection of $x_\alpha$ on the hyperplane. Let $d = x_\alpha - x_h$, then $x_\alpha$ can be written as:

$$x_\alpha = x_h + d$$

$$= x_h + d\frac{\beta}{\|\beta\|},$$

where $d$ is the directed distance of the point $x_\alpha$ from $x_h$ in terms of the unit weight vector $\frac{\beta}{\|\beta\|}$. Note that $d$ is positive if $d$ is in the same direction as the weight vector $\beta$, but negative if in a direction opposite to $\beta$.

Figure 2.3: Illustration of the distance from any arbitrary point $x_\alpha$ say, to $x_h$ on the Hyperplane. The hyperplane is represented by a solid black line separating the red and blue classes.

Substituting $x$ for $x_\alpha$ in (2.19),

$$h(x_\alpha) = h\left(x_h + d\frac{\beta}{\|\beta\|}\right)$$

$$= \beta^T (x_h + d\frac{\beta}{\|\beta\|}) + \beta_0$$

$$= \beta^T x_h + \beta_0 + d\frac{\|\beta\|^2}{\|\beta\|}$$
\[ d = \frac{h(x_n)}{\|\beta\|}. \]  

(2.21)

The absolute value of the numerator of (2.21) can be taken, to ensure that the distance is always a positive value.

We can use (2.21) to confirm that the directed distance of the origin (0) to the hyperplane is \( \frac{\beta_0}{\|\beta\|} \). For instance, we denote this distance by \( d_0 \), meaning that

\[ d_0 = \frac{h(0)}{\|\beta\|} = \frac{\beta^T 0 + \beta_0}{\|\beta\|} = \frac{\beta_0}{\|\beta\|}. \]  

(2.22)

2.2.1.5 Determination of SVM Margin

In line with (2.21), the distance of each data point to the separating hyperplane is

\[ d_i = \frac{y_i h(x_i)}{\|\beta\|} = \frac{y_i (\beta^T x_i + \beta_0)}{\|\beta\|}. \]  

(2.23)

The multiplication of the numerator by \( y_i \) has the same effect as taking the absolute value. While the numerator is regarded as the absolute distance, the denominator makes it a relative distance. Over all \( n \) data points, the vector with minimum distance to the separating hyperplane is such that

\[ d^* = \frac{|\beta^T x^* + \beta_0|}{\|\beta\|}. \]  

(2.24)
where \( x^* \) stands for the +\( ve \) or −\( ve \) support vector.

The numerator of (2.24) specifies that the absolute distance between the hyperplane, and the +\( ve \) or −\( ve \) support vector is \( |\beta^T x^* + \beta_0| \). We have the option of making this distance to be equal to 1 by choosing a constant \( k \), such that

\[
k |\beta^T x^* + \beta_0| = 1. \tag{2.25}
\]

This amounts to rescaling the hyperplane or shifting it to the right or left by one unit. Now based on (2.21), and in line with (2.25), the distance from the hyperplane to the support vector (+\( ve \) or −\( ve \)) is

\[
d^* = k |\beta^T x^* + \beta_0| = \frac{1}{\|\beta\|} = \frac{1}{\|\beta\|}. \tag{2.26}
\]

We obtain the margin by multiplying (2.26) by 2, hence

\[
\text{Margin} = 2 \times \frac{1}{\|\beta\|} = \frac{2}{\|\beta\|}. \tag{2.27}
\]

### 2.2.2 SVM Optimization

We shall discuss SVM optimization under two scenarios, namely linearly separable and linearly non-separable SVM. The procedure for optimization is similar in both cases, and includes maximization of an objective function, subject to some linear constraints. The objective function for the linearly separable SVM is different from the objective function for the linearly non-separable case. Their respective linear constraints are also different.
Thereafter comes the primal formulation of SVM optimization problem, then the dual formulation. At this point, the SVM vector of coefficients can be obtained, hence we can construct a classification function based on the SVM. The use of the SVM classification function helps to allocate new unlabelled input to their respective classes.

2.2.2.1 Linearly Separable or Hard Margin SVM

A classification problem is linearly separable if a separating hyperplane $h(x)$ exists, such that for each $x_i$,

$$
h(x_i) \begin{cases} 
  \geq +1, & \text{if } y = +1 \\
  \leq -1, & \text{if } y = -1.
\end{cases} \tag{2.28}
$$

Alternatively,

$$
\exists \beta, \beta_0 \text{ s.t. } y_i h(x_i) = y_i (\beta^T x_i + \beta_0) \geq 1 \forall i. \tag{2.29}
$$

The statement expressed in (2.28) or (2.29) shows that each data point is either on the margin lines or outside the margin lines, but not within the margin. It also means that every data point is in its correct $+ve$ or $-ve$ class. An example of a linearly separable classification problem is illustrated in Figure 2.1.

One problem with linearly separable SVM is that infinitely many hyperplanes can be used to separate the two classes, but SVM seeks for the hyperplane that is optimum. The search for the optimum hyperplane thus leads to SVM optimization.

The objective function to optimize is the margin, subject to some constraints. In
particular, we

\[
\text{maximize } \frac{2}{\|\beta\|^2} \quad \text{subject to } y_i (\beta^T x_i + \beta_0) \geq 1, \ i = 1, 2, \ldots, n.
\]

Objective Function

Linear Constraints

\[ (2.30) \]

Alternatively, we

\[
\text{minimize } \frac{1}{2} \|\beta\|^2 \quad \text{subject to } y_i (\beta^T x_i + \beta_0) \geq 1, \ i = 1, 2, \ldots, n.
\]

Objective Function

Linear Constraints

\[ (2.31) \]

The constraints specify that there is no margin violation, meaning that each data point is on the correct side of the plane. For this reason, the optimization procedure here is also called the hard margin or maximum margin SVM.

The primal formulation obtains by multiplying the constraints in (2.31) by a positive Lagrangian \( \alpha_i \), based on Karush-Kuhn-Tucker (KKT) conditions to get

\[
\alpha_i \left( y_i (\beta^T x_i + \beta_0) - 1 \right) = 0.
\]

\[ (2.32) \]

The KKT conditions (also known as the Kuhn-Tucker conditions) are first order necessary conditions for a solution in non-linear programming to be optimal (Wang, 2005), provided that some regularity conditions are satisfied. The author further noted that the KKT approach to non-linear programming generalizes the method of Lagrange multipliers, which allows only equality constraints.
CHAPTER 2. CLASSIFICATION

We now subtract (2.32) from the object function to get

\[
L(\beta, \beta_0, \alpha) = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i \left( \beta^T x_i + \beta_0 \right) - 1 \right),
\]  

(2.33)

where \( \alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)^T \geq 0 \) are non-negative Lagrangian coefficients.

Next, we obtain the primal variables, namely \( \beta \) and \( \beta_0 \), by minimizing (2.33) with respect to each primal variable, and setting the result to zero.

\[
\frac{\partial}{\partial \beta} L(\beta, \beta_0, \alpha) = \beta - \sum_{i=1}^{n} \alpha_i y_i x_i = 0,
\]

or \( \beta = \sum_{i=1}^{n} \alpha_i y_i x_i \).

(2.34)

Similarly,

\[
\frac{\partial}{\partial \beta_0} L(\beta, \beta_0, \alpha) = -\sum_{i=1}^{n} \alpha_i y_i = 0.
\]

(2.35)

The problem with the primal variables at this stage is that they are not explicitly determined, because they still depend on \( \alpha \). In order to determine them explicitly, we shall first determine the non-negative values of \( \alpha_i \). This will lead us to dual formulation of the SVM optimization problem, and it is the result of substituting \( \beta \) and \( \beta_0 \) for their respective values in (2.33), to get

\[
L(\beta, \alpha) = \frac{1}{2} \|\beta\|^2 - \beta^T \left( \sum_{i=1}^{n} \alpha_i y_i x_i \right) - \beta_0 \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i
\]

\[
= \frac{1}{2} \|\beta\|^2 - \beta^T \beta + \sum_{i=1}^{n} \alpha_i
\]
\[ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \beta^T \beta. \]

Hence,
\[
L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j).
\] (2.36)

The optimum \( \alpha \) is the result of maximizing
\[
\text{maximizing } L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]
subject to \( \alpha_i \geq 0 \) and \( \sum_{i=1}^{n} \alpha_i y_i = 0. \) (2.37)

In matrix notation, we
\[
\text{maximize } L(\alpha) = 1_n^T \alpha - \frac{1}{2} \alpha^T H \alpha
\]
subject to \( \alpha \geq 0, \alpha^T y = 0, \) (2.38)

where,
\[
H = \begin{pmatrix}
y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \cdots & y_1 y_n x_1^T x_n \\
y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \cdots & y_2 y_n x_2^T x_n \\
\vdots & \vdots & \cdots & \vdots \\
y_n y_1 x_n^T x_1 & y_n y_2 x_n^T x_2 & \cdots & y_n y_n x_n^T x_n
\end{pmatrix}.
\]

A quadratic programming package in R, like quadpro, can be used to solve for \( \alpha. \)

**The Weight Vector (\( \beta \)) and Bias (\( \beta_0 \))**

In (2.32), \( \alpha_i \) is either 0 or greater than 0, but if \( \alpha_i > 0, \) then \( y_i (\beta^T x_i + \beta_0) = 1, \)
which means that \( x_i \) is on the margin line. Hence, \( x_i \) is a support vector. If \( x_i \) is not a support vector, then \( \alpha_i = 0 \).

Let \( sv \) denote indices for support vectors and non-zero \( \alpha_i \), so that \( sv \subset \{1, 2, \ldots, n\} \).

Based on (2.34),

\[
\beta = \sum_{i \in sv} \alpha_i y_i x_i. \tag{2.39}
\]

Regarding \( \beta_0 \),

\[
y_i \left( \beta^T x_i + \beta_0 \right) = 1
\]

\[
\beta^T x_i + \beta_0 = \frac{1}{y_i} = y_i
\]

\[
\beta_0 = y_i - \beta^T x_i, \quad \forall i \in sv. \tag{2.40}
\]

Since \( \beta_0 \) depends on \( i \in sv \), the average is taken. Hence,

\[
\beta_0 = \frac{1}{n(sv)} \sum_{i \in sv} (y_i - \beta^T x_i), \tag{2.41}
\]

where \( n(sv) \) is the total number of data points that are the support vectors.

### 2.2.2.2 Linearly Non-separable or Soft Margin SVM

The hard margin constraints have a weakness of not making provision for margin violation. Since many classification datasets are linearly non-separable, meaning that they overlap; satisfying the constraints is not always possible. For instance if two classes overlap, the optimization problem of (2.31) is disregarded because of violation of the constraints that are associated with it. Instead, we use a more robust procedure as we shall later find in (2.43).
An example of a linearly non-separable dataset is illustrated in Figure 2.4. The graph consists of three separating hyperplanes chosen to separate the two classes. If the solid black line is used, one data point will be misclassified. If we use the green line, we run the risk of misclassifying four data points. We also wrongly classify two data points if the dotted black line is used as a separating hyperplane. In a like manner, we can experiment with as many separating hyperplanes as possible, but none will bring about perfect separation. By perfect separation, we mean a situation where all the data points belonging to the red class will be on one side of the plane, and those belonging to the blue class on the other side of the plane. We then position the hyperplane in such a way that no data point is wrongly classified. Whenever it is not possible to obtain a perfect separation as described, we say that the two classes datasets are linearly non-separable.

Figure 2.4: A linearly non-separable classification problem, because no hyperplane can separate the two classes without the possibility of misclassifying at least one data point.
If datasets are linearly non-separable, slack variables are introduced to contain the effect of violation of the hard margin constraints. This leads to new constraints in the form of

\[ y_i \left( \beta^T x_i + \beta_0 \right) \geq 1 - \xi_i; \quad i = 1, 2, \ldots, n, \]  

where \( \xi_i \geq 0 \) is a slack variable for the data point \( x_i \). If \( \xi_i = 0 \), then \( x_i \) is on the correct side of the plane, and the same applies when \( 0 < \xi_i < \frac{1}{\|\beta\|} \). If \( \xi_i \geq \frac{1}{\|\beta\|} \), data points are misclassified because they are on the wrong side of the plane. The graph of Figure 2.5 gives further illustration. Here, data points with slack variables \( \xi_1 \) and \( \xi_2 \) are correctly classified, but have the problem of margin violation. As a result,

\[
0 < \xi_1, \xi_2 < \frac{1}{\|\beta\|}.
\]

For slack variables \( \xi_3, \xi_4 \) and \( \xi_5 \), data points are misclassified because they are on the wrong side of the plane. Here, \( \frac{1}{\|\beta\|} \leq \xi_3, \xi_5 < \frac{2}{\|\beta\|} \), but
ξ_i \geq \frac{2}{\|\beta\|}. All data points without the problem of margin violation have \( \xi_i \) equal to zero.

The presence of slack variables gives rise to the soft margin SVM, with a different objective function as follows:

\[
\begin{align*}
\text{minimize} & \quad \left( \frac{1}{2} \|\beta\|^2 + \lambda \sum_{i=1}^{n} \xi_i \right) \\
\text{subject to} & \quad y_i (\beta^T x_i + \beta_0) \geq 1 - \xi_i; \quad \xi_i \geq 0.
\end{align*}
\] (2.43)

Note that \( \sum_{i=1}^{n} \xi_i \) is the loss arising from the failure of linear separability, and the smaller it is, the better. Also, \( l \) is a constant with value 1 or 2, and if it is equal to 1, we have the hinge loss. The hinge loss is concerned with minimization of slack variables. If \( l \) is equal to 2, the quadratic loss obtains; it minimizes the sum of squared slack variables.

Also, \( \lambda \) is a regularization parameter that controls the trade-off between margin maximization and loss minimization. If \( \lambda = 0 \), the loss term disappears, and the objective function will aim to maximize the margin. Conversely, if \( \lambda = \infty \), the margin effect will disappear and the objective function will mainly minimize the loss. A well informed choice of \( \lambda \), often aided by the use of cross validation, brings about a trade-off between margin maximization and loss minimization. In Figure 2.6, pertinent graphical illustrations are provided.

Given the hinge loss, the constraints in (2.43) are multiplied by positive Lagrange
multipliers $\alpha$ and $\eta$, to get

$$
\alpha_i \left( y_i \left( \beta^T x_i + \beta_0 \right) - 1 + \xi_i \right) = 0; \quad \alpha_i \geq 0 $$
$$
\eta_i \xi_i = 0; \quad \eta_i \geq 0.
$$

(2.44)

Subtracting (2.44) from the objective function in (2.43) leads to

$$
L (\beta, \beta_0, \alpha_i, \xi_i, \eta_i) = \frac{1}{2} \| \beta \|^2 + \lambda \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \left( y_i \left( \beta^T x_i + \beta_0 \right) - 1 + \xi_i \right) - \sum_{i=1}^n \xi_i \eta_i.
$$

(2.45)

The primal variables including $\xi_i$ and $\eta_i$ are similarly obtained as in (2.34) - (2.35).

Now,

$$
\frac{\partial}{\partial \beta} L (\beta, \beta_0, \alpha_i, \xi_i, \eta_i) = \beta - \sum_{i=1}^n \alpha_i y_i x_i = 0 \text{ or } \beta = \sum_{i=1}^n \alpha_i y_i x_i,
$$

(2.46)

$$
\frac{\partial}{\partial \beta_0} L (\beta, \beta_0, \alpha_i, \xi_i, \eta_i) = - \sum_{i=1}^n \alpha_i y_i = 0,
$$

(2.47)

$$
\frac{\partial}{\partial \xi_i} L (\beta, \beta_0, \alpha_i, \xi_i, \eta_i) = \lambda - \alpha_i - \eta_i = 0 \text{ or } \lambda = \eta_i + \alpha_i.
$$

(2.48)

From (2.48), $\eta_i = \lambda - \alpha_i$, and by substituting this value in (2.44),
\[(\lambda - \alpha_i)\xi_i = 0. \tag{2.49}\]

Observe that the presence of \(\eta_i\) is the reason why \(\xi_i\) is non-negative because in (2.49), the value of \(\xi_i\) is controlled by \((\lambda - \alpha_i)\). If \(\alpha_i < \lambda\), \(\xi_i = 0\) and if \(\alpha_i = \lambda\), \(\xi_i > 0\). Graphically, the relationships among \(\alpha_i\), \(\xi_i\) and \(\lambda\) are shown in Figure 2.7.

![Figure 2.7](image)

**Figure 2.7:** A graphical display of relationships among \(\alpha_i\), \(\xi_i\) and \(\lambda\).

Now, we obtain the dual by plugging (2.46) - (2.48) into (2.45) to get

\[
L = \frac{1}{2}\beta^T\beta - \beta^T\sum_{i=1}^{n} \alpha_i y_i x_i - \beta_0 \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i + \lambda \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} (\alpha_i + \lambda - \alpha_i) \xi_i
\]

\[
= \frac{1}{2}\beta^T\beta - \beta^T\beta + \sum_{i=1}^{n} \alpha_i
\]

\[
= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\beta^T\beta
\]

\[
= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j.
\]
The maximum $\alpha_i$ is obtained by

$$\text{maximizing } L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to $0 \leq \alpha_i \leq \lambda$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$. \hfill (2.50)

**The Weight Vector ($\beta$) and Bias ($\beta_0$)**

As applicable to the hard margin case, the solution to $\alpha_i$ gives rise to two possibilities; $\alpha_i = 0$ and $\alpha_i > 0$. For $\alpha_i \neq 0$, $x_i$ is a support vector such that

$$y_i (\beta^T x_i + \beta_0) \geq 1 - \xi_i; \quad i = 1, 2, \cdots, n.$$  

Since the value of $\xi_i$ is at least zero, support vectors now include data points $x_i$ that are outside the margin.

The weight vector and bias are obtained similarly as in (2.39) and (2.41) respectively. As a result,

$$\beta = \sum_{i \in sv} \alpha_i y_i x_i,$$ \hfill (2.51)

and

$$\beta_0 = \frac{\sum_{i \in sv} y_i - \beta^T x_i}{n (sv)}.$$ \hfill (2.52)

Given a new input $x_{\text{new}}$, say, an SVM based classification function can be written as

$$y = \pm (\beta^T x_{\text{new}} + \beta_0).$$ \hfill (2.53)

If $y \geq 0$, $x_{\text{new}}$ is allocated to class $+1$, otherwise to class $-1$. 
2.2.2.3 Non-Linear SVM

Our discussions so far centred on linearly separable, and linearly non-separable datasets. It is possible to find datasets that are neither linearly separable nor linearly non-separable. Such datasets are either non-linearly separable or non-linearly non-separable. In this section, we shall be concerned with non-linearly separable datasets, hence non-linear SVM.

When datasets are non-linearly separable, they are first transformed before the process of SVM optimization takes place. Here, we transform from a low dimension input space to a high dimension feature space. In the feature space, we can use either the hard or soft margin SVM depending on the dataset. In other words, SVM has no separate optimization procedure when datasets are non-linearly separable. The idea of dataset transformation is to put data into a form that can enable the process of SVM optimization. When this is done, we end up having a non-linear decision boundary in the input space. In Figure 2.8, we present graphical illustrations of non-linear separable datasets.

The datasets of Figure 2.8 are separable, but not with a linear classifier. For instance, if we apply a linear classifier on either Figure 2.8(a) or Figure 2.8(b), we run the risk of misclassifying at least one data point. On the other hand, the application of a non-linear classifier is capable of obtaining best separation, in the sense that no data point will be misclassified.

If we transform say, Figure 2.8(a) from its low dimension (one dimensional) input space, to a higher dimension (two dimensions) feature space, a linear classifier can be constructed in the feature space. We then observe a non-linear decision
Figure 2.8: Non-linearly separable datasets.

boundary in the input space. In Figure 2.9, a further illustration is given.

**Transformation from Input to Feature Space**

In Figure 2.9, we saw a feature space that is only two dimensional. In some cases, the number of dimensions is much higher, and can be infinite. For example, consider a two dimensional input space with features, \( X_1 \) and \( X_2 \). A higher dimensional quadratic feature space can be given as \( X_1^2, X_2^2, \sqrt{2}X_1X_2 \) (Zaki et al., 2014). Define,

\[
\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T \in \mathbb{R}^3,
\]

where \( \phi(x) \) is a non-linear map that takes the input vector \( x_i = (x_1, x_2)^T \in \mathbb{R}^2 \) and maps it to a feature space in \( \mathbb{R}^3 \).

It is possible to show that the dot product of say, \( \phi(x) \) and \( \phi(z) \), based on
(2.54) defines a new function called a kernel function. For instance, let $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$ and $\phi(z) = (z_1^2, z_2^2, \sqrt{2}z_1z_2)$.

$$
\phi(x)^T \phi(z) = x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2
$$

$$
= (x_1z_1 + x_2z_2)^2
$$

$$
= (x^T z)^2
$$

$$
= K(x, z)
$$

(2.55)

The function $K(x, z)$ is called a kernel function.

**Kernel Function**

A kernel function is a function that takes any two vectors of the same dimensions
as the input, and returns a dot product of the vectors (Kim, 2013). For any two
vectors, say $x_i, x_j$, a kernel function satisfies the condition,

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j).$$  \hfill (2.56)

The function specified in (2.56) shows that the value of the dot products can be
constructed in the input space using input vector $x$, without recourse to the non-
linear map $\phi(x)$. Thus, a computation of all possible pairwise similarity functions
$K(x_i, x_j); i, j = 1, 2, \ldots, n$ gives rise to the kernel matrix ($K$). Hence,

$$K = \begin{pmatrix}
K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_n) \\
K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_n) \\
\vdots & \vdots & \ddots & \vdots \\
K(x_n, x_1) & K(x_n, x_2) & \cdots & K(x_n, x_n)
\end{pmatrix}.$$

Now, we shall consider a numerical example to show that (2.56) holds.

**Example 2.2.1.**

Let $x_1 = (5.1, 3.0)$, $x_2 = (6.0, 3.2)$, and $\phi(x) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$. Then,

$$\phi(x_1) = (5.1^2, 3.0^2, \sqrt{2} \times 5.1 \times 3.0)^T = (26.01, 9.0, 21.64)^T$$

$$\phi(x_2) = (6.0^2, 3.2^2, \sqrt{2} \times 6.0 \times 3.2)^T = (36.0, 10.24, 27.15)^T$$

$$\phi(x_1)^T \phi(x_2) = 26.10 \times 36.0 + 9.0 \times 10.24 + 21.64 \times 27.15 = 1616.04$$

$$K(x_1, x_2) = (x_1^T x_2)^2 = (5.1 \times 6.0 + 3.0 \times 3.2)^2 = (40.2)^2 = 1616.04$$

**Types of Kernel Functions**

Different types of kernel functions have been identified (Zeileis et al., 2004), and
a number of them are provided in Table 2.1. In addition to these functions are user defined kernel functions, that must necessarily satisfy the condition specified in (2.56).

<table>
<thead>
<tr>
<th>S\No.</th>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linear kernel</td>
<td>$K(x_1, x_2) = x_1^T x_2$</td>
</tr>
<tr>
<td>2</td>
<td>Polynomial kernel</td>
<td>$K(x_1, x_2) = (c x_1^T x_2 + b)^d$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c$ is a scale parameter, whereas $b$ is the offset. $d$ is the degree of the polynomial.</td>
</tr>
<tr>
<td>3</td>
<td>Gaussian radial basis function</td>
<td>$K(x_1, x_2) = \exp\left(-\frac{|x_1-x_2|^2}{2\sigma^2}\right)$</td>
</tr>
<tr>
<td>4</td>
<td>Exponential kernel</td>
<td>$K(x_1, x_2) = \exp\left(-\frac{|x_1-x_2|^2}{2\sigma^2}\right)$</td>
</tr>
<tr>
<td>5</td>
<td>Hyperbolic tangent (sigmoid kernel)</td>
<td>$K(x_1, x_2) = \tanh(bx_1^T x_2 + c)$</td>
</tr>
<tr>
<td>6</td>
<td>Quadratic kernel</td>
<td>$K(x_1, x_2) = (x_1^T x_2 + 1)^2$</td>
</tr>
</tbody>
</table>

Table 2.1: Different types of kernel functions.

Kernel SVM

Kernel SVM is concerned with dataset transformation using a kernel function, prior to carrying out SVM optimization. The kernel function to use must be suitable for the dataset in question. For instance, a linear kernel function cannot be suitable for a dataset that shows evidence of a polynomial pattern. In this case, a polynomial kernel function is more appropriate.

The advantage of transforming a dataset via a kernel, as against the use of a non-linear map $\phi(x)$, is that the kernel function can be constructed in the input space.
CHAPTER 2. CLASSIFICATION

Here, the objective function is,

$$\text{minimize } \left( \frac{1}{2} \| \beta \|^2 + \lambda \sum_{i=1}^{n} \xi_i \right)$$

subject to $$y_i \left( \beta^T \phi(x_i) + \beta_0 \right) \geq 1 - \xi_i; \quad \xi_i \geq 0.$$  

(2.57)

Note that (2.57) is similar to (2.43) except for the introduction of $$\phi(x_i)$$. The primal and dual formulations are equally similar. However, for the dual, the role played by the kernel function is clearly shown. For instance, we

$$\text{maximize } L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to $$0 \leq \alpha_i \leq \lambda$$ and $$\sum_{i=1}^{n} \alpha_i y_i = 0.$$  

(2.58)

(2.59)

On comparing (2.58) and (2.59), we observe a shift from a dot product in the feature space to a kernel function in the input space. The introduction of a kernel function makes it possible for every operation to take place in the input space. Also, it makes it relatively easier to obtain the non-zero alpha solution, because the feature space can be infinite dimensional, particularly when $$\phi(x)$$ is used.

The Weight Vector ($$\beta$$) and Bias ($$\beta_0$$)

Both $$\beta$$ and $$\beta_0$$ are obtained similarly as in (2.39) and (2.41) respectively. Hence,

$$\beta = \sum_{i \in sv} \alpha_i y_i \phi(x_i).$$

(2.60)
Note that $\beta$ is not explicitly determined yet, but a clearer insight on what obtains eventually is given in (2.61). In the case of the bias or intercept,

$$
\beta_0 = \frac{1}{n(sv)} \left( \sum_{i \in sv} y_i - \sum_{j \in sv} \beta^T \phi(x_j) \right)
$$

$$
= \frac{1}{n(sv)} \left( \sum_{i \in sv} y_i - \sum_{i \in sv} \sum_{j \in sv} \alpha_i y_i \phi(x_i)^T \phi(x_j) \right) \tag{2.61}
$$

$$
= \frac{1}{n(sv)} \left( \sum_{i \in sv} y_i - \sum_{i \in sv} \sum_{j \in sv} \alpha_i y_i K(x_i, x_j) \right).
$$

Now given a new input, say $x_{\text{new}}$, the procedure for allocation is as follows;

$$
y = \pm \left( \beta^T \phi(x_{\text{new}}) + \beta_0 \right)
$$

$$
= \pm \left( \sum_{i \in sv} \alpha_i y_i \phi(x_i)^T \phi(x_{\text{new}}) + \beta_0 \right)
$$

$$
= \pm \left( \sum_{i \in sv} \alpha_i y_i K(x_i, x_{\text{new}}) + \beta_0 \right)
$$

We finally allocate $x_{\text{new}}$ to the $+ve$ class if $y \geq 0$, otherwise allocation is in favour of class $-1$.

### 2.3 Evaluation of Classifiers

We have so far reviewed two different classifiers namely, FDA and SVM. At this point, we shall discuss different ways their performances can be evaluated or assessed.
In the first place, we define a test set, say;

\[ D' = \{(x_i, y_i)\}_{i=1}^m, \]  

(2.62)

where \( x_i \in \mathbb{R}^p \), \( y_i \in \{+1, -1\} \), and \( m \) is the sample size of the test set. Let \( c_k; k = 1, 2 \) denote the class labels, and let \( D'_k \) be the \( k^{th} \) partition of \( D' \). Note that \( m = \sum_{k=1}^{2} m_k \), where \( m_k \) is the sample size of \( D'_k \) in respect of class \( c_k \). For \( x_i \in D' \), let \( y_i \) denote its true class, and let \( r(x_i) = \hat{y}_i \) denote the class prediction of \( r \) given \( x_i \). If \( y_i = \hat{y}_i \), it means that the prediction of \( r \) given \( x_i \) is correct, otherwise \( x_i \) is wrongly predicted.

### 2.3.1 The Error Rate

Error rate is the fraction of incorrect predictions for a classifier over the test set (Zaki et al., 2014). It is defined as:

\[ \text{Error Rate} = \frac{1}{m} \sum_{i=1}^{m} I(y_i \neq \hat{y}_i), \]

where \( I \) is an indicator function with a value 1 if the argument is true, otherwise it is 0. Error rate can be used to estimate the probability of misclassification for a given classifier. Also, the classifier that gives the smallest error rate may be preferred to any other classifier considered for the same classification problem.

One disadvantage of the error rate is that it does not provide information on the performance of a classifier in each class separately. It rather gives information on the overall performance of a classifier given the test set. As a result, we lack the
ability to assess how a classifier performs in each class.

2.3.2 Accuracy Rate

Accuracy refers to the fraction of correct predictions of a classifier, over the entire test set. It can be seen as the opposite of error rate, but both of them provide the same information about the strength or weakness of a classifier. We define it as follows:

\[
\text{Accuracy Rate} = \frac{1}{m} \sum_{i=1}^{m} I(y_i = \hat{y}_i).
\]

As in error rate, if the argument here is true, the value of \( I \) is 1, otherwise, it is 0. Equivalently, accuracy rate can be defined as \( 1 - \text{Error Rate} \).

**Confusion Matrix**

A confusion matrix for the test set defined in (2.62), is a \( K \times K \) table used to describe the performance of a classification function. On the two class case, it consists of information on true positive class prediction, true negative class prediction, false positive class prediction, and false negative class prediction. A sample confusion matrix is presented in Table 2.2, and we can use it to also obtain error rate and accuracy.

<table>
<thead>
<tr>
<th>Predicted Class (( \hat{y} ))</th>
<th>True Class (( y ))</th>
<th>(+ve)</th>
<th>(-ve)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+ve)</td>
<td>True Positive (( m_{11} ))</td>
<td>False Positive (( m_{12} ))</td>
<td></td>
</tr>
<tr>
<td>(-ve)</td>
<td>False Negative (( m_{21} ))</td>
<td>True Negative (( m_{22} ))</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: A sample confusion matrix.
Let 
\( m_{11} \) denote the number of true positive class predictions, 
\( m_{12} \) denote the number of false positive class predictions, 
\( m_{21} \) denote the number of false negative class predictions, 
\( m_{22} \) denote the number of true negative class predictions and 
\( m = m_{11} + m_{12} + m_{21} + m_{22} \).

Then, 
\[
\text{Error Rate} = \frac{m_{12} + m_{21}}{m},
\]
and
\[
\text{Accuracy Rate} = \frac{m_{11} + m_{22}}{m}.
\]

### 2.3.3 Receiver Operating Characteristic (ROC)

The development of the ROC can be traced to the field of statistical decision theory. During World War II, the ROC was used in signal detection for analysing radar images. Through the use of the curve it produces, the radar operators could distinguish between the enemy’s target, a friendly ship or noise (Morrison, 2005).

The ROC is a plot of False Positive Rate (FPR) versus True Positive Rate (TPR) for every possible classification threshold. A classification threshold can be defined as a cut-off point that determines the class to assign the observation \( x_i \). For instance, a classification threshold, say \( \psi = 0.054 \), means that for values of a
classification function $f(x_i) \geq 0.054$, $x_i$ is assigned to class $+1$, otherwise to class $-1$. Thresholds often differ from one classifier to another. For instance regarding FDA, a classification threshold is given as $m = d^T\hat{\Sigma}^{-1}\bar{x}$, where $\bar{x} = \frac{1}{2}(\bar{x}_1 + \bar{x}_2)$. In the case of SVM, it is 0.

With reference to Table 2.2, the two rates can be defined in the following ways:

$$FPR = \frac{m_{12}}{m_{12} + m_{22}},$$

and

$$TPR = \frac{m_{11}}{m_{11} + m_{21}}.$$

When a threshold is chosen for a classifier, ROC considers all possible values of the threshold and plots them. In particular given each value in the range of the threshold, the ROC curve plots FPR on the X-axis versus TPR on the Y-axis. The curve that obtains is called the ROC curve or ROC plot for the given classifier.

An ROC curve can provide a visual assessment of the performance of a classifier, by considering the area under the curve. If a ROC curve produces a diagonal line as in Figure 2.10, the area under the curve is 0.5. Any classifier with such an area or less, is generally regarded as a very poor classifier given the classification problem under investigation. This is because at any point on the dotted line, the TPR equals the FPR. Ideally, a situation where the TPR is consistently higher than the FPR is preferred.
If the area under a ROC curve is 1.0, the classifier concerned is regarded as a good one because it correctly classified all data points in the test set. It also shows that a perfect separation exists between the two classes involved. Assuming there is overlap of classes, the area under an ROC curve will be less than 1.0. In such a situation, a better classifier is one whose area under an ROC curve is as close to 1.0 as possible. For this reason, we can compare two or more different classifiers by visually inspecting the ROC curve each classifier produces, or by computing the area under their respective curves. A classifier that is preferred is the one whose area under the ROC curve is highest. The area under the ROC curve can also be extended to more than two classes (Hand et al., 2001).

2.3.4 Sensitivity and Specificity

With reference to Table 2.2, sensitivity and specificity can be defined as follows:
Sensitivity = $P(\text{Class prediction is } +ve \mid \text{True class is } +ve)$

$$= \frac{m_{11}}{m_{11} + m_{21}} = TPR,$$

and

Specificity = $P(\text{Class prediction is } -ve \mid \text{True class is } -ve)$

$$= \frac{m_{22}}{m_{22} + m_{12}} = 1 - FPR.$$ 

Consider a situation where sensitivity is 80%, and specificity 90% for a classification problem. The interpretation is that the classifier in question is able to predict $+ve$ class, when the true class is $+ve$ 80% of the time, and also predicts $-ve$ class when the true class is $-ve$ 90% of the time. Assuming there is a perfect separation of the training instances, a good classifier will normally output 100% sensitivity and 100% specificity. In other words, if we compare the performances of two or more classifiers using sensitivity and specificity, a classifier with highest sensitivity and specificity at the same time, is usually preferred.

Sometimes we may reasonably suspect that the error rate of a given classifier is not correct. In such a circumstance, sensitivity and specificity can provide a clearer picture of the classifier’s performance. For instance, we consider again Table 2.2, and assuming that $m_{11} = 200$, $m_{12} = 20$, $m_{21} = 0$ and $m_{22} = 0$. Then,

Error Rate = $\frac{20}{220} = 0.0909$,

Sensitivity = 100%, and

Specificity = 0%.
CHAPTER 2. CLASSIFICATION

The error rate of 9.09% can be regarded as reasonable for a classifier, but since specificity is zero, it stands to reason that the classifier in question may not have performed reasonably well. We remark that although this illustration is hypothetical, it is still useful in drawing attention to the fact that this kind of situation may be possible, given a real world dataset.

2.4 Class Predictions in Logistic Regression

In Section 1.4, we reviewed the logistic regression. We mentioned that the tool can be used on dataset that is discrete, continuous or combination of both. We also saw how the prediction function of the logistic regression was derived. Here, we shall discuss how the predictions of the logistic regression function can be tied to their true respective classes.

We note that the prediction function returns output in the interval $[0, 1]$. Unlike classifiers such as FDA or SVM, the cut-off point or classification threshold is not well established in the sense that there is no direct formula for obtaining it. As a result, we often resort to the use of ROC. The ROC helps to obtain the optimal classification threshold, and we can use it to construct a confusion matrix. From the confusion matrix, we can obtain the error rate assuming we aim to do so. Alternatively, the area under the ROC curve can be helpful in providing approximations of the error rate.

By definition, the optimal classification threshold is the threshold which in comparison with all other thresholds available for the same classification problem, gives the maximum sensitivity and specificity.
Figure 2.11: ROC Curve for simulated dataset. The optimal point on the curve is $\tau_1$, where the vertical line intersects the curve. A corresponding value on the x-axis is 0.6410. At this point, sensitivity plus specificity is at the maximum.

As a demonstration of how to obtain the optimal threshold, we simulate two datasets, namely true classes and predictions. Thereafter we construct a ROC model and 15 different outputs are tied to the model based on the pROC package (Robin et al., 2011) in R. The outputs we are interested in are sensitivities, specificities and thresholds. Since we target the threshold that gives maximum sensitivity and specificity, we added both sensitivity and specificity and found a threshold corresponding to the highest value. The R code contained in Appendix A.1 gives further insight on this.

In Figure 2.11, the point where the vertical line intersects the curve is optimal, and a corresponding value on the x-axis is 0.6410. This means that values of $\hat{p} \geq 0.6410$ is classified to class $+1$, otherwise to class 0 (or $-1$ as may be denoted).
As previously mentioned, $\hat{p}$ are the predictions of the logistic regression function.

We note that this procedure is not specific to the logistic regression, but can be used as an alternative way of finding a classification threshold for any given classifier.

### 2.5 Cost-Sensitive Classification

The cost of false positive or false negative can differ from one classification problem to another. For medically related cases, it can be costly to diagnose a disease as present whereas in fact, it is not. Also, the cost can be high to diagnose absence of a disease when it is truly present. For instance, it can be costly to diagnose a patient as not suffering from cancer when in fact the patient has some cancerous infections. Conversely, to subject a healthy person to cancer medication as a result of wrong diagnoses can be very costly.

The regular classification procedure will aim to minimize the misclassification error, via optimal choice of classification threshold. Thus leaving behind an implied assumption that all misclassification errors are equally severe. When we consider the cancer diagnosis cited earlier, it becomes clear that such assumption is not always helpful. For this reason, a cost sensitive classification procedure is called for. In this case, the costs caused by different kinds of errors are not assumed to be equal, and the objective we aim to minimize is the expected cost.

Misclassification costs are weights applied to specific outcomes. These weights form integral part of the classification model, and usually change the pattern of
prediction by offering protection against making costly mistakes. We can specify the classification costs using a cost matrix as in the following example:

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>True Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>−1</td>
<td>$c_2$</td>
</tr>
</tbody>
</table>

The main diagonal entries have zero costs because we make no mistakes for correctly predicting an outcome. The off-diagonal entries $c_1$ and $c_2$ are costs we incur on false positive and false negative predictions respectively. Note that $c_1 \neq c_2$, which means that different costs are associated with both false positive and false negative predictions in this example. However, the choice of the costs are usually based on prior experience, and also depends on the classification problem in question. In other words, different classification problems can give rise to different choices of costs.

To simplify, suppose equal group sizes, then we can define the expected cost as follows:

$$\text{Expected Cost} = (1 - \text{Sensitivity}) \times c_2 + (1 - \text{Specificity}) \times c_1.$$  \hspace{1cm} (2.63)

The effects of minimum expected cost on the ROC curve, for example Figure 2.11, can be summarized in the following ways:

a. Since we assume unequal costs, with equal group sizes, the gradient of the
CHAPTER 2. CLASSIFICATION

line at the tangent (optimal) point changes.

i. If $c_1 > c_2$, the new optimal point on the curve is found to the right of $\tau_1$.

ii. If $c_1 < c_2$, we obtain an optimal point to the left of $\tau_1$.

b. In general, we note that the minimization of (2.63) has the same effect as finding the maximum of

$$\text{Sensitivity} \times c_2 + \text{Specificity} \times c_1,$$

irrespective of the values chosen for $c_1$ and $c_2$. 
Chapter 3

Investigative and Analytical Study of FDA and SVM

3.1 Introduction

FDA and SVM are two important tools for binary classification. While FDA is the work of (Fisher, 1936), a statistician, SVM is credited to the machine learning community and in particular, to (Cortes et al., 1995). Both classifiers have been widely applied to numerous areas of research. For instance, FDA has been used in face recognition (Li et al., 2003; Q. Liu et al., 2004) and face detection (Kurita et al., 2002; Feng et al., 2004). It has also been used in classification of malignant and benign cluster micro-calcifications (Wei et al., 2005), and seed classification (Baudat et al., 2000) etc.

In a like manner, SVM has been used in studies including image analysis (J. J. Liu
et al., 2005), drug design (Burbidge et al., 2001; Norinder, 2003; Bao et al., 2002) and time series (Kyoung-jae, 2003; Thissen et al., 2003). It has also been applied in food quality control (Xie et al., 2009) and environmental sciences (Gilardi et al., 1999; Kanevski et al., 2005) etc.

Questions pertaining to the individual relative merits of both FDA and SVM are likely to depend on who you ask. It may not be out of place to reason that while SVM will appeal to a majority of researchers in machine learning, FDA will remain preferred by those researchers who have relatively no interest in big data.

This chapter, therefore, will focus on the assessment of the relative merits of both classifiers. A similar study in this respect is credited to (Gokcen et al., 2002). The authors compared the Linear Discriminant Analysis (LDA) and Support Vector Machine. They mainly focused on the different ways LDA and SVM compute their respective hyperplanes, and what happens in high dimensions. They also defined a statistic

\[ \beta = \frac{\| w_{SVM} \|}{\| w_{LDA} \|}, \]

where \( w_{SVM} \) and \( w_{LDA} \) are the weight vectors of SVM and LDA classification functions respectively. They claimed that if \( \beta = 1 \), then LDA hyperplane maximizes the margin as much as SVM, since \( \| w_{SVM} \| \leq \| w_{LDA} \| \). The authors noted that SVM hyperplane gives maximum margin and that the margin is proportional to the inverse of the magnitude of the norm. Lastly, they carried out an empirical examination with the aid of some datasets. They observed that SVM outperform LDA except on the simulated dataset. Also for linearly non separable datasets, they noted that SVM tends to perform badly.
In like manner, (Khondoker et al., 2016) compared the classification performances of a number of machine learning algorithms, namely Linear Discriminant Analysis, Support Vector Machine, Random Forests (RF) and k-Nearest Neighbour (kNN). Aspects of their observations that are relevant to us include:

- For smaller number of correlated features, number of features not exceeding approximately half the sample size, they discovered that LDA performed better in terms of average generalisation errors. Based on their findings, the region of strength of LDA appears to be $\frac{p}{n} < 0.5$ (number of features smaller than approximately half the sample size) and higher correlation ($\rho > 0.6$) between features.

- As the feature set gets larger ($\frac{p}{n} \geq 0.5$) SVM (with RBF kernel) outplays LDA. They noted that the sample size here should be at least 20, irrespective of the number of features ($p$) for SVM to achieve its superior performance.

In our study, we examine the differences and similarities between FDA and SVM (Section 3.2). They include the examination of differences based on different theories behind the two classifiers. We also look into the ways regularization is carried out by both classifiers, and further examine how FDA and SVM respond to linear transformations. We conclude the section with examination of the behaviour of FDA and SVM on data, given different scenarios, and in high dimensions too. In Section 3.3, we explore FDA variants, and in particular investigate conditions where FDA or its variants perform better than SVM.
CHAPTER 3. INVESTIGATIVE AND ANALYTICAL STUDY OF ···

3.2 Differences and Similarities between Fisher’s Discriminant Analysis and the Support Vector Machine

3.2.1 Differences based on Theoretical Background

One basic difference in the theory underlying FDA and SVM, is the distance measure each of the classifiers is based on. While FDA is based on Mahalanobis distance, SVM is based on Euclidean distance. The Mahalanobis distance between any two vectors, say \( x \) and \( y \), depends on a positive definite covariance matrix \( \Sigma \), i.e.

\[
d_\Sigma(x, y) = (x - y)^T \Sigma^{-1} (x - y). \tag{3.2}
\]

One advantage of the Mahalanobis distance is that it does not depend on the units in which the variables may be measured. In other words, interpretation of result is more meaningful with a distance measure based on the Mahalanobis distance.

On the other hand, for the vectors \( x, y \in \mathbb{R}^p \), the Euclidean distance is given as

\[
d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_p - y_p)^2}. \tag{3.3}
\]

The Euclidean distance depends on the units in which variables may be measured, and for this reason, it is strongly recommended that variables are scaled prior to using SVM (Hsu et al., 2003). As noted by the authors, scaling helps to avoid a situation where attributes in greater numeric ranges dominate those in smaller numeric ranges. It removes the effects of the different units in which variables may
be measured.

As regards the Mahalanobis distance, data are used in the distance measure via sufficient statistics (examples are $\hat{\Sigma}$, $\bar{x}_1$, and $\bar{x}_2$). In contrast, the Euclidean distance directly involves the individual data points in constructing the distance measure. Thus, SVM can easily determine data points that are closest (support vectors) to the separating hyperplane.

The Mahalanobis and Euclidean distances help to determine how FDA and SVM respectively achieve class mean separation. For instance, FDA relies on a decision boundary passing through the midpoint between the two class means. On the other hand, SVM relies on a separating hyperplane passing midway through the support vectors of both classes. In essence, SVM depends only on the support vectors in determining the separating hyperplane. The implication is that outliers are unlikely to influence the alignment of the separating hyperplane, because they may not be the closest data points. On the other hand, outliers can influence the alignment of the decision boundary of FDA, more easily through their effect on the class mean and covariance matrix of the class containing outliers.

On the assumptions beneath the use of FDA and SVM, FDA assumes that each individual group multivariate samples are drawn from their respective multivariate normal populations (Wikipedia, 2016a), with equal covariance matrix. To put it another way, if parameters are known, and data are normal with different means and a common covariance matrix ($\Sigma$), then FDA is optimal. If any of these assumptions fails, FDA may no longer be a preferred classifier, and instead SVM can be used. Also, the downside of FDA is the inability to provide a solution when $p >> n$, even if the underlying assumptions are met. This failure can be
attributed to the presence of multicollinearity which often leads to a singular covariance matrix, hence no unique solution for FDA. Conversely, SVM makes no assumptions of normality and equal covariance matrices, and is suitably adapted for large $p$, because it depends solely on support vectors. However, it assumes that the units of measurement of data are essentially the same. More importantly, SVM is concerned about linear separability of the training instances. If data are linearly separable, by even a smallest possible margin, SVM is optimal (see Figure 3.4(b)), and preferred to FDA because under such circumstance, it is possible for FDA to wrongly classify at least one data point, whereas SVM consistently returns 0% error rate.

Data preprocessing can be useful to both FDA and SVM. Regarding FDA, preprocessing may aim to achieve dimensionality reduction, particularly when $p >> n$. In this regard, variable selection or the use of principal component analysis can help to reduce the dimensions of a dataset. The use of variable selection technique, for instance, can help to eliminate redundant variables to the extent that a situation where $p \leq n$ obtains. When this happens, we can use FDA in a classification problem. On the side of SVM, data preprocessing can aim to carry out data normalization. It helps to eliminate the effects of the different units in which variables may be measured. We can achieve this through the use of scaling, rescaling or standardization.

### 3.2.2 Regularization

Assuming that the covariance matrix in (2.6) is singular, RFDA (J. H. Friedman, 1989; Witten et al., 2011; Bickel et al., 2004) can be used, and it is the result of
adding $\lambda I$ to $\hat{\Sigma}$, to get

$$f(x) = (\bar{x}_1 - \bar{x}_2)^T \left( \hat{\Sigma} + \lambda I \right)^{-1} \left( x - \frac{1}{2} (\bar{x}_1 + \bar{x}_2) \right).$$  

(3.4)

Here, $\lambda$ is a regularization parameter, and the value it takes determines how close to optimality or otherwise, the classifier is. For instance if $\lambda = 0$, we are back to the unregularized condition where $\hat{\Sigma}$ is singular. If $\lambda \to \infty$, the effect of $\hat{\Sigma}$ in (3.4) depletes and such an outcome is undesirable. Somewhere between 0 and $\infty$ gives rise to an optimal classifier. Therefore, a careful choice of $\lambda$, often aided by the use of cross validation, helps to obtain a classifier that is optimal for the dataset in question.

In the case of SVM, we state that regularization is inherent in the construction of the classifier. For instance, the optimal SVM classifier is a function of the support vectors. Usually the number of support vectors, to a great extent, is smaller than the number of training instances. Through the use of support vectors, SVM is able to prevent problems arising for numerically unstable datasets, where particularly the entire dataset is used for constructing classifiers.

Another form of regularization in SVM is in respect of trade-off between margin maximization, and loss minimization. We can find this in (2.43), and $\lambda$ particularly controls the trade-off. For instance, if $\lambda$ takes the value 0, the slack term disappears, and the objective function defaults to margin maximization. This development is undesirable because the influence of slack variables is not accounted for. On the other hand, if $\lambda \to \infty$, another unwanted scenario obtains because margin effect depletes, and the objective function will aim to minimize the loss.
term only. It therefore follows that a well-informed choice of $\lambda$, often through the use of cross validation, brings about a trade-off between margin maximization, and loss minimization. The classifier that obtains, as a consequence, is optimal in accordance with the data based criterion.

3.2.3 Linear Transformation

We state that FDA is invariant under all non-singular linear transformations, whereas SVM is orthogonally invariant.

We shall prove this first statement, and further use some toy datasets to show that SVM is only orthogonally invariant.

(a) Proof

Let $X_{(n \times p)}$ be a training set, and $x \in \mathbb{R}^p$, the Fisher’s rule uses

$$f_x(x) = (\bar{x}_1 - \bar{x}_2)^T \hat{\Sigma}_x^{-1} \left[ x - \frac{1}{2}(\bar{x}_1 + \bar{x}_2) \right].$$  \hspace{1cm} (3.5)

Let $X \rightarrow Y$ be a transformation from $X$ to $Y$, then

$$Y = X A^T,$$

$$\hat{\Sigma}_y = A \hat{\Sigma}_x A^T$$ and

$$\bar{y} = A \bar{x}.$$

The rule in transformed coordinates is

$$f_y(y) = (\bar{y}_1 - \bar{y}_2)^T \hat{\Sigma}_y^{-1} \left[ y - \frac{1}{2}(\bar{y}_1 + \bar{y}_2) \right]$$

$$= (A\bar{x}_1 - A\bar{x}_2)^T A^{-T} \hat{\Sigma}_x^{-1} A^{-1} \left( A x - \frac{1}{2} (A\bar{x}_1 + A\bar{x}_2) \right)$$
\[ (\bar{x}_1 - \bar{x}_2)^T A^T A^{-T} \Sigma^{-1} A^{-1} A \left( x - \frac{1}{2} (\bar{x}_1 + \bar{x}_2) \right) \]
\[ = (\bar{x}_1 - \bar{x}_2)^T \Sigma^{-1} \left( x - \frac{1}{2} (\bar{x}_1 + \bar{x}_2) \right) = f_x(x). \]

Hence, FDA is invariant under non-singular linear transformations as claimed.

(b) **The effects of rotation, and shearing on FDA and SVM**

In addition to the proof that FDA is invariant under all non-singular linear transformations, we shall here use some toy datasets to further explore the responses of both classifiers, to transformations with respect to rotation and shearing. To this end, we present Figure 3.1.

![Figure 3.1](image)

Figure 3.1: Linear transformations with respect to rotation and shearing. The solid green and black lines are respectively the decision boundary of FDA and SVM separating hyperplane. The two lines are superimposed on each other in both (a) and (b). The dotted lines are the SVM margin lines, and data points on the lines are the support vectors.

Figures 3.1(b) and 3.1(c) are respectively the outcome of orthogonal transformation of the dataset plotted in Figure 3.1(a), and the effect of shearing on the same dataset. In Figure 3.1(b), the FDA decision boundary (solid green line) is superimposed on the separating hyperplane (solid black line) of SVM. It shows that in this case, the behaviour of both classifiers are identical. We also see
that in comparison with Figure 3.1(a), the number of support vectors has not changed. It therefore follows that rotating the original dataset failed to change the behaviour of SVM. Recalling that a rotation matrix is one example of an orthogonal matrix, the development we observed confirms our earlier statement that SVM is orthogonally invariant.

With respect to Figure 3.1(c), SVM is not invariant. Our argument is based on the ground that we now have only two support vectors, compared to four support vectors in Figure 3.1(a). We attribute this development to the fact that when a dataset is sheared, the positions of data points on the coordinate plane are altered. In effect, new data points, not necessarily the old ones, are now closest to the separating hyperplane. In the end, we have new data points as support vectors.

(c) **Alternative Method**

Another way to understand that SVM is not invariant under all linear transformations, is to consider three non-collinear points on a plane as shown in Figure 3.2.

We treat Figure 3.2(b) as the result of transforming the dataset of Figure 3.2(a), or vice versa. It then follows that it is possible to transform any three non-collinear points, to any other three non-linear points on a plane. In Figure 3.2(a), each SVM margin line passes through one support vector in both classes, whereas in Figure 3.2(b) one margin line passes through two support vectors, and the other passes through one support vector. That is to say, Figure 3.2(a) has two support vectors, whereas Figure 3.2(b) has three support vectors. The variation in the number of support vectors after transformation is an indication
(a) SVM: Both margin lines pass through support vector
(b) SVM: One margin line passes through two support vectors

Figure 3.2: A change in behaviour of SVM following a dataset transformation. In (a), we have two data points as support vectors and following a dataset transformation, we obtained three data points support vectors as shown in (b). This shows that SVM is not invariant given some linear transformations.

that SVM is not invariant under all linear transformations.

3.2.3.1 Effect of Change in Coordinates

In the preceding discussions, we provided different illustrations to show that FDA is invariant under all non-singular linear transformations, whereas SVM is orthogonally invariant. In this section, we shall consider additional clarification involving effect of change in coordinates. For instance, we know that FDA and SVM classification functions have largely been constructed in $X$-coordinates, but sometimes we may need to construct them in $Y$-coordinates. Based on earlier discussions, we can easily say that the FDA function constructed in $X$-coordinates is the same as the FDA function constructed in $Y$-coordinates. We can also say that the same is not always true with SVM. On the other hand, assuming there are changes in
some parameters, say $\hat{\Sigma}$ is given as the identity matrix, what then do we observe 
under a change of coordinates?

Consider a linear discriminant function given as:

$$f(x) = \delta_x^T x,$$  \hspace{1cm} (3.7)

where both $\delta_x$ and $x$ are $p$ dimensional vectors. The discriminant rule takes the 
form of some constant $c$. New instances are classified in favour of class $+1$ if 
$f(x) \geq c$, otherwise to class $-1$.

**Case 1**

Supposing we transform from $X$ to $y = Bx$, where $B$ is a $p \times p$ nonsingular matrix. 
Define,

$$\delta_y = B^{-1} \delta_x.$$  \hspace{1cm} (3.8)

A new classification function for $y$ obtains as

$$g(y) = \delta_y^T y.$$  \hspace{1cm} (3.9)

The rule stays the same, and new instances are classified to class $+1$ if $g(y) \geq c$, 
otherwise to class $-1$. Note that (3.7) and (3.9) are identical, and in line with the 
proof given in Section 3.2.3, we say that:

(i) Fishers “classic” discriminant rule constructed in the $X$-coordinates is the same 
as Fisher’s rule constructed in $Y$-coordinates. In other words, Fisher’s rule is 
invariant to the choice of coordinate system.
(ii) We can define Fisher’s “Euclidean” rule by

\[ \delta_x = \bar{x}_1 - \bar{x}_2. \] (3.10)

Here, we let \( \Sigma_x = I \) or equivalently, the presence of \( \Sigma_x \) is ignored in the classic formula. The problem with Fishers Euclidean rule is that discriminant functions constructed in \( X \)-coordinates is not, in general, the same as discriminant functions constructed in \( Y \)-coordinate. For instance under a change of coordinates;

\[ \Sigma_y = B \Sigma_x B^T = B B^T, \]

and

\[ \bar{y}_1 - \bar{y}_2 = B (\bar{x}_1 - \bar{x}_2). \]

Therefore,

\[ \delta_y = \Sigma_y^{-1} (\bar{y}_1 - \bar{y}_2) \]
\[ = (B B^T)^{-1} B (\bar{x}_1 - \bar{x}_2) \]
\[ = B^{-T} B^{-1} B (\bar{x}_1 - \bar{x}_2) \]
\[ = B^{-T} (\bar{x}_1 - \bar{x}_2) \]
\[ \neq \delta_x \]

(iii) The SVM rule constructed in \( X \)-coordinates is different from the one constructed in \( Y \)-coordinates unless \( B \) is orthogonal. This development is connected to the fact that the SVM rule constructed in \( X \)-coordinates depends on the Euclidean distances between \( x \) vectors. Also, the Euclidean distance
between \( x \) vectors is different from Euclidean distance between \( y \) vectors.

**Example**

Let \( x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ x_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \) and \( x_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \)
then the Euclidean distances between \( x_1 \) and \( x_3 \), and between \( x_2 \) and \( x_3 \), are the same. For instance, \( \|x_1 - x_3\|^2 = (\sqrt{1^2 + 0^2})^2 = 1 \) and \( \|x_2 - x_3\|^2 = (\sqrt{0^2 + 1^2})^2 = 1. \)

Now, let \( B = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \), such that \( y_1 = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \)
\( y_2 = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \) and \( y_3 = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \)
Observe that the corresponding Euclidean distances in \( Y \)-coordinates are not the same. For instance, \( \|y_1 - y_3\|^2 = (\sqrt{2^2 + 0^2})^2 = 4 \) and \( \|y_2 - y_3\|^2 = (\sqrt{0^2 + 1^2})^2 = 1. \)

**Some Implications:**

By comparing the responses of FDA and SVM to linear transformation, we are more informed to understand the result we are likely to get, in the event of linearly transforming datasets because we intend to improve the classification result. For instance, if we aim to minimize the error rates using FDA, it is clearer now that no form of linear transformation can be helpful in this regard. On the other hand, we saw that Fisher’s Euclidean rule is not invariant following changes of coordinates, which means that there is a transformation effect on the rule. In Section 3.3, we
shall carry out further investigation with the aim to determining how useful, or otherwise, the rule is.

In the case of SVM, we can obtain differences in the classification result if the transformed datasets lead to an overlapping situation (Section 3.2.4.1). Conversely, assuming a linear transformation gives rise to linearly separable transformed datasets, SVM still returns the same 0% error rate, as in the linearly separable original dataset. In other words, a change of coordinates in this regard has no effect on the classification result of SVM.

3.2.4 Behaviour of FDA and SVM on Data

In this section, some toy datasets will be used to investigate the responses of FDA and SVM on data. They include overlapping datasets, datasets with outliers and non-linearly separable datasets. We shall also assess the usefulness of both classifiers in high dimensions, since some high dimensional transformations will be involved.

3.2.4.1 Responses of FDA and SVM to Class’s Overlap

Two classes overlap if they are linearly non-separable (Figure 2.4). It is usually the case, when measurements on some variables in one class take the same or closely related values of similar measurements in the other class. For instance, assuming we are interested in male and female college students in a certain locality, and the measurable variable is the weights of the students, it is possible that the weights of some female students are similar to the weights of the majority of male students,
or vice versa. In this case, the two classes will overlap.

![Graph showing various degrees of class overlap](image)

(a) Linearly separable datasets  
(b) Classes overlap 1  
(c) Classes overlap 2  
(d) Classes overlap 3  
(e) Classes overlap 4  
(f) Classes overlap 5

Figure 3.3: Illustration of various degrees of class overlap. The dotted black lines are the SVM margin lines and the solid black lines are the separating hyperplanes of SVM. The green solid lines are the decision boundaries of FDA. The support vectors are the data points overlaid by small triangles. In (a), we have a linearly separable dataset but as we go from (b) to (f), we obtain different forms of class overlap.

When classes overlap, the FDA rule stays the same, and the decision boundary still passes through the midpoint between the two class means. In the case of SVM, the hard margin will no longer be useful because the constraints are now violated. Alternatively, the SVM uses the soft margin on a classification problem. For this reason, this investigation will largely focus on the behaviour of the SVM as we vary the degree of class overlap using the toy datasets. Put differently, the investigation is not about how FDA or SVM solves problems involving class
overlap, instead, we aim to understand the changes in SVM separating hyperplane following various degrees of class overlap, since FDA’s rule is invariant. At the end, we would also like to understand the conditions in which both classifiers can behave exactly alike.

Figure 3.3 consists of illustrations on the behaviour of SVM following different degrees of class overlap. In Figure 3.3(a), we have non-overlapping classes, hence no data point was misclassified and both classifiers recorded 0% error rates. Figure 3.3(b) is the result of transferring two data points from the red class of Figure 3.3(a), to the black class in order to obtain class overlap. In response, the SVM generated more support vectors to obtain the separating hyperplane. The support vectors that were formerly 10% of the whole data points rose to 60% in Figure 3.3(b). We also observed that, following this increase, the SVM separating hyperplane is now closer to the decision boundary of FDA. At this point, both classifiers misclassified two data points.

We obtained Figure 3.3(c) by further extending one of the misclassified data points of Figure 3.3(b) beyond the left margin line. In response, SVM generated more support vectors making up 77% of the entire data points. We also observed that the SVM separating hyperplane is now closer to the decision boundary of FDA than in the two previous cases. Also, the error rates are the same for both classifiers since there is no extra misclassified data point.

Figure 3.3(d) is the result of dragging two data points from the black class in Figure 3.3(c) to the red class. Also, six data points of the red class were brought very close to each other. Following this alteration, SVM responded by marginally increasing the number of support vectors to 80% of the whole data points. This
time, SVM misclassified five data points whereas FDA misclassified four.

We continued to vary the positions of some data points of both classes, and subsequ- nutly obtained the results shown in Figures 3.3(e) and 3.3(f). The percentages of datasets used as support vectors in both Figures are respectively 90% and 96.67%. Also, the SVM separating hyperplane was able to locate the decision boundary of FDA in both Figures 3.3(e) and 3.3(f), because one is superimposed on the other. Thus in both cases, the two classifiers behaved in identical manners.

In general, it is our position that FDA can out-perform SVM only in instances of classes overlap (Figure 3.3(d)). As more data points are used as support vectors, the SVM separating hyperplane appears to get closer to the decision boundary of FDA. In particular, where at least 90% of the entire data points are support vectors, it is possible to obtain a situation where the SVM separating hyperplane is superimposed on the FDA decision boundary. This likely development can be explained intuitively. For instance, since we have almost the entire set of data points as support vectors, the SVM separating hyperplane still passes midway through the support vectors. Technically, it is similar to passing through the mid point of the two class means. In essence, it mimics the behaviour of FDA, and in that case, both classifiers behave in identical manners.

3.2.4.2 Effect of Outliers on FDA and SVM

Outliers are observations or data points that are considerably different from the rest of the observations (Taylor, 2012). They may be caused by variability in the dataset, or as a result of experimental error. The investigation we carry out shall
consider three different types of outliers in two dimensions, as contained in Figure 3.4. We identify them as $Y$-outliers, $X$-outliers on the correct side of the plane, and $X$-outliers on the wrong side of the plane.

Figure 3.4: Responses of the FDA and SVM to different types of outliers. The outliers include: (a) $Y$-outliers on the correct side of the plane, (b) $X$-outliers on the correct side of the plane, and (c) $X$-outliers on the wrong side of the plane.

Figure 3.4(a) consists of two data point outliers with no influence on the behaviour of the classifiers, because both data points are on the correct side of the plane. Here, the two classes are linearly separable, and each classifier returned 0% error rate since no data point is misclassified. The outliers contained in Figure 3.4(b) are on the correct side of the plane. Although the two classes are still linearly separable, FDA misclassified two data points whereas the SVM maintained a 0% error rate. This observation may be attributed to the fact that the outliers here influenced the alignment of FDA decision boundary towards it. As a result, two data points were misclassified. On the other hand, since the SVM separating hyperplane will rely on the closest data points to it, outliers of this kind will have no effect on the separating hyperplane. For this reason, the SVM returned a 0% error rate.
In Figure 3.4(c), outliers were positioned on the wrong side of the plane, thereby causing class overlap. In response, the SVM generated more support vectors to obtain the separating hyperplane. It eventually misclassified seven data points whereas FDA misclassified five.

In summary, Figure 3.4(b) shows that outliers on the correct side of the plane do not affect the classification result of the SVM. Instead, the classification result of FDA can be adversely affected. Based on the illustrations, we state that even when two classes do not overlap, it is still possible for FDA to wrongly classify at least one data point, whereas the SVM will consistently return a 0% error rate. Conversely, outliers on the wrong side of the plane causes class overlap, assuming that the classes were previously linearly separable. Such outliers can affect the classification results of both FDA and the SVM. In such a situation, either of the classifiers can out-perform the other.

3.2.4.3 Non-Linearly Separable Datasets, and Usefulness of FDA and SVM in High Dimensions

FDA and SVM are linear classifiers, meaning that they perform badly when used on non-linearly separable datasets. As we explained in Section 2.2.2.3, it is still possible to use a linear classifier on non-linearly separable datasets after transformation. We routinely transform from low dimension input space, to high dimension feature space. This procedure helps to obtain conditions favourable for application of linear classifiers in the feature space. Assuming we are in two dimensions, if we apply a linear classifier in the feature space, we easily notice a non-linear decision boundary in the input space. In dataset transformation, the option of using either
a non-linear map (2.54), or kernel function (2.55) is available.

In this section, we shall use toy datasets to investigate the adaptability of both classifiers for non-linearly separable classification problems. Also, since there is a connection to high dimensions because transformation is involved, we would equally explore the prospects of both classifiers in high dimensions.

Figure 3.5: The behaviour of FDA and SVM on the non-linearly separable datasets in (a). In (b), we observe that the classifiers performed poorly as many data points are misclassified by both classifiers.

Figure 3.5(a) is one example of non-linearly separable datasets, because only a non-linear classifier can separate the two classes, without the risk of misclassifying at least one data point. As illustrated in Figure 3.5(b), the application of a linear classifier on non-linearly separable datasets will lead to misclassification of several data points. For instance, Figure 3.5(b) shows that FDA misclassified 13 data points with an error rate of 0.2167, whereas SVM misclassified 12 data points with an error rate of 0.20. Both error rates are high, which reflects the bad performances
of the classifiers given the datasets.

In order to enhance the performances of FDA and SVM, in turn we transformed the datasets using a quadratic non-linear map,

\[ \phi(x) = \left( \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad x_1^2 \quad x_2^2 \quad \sqrt{2}x_1x_2 \right)^T \]

(Schütze, 2009), and a radial basis function kernel. Using the code in Appendices B.1 and B.2, the effects of both transformation procedures resulted in a 0\% error rate using each of the classifiers.

Considering that the toy datasets are in two dimensions, the use of \( \phi(x) \) in transformation was relatively straightforward vis-a-vis the kernel function. For instance, the dimensions of the transformed datasets using \( \phi(x) \) is \( 60 \times 5 \), whereas using a radial basis function kernel, it is \( 60 \times 60 \). Assuming that the dimensions of the toy datasets were higher, the use of \( \phi(x) \) may not be advised because the dimensions of the transformed datasets in the feature space this time may be extremely large. In that regard, a kernel function is preferred.

When we carry out transformation with \( \phi(x) \), FDA can be used as a classifier, but if the dimensions of the transformed datasets are very high, FDA may no longer be useful because multicollinearity is likely to set in. In that case, RFDA can be used. The use of a kernel function does not support direct application of FDA, because of the dimensions of the kernel matrix. For this reason, we also use RFDA any time a kernel function is used in transformation, and FDA is the preferred classifier. In high dimensions, the use of FDA or its variant RFDA can take relatively more time in computing \( \hat{\Sigma}^{-1} \), often because a high dimensional \( n \times n \) kernel matrix is
involved.

On the other hand, when either \( \phi(x) \) or a kernel function is used in transformation, the application of SVM comparatively goes without a hitch. This may be connected with the fact that SVM does not depend on the entire datasets in classification, but only on the support vectors. As a result, problems associated with multicollinearity do not count here. Also, since reliance on support vectors equally connotes reduction in dimensions of the datasets, classification here can take less time, comparatively.

In summary, when low dimensional non-linearly separable datasets are involved in a classification problem, the use of \( \phi(x) \) is preferred to the use of a kernel function. In this case, the choice of either FDA or SVM is adequate. With a high dimensional alternative, it is preferable to use a kernel function in transformation, followed by the application of SVM. By implication therefore, in high dimensions, the use of SVM as a preferred classifier is recommended.

### 3.3 Breaking the SVM

In Figure 3.4, we saw that FDA can wrongly classify at least one data point, even when the two classes in question are linearly separable. In that reference, SVM maintained dominance by returning a 0% error rate. In both Figures 3.3 and 3.5, described as instances of class overlap, we saw that either FDA or SVM can at least marginally perform better than the other. In other words, when datasets overlap, it is possible that either of the classifiers can outperform the other. By and large, SVM is in the lead, and can be credited with superior performance over
In order to discover scenarios where FDA consistently performs better than SVM, we shall study the performances of FDA variants in comparison with SVM. The FDA variants will include Isotropic FDA (IFDA), Regularized FDA (RFDA) and Equicovariance FDA (EFDA). It is our view that since the variants of FDA represent different informed modifications to standard FDA, classification results may generally improve.

3.3.1 Isotropic FDA

Let \( \hat{\Sigma} \) be an estimate of a pooled covariance matrix. Given \( \hat{\Sigma}_I = \hat{\alpha} I_p \) for some \( \hat{\alpha} \), then \( \hat{\Sigma}_I \) is called isotropic. We assume here that the variances of the predictor variables are the same, and covariances among them are zero. This implies that each predictor variable has the same contribution to the classification result.

The discriminant function for IFDA is:

\[
 f_I (x) = d^T \hat{\Sigma}_I^{-1} (x - \bar{x}),
\]  

(3.11)

where \( \hat{\Sigma}_I \) is as previously defined, and \( \hat{\alpha} = \frac{1}{p} \text{tr} (\hat{\Sigma}) \). Note that \( d \) is also as previously defined, and the classification rule is the same as standard FDA.
3.3.2 Regularized FDA

We discussed RFDA in Section 3.2.2, and in addition, we note that \( \hat{\Sigma}_R = \hat{\Sigma} + \lambda I_p; \lambda \geq 0 \). The discriminant function is:

\[
    f_R(x) = d^T \hat{\Sigma}_R^{-1} (x - \bar{x}).
\]  

Like in standard FDA, \( x \) is allocated to class +1 if \( f_R(x) \geq 0 \), otherwise to class -1.

3.3.3 Equicovariance FDA

EFDA is based on the assumption that covariances among pairs of predictor variables, are not significantly different from one another. In our view, in this situation replacing the covariance of each pair, with the average of all the covariances, and replacing the variance of each variable with the average of all variances, the classification result may be improved.

In the construction of EFDA, we first generate datasets for two classes using an equicorrelation matrix. Thereafter, we obtain an estimate of the pooled covariance matrix \( \hat{\Sigma} \), and subsequently the equicovariance matrix \( \hat{\Sigma}_E \). Note that \( \hat{\Sigma}_E \) is obtained from \( \hat{\Sigma} \), in such a way that the average of the diagonal entries of \( \hat{\Sigma} \), constitutes each diagonal entry of \( \hat{\Sigma}_E \). Similarly, the average of all the off-diagonal entries of \( \hat{\Sigma} \) will constitute each off-diagonal entry of \( \hat{\Sigma}_E \). We can symbolically
represent this description as follows:

\[
\begin{align*}
\left( \hat{\Sigma}_E \right)_{jj} &= \theta \text{ say, where } \theta = \frac{1}{p} \sum_{j=1}^{p} \sigma_{jj}; \sigma_{jj} \in \hat{\Sigma} \\
\left( \hat{\Sigma}_E \right)_{ij} &= \omega \text{ say, where } \omega = \frac{p(p-1)}{2} \sum_{i,j=1}^{p} \sigma_{ij} | i > j; \sigma_{ij} \in \hat{\Sigma}.
\end{align*}
\]

The discriminant function here is the same with other FDA variants, except for changes in the covariance structure. Hence,

\[f_E(x) = d^T \hat{\Sigma}_E^{-1} (x - \bar{x}).\]

Similarly to the standard FDA, \(x\) is classified to class +1 if \(f_E(x) \geq 0\), otherwise to class \(-1\).

### 3.3.4 Investigation

We shall use two randomly generated multivariate datasets; one for a training set, and the other for a test set. Both sets are obtained using an equicovariance matrix for \(\Sigma\), with \(\rho = 0.8\), and a proportional mean difference for \(\mu\) as shown in Appendix C. The dimensions of the test set are \(1000 \times 20\), whereas the training set consists of three different sets with dimensions: \(30 \times 20\), \(100 \times 20\) and \(1000 \times 20\). The reason for using different training set sizes is to help us study the behaviour of each classifier in different scenarios depicted by different training sample sizes.

Thereafter, we train each classifier three times; each time a different training set size is used. After a classifier is trained, it is tested on 100 different randomly generated test sets of the same dimensions \((1000 \times 20)\), 500 observations from
each class. This procedure helps to eliminate biased conclusions, because different aspects of the data are accounted for. The error rate of a classifier for each training sample size, is the average error rate over 100 different test sets. In Table 3.1, we present the average error rates, compared to the SVM.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Sample size</th>
<th>IFDA</th>
<th>EFDA</th>
<th>RFDA</th>
<th>FDA</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n = 30$</td>
<td>0.2528</td>
<td>0.013</td>
<td>0.0483</td>
<td>0.0482</td>
<td>0.0289</td>
</tr>
<tr>
<td>2</td>
<td>$n = 100$</td>
<td>0.0671</td>
<td>0.0105</td>
<td>0.0218</td>
<td>0.0213</td>
<td>0.0230</td>
</tr>
<tr>
<td>3</td>
<td>$n = 1000$</td>
<td>0.0154</td>
<td>0.0091</td>
<td>0.0093</td>
<td>0.0091</td>
<td>0.0161</td>
</tr>
</tbody>
</table>

Table 3.1: Average error rates for each classifier, given different sizes of the training sets. A linear kernel was considered for SVM, which in effect, is as good as using the datasets in their untransformed form. This way, we eliminate the introduction of bias in the error rates of SVM, in comparison with the error rates of the other classifiers.

Differences in error rates for each training sample size can be regarded as marginal, except in the case of IFDA for the $n = 30$ sample size. Here, we observed a marked difference in comparison with the error rates of other classifiers. By and large, IFDA is the least well performing classifier when compared with the other classifiers, and we link this development to absence of estimates of covariances among pairs of explanatory variables.

Both FDA and RFDA are neck and neck in their error rates, and we attribute this development to the fact that we used only numerically stable datasets in the investigation. As a result, the value of regularization parameter $\lambda$ required for RFDA is as small as 0.0001, which in general is not enough to initiate major changes in the structure of the covariance matrix. In other words, the effect on the covariance matrix is similar to when $\lambda = 0$. This development further shows that RFDA is superior to FDA only when datasets are numerically unstable.

FDA and SVM behaved in manners consistent with our position in Section 3.2.1,
where we remarked that if data are normal with different means, and a common covariance matrix, then FDA is ideal. In the case of \( n = 30 \), we find a situation where the parameters are poorly estimated, unlike in the cases involving 100 and 1000 sample sizes. For this reason, it was not out of place that SVM performed better when \( n = 30 \), unlike in the other two sample sizes where FDA performed relatively better.

The error rates of EFDA is consistently lower irrespective of the size of the training set. It shows that informed modification of FDA, under some assumptions, can give rise to a classifier that consistently performs better than SVM. The problem with EFDA is the inability to replicate the same effect given real world datasets. For instance, we applied the classifier on three datasets, namely Appendicitis, Australia and Coil2000 datasets described in Chapters 4 and 5, and the error rates are 0.2188, 0.3140 and 0.3424 respectively. Also, the error rates of SVM on the same datasets are 0.1875, 0.1400 and 0.0597 respectively. In comparison, EFDA performed rather poorly. But in spite of this outcome on real world datasets, the performances of EFDA on the simulated datasets show that the classifier can consistently perform better than SVM, on any real world dataset that meets the underlying assumptions.
Chapter 4

Regression Discriminant Analysis (RDA)

4.1 Introduction

In Chapter 1, we identified regression and classification as valid tools for prediction, and further provided a number of commonly shared characteristics that the two prediction tools have in common. In particular, we mentioned that:

(a) There exist a matrix of input data, and vector of output required for training and testing of both regression and classification models.

(b) We stated that since the dimensions of the input data in most cases is at least $n \times 2$, there can be concern about numerical stability of methods in both cases.

(c) Also, in instances where the input data is high dimensional, we noted that to obtain a regression or classification function that optimally performs may be a
challenging task.

(d) We added that the assessment of prediction tools is via prediction error, and the method of calculating such error depends on the prediction tool in question. For instance in regression we use the mean square error, whereas in classification we often use the error rate.

(e) Finally, we remarked that a very important characteristic is the fact that equations describing both regression and classification functions, can be similarly expressed (see (1.2) and (1.3)).

In the light of these, we claim that it is possible to use regression as a tool for classification. We therefore propose a classification function based on multiple regression, and claim that it is identical to FDA; hence we name this classifier Regression Discriminant Analysis (RDA). We further claim (Section 4.3) that regression variants, namely ridge regression and Lasso can be used as valid binary classifiers.

In the section that follows, we shall provide a mathematical backing to the claim that RDA is identical to FDA. Also, some data-based illustrations will be provided.

4.2 Mathematical Examination

1. Data and some notation

Let $X_1 (n_1 \times p)$ and $X_2 (n_2 \times p)$ be datasets for two populations $\Pi_1$ and $\Pi_2$, and
let $n = n_1 + n_2$. Let

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (n \times p)$$

denote the whole dataset, and $H = I_n - (1/n) 1_n 1_n^T$ denote the $n \times n$ centring matrix. In a similar way, let $H_1$ and $H_2$ denote the $n_1 \times n_1$, and $n_2 \times n_2$ centring matrices respectively.

Let $\bar{x}_1$, $\bar{x}_2$ and $\bar{x}$ denote the sample mean vectors of $X_1$, $X_2$ and $X$ respectively. Note that

$$\bar{x} = (n_1 \bar{x}_1 + n_2 \bar{x}_2) / n$$

is a weighted average of the two group means. We also need the unweighted average

$$x_{\text{av}} = (\bar{x}_1 + \bar{x}_2) / 2,$$

and the difference,

$$\delta = \bar{x}_1 - \bar{x}_2.$$ (4.1)

2. Fisher’s allocation rule

Several matrices are of interest in discriminant analysis:

$$T = X^T H X,$$

$$B = (n_1 n_2 / n) \delta \delta^T,$$

$$W = X_1^T H_1 X_1 + X_2^T H_2 X_2.$$
A classic result (Hewson, 2009) states that

\[ T = W + B, \]

where \( T \) is the total sample matrix of sum of squares and cross products, \( W \) is the within sample matrix of sum of squares and cross products and \( B \), the between-groups sample matrix of sum of squares and cross products.

As shown in Chapter 2, the Fisher’s allocation rule is based on Fisher’s linear discriminant function

\[ f(x) = \delta^T W^{-1} (x - x_{av}). \]

The allocation rule in respect of a new input \( x \) says: allocate \( x \) to \( \Pi_1 \) if \( f(x) \geq 0 \), and to \( \Pi_2 \) otherwise.

It is important to note that sometimes \( f(x) \) is constructed using \( S_{pooled} = W/(n - 2) \) instead of \( W \), but the allocation rule is the same. Since \( W \) is symmetrical, write

\[ \gamma = W^{-1}\delta; \quad (4.2) \]

then Fisher’s discriminant function simplifies to

\[ f(x) = \gamma^T (x - x_{av}). \]

3. Multiple Regression

Let \( y = \begin{pmatrix} +1_{m1 \times 1} \\ -1_{m2 \times 1} \end{pmatrix} \) denote a response vector of length \( n \), and consider a regression of \( y \) on \( X \). Then, the ordinary least squares regression function can be
written as
\[ g(x) = \hat{\alpha} + \hat{\beta}^T x, \]
where \( \hat{\alpha} = \bar{y} - \hat{\beta}^T \bar{x}, \) and
\[ \hat{\beta} = (X^T H X)^{-1} X^T H y = T^{-1} X^T (H y). \]

Note that \( \hat{\beta} \) is estimated using the centred matrix \( H, \) and it is idempotent, i.e. \( H = H^T \) and \( H^2 = H. \) We then claim that:
\[ \hat{\beta} \propto \gamma, \] (4.3)
where \( \gamma \) is as defined in (4.2).

4. Proof

First note that the centred vector \( H y \) has entries \(+1 - \bar{y}\) in the first \( n_1 \) places and \(-1 - \bar{y}\) in the final \( n_2 \) places. Since \( \bar{y} = (n_1 - n_2) / n, \) \( H y \) simplifies to \( 2n_1 n_2 / n \) times a vector with \(+1/n_1\) in the first \( n_1 \) places and \(-1/n_2\) in the final \( n_2 \) places.

Hence,
\[ X^T(H y) = \frac{2n_1 n_2}{n} \left( (1/n_1) X_1^T 1_{n_1} - (1/n_2) X_2^T 1_{n_2} \right) = \frac{2n_1 n_2}{n} (\bar{x}_1 - \bar{x}_2) = \frac{2n_1 n_2}{n} \delta, \]
where \( \delta \) is as defined in (4.1).

Let \( z = \frac{2n_1 n_2}{n}; \) showing that \( \hat{\beta} \propto \gamma \) is equivalent to showing that \( z T^{-1} \delta \propto \gamma, \)
which is true if and only if
\[ z \delta \propto T \gamma \propto T W^{-1} \delta. \]
\( \propto (W + B) W^{-1} \delta \)
\( \propto (I + (n_1 n_2/n) \delta \delta^T W^{-1}) \delta \)
\( \propto \delta + (n_1 n_2/n) \delta (\delta^T W^{-1} \delta) \)
\( = \frac{1}{z} \{ 1 + (n_1 n_2/n) (\delta^T W^{-1} \delta) \} \delta \)
\( = u \delta, \)

where \( u = \frac{1}{z} \{ 1 + (n_1 n_2/n) (\delta^T W^{-1} \delta) \} \) is a constant. Hence, the result is proved.

5. Regression rule

Set

\[
g(x) = \hat{\alpha} + \hat{\beta}^T x \\
= \bar{y} - \hat{\beta}^T \bar{x} + \hat{\beta}^T x \\
= \bar{y} + \hat{\beta}^T (x - \bar{x}), \tag{4.4}
\]

and allocate to \( \Pi_1 \) if \( g(x) \geq 0 \), otherwise to \( \Pi_2 \). If on the other hand we set \( x = x_{av} \), then,

\[
g(x_{av}) = \bar{y} + \hat{\beta}^T (x_{av} - \bar{x}) \neq 0,
\]

unless \( n_1 = n_2 \). Hence, the naive regression is different from Fisher’s rule. We have used the term naive regression to explain that the function \( g \), specified in (4.4), is identical to FDA if and only if \( n_1 = n_2 \).

6. Alternative rule

Alternatively, we can shift the regression predictor by a constant value to

\[
g^*(x) = g(x) - \left( \hat{\alpha} + \hat{\beta}^T x_{av} \right)
\]
\[ = \hat{\alpha} + \hat{\beta}^T \mathbf{x} - \hat{\alpha} - \hat{\beta}^T \mathbf{x}_{av} \]
\[ = \hat{\beta}^T (\mathbf{x} - \mathbf{x}_{av}) , \]  \hfill (4.5)

and define another rule: allocate \( \mathbf{x} \) to \( \Pi_1 \) if \( g^* (\mathbf{x}) \geq 0 \) and to \( \Pi_2 \) otherwise.

The allocation rule given by \( f \) and \( g^* \) are identical, hence we use \( g^* \) instead of \( g \) and refer to it as RDA. In summary, the Fisher’s allocation rule based on \( f \) is identical to the regression-based allocation rule based on \( g^* \).

### 4.2.1 Graphical Illustration

Two datasets, Australia and Ionosphere, will be used for this illustration. The Australia dataset concerns credit card applications, and all attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. It has dimensions \( 690 \times 14 \), with two classes representing approved and not approved. The data source and website are (Alcalá et al., 2010), and http://sci2s.ugr.es/keel/dataset.php?cod=53 respectively.

Ionosphere is a radar data set collected by a system in Goose Bay, Labrador. The system consists of a phased array of 16 high-frequency antennas with a total transmitted power of the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. “Good” radar returns are those showing evidence of some type of structure in the ionosphere. “Bad” returns are those that do not; their signals pass through the ionosphere.

Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for
CHAPTER 4. REGRESSION DISCRIMINANT ANALYSIS (RDA)  

Figure 4.1: Graphical illustrations showing that the classification results of FDA and RDA are the same.

The Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal. The dataset is included in
 CHAPTER 4. REGRESSION DISCRIMINANT ANALYSIS (RDA)

mlbench package, R.

The ROC-Curves of FDA and RDA on the Australia dataset are contained in Figures 4.1(a) and 4.1(b) respectively. The curves of both classifiers are identical, and the areas under the curves are exactly the same in both cases. We also obtained the same error rate of 13.53\% using the two classifiers on the dataset.

Figures 4.1(c) and 4.1(d) respectively consist of ROC-Curves of FDA and RDA given the Ionosphere dataset. Again, the two curves are exactly alike, and the areas under the curves are the same for both classifiers. Also, error rate of 18.10\% was obtained using each of the classifiers. The R codes used here are contained in Appendices D.1 and C.4.

In summary, FDA and RDA are identical classifiers, hence either of the classifiers can be used in place of the other, in a binary classification problem.

4.3 Regression Discriminant Analysis Variants

In line with the proof that FDA and RDA are two identical classifiers (Section 4.2), we further claim that regression variants, namely ridge regression and Lasso can be used as binary classifiers. We also claim that the error rates output by these regression tools are not statistically different from the error rates of FDA, when applied to the same datasets. Hence, we shall refer to the tools as Ridge Regression Discriminant Analysis (RRDA), and Lasso Discriminant Analysis (LaDA) respectively.

Prior to examining their mathematical framework, i.e the geometry of RRDA and
LaDA, we would like to re-examine (4.5) since it is the basis for our claims. Here,

\[ g^*(x) = \hat{\beta}^T (x - x_{av}), \]

where \( \hat{\beta} \) is a vector of regression coefficients based on the least squares. Recall that \( \hat{\beta} \) depends on both \( X \) and \( y \), and that \( y \in \{+1, -1\} \). If \( \hat{\beta}_{\text{ridge}} \) and \( \hat{\beta}_{\text{lasso}} \) are respectively vectors of regression coefficients based on ridge regression and Lasso, given that \( y \in \{+1, -1\} \), a substitution of either of them for \( \hat{\beta} \) in (4.5), will give rise to a classifier after ridge regression or Lasso. The advantages of using these regression tools as binary classifiers are tied to their abilities to double as tools for regularization.

For instance, recall also that when the number of predictor variables \( (p) \) is larger than the sample size \( (n) \), we have problems using standard Fisher’s DA as a classifier. However, the use of RRDA or LaDA can solve a classification problem given such an instance. This is because, as regularization tools, both classifiers are capable of shrinking the model coefficients. Depending on which one is used, some of the coefficients could be set to zero.

Concerning how regularization is carried out here, we shall re-visit the optimization procedure for ridge regression and Lasso discussed in Sections 1.3.1.1 and 1.3.2 respectively. In the first place, throughout this discussion we assume that \( y \in \{+1, -1\} \).

In respect of RRDA, the constraint \( \sum_{j=1}^{p} \beta_j^2 \leq s \) in (1.17) prevents the sum of \( \beta_j^2 \) from increasing beyond the specified threshold \( s \). Since \( s \) is a very small positive constant, the coefficients are thus constrained or shrunk depending on the value
chosen for $s$. The effects are coefficients that are relatively smaller in size, and able to correct the inverse problems that are common with numerically unstable datasets. It is important to note that no matter the value chosen for $s$, it is not possible that some coefficients can be set to zero, since a quadratic optimization problem is involved.

On the other hand, the inability of RRDA to set some coefficients to zero can be corrected through the use LaDA. For instance, examining the constraint $\sum_{j=1}^{p} |\beta_j| \leq s$ in (1.22), we observe a similarity to the constraint for RRDA, except that here, the sum of $|\beta_j|$ is at most $s$. Also, depending on the value chosen for $s$, it is possible that some coefficients can be set to zero, and the rest shrunk. We can attribute this development to the presence of $|\beta_j|$ in the optimization problem.

To demonstrate that Lasso can move a weight to zero, whereas ridge regression cannot, let $L_1(\beta) = \sum_j |\beta_j|$, and $L_2(\beta) = \frac{1}{2} \sum_j \beta_j^2$ be $L_1$ and $L_2$ penalty functions respectively for Lasso and ridge regression. Expanding both functions, we obtain:

$$L_1(\beta) = |\beta_1| + |\beta_2| + \cdots + |\beta_p|$$

and

$$L_2(\beta) = \frac{1}{2} (\beta_1^2 + \beta_2^2 + \cdots + \beta_p^2).$$

If we consider a regression function with a single parameter, then

$$L_1(\beta) = |\beta_1| \quad \text{and} \quad \frac{dL_1(\beta)}{d\beta} = \frac{\beta_1}{|\beta_1|},$$

$$L_2(\beta) = \frac{1}{2} \beta_1^2 \quad \text{and} \quad \frac{dL_2(\beta)}{d\beta} = \beta_1.$$
Further, we plot each function, and also the corresponding derivative against $\beta_1$ in turn; the results obtained are presented in Figure 4.2.

![Graphs for $L_1$ and $L_2$ functions](image)

(a) Plot of $\beta_1$ against $L_1$ function, where $\beta_1$ is on the x-axis and $L_1$ on the y-axis.

(b) Plot of $\beta_1$ against derivative of $L_1$ function, where $\beta_1$ is on the x-axis and derivative of $L_1$ on the y-axis.

(c) Plot of $\beta_1$ against $L_2$ function, where $\beta_1$ is on the x-axis and $L_2$ function on the y-axis.

(d) Plot of $\beta_1$ against derivative of $L_2$ function, where $\beta_1$ is on the x-axis and $L_2$ function on the y-axis.

Figure 4.2: Plots for $L_1$ and $L_2$ functions in a one parameter model.

With reference to Figure 4.2(b), we notice that the gradient is either 1 or $-1$, except for $\beta_1 = 0$ which is the case on the vertical axis between 1 and $-1$. It
shows that the $L_1$ norm is capable of moving any weight to zero, thus creating a sparse solution in the end.

In contrast, Figure 4.2(d) shows that the $L_2$ function is differentiable even at point zero. The size of the gradient linearly decreases towards zero, as the absolute weight moves towards zero. For this reason, the $L_2$ norm can move a weight towards zero, but it does not mean that the weight can take the value zero.

The intuition gained here will help us in Section 4.4, to understand the discussion on estimation of vectors of coefficients using RRDA and LaDA. In particular, we shall understand that any solution terminating with a value on the vertical axis, returns a zero coefficient for LaDA, and a coefficient very close to zero for RRDA.

### 4.4 Estimation of Vector of Coefficients using RRDA and LaDA

The knowledge gained in the preceding discussion will be applied here, in explaining how LaDA can set some coefficients to zero, whereas RRDA is unable to do so. The description we shall present, including Figure 4.3, are patterned after a discussion on ridge regression and Lasso contained in (T. Hastie et al., 2015).

In Figure 4.3, we present illustrations in two dimensions of estimation of coefficients using RRDA and LaDA. The solid blue areas are the constraint regions $\beta_1^2 + \beta_2^2 \leq s$ and $|\beta_1| + |\beta_2| \leq s$ respectively for RRDA and LaDA. The red ellipses are examples of the error function based on RDA. The solid black dot in the middle of the contours is the smallest possible value for the error function.
Concerning Figure 4.3(a), if the constraint is very large, in other words we allow the coefficients to be as big as possible, the solid blue circle will enlarge and this causes the red contours to shrink. Thus, the point of intersection between the circle and red contours will move closer to the minimum point. This minimum point gives a solution, and it is the same as the RDA solution. If on the other hand the constraint is smaller, the circle will shrink and the contours will enlarge. The point where both red contours and the circle intersect is the RRDA solution, in the form of RRDA coefficient. The RRDA solution obtained can be close to zero depending on the value chosen for $s$, but not exactly zero, as we had earlier indicated.

In the case of Figure 4.3(b), the constraint region and the red contours can intersect either on the vertical axis or on the surface. If the point of intersection takes place
on the vertical axis, a solution obtains but the LaDA coefficient in question is set to zero. Intersection that takes place on the surface provides a solution in the form of non-zero LaDA coefficients.

### 4.5 RRDA and LaDA Trace

RRDA and LaDA traces are graphical illustrations that explain how the coefficients change, as we vary the values of regularization parameter $\lambda$. It also helps to visualize how some variables enter into the classification algorithm, and others drop out as $\lambda$ assumes various values. It was first used by (T. Hastie et al., 2015)

![Graphs showing RRDA and LaDA traces](image)

(a) RRDA trace for the first eight variables of Ionosphere dataset  
(b) LaDA trace for the first eight variables of Ionosphere dataset

**Figure 4.4:** RRDA and LaDA traces using a classification based dataset.

on a regression dataset, but we adapt it for a classification dataset, since the same explanation applies to RRDA and LaDA.
CHAPTER 4. REGRESSION DISCRIMINANT ANALYSIS (RDA)

Figure 4.4 consists of RRDA and LaDA traces for the first eight variables of the Ionosphere dataset, described in Section 4.2.1. Each curve here represents a variable, and it shows a path for the coefficient as $\lambda$ changes value. The y-axis indicates the number of non-zero coefficients at the current value of $\lambda$. In Figure 4.4(a), we observe that the coefficients of the variables can be very small or even close to zero, but not exactly zero. This is a unique feature of RRDA (Section 1.3.1.1), which makes the classifier able to handle numerically unstable datasets. On the other hand, when the number of dimensions becomes very high, RRDA may not be a preferred classifier because all explanatory variables are retained in the classification algorithm.

In the case of Figure 4.4(b), we observe that at certain values of $\lambda$, some variables do not stand a chance of entering into the classification algorithm, since the coefficients of such variables are exactly zero. For instance, if we consider Log Lambda = $-3$ as a threshold for adding variables into the model, two variables will be dropped because at the value of $\lambda$ their coefficients are exactly zero. The ability of LaDA to set some coefficients to zero can be seen as one advantage it has over RRDA. Particularly in high dimensions, this additional feature of LaDA pays off, and as a result, the classifier may be preferred to RRDA.

4.5.1 Some Illustrations

In this illustration, we aim to show that both RRDA and LaDA are valid binary classifiers, and that their error rates may not be significantly different. We shall use four different datasets, namely datasets Australia and Ionosphere, a 40-dimensional simulated dataset, and a hill valley without noise dataset.
Australia and Ionosphere datasets were already described in Section 4.2.1. The 40-dimensional dataset was generated using equicorrelation matrix for Σ, with ρ = 0.19, and a proportional mean difference for μ. Altogether, 25% of the variables were made to correlate with other variables. The idea is to make the dataset numerically unstable, and also to help investigate the ability of LaDA to particularly isolate the correlated variables without compromising accuracy.

The hill valley without noise (HVWON) data consists of 606 instances and 100 features, for both training and test sets. Noise contamination of the dataset has been removed, thereby differentiating it from the hill valley with noise dataset (see Section 5.2). Each instance (sample) represents 100 points on a two dimensional graph. The points will create either a Hill (a “bump” in the terrain) or a Valley (a “dip” in the terrain). Hills are represented by class −1 and Valley represented by class +1. The data is sourced from the UCI Machine Learning Repository.

In respect of the simulated dataset, the size of the training set is 600 and the test set is 400. With the exception of the hill valley dataset, 70% of each of the other datasets was used as the training set, and the remaining 30% used as the test set. In the case of the hill valley dataset, both training and test datasets were provided by the authors. We included the two unstable datasets (simulated and hill valley datasets) in order to demonstrate the ability of both classifiers to handle dataset instability, often arising from correlation among variables. Also, we used the glmnet package (Simon et al., 2011) in R, to obtain the coefficient vectors in respect of RRDA and LaDA (Appendices D.3 and D.2), because the package is adapted for both ridge regression and Lasso. We used a two fold cross validation, and the choice of λ is based on a grid search. We implemented the grid search
using the glmnnet package in R, also contained in Appendices D.2 and D.3.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>LaDA</th>
<th>RRDA</th>
<th>Diff</th>
<th>No. of Variables Used by LaDA</th>
<th>Percentage of Variables Used by LaDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
<td>0.1353</td>
<td>0.1353</td>
<td>0</td>
<td>8</td>
<td>57.14%</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.1619</td>
<td>0.1714</td>
<td>-0.0095</td>
<td>20</td>
<td>62.50%</td>
</tr>
<tr>
<td>HVWON</td>
<td>0.3069</td>
<td>0.3300</td>
<td>-0.0231</td>
<td>38</td>
<td>38%</td>
</tr>
<tr>
<td>Simulated dataset</td>
<td>0.0425</td>
<td>0.0362</td>
<td>0.0063</td>
<td>30</td>
<td>75%</td>
</tr>
</tbody>
</table>

Table 4.1: Error rates of RRDA and LaDA on four different datasets. The error rates were obtained using a two fold cross validation, consisting of the training and test sets. For both Australia and Ionosphere datasets, we used 70% of each dataset to train the classifiers and the remaining 30% used to test their performances. Regarding the simulated dataset, 60% of the dataset was used for training the classifiers and the remaining 40% used to test their performances. The data source for HVWON supplied both training and test samples.

The error rates of RRDA and LaDA on the datasets are contained in Table 4.1. On considering the small differences in the observed error rates, one can argue that both classifiers have not shown significant differences in their classification results given the datasets. A valid test of hypothesis would have confirmed or otherwise such argument, but since only a small number of datasets were used, in Chapter 5, more detailed comparisons will be considered, mainly for inferential purposes. In respect of the Australia dataset, the error rate of 0.1353 tallied with the result obtained using FDA and RDA (Section 4.2.1). In the case of the Ionosphere dataset, we observed that the error rates of RRDA and LaDA are marginally lower than what was obtained using both FDA and RDA. These give us the impression that RRDA and LaDA can be competitive in comparison with FDA.
On examining the number of explanatory variables used by LaDA to obtain its error rates, it can be argued that, overall the classifier is preferred to RRDA. Our argument is based on considering some cost implications. For instance, assuming the costs of measuring the explanatory variables are expensive, the use of RRDA as a classifier will entail more expenses because every variable must be included in the model. Also, assuming in Table 4.1, the costs of measuring the explanatory variables are the same for each given dataset, the use of LaDA will lead to reduction in costs of 42.86% for the Australia dataset, 37.50% for the Ionosphere dataset and 62% for the hill valley dataset.

4.5.2 Benefits of Using LaDA as a Classifier

The use of LaDA brings about sparsity, referring to the presence of fewer explanatory variables in the predictive model than in the input data. Sparsity is present when a classifier sets some coefficients to zero, meaning that the explanatory variables connected to the coefficients are not of predictive relevance. LaDA is comparable to the work of (Clemmensen et al., 2011), which is about sparse discriminant analysis.

In Table 4.1, the percentages of explanatory variables used by LaDA to obtain the error rates show that some coefficients were set to zero, thus resulting in a reduced number of explanatory variables in the classification algorithm. We note that it is not in all cases that LaDA will set some coefficients to zero. If for instance all explanatory variables in the training set contain predictive information, they will all be retained. In most cases however, this will not happen.
Regarding the simulated dataset, we discovered that initially only two coefficients were exactly zero, and the error rate using 38 variables was 0.0425. Since we originally correlated 25% of the explanatory variables with each other, we expected at least 10 explanatory variables to be set to zero by LaDA. Since the number of explanatory variables dropped by the classifier fell short of expectation, we investigated further and discovered that some coefficients were very close to zero. We considered these coefficients for removal, and eventually removed all the coefficients that were less than 0.00042. In the end, an additional 8 explanatory variables were removed. With the remaining 30 explanatory variables, we still obtained the same error rate of 0.0425.

This finding shows that if we aim to use as few a number of explanatory variables as possible in a classification problem, then the LaDA coefficients that are very close to zero can be removed, in addition to those that are exactly zero. The classification result that obtains eventually may not change noticeably, compared to when only coefficients that are exactly zero are removed.
Chapter 5

Statistical Investigations

5.1 Introduction

In Chapter 4, we claimed that regression variants, namely ridge regression and Lasso, could be used in binary classification. We later named the binary classification algorithms after the ridge regression and Lasso, RRDA and LaDA respectively, and further claimed that their different error rates are not statistically different from the error rate of FDA, when used on the same dataset (Section 4.3). In Section 4.5.1, the results we obtained suggested that differences in the error rates of RRDA and LaDA are not statistically different from 0. We also saw in the same Section that RRDA and LaDA arguably compete favourably with FDA. However, we did not carry out relevant tests of hypothesis to confirm our claims or otherwise, because we used a small number of datasets in the investigation.

For this reason, we shall revisit the problem with a view to investigate whether
the error rates of FDA, RRDA and LaDA significantly differ or otherwise. We shall further expand the classifiers to include RFDA, since it is a variant of FDA. Moreover, RFDA can be used to correct the limitation of FDA when \( p >> n \). Hence, we shall investigate the nature of differences among RFDA, RRDA and LaDA, since the three classifiers can be used in place of FDA when datasets are numerically unstable. We would also like to know if there are peculiar features which each of the classifiers possesses that can distinguish each one. Specifically, we shall investigate the execution time for each classifier, as the number of explanatory variables becomes larger. We hope the investigation will help us gain insight on the usefulness of the classifiers, particularly in a high dimensional setting.

On the benefits of using LaDA as a classifier, contained in Section 4.5.2, we mainly talked about the sparse model it is able to produce. Aware that sparsity can also connote variable selection, we shall, in this Chapter too, verify how non-informative, in terms of usefulness in the construction of classification functions, the variables discarded by LaDA are likely to be. Through this investigation, we would like to know if LaDA can be recommended as a valid variable selection tool, if we intend to use a different classifier, or whether the variables LaDA selects are only useful for LaDA itself. Summarily, we shall carry out three different investigations in this chapter.

### 5.2 Datasets

The majority of the datasets used in the investigation were sourced from the UCI Machine Learning Repository (Lichman, 2013), and KEEL dataset repository
(Alcalá et al., 2010). We preprocessed all the datasets to ensure that each class label is identified with the name “class”, and consists of a vector of +1 and −1 discrete variables. The datasets include:

**Appendicitis**

The data represents 7 medical measures taken over 106 patients on which the class label represents whether the patient has appendicitis (class label +1) or not (class label −1). We have a total of 21 samples in class +1 whereas class −1 consists of 85 samples. The dataset was sourced from KEEL dataset repository.

**CoIL 2000**

The dataset was used in CoIL 2000 challenge, and contains information on customers of an insurance company. It is a binary classification dataset, and consists of 85 variables including product usage data, and socio-demographic data. The number of samples involved is 9822, with a total of 9236 in class +1 and 586 in class −1. It was sourced from the UCI Machine Learning Repository.

**Colon**

Colon is a gene expression dataset from the microarray experiments of colon tissue samples (Alon et al., 1999). The dataset consists of 62 samples and 2000 genes (features). It has two classes, namely tumour tissue with 40 samples, and normal tissue with 22 samples. It is contained in the plsgenomics package in R. The names of the genes were not given and we represented them conveniently.

**Gisette**

The dataset is one of five datasets used in the NIPS 2003 feature selection challenge, and it was put together by (Guyon, 2003). The sample size is 7000, with 5000
features and each of the two classes has 3500 samples. The dataset is also contained in the UCI Machine Learning repository.

**Handheight**

The Handheight dataset is two dimensional, and consists of heights and stretched hand span of 167 male and female college students. Each student decided which of their hands to measure. Class +1 has 89 samples whereas class −1 consists of 78 samples. The source of the data is (Utts et al., 2011).

**Heart**

This is a real world binary classification heart disease dataset, and the task is to detect the absence (−1) or presence (1) of heart disease. It contains 270 samples and 13 features, with 120 samples in class +1 and 150 samples in class −1. The data was sourced from the UCI Machine Learning Repository.

**Heberman**

This dataset contains cases from a study that was conducted between 1958 and 1970, at the University of Chicago’s Billings Hospital, on the survival of patients who had undergone surgery for breast cancer. The task is to determine if the patient survived 5 years or longer (positive) or if the patient died within 5 year (negative). The sample size is 306 with 3 features, and class +1 has 225 samples whereas class −1 contains 81 samples. The dataset was sourced from the KEEL dataset repository.

**Hepatitis**

Hepatitis is a real world dataset; it contains a mixture of integer and real valued attributes, with information about patients affected by the hepatitis disease. It
consists of 80 samples and 19 features. Class +1 has 67 samples whereas class −1 has 13 samples, and the task is to predict if these patients will die (−1) or survive (1). It was sourced from the UCI Machine Learning Repository.

**Hill valley with noise (HVWN)**

The hill valley with noise dataset consists of 606 instances, and 100 features for both training and test sets. Noise contamination of the dataset is retained, thereby differentiating it from the hill valley without noise dataset. The data was sourced from the UCI Machine Learning Repository.

**Leukemia**

The leukemia dataset is a gene expression data consisting of 3051 genes, with 38 tumour MRNA samples from the leukemia microarray study (Golub et al., 1999). The tumour MRNA samples are of two cancerous classes, here denoted as −1 and +1. Since the number of training samples is small, we used the same dataset for training to also test the classifiers. Our interest here is merely in the performances of the classifiers given such scenarios. The dataset is contained in R package plsgenomics. Although the gene names were given, we have represented them conveniently because they are lengthy, and we have no intrinsic interest in the gene names.

**Magic**

This dataset was used to simulate registration of high energy gamma particles, in a ground-based atmospheric Cherenkov gamma telescope, using the imaging technique. The dataset was generated by a Monte Carlo program (Bretz et al., 2009), and the task is to discriminate statistically images generated by primary gammas, from the images of hadronic showers initiated by cosmic rays in the
upper atmosphere. It contains 19020 samples and 10 features; the source is the UCI Machine Learning Repository.

**Mammographic**

This dataset was used to predict the severity (benign or malignant) of a mammographic mass lesion from BI-RADS attributes and the patient’s age. It contains a BI-RADS assessment, the patient’s age and three BI-RADS attributes together with the ground truth (the severity field, which is the target attribute). The dataset was collected at the Institute of Radiology of the University Erlangen-Nuremberg between 2003 and 2006. It has dimensions $830 \times 5$, and the source is the KEEL dataset repository.

**Parkinsons**

The Parkinsons dataset is of dimension $195 \times 23$, and involves a range of biomedical voice measurements of some people with and without Parkinson’s disease (PD). It was sourced from the UCI Machine Learning Repository. Documentation on the dataset shows that each column is a particular voice measure, and each row corresponds to one of 195 voice recordings from these individuals.

**Prostate**

The prostate dataset is a gene expression dataset (Singh et al., 2002). The dataset is contained in R package spls and consists of two classes, namely 52 prostate tumour and 50 normal classes. The number of genes involved is 6033. The names of the genes were not given and as a result, we represented them conveniently.

**Ringnorm**

Ringnorm is a 20 dimensional, 2 class classification dataset. Each class is drawn
from a multivariate normal distribution, and class 1 has mean 0 and covariance 4 times the identity. Class 2 has mean \((a, a, \cdots, a)\) and unit covariance \((a = 2/\sqrt{20})\).

The number of instances is 7400, and like most simulated datasets, the dataset is useful for testing performances of binary classifiers. The source is the KEEL dataset repository.

**Saheart**

The Saheart dataset pertains to a retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa. There are roughly two controls per case of CHD. Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their CHD event. In some cases the measurements were made after these treatments. The saheart data were taken from a larger dataset described in (Rousseauw et al., 1983). The class label indicates whether the person has a coronary heart disease: negative (−1) or positive (+1). The dataset has dimensions \(462 \times 9\), and is contained in the ElemStatLearn package in R.

**Sonar**

This dataset contains how many signals obtained from a variety of different aspect angles, spanning 90 degrees for mines and 180 degrees for rocks. Each pattern is a set of 60 numbers in the range 0.0 to 1.0, where each number represents the energy within a particular frequency band, integrated over a certain period of time. The output attribute contains the letter +1 if the object is a rock and −1 if it is a mine (metal cylinder). The source is the UCI Machine Learning Repository.

**Spambase**

The spambase dataset contains information about 4597 e-mail messages. The
task is to determine whether a given email is spam (class +1) or not (class –1), depending on its contents (4 duplicated instances have been removed from the original data set). Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail. The dataset was sourced from the UCI Machine Learning Repository.

**SPECTF Heart**

The SPECTF Heart dataset is of dimension 267 × 44, and consists of diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient involved in the study is classified into one of two categories: normal and abnormal. Altogether, 44 continuous feature patterns were created for each patient. The source of the dataset is UCI Machine Learning Repository.

**Twonorm**

This dataset is 20 dimensional, and consists of 2 classes. Each class is drawn from a multivariate normal distribution. Class +1 has mean \((a, a, ..a)\) while Class –1 has mean \((-a, -a, .. -a)\). \(a = 2/sqrt(20)\). The dataset has dimensions 7400 × 20, and is contained in the KEEL dataset repository.

**Wisconsin Diagnostic Breast Cancer (WDBC) Dataset**

WDBC is a real world dataset, and contains 30 features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The number of instances is 569 and the task is to determine if a tumour found is benign or malignant (–1 = malignant, and 1 = benign). It was sourced from the UCI Machine Learning Repository.
5.3 Comparison of the Error Rates of FDA, RRDA and LaDA

In this study, we shall use only numerically stable datasets because FDA is involved. Moreover, we require to obtain the error rate of each classifier, given each dataset considered in the study in order to carry out a detailed comparison. In Table 5.1, we present the error rates of the three classifiers.

The error rates were obtained using a two fold cross validation (training and test sets). With the exception of HVWN dataset, 70% of each dataset was used to train each classifier, and the remaining 30% used as test set. In the case of HVMV dataset, both the training and test sets were provided by the dataset source. Regarding RRDA and LaDA, the choice of the regularization parameter $\lambda$ was based on a grid search. The grid search was carried out using a glmnet package in R, as contained in Appendix D.

On preliminary investigation of Table 5.1, we observed that the error rates of the classifiers are fairly close to each other, in most of the samples considered. What may appear as an exception is the Sonar dataset. Here, the difference in error rate between LaDA and the other classifiers is 0.0807. On the other hand, out of 60 explanatory variables the Sonar dataset consists of, only 17 (28.33%) of the variables were used to construct the LaDA classification algorithm. In other words, with a saving of 71.67%, assuming that cost is attached to variables measurement, we cannot say that LaDA performed badly given the dataset.

Similarly, the error rates of FDA, LaDA and RRDA on HVWN dataset appear
Table 5.1: Error rates of FDA, LaDA and RRDA on different datasets

to be slightly unusual. The three error rates are all above 30%, with FDA error rate marginally higher than others, followed by LaDA error rate and lastly, RRDA error rate. However, differences in the error rates between FDA vs LaDA, FDA vs RRDA and LaDA vs RRDA are 0.0495, 0.0808 and 0.0333 respectively. It can be argued that the differences are marginal, and this suggests that the classifiers
may not have differed in their performances.

In Figure 5.1, the box plots of the error rates of FDA, LaDA and RRDA are given. The plots show that the median error rates of LaDA and RRDA slightly differ, whereas the median error rate of FDA seems to be a little lower comparatively. Based on the sizes of the boxes, we have more variations in the error rates of LaDA than with both FDA and RRDA. Variations in the error rates of RRDA appear to be in the least, which could suggest that the classifier may be having the upper hand vis-a-vis FDA and LaDA.

A preferred statistical tool to investigate if the error rates of the classifiers significantly differ from each other, or otherwise, is the one way ANOVA. On the other hand, since we used the three classifiers each time on the same dataset, the error
rates of the classifiers will consist of correlated observations. This means that inde-
pendence assumption necessarily for the use of one way ANOVA has particularly
failed. Alternatively, we shall use one way repeated measure ANOVA under the
assumptions that:

- The error rates of each classifier consist of independent observations.
- The error rates of each classifier follow a normal distribution.
- The variances of differences in error rates between all combinations of the
classifiers are equal.

5.3.1 Compliance with the Underlying Assumptions

We can confirm that the first assumption is met because each dataset is indepen-
dent of the other. As a result, the error rates consist of independent observations
within each classifier, but not independent observations between the classifiers,
since the same dataset is used by the classifiers one after the other.

Figure 5.2: Normal Q-Q plots in respect of FDA, LaDA and RRDA.

On the second assumption, the Q-Q plots of Figure 5.2 suggest that there is com-
CHAPTER 5. STATISTICAL INVESTIGATIONS

Compliance with the normality assumption given the classifiers error rates. It seems that the Q-Q plot of FDA shows evidence of slight departure from normality, but on the other hand, we do not have strong evidence to conclude that normality assumption is violated here. Furthermore, the Shapiro-Wilk normality test could not reject compliance with the normality assumption at p-values of 0.6001, 0.6760 and 0.7825 in respect of FDA, LaDA and RRDA respectively. We therefore conclude that normality assumption is also met.

Concerning the third assumption, the Bartlett’s test failed to reject compliance with the homogeneity of variances at a p-value of 0.8531.

Having complied with the required assumptions, we carried out repeated measures ANOVA using R (R Core Team, 2016), and obtained the following output:

Error: Datasets

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residuals</td>
<td>18</td>
<td>0.4627</td>
<td>0.0257</td>
<td></td>
</tr>
</tbody>
</table>

Error: Datasets:Classifiers

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifiers</td>
<td>2</td>
<td>0.000879</td>
<td>0.0004395</td>
<td>1.15</td>
</tr>
<tr>
<td>Residuals</td>
<td>36</td>
<td>0.013755</td>
<td>0.0003821</td>
<td></td>
</tr>
</tbody>
</table>

Note that the treatment we have interest in is the classifiers, and this treatment effect is visible within each dataset, since we repeated the treatment on each dataset. Thus, the proper Error term (see R codes in Appendix D.5) is Dataset/Classifiers, which is read as classifiers within datasets. This explains the presence of Error:
Datasets: Classifiers in the output.

At a p-value of 0.328, there is no reason to reject the hypothesis that the mean error rates of the classifiers are not significantly different from each other. Thus we conclude that the classifiers are as good as each other.

5.4 Comparison of the Error Rates of RFDA, RRDA and LaDA

In this investigation, we shall involve different numerically stable and unstable datasets. It differs from the former investigation because the three classifiers in question are able to handle both numerically stable and unstable datasets. Since the classifiers have something in common with FDA, any of them can be used in place of FDA when $p >> n$. Consequently, we would like to know which of them, if any, significantly differs in its error rates given the different datasets. Put differently, we aim to find out if any of the classifiers can be recommended as a preferred alternative to FDA, when $p >> n$. This means we shall investigate the responses of the classifiers, as the number of explanatory variables becomes larger. Such responses will involve looking into the system time it takes each classifier to finish execution. In the end, we hope the investigation will help to identify the classifiers that may be more useful in high dimensions.

The error rates of the three classifiers are contained in Table 5.2. Similar to the experiment conducted in Section 5.3, we obtained the error rates using a two fold cross validation. With the exception of Colon, Hepatitis, HVWN, HVWON, and
Leukaemia datasets, 70% of each dataset was used as the training set and the remaining 30% used as test set. In respect of Colon and Leukaemia datasets, the same datasets used for training the classifiers were also used as test sets. We took that decision because the sizes of the two datasets are comparatively smaller. Moreover, our interest is in the responses of the classifiers given such scenarios. Concerning Hepatitis dataset, 55% of the dataset was used as training set and the remaining 45% used as test set. As we earlier remarked, the source of HVWN and HVWON datasets provided both training and test sets.

The choice of the regularization parameter $\lambda$, for each classifier, was based on a grid search with the help of a glmnet package in R. The discovery of the usefulness of the package, in choosing the best value of lambda for RFDA was important to us because until now, the method we used took comparatively more time. It involved training a given classifier with each value of $\lambda$ in the grid, and evaluating the performances of the classifier on the test set. In the end, the value of $\lambda$ that gives the smallest error rate is preferred for the classifier. This procedure takes a lot of time particularly if we have too many values of $\lambda$ in the grid.

On examining Table 5.2, it appears that differences in the error rates of the classifiers on most datasets are not significant. Exceptions may include six datasets, namely the Colon, Hepatitis, HVWN, HVWON, SPECT Heart and Sona datasets. The error rates of these datasets are comparatively higher. It seems that the RRDA error rates are marginally smaller on Hepatitis, HVWN, SPECTF Heart and Sona datasets. LaDA seems to have marginally smaller error rates on Colon dataset, whereas RFDA recorded a marginally smaller error rates on HVWON dataset. Altogether, it appears that RRDA performed relatively better on the six datasets,
Table 5.2: Error rates of RFDA, LaDA and RRDA, and the system time it took each classifier to finish execution given different datasets.

but it can still be argued that the different error rates between RRDA and others are not significant.

The box plots of Figure 5.3(a) show that LaDA has a marginally smaller median error rate, in comparison with RRDA and RFDA. The median error rates of RRDA
and RFDA appear to be similar. However, based on the size of the LaDA box, it seems that we have more variations in error rates of LaDA than in the error rates of RRDA and RFDA.

(a) Box plots of the error rates of LaDA, RRDA and RFDA
(b) Box plots of the system time for LaDA, RRDA and RFDA

Figure 5.3: Box plots in respect of the error rates and system time for LaDA, RRDA and RFDA.

On the other hand, differences in the time it took each classifier to finish execution may be seen as insignificant with some datasets, and very significant with other datasets. For instance, with datasets Australia, Handheight, Heart, Mammographic, Parkinsons, Saheart, Sonar, Twonorm and WDBC, differences in system time appear insignificant. The same may not be true with datasets like Gisette, Prostate, and possibly, Colon and Leukaemia. It seems that in the instances where \( n > p \), the system time is relatively smaller vis-a-vis when \( p > n \). An exception here is the dataset Gisette, because despite the fact that \( n > p \) the system time for the classifiers are still very high. The RFDA particularly recorded the highest system time on datasets Gisette and Prostate. They constitute the two most prominent outliers in Figure 5.3(b). Comparatively, it seems we have more out-
liers with the system time for RFDA. The implication is that RFDA may not be a preferred classifier in high dimensions.

We shall use repeated measures ANOVA to test for existence of significant differences in the error rates of the classifiers, or otherwise. As we saw in Section 5.3, we shall first ensure that all the underlying assumptions are met.

### 5.4.1 Compliance with the Underlying Assumptions

The error rates consist of independent observations given each classifier, because each classifier’s error rate is dependent on a corresponding dataset. Since the datasets are independent of each other, the corresponding error rates given a classifier are independent of each other. We therefore confirm that the assumption of independence is met.

![Normal Q-Q plots](image)

(a) Normal Q-Q plot for RFDA on 25 datasets.  
(b) Normal Q-Q plot for LaDA on 25 datasets.  
(c) Normal Q-Q plot for RRDA on 25 datasets.

Figure 5.4: Normal Q-Q plots in respect of RFDA, LaDA and RRDA on 25 datasets.

The normal Q-Q plots of Figure 5.4 suggest that the error rates of the classifiers are normally distributed. It may be argued that the plots for LaDA and RRDA show signs of slight departures from normality, but we do not have enough evidence
to argue that normality assumption has been violated given the two classifiers. Furthermore, the Shapiro Wilks normality test failed to reject compliance with normality assumption is respect of LaDA, RRDA and RFDA at p-values 0.4059, 0.383 and 0.9509 respectively. We therefore confirm that we have complied with the normality assumption.

Also, the Bartlett’s test we carried out failed to reject compliance with the homogeneity of variances at a p-value of 0.9590. We similarly confirm that we met this assumption.

Having satisfied all the required assumptions, we carried out repeated measures ANOVA and obtained the following output in R:

Error: Datasets

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residuals</td>
<td>24</td>
<td>0.7426</td>
<td>0.03094</td>
<td></td>
</tr>
</tbody>
</table>

Error: Datasets:Classifiers

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifiers</td>
<td>2</td>
<td>0.003743</td>
<td>0.0018714</td>
<td>4.069</td>
</tr>
<tr>
<td>Residuals</td>
<td>48</td>
<td>0.022075</td>
<td>0.0004599</td>
<td></td>
</tr>
</tbody>
</table>

Based on the p-value of 0.0233, the hypothesis that differences in the error rates of the classifiers are non significant is rejected. Tukey’s post hoc test would have been carried out at this stage, but it does not work with the repeated measures ANOVA model. Alternatively, we considered a post hoc test based on paired t-test, and obtained the following output in R:

Pairwise comparisons using paired t tests
data: Values and Classifiers

LaDA  RFDA
RFDA  0.057  -
RRDA  0.437  0.013
P value adjustment method: none

This result shows that differences in the error rates between LaDA and RRDA are non significant at a p-value of 0.437. We narrowly rejected the hypothesis that differences in error rates between RFDA and LaDA are significant at a p-value of 0.057. Lastly, we failed to reject the hypothesis that differences in error rates between RFDA and RRDA are significant at a p-value of 0.013. By considering the p-value at which we rejected the existence of significant differences between the error rates of RFDA and LaDA, one may be cautious to assume that both classifiers are as good as the other. If we further consider the number of variables LaDA used, we are therefore of the view that LaDA performed better than RFDA. In all, based on the error rates, we argue that RRDA relatively performed better, followed by LaDA and lastly, RFDA.

Assuming we considered a paired t-test with a Bonferroni adjustment, we would have obtained the following result:

Pairwise comparisons using paired t tests
data: Values and Classifiers

<table>
<thead>
<tr>
<th></th>
<th>LaDA</th>
<th>RFDA</th>
<th>RRDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaDA</td>
<td></td>
<td>0.17</td>
<td>-</td>
</tr>
<tr>
<td>RFDA</td>
<td>0.17</td>
<td></td>
<td>0.04</td>
</tr>
<tr>
<td>RRDA</td>
<td>1.00</td>
<td>0.04</td>
<td></td>
</tr>
</tbody>
</table>

P value adjustment method: bonferroni

The Bonferoni adjustment gives a result similar to the paired t-test without adjustment. For this reason, our interpretation of the performances of the classifiers remains unaltered.
In terms of the system time, it is our view that since we have heavy presence of outliers in the box plot for RFDA, differences in the system time will likely be significant. We also suspect that following the heavy presence of outliers too, both normality and homogeneity of variances assumptions would be violated, hence the use of repeated measures ANOVA may give a misleading result.

In the same vein, the histograms of Figure 5.5 strongly suggest that the system time in respect of each classifier does not follow a normal distribution. It is clearly caused by the heavy presence of outliers. Also confirming our suspicion of the violation of normality assumption, the Shapiro Wilks normality test rejected compliance with the normality assumption in respect of system time of LaDA, RRDA and RFDA at p-values of $1.48e-10$, $1.429e-10$ and $1.012e-09$ respectively.

Similarly, the Bartlett’s test on homogeneity of variances rejected compliance with the homogeneity of variance assumption at a p-value of $2.2e-16$. In view of these violations, we therefore resort to the use of a non parametric alternative; the Friedman’s test.
The output of the Friedman’s test in R gives:

Friedman rank sum test
data:  datx

Friedman chi-squared = 8.7475, df = 2, p-value = 0.0126

With a p-value of 0.0126, at 5% level of significance, we reject the null hypothesis of no difference in the system time of the three classifiers. This means that the observed differences are significant, hence, a post hoc analysis for the Friedman’s test will follow.

In this regard, we used the Nemenyi post hoc test (Nemenyi, 1962) and obtained the following output in R, at a 5% level of significance:

Pairwise comparisons using Nemenyi multiple comparison test
data:  datx

<table>
<thead>
<tr>
<th></th>
<th>LaDA.SysTime</th>
<th>RRDA.SysTime</th>
<th>RRDA.SysTime</th>
<th>RFDA.SysTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaDA.SysTime</td>
<td></td>
<td>0.989</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>RRDA.SysTime</td>
<td>0.036</td>
<td>0.024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFDA.SysTime</td>
<td>0.036</td>
<td>0.024</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Nemenyi test shows that we have significant differences in system time between LaDA and RFDA, and between RRDA and RFDA. Differences in the system time between LaDA and RRDA are non significant at a p-value of 0.989.

Because we suspected that datasets like Colon, Gisette, Leukaemia and Prostate may be responsible for significant differences in the system time between RFDA and others, we removed them and carried out the Friedman test again. At a p-value of 8.598e-05, at the same level of significance, we equally rejected the null hypothesis of no difference in the system time between RFDA and other two
classifiers.

A post-hoc Nemenyi test similarly rejected the null hypothesis of no difference in system time between RFDA and LaDA, and between RFDA and RRDA at p-values of 0.00025 and 0.00150 respectively. Also at a p-value of 0.88862, we failed to reject differences in the system time between LaDA and RRDA.

We equally observed that the dimensions of the datasets, where RFDA recorded increased number of system time are relatively higher. It suggests that increase in the dimensions of a dataset, would mean more system time for RFDA.

On account of the foregoing, we state that based on the system time, either of LaDA or RRDA can be preferred to the use of RFDA. In high dimensions, we recommend the use of LaDA as a preferred classifier in place of FDA, primarily for its additional feature as a variable selector.

### 5.5 Verification of the Effectiveness of Variable Selection Using LaDA

The investigation here will involve the use of FDA, RRDA and LaDA. In the first place, we shall use LaDA to carry out variable selection. Thereafter, FDA and RRDA will be used in turn for classification with the full set of variables, and variables selected using LaDA. We shall then compare the differences in error rates between using all variables, and using only variables selected by LaDA for each classifier. Our argument is that if most of the useful information is contained in the variables selected by LaDA, we do not expect to find significant differences in
the two error rates. Otherwise, differences in error rates will be very significant, and in that case, we shall argue that LaDA is not a suitable instrument for variable selection.

Table 5.3: Error rates using all the explanatory variables (Full Var), and variables selected with LaDA (LaDA Var) for both FDA and RRDA.

Table 5.3 shows differences in the error rates (between using full variables and LaDA selected variables) for both FDA and RRDA. On examination, it appears that differences between the two error rates are generally non significant for both classifiers. On a number of datasets, differences in the error rates are even zero,
meaning that given those datasets, the variables discarded by LaDA do not contain any predictive information. With some datasets, we observed very small negative differences. For those datasets, variables discarded by LaDA have very small contributions to prediction of classes. Datasets that fall into this category with at least 4% negative difference include Sona and Hepatitis. Given the two datasets, we can argue that LaDA variables marginally performed worse than the full variables.

We equally observed positive differences on some datasets. Here, the implication is that dropping some variables using LaDA further improved classification results. Therefore given the datasets in this category, LaDA variables contain more predictive information vis-a-vis the full variables. The datasets where we observed at least 4% positive difference include HVWN and Sona.

We attribute this development to possible noisy contamination of some explanatory variables, which could further reduce their predictive relevance. Since LaDA will detect those variables, and eventually removes them, we are then likely to observe reduced error rates with LaDA selected variables.

In spite of the evidence to accept that variables selected using LaDA are as good as using all the explanatory variables, a valid test of hypothesis is nevertheless necessary. A paired t-test is suitable in this regard, but the normality assumption on differences in error rates are not satisfied in both cases. For instance, the p-values for Shapiro Wilk’s normality test, on differences in error rates between using the full variables, and using variables selected by LaDA for both FDA and RRDA, are 0.001696 and 0.0001544 respectively. We therefore reject the assumption of normality in both cases, at a 5% level of significance. For this reason, we shall use a non-parametric alternative, namely the Wilcoxon signed rank test (Woolson,
In respect of FDA, the output of Wilcoxon signed rank test in R is:

```
Wilcoxon signed rank test with continuity correction
data:  FDA_Full.Var and FDA_LaDA.Var
V = 35, p-value = 0.7837
alternative hypothesis: true location shift is not equal to 0
```

With a p-value of 0.7837, at a 5% level of significance, the null hypothesis that there is no observed difference in the two error rates cannot be rejected.

Similarly in the case of RRDA, we obtained the following output:

```
Wilcoxon signed rank test with continuity correction
data:  RRDA.Full.Var and RRDA_LaDA.Var
V = 36, p-value = 0.1033
alternative hypothesis: true location shift is not equal to 0
```

Also with a p-value of 0.1033, at a 5% level of significance, we do not reject the null hypothesis that there is no observed difference in the two error rates.

Based on this result, we conclude that LaDA is also a valid tool for variable selection, and can be used for variable selection when our intention is to use either FDA or RRDA for classification. Also by reason of the result, we are of the view that we can extend the use of LaDA as a tool for variable selection, if we intend to use any other classifier other than FDA or RRDA for classification.
Chapter 6

Summary and Conclusions

We have shown in this study that central to the objectives of prediction is the minimization of prediction error. In Chapters 1 and 3, we demonstrated the usefulness of regularization in this regard. In particular, we provided illustrations on the gains of regularization techniques (ridge regression and Lasso) and the Regularized Fisher’s DA (RFDA). We saw that through the use of RFDA, for instance, we could handle cases involving \( p \gg n \). By optimally choosing the value of \( \lambda \), the error rates are brought to their barest minimum. Also in Chapter 3, we saw that regularization is inherent in the construction of SVM, essentially through its reliance on the support vectors. Further in Chapter 5, we explored the usefulness of RRDA and LaDA as tools for regularization in classification.

For these reasons, we therefore lay emphasis on the importance of regularization in accessing a classification function that performs optimally, especially in high dimensions.
On the comparison between FDA and SVM, it is our position that SVM performs relatively better than FDA. For instance, in a linearly separable classification problem, while it is true that SVM will always output a 0% error rate, it is possible for FDA to wrongly classify at least one data point. A case in point here is provided in Figure 3.4(b), where the presence of outliers gave rise to incorrect classification of two data points by FDA, whereas SVM returned a 0% error rate. It is true that in this case, outliers facilitated the poor performance of FDA, but it is still possible to find a situation where data points are not necessarily outliers, and at least one data point will be wrongly classified by FDA. When datasets overlap, either FDA or SVM can outperform the other, as shown in both Figures 3.3 and 3.4(c). We therefore state that FDA can only outperform SVM in instances of class overlap, whereas SVM can outperform in both linearly separable instances and class overlap situations. This position is consistent with the views of (Gökçen et al., 2002), because SVM performed badly on the linearly non separable datasets they investigated.

We also note that both classifiers can behave in an identical manner, when particularly we find the decision boundary of FDA superimposed on the separating hyperplane of SVM. Instances of this were illustrated in Figures 3.3(e) and 3.3(f).

On non-linearly separable classification problems, we showed that transforming datasets using either a non-linear map (2.54), or kernel function (2.55) enhances the performances of FDA and SVM (Zaki et al., 2014). For instance, transforming the datasets of Figure 3.5(a) with a quadratic non-linear map, and Gaussian kernel together resulted in 0% error rates, using both FDA and SVM. We further note that SVM is more adaptive to the use of kernel function than FDA, because a
kernel function returns an $n \times n$ kernel matrix. Hence, direct application of FDA is not possible, instead RFDA is used. On the other hand, we saw in Chapter 5 that the system time taken to run RFDA programs increases as dataset dimensions increase. If we have a very high dimensional kernel matrix, the system time may take more hours, or at least one day, but whichever is the case, this is a weakness on the part of RFDA.

One disadvantage of using a non linear map is that it can be very high or infinite dimensional, depending on the dimensions of the datasets in the input space. If this is the case, the use of a kernel function in transformation is preferred, hence SVM is used for classification. For these reasons, our conclusion is that concerning non linearly separable classification problems, and in high dimensions, the use of SVM is preferred to FDA. Expressed differently, SVM is more suitably adapted for non linear, and high dimensional classification problems, than FDA. In like manner, (Khondoker et al., 2016) are in agreement with our position because they observed that as the number of explanatory variables become larger ($\frac{p}{n} \geq 0.5$), SVM (with RBF kernel) outperform LDA.

In Chapter 4, we introduced regression discriminant analysis (RDA), and provided a proof to show that it is identical to FDA. We further introduced other classification functions, namely RRDA and LaDA after multiple regression variants (ridge regression and Lasso). Following the results of various investigations carried out in Chapter 5, we state that the error rates arising from the use of either RRDA or LaDA, are not statistically different from the error rates using FDA. We also showed in Chapter 5 that LaDA has a peculiar role in classification, by doubling as a variable selector. The illustration we provided using FDA or RRDA clearly
showed that when we apply a different classifier on the variables selected using LaDA, the error rate we get is as good as the error rate obtained by applying the same classifier on all input variables. Put it another way, the variables selected using LaDA have strong discriminatory power.

We also saw in Chapter 5 that if $p >> n$, the performances of RRDA are as good as LaDA. Based on the system time, the use of either RRDA or LaDA is recommended in place of FDA for high dimensional classification problem. We further state that if the objective of a given classification problem is to select a few important variables, whether in high or low dimensions, the use of LaDA as a classifier is highly recommended.

On comparison between the logistic regression and FDA, (Wikipedia, 2016a) is of the view that discriminant function analysis (e.g. FDA) is very similar to logistic regression and that both of them answer the same research question. It noted that the logistic regression does not have as many assumptions and restrictions as discriminant analysis. When the assumptions of FDA are met, it is more powerful than the logistic regression (Efron, 1975). Also since parameter estimation for logistic regression is based on the MLE, the discriminant analysis is more powerful with small sample sizes. Similarly, when sample sizes are equal, and assumption of homogeneity of variances hold, then the discriminant analysis is preferred (Cohen et al., 2003; Büyüköztürk et al., 2008). On the other hand, both authors are also of the view that there is preference for the logistic regression since the assumptions of discriminant analysis rarely hold.
6.1 Future Work

(1) Lasso is known to have a number of shortcomings which led to the introduction of the Elastic net (Zou et al., 2005). For instance, when $p > n$, the authors noted that Lasso selects at most $n$ variables despite the fact that more variables are associated with the outcome. Also, when there is a set of highly correlated predictor variables, Lasso selects one predictor variable from the set. They further added that even when $n > p$, and the predictor variables are strongly correlated, ridge regression tends to do better comparatively.

The Elastic net takes care of the shortcomings by extending Lasso through adding an $L_2$ penalty term. Thus we use

$$
\min_{\beta} \| y - X\beta \|^2 + \lambda_1 \| \beta \|_1 + \lambda_2 \| \beta \|_2^2,
$$

which is similar to writing

$$
\min_{\beta} \| y - X\beta \|^2 \text{ subject to } (1 - \alpha) \| \beta \|_1 + \alpha \| \beta \|_2^2 \leq s,
$$

where $\alpha = \frac{\lambda_2}{\lambda_1 + \lambda_2}$.

Since we are concerned with a response variable that is discrete, we would like to know if the observations of (Zou et al., 2005) are applicable to classification problems. We shall therefore carry out further investigation (in future) with the aim of addressing the following:

- If $p > n$, can we confirm or otherwise that LaDA will select only $n$ predictor variables, despite the fact that more variables could be associated
with the response variables?

- Can we confirm or otherwise that when there is a set of highly correlated predictor variables, LaDA selects only one predictor variable from the set?

- We would also like to know how LaDA compares with RRDA when \( n > p \), and predictor variables are strongly correlated.

Regardless of the outcome of our investigation, we shall further explore the possibility of using the Elastic net as a linear classifier.

(2) In our concluding remarks of Section 3.3, we pointed out that the performances of EFDA on simulated datasets show that the classifier can consistently perform better than SVM, on any real world dataset that meets the underlying assumptions. We shall therefore carry out further investigation to find out the kind of real world datasets that are relevant here. We may try various data preprocessing techniques, with the aim of determining if the underlying assumptions for the use of EFDA can be met.

(3) We would equally like to develop R packages to automate the use of RDA, RRDA, RFDA and LaDA.

(4) Lastly we note that in this study, our discussions were restricted to binary classification problems. In future, we intend to explore the possibility of extending applications (of RDA, RRDA and LaDA) to more than two classes.
Appendix A

Program in Connection with Evaluation of Classifiers

A.1 Class Predictions in Logistic Regression (Section 2.4)

```r
##Simulate Data
set.seed(12)
n = 10000
q = 0.78

#Simulate true classes
Class = c(sample(c(0,1), n/2, replace = TRUE, prob = c(1-q,q)),
           sample(c(0,1), n/2, replace = TRUE, prob = c(0.88,0.12)))
```

155
APPENDIX A. PROGRAM IN CONNECTION WITH EVALUATION OF CLASSIFIERS

10 #Simulate predictions
11 pre = c(rep(seq(0.45, 0.95, length=100), 50),
12 rep(seq(0.24, 0.64, length=100), 50))
13 pre = data.frame(cbind(Class, pre))

15 library(pROC)
16 ruc_mod = roc(response=pre$Class, predictor=pre$pre)
17 dat = cbind(ruc_mod$thresholds, ruc_mod$sensitivities+ruc_mod$specificities)
18 cut_off = subset(dat, dat[,2]==max(dat[,2]))[,1]
19 #Plot ROC Curve
20 plot(1-ruc_mod$specificities, ruc_mod$sensitivities, type="l",
21 ylab="Sensitivity", xlab="1-Specificity", col="black", lwd=2,
22 main = "ROC Curve for Simulated Data")
23 abline(a=0, b=1)
24 abline(v = cut_off)
Appendix B

Non-linear Classification Program

B.1 Non-Linear Transformation using Radial Basis Function or Gaussian Kernel (Section 3.2.4.3)

```r
load(file.choose()) ## The file to load is NonLinearSemiCircularToyDat.Saved
ndat = rbind(pop1, pop2) ## pop1 and pop2 make up the training set for the loaded dataset
library(kernlab)
K.mat = kernelMatrix(rbf(), as.matrix(ndat), y = NULL)  ## Here we used the kernlab package to generate radial basis function or Gaussian kernel
dim(K.mat)
pop1 = K.mat[1:30, ]  ## kernel matrix for pop1
pop2 = K.mat[31:60, ]  ## kernel matrix for pop2
```
APPENDIX B. NON-LINEAR CLASSIFICATION PROGRAM

\[ X.1 = \text{colMeans}(\text{pop1}) \]
\[ X.2 = \text{colMeans}(\text{pop2}) \]
\[ n1 = \text{nrow}(\text{pop1}) \]
\[ n2 = \text{nrow}(\text{pop2}) \]

## RFDA
\[ S = \frac{(n1-1) \ast \text{var}(\text{pop1}) + (n2-1) \ast \text{var}(\text{pop2})}{n1+n2-2} \]
\[ S = S + 0.0001 \ast (\text{diag}(\text{nrow}(S))) \]
\[ S.p = \text{solve}(S) \]
\[ b.hat = (X.1 - X.2) \ast S.p \]
\[ m = \frac{1}{2} \ast (b.hat \ast \text{matrix}((X.1 + X.2))) \]

# pop1.correct
\[ \text{class1.C} = 0 \]
\[ \text{for} (i \text{ in } 1 : \text{nrow}(\text{pop1})) \{ \]
\[ \quad \text{disA} = b.hat \ast \text{pop1}[i, ] \]
\[ \quad \text{if}(\text{disA} > m) \{ \]
\[ \quad \quad \text{class1.C} = \text{class1.C} + 1 \]
\[ \quad \} \]
\[ \} \]

# pop2.correct
\[ \text{class2.C} = 0 \]
\[ \text{for} (j \text{ in } 1 : \text{nrow}(\text{pop2})) \{ \]
\[ \quad \text{disB} = b.hat \ast \text{matrix}(\text{as.numeric}(\text{pop2}[j, ])) \]
\[ \quad \text{if}(\text{disB} < m) \{ \]
\[ \quad \quad \text{class2.C} = \text{class2.C} + 1 \]
\[ \quad \} \]
\[ \} \]
**APPENDIX B. NON-LINEAR CLASSIFICATION PROGRAM**

```r
FDA.error = 1 - ((class1.C + class2.C)/(n1+n2))

## SVM
library(e1071)
dat = cbind(y, K.mat)  # Create a data matrix including the response output y
svmfit = svm(as.factor(y) ~ ., data = dat, kernel = "linear",
             cost=100, scale=T)
ypred = predict(svmfit, dat[,-1])
svm.tab = table(dat[,1], ypred)
SVM.error = 1 - sum(diag(svm.tab))/sum(svm.tab)
```

### B.2 Non-Linear Transformation using a Quadratic Non-Linear Map $\phi(x)$ (Section 3.2.4.3)

```r
X = rbind(pop1, pop2)  # Training set for the NonLinearSemiCircularToyDat dataset we loaded
attach(X)
Var1 = sqrt(2)* X1
Var2 = sqrt(2)* X2
Var3 = X1^2
Var4 = X2^2
Var5 = sqrt(2)* X1*X2
dat = cbind(Var1, Var2, Var3, Var4, Var5)
ndat = data.frame(dat, y = as.factor(y))
library(e1071)
```
APPENDIX B. NON-LINEAR CLASSIFICATION PROGRAM

svmfit = svm(y ~., data = ndat, kernel = "linear", cost=100,
    scale=F) ## Observe that we use a linear kernel function
    because datasets have been transformed

ypred = predict(svmfit, ndat[,-6])

svm.tab = table(ndat[,6], ypred)
SVM.error = 1 - sum(diag(svm.tab))/sum(svm.tab)

library(MASS)

FDA.mod = lda(y ~., data = ndat)

lda.pre = predict(FDA.mod, ndat[, -6])

fda.tab = table(ndat[,6], lda.pre$class)
FDA.error = 1 - sum(diag(fda.tab))/sum(fda.tab)
Appendix C

R Codes in Connection with Breaking the SVM (Section 3.3)

C.1 Isotropic FDA

c = 1.5/sqrt(20)
u1 = matrix(rep(c(1, -1, 0), c(5, 5, 10)))
u1 = c*u1
u2 = matrix(rep(c(-1, 1, 0), c(5, 5, 10)))
u2 = c*u2
p = 20
rho = 0.80
Sig = (1-rho)*diag(p) + rho*matrix(1,p,p)

library(MASS)
```r
load(file.choose()) ## Three files will be loaded differently; they include SimuCh3_30n.Saved, SimuCh3_100n.Saved and SimuCh3_1000n.Saved

X.1 = colMeans(pop1)
X.2 = colMeans(pop2)
n1 = nrow(pop1)
n2 = nrow(pop2)
S = ((n1-1)*var(pop1) + (n2-1)*var(pop2))/(n1+n2-2)
aph = (sum(diag(S)))/p
SI = aph*diag(p) ## Isotropic covariance matrix
S.p = solve(SI)
b.hat = t(X.1 - X.2)%*%S.p
m = 1/2 * (b.hat %*% matrix((X.1 + X.2)))
m

class.IFDA = numeric(0)
for (q in 1 : 100){
  pop1.test = mvrnorm(500, mu = u1, Sigma = Sig)
  pop2.test = mvrnorm(500, mu = u2, Sigma = Sig)
  class1 = 0
  for (i in 1 : nrow(pop1.test)){
    disA = b.hat %*% pop1.test[i, ]
    if(disA > m) {
      class1 = class1 + 1
    }
  }
}
```
APPENDIX C. R CODES IN RESPECT OF BREAKING THE SVM

# pop2.correct
class2 = 0
for (j in 1 : nrow(pop2.test)){
  disB = b.hat %*% matrix(as.numeric(pop2.test[j, ]))
  if(disB < m) {
    class2 = class2 + 1
  }
}
class.IFDA[q] = 1 - (class1 + class2)/1000
}
Ave.IFDA = (sum(class.IFDA))/100

class.IFDA = round(class.IFDA, digits = 4)
Ave.IFDA = round(Ave.IFDA, digits=4)
class.IFDA
Ave.IFDA

C.2 Equicovariance FDA

c = 1.5/sqrt(20)
u1 = matrix(rep(c(1, -1, 0), c(5, 5, 10))
u1 = c*u1
u2 = matrix(rep(c(-1, 1, 0), c(5, 5, 10))
u2 = c*u2
p = 20
rho = 0.80
Sig = (1-rho)*diag(p) + rho*matrix(1,p,p)
library(MASS)
load(file.choose())  ## Files are similarly loaded as in Isotropic FDA
ls()
X.1 = colMeans(pop1)
X.2 = colMeans(pop2)
n1 = nrow(pop1)
n2 = nrow(pop2)
S = ((n1-1)*var(pop1) + (n2-1)*var(pop2))/(n1+n2-2)

## From here, we generate the equicovariance matrix
up.tri = S[upper.tri(S)]
lo.tri = S[lower.tri(S)]
ave = sum(lo.tri)/length(lo.tri)
dd = sum(diag(S))/p
SE = matrix(diag(S))/p
SE[upper.tri(SE)] = rep(ave, length(up.tri))
SE[lower.tri(SE)] = rep(ave, length(lo.tri))
diag(SE) = dd  ## SE is now equicovariance matrix
S.p = solve(SE)
b.hat = t(X.1 - X.2) %*% S.p
m = 1/2 * (b.hat %*% matrix(X.1 + X.2))

class.EFDA = numeric(0)
for (q in 1 : 100){
    pop1.test = mvrnorm(500, mu = u1, Sigma = Sig)
    pop2.test = mvrnorm(500, mu = u2, Sigma = Sig)
    class1 = 0
    for (i in 1 : nrow(pop1.test)){
        disA = b.hat %*% pop1.test[i, ]
        if(disA > m) {
            class1 = class1 + 1
        }
    }
    class2 = 0
    for (j in 1 : nrow(pop2.test)){
        disB = b.hat %*% matrix(as.numeric(pop2.test[j, ]))
        if(disB < m) {
            class2 = class2 + 1
        }
    }
    class.EFDA[q] = 1 - (class1 + class2)/1000
}
Ave.EFDA = (sum(class.EFDA))/100
class.EFDA = round(class.EFDA, digits = 4)
Ave.EFDA = round(Ave.EFDA, digits=4)
class.EFDA
Ave.EFDA
C.3 Regularized FDA

```r
C = 1.5/sqrt(20)
ul = matrix(rep(c(1, -1, 0), c(5, 5, 10))
ul = C * ul
u2 = matrix(rep(c(-1, 1, 0), c(5, 5, 10)))
ul = C * u2
p = 20
rho = 0.80
Sig = (1-rho)*diag(p) + rho*matrix(1,p,p)

library(MASS)
load(file.choose()) ## We load files as in the case of
      # Equicovariance FDA
X.1 = colMeans(pop1)
X.2 = colMeans(pop2)
n1 = nrow(pop1)
n2 = nrow(pop2)
n1; n2
S = ((n1-1)*var(pop1) + (n2-1)*var(pop2))/(n1+n2-2)
S = (S + diag(p) * 0.0001) ## S is now a regularized covariance matrix
S.p = solve(S)
b.hat = t(X.1 - X.2)%*%S.p
m = 1/2 * (b.hat %*% matrix((X.1 + X.2)))
m
```


APPENDIX C. R CODES IN RESPECT OF BREAKING THE SVM

```r
class.RFDA = numeric(0)
for (q in 1 : 100){
  pop1.test = mvrnorm(500, mu = u1, Sigma = Sig)
pop2.test = mvrnorm(500, mu = u2, Sigma = Sig)
class1 = 0
for (i in 1 : nrow(pop1.test)){
disA = b.hat %*% pop1.test[i, ]
if(disA > m) {
class1 = class1 + 1
}
}
}

# pop2.correct
class2 = 0
for (j in 1 : nrow(pop2.test)){
disB = b.hat %*% pop2.test[j, ]
if(disB < m) {
class2 = class2 + 1
}
}
class.RFDA[q] = 1 - (class1 + class2)/1000
}
Ave.RFDA = (sum(class.RFDA))/100
class.RFDA = round(class.RFDA, digits = 4)
Ave.RFDA = round(Ave.RFDA, digits=4)
class.RFDA
```
C.4 FDA

```r
# C = 1.5/sqrt(20)
u1 = matrix(rep(c(1, -1, 0), c(5, 5, 10)))
u1 = c*u1
u2 = matrix(rep(c(-1, 1, 0), c(5, 5, 10)))
u2 = c*u2
p = 20
rho = 0.80
Sig = (1-rho) * diag(p) + rho * matrix(1, p, p)

library(MASS)
load(file.choose())
X.1 = colMeans(pop1)
X.2 = colMeans(pop2)
n1 = nrow(pop1)
n2 = nrow(pop2)
n1;n2
S = ((n1-1)*var(pop1) + (n2-1)*var(pop2))/(n1+n2-2)
S.p = solve(S)
b.hat = t(X.1 - X.2) %*% S.p
m = 1/2 * (b.hat %*% matrix((X.1 + X.2)))
m
```
class.FDA = numeric(0)
for (q in 1:100) {
  pop1.test = mvrnorm(500, mu = u1, Sigma = Sig)
  pop2.test = mvrnorm(500, mu = u2, Sigma = Sig)
  class1 = 0
  for (i in 1:nrow(pop1.test)) {
    disA = b.hat %*% pop1.test[i, ]
    if (disA > m) {
      class1 = class1 + 1
    }
  }
  
  # pop2.correct
  class2 = 0
  for (j in 1:nrow(pop2.test)) {
    disB = b.hat %*% pop2.test[j, ]
    if (disB < m) {
      class2 = class2 + 1
    }
  }
  class.FDA[q] = 1 - (class1 + class2)/1000
}
Ave.FDA = (sum(class.FDA))/100
class.FDA = round(class.FDA, digits = 4)
Ave.FDA = round(Ave.FDA, digits=4)
class.FDA
Ave.FDA
APPENDIX C. R CODES IN RESPECT OF BREAKING THE SVM

## FDA Using a Package

c = 1.5/sqrt(20)
u1 = matrix(rep(c(1, -1, 0), c(5, 5, 10)))
u1 = c*u1
u2 = matrix(rep(c(-1, 1, 0), c(5, 5, 10)))
u2 = c*u2
p = 20
rho = 0.80
Sig = (1-rho)*diag(p) + rho*matrix(1,p,p)
library(MASS)
load(file.choose())
y = as.factor(y)
Xdat = rbind(pop1,pop2)
dat = data.frame(cbind(y, Xdat))
FDA.mod = lda(y ~., data = dat)

FDA.error = numeric(0)
for (q in 1 : 100){
  pop1.test = mvrnorm(500, mu = u1, Sigma = Sig)
pop2.test = mvrnorm(500, mu = u2, Sigma = Sig)
y.test = rep(c(1, -1), each = nrow(pop1.test))
y.test = as.factor(y.test)
dat.test = rbind(pop1.test, pop2.test)
Xdat.test = data.frame(cbind(y.test, dat.test))
APPENDIX C. R CODES IN RESPECT OF BREAKING THE SVM

 lda.pre = predict(FDA.mod, Xdat.test[, -1])
 fda.tab = table(Xdat.test[,1], lda.pre$class)
 FDA.error[q] = 1 - sum(diag(fda.tab))/sum(fda.tab)
 }
 Ave.FDA.error = (sum(FDA.error))/100
 FDA.error
 Ave.FDA.error

C.5 SVM

c = 1.5/sqrt(20)
 u1 = matrix(rep(c(1, -1, 0), c(5, 5, 10)))
 u1 = c*u1
 u2 = matrix(rep(c(-1, 1, 0), c(5, 5, 10)))
 u2 = c*u2
 p = 20
 rho = 0.80
 Sig = (1-rho)*diag(p) + rho*matrix(1,p,p)

library(MASS)
library(e1071)
load(file.choose())
X = rbind(pop1, pop2)
dat = data.frame(X, y = as.factor(y))
svmfit = svm(y ~ ., data = dat, kernel = "linear", cost=100, scale=F)
print(svmfit)

SVM.error = numeric(0)

for (q in 1 : 100){
  pop1.test = mvrnorm(500, mu = u1, Sigma = Sig)
  pop2.test = mvrnorm(500, mu = u2, Sigma = Sig)
  X.test = rbind(pop1.test, pop2.test)
  y.test = rep(c(1,-1), each = 500)
  ypred = predict(svmfit, X.test)
  svm.tab = table(y.test, ypred)
  SVM.error[q] = 1 - sum(diag(svm.tab))/sum(svm.tab)
}

SVM.error = round(SVM.error, digits = 4)

Ave.SVM = sum(SVM.error)/100

SVM.error

Ave.SVM
Appendix D

R Codes used in Chapters 4 and 5

D.1 Regression Discriminant Analysis (RDA) ROC Curve

```r
load(file.choose()) ## Australia and Ionosphere datasets are loaded in turn
pop1 = pop1[, -1]
pop2 = pop2[, -1]
n1 = nrow(pop1)
n2 = nrow(pop2)
dat = rbind(pop1, pop2)
I = rep(1, (n1+n2))
X = as.matrix(cbind(I, dat))
y = rep(c(1, -1), c(n1, n2))
B = solve(t(X) %*% X) %*% t(X) %*% y
```
```r
pop1.test = pop1.test[, -1]
pop2.test = pop2.test[, -1]
m1 = nrow(pop1.test)
m2 = nrow(pop2.test)
dat.test = rbind(pop1.test, pop2.test)
I.test = rep(1, (m1 + m2))
X.test = as.matrix(cbind(I.test, dat.test))
mid = colMeans(X.test)
test.pop1 = X.test[1:m1, ]
test.pop2 = X.test[(m1 + 1):(m1 + m2), ]

# classification check
CorrectClass.pop1 = 0
yestA = numeric(0)
for (s in 1: m1)
  yestA[s] = t(B) %*% matrix(test.pop1[s,] - mid)
  if (yestA[s] > 0)
    CorrectClass.pop1 = CorrectClass.pop1 + 1
}
CorrectClass.pop2 = 0
yestB = numeric(0)
for (k in 1: m2)
  yestB[k] = t(B) %*% matrix(test.pop2[k,] - mid)
  if (yestB[k] < 0)
    CorrectClass.pop2 = CorrectClass.pop2 + 1
```
APPENDIX D. R CODES USED IN CHAPTERS 4 AND 5

RDA.error = 1 - ((CorrectClass.pop1+CorrectClass.pop2)/(m1+m2))
RDA.error

## From here we aim to obtain the ROC Curve
## We also present alternative method of finding RDA error rate

def.RDA = c(yestA, yestB)  ## Predictions for both classes
res.RDA = rep(c(1, -1), c(m1, m2))  ## Actual class responses
test = data.frame(pred = c(yestA, yestB), group = c(rep("pop1", m1), rep("pop2", m2)))
Tab = table(test$pred > 0, test$group)
RDA_Error = 1 - (sum(diag(Tab))/sum(Tab))
RDA_Error

library(ROCR)
library(pROC)
pred = prediction(pred.RDA, res.RDA)
perf = performance(pred,"tpr","fpr")
plot(perf, col="red", lwd=2)
auc(res.RDA, pred.RDA)
abline(0, 1, lty = 2, col = "black", lwd=1.80)
legend(locator(1), legend = "AUC = 0.8397", col = "black", box.
   lty = 0, cex=1.10)
D.2 Lasso Discriminant Analysis

```r
load(file.choose()) ## Different saved datasets can be loaded one at a time
t1 = Sys.time()
dat.train = rbind(pop1, pop2)
dat.test = rbind(pop1.test, pop2.test)
grid = 10^seq(10,-3,length=200) ## grid here refers to values for Lambda
library(glmnet)
X = as.matrix(dat.train[, -1])
y = as.vector(dat.train[, 1])
lasso.mod = glmnet(X, y, alpha = 1, lambda = grid, standardize = T) ## alpha = 1 means we are concerned with Lasso model
set.seed(32) ## Helps to maintain constant value for cv.out
cv.out = cv.glmnet(X, y, alpha = 1)
bestlam = cv.out$lambda.min ## The best lambda value
bestlam
lasso.coef = predict(lasso.mod, type = "coefficients", s = bestlam)[1:ncol(dat.train),] ## The model coefficients are predicted here
tt = which(lasso.coef != 0) ## Here we track the nonzero coefficients
tt = as.numeric(tt)
length(tt)
tt
b0 = lasso.coef[tt]
```
n1 = nrow(pop1)
n2 = nrow(pop2)
## tt includes the intercept and in order to retain it, the
class entries at this stage are replaced with a vector of
ones as in lines 24 and 25.
pop1$class = matrix(1, n1, 1)
pop2$class = matrix(1, n2, 1)
pop1 = pop1[, tt] # selection of columns with non zero
coefficients, including the intercept
pop2 = pop2[, tt]
X1_mean = colMeans(pop1)
X2_mean = colMeans(pop2)
X.av = as.matrix((X1_mean + X2_mean)/2) ## See Eq (4.5)
B = as.matrix(as.numeric(b0))
m1 = nrow(pop1.test)
m2 = nrow(pop2.test)
test.new = rbind(pop1.test, pop2.test)
test.new$class = matrix(1, (m1 + m2), 1) ## a vector of ones
replaces the class entries in the test set
test.new = test.new[,tt] ## selection of columns with non zero
coefficients for test set
test_A = test.new[1:m1, ]
test_B = test.new[(m1 + 1):(m1 + m2), ]
CorrectClass.pop1 = 0
yestA = numeric(0)
for (m in 1 : nrow(test_A)){
    yestA[m] = t(B) %*% (matrix(as.numeric(test_A[m,])) - X.av)
    if (yestA[m] > 0){
        CorrectClass.pop1 = CorrectClass.pop1 + 1
    }
}
CorrectClass.pop1

CorrectClass.pop2 = 0
yestB = numeric(0)
for (k in 1:nrow(test_B)){
    yestB[k] = t(B) %*% (matrix(as.numeric(test_B[k,]))-X.av)
    if (yestB[k] < 0){
        CorrectClass.pop2 = CorrectClass.pop2 + 1
    }
}
CorrectClass.pop2
LaDAerror = 1 - (((CorrectClass.pop1+CorrectClass.pop2)/(m1+m2))
LaDAerror

t2 = Sys.time()
difftime(t2,t1)

D.3 Ridge Regression Discriminant Analysis

load(file.choose()) ## Datasets are similarly loaded as in LaDA
t1 = Sys.time()
dat.train = rbind(pop1, pop2)
dat.test = rbind(pop1.test, pop2.test)
grid=10^seq(10,-3,length=200)
library(glmnet)
X = as.matrix(dat.train[, -1])
y = as.vector(dat.train[, 1])
ridge.mod = glmnet(X, y, alpha = 0, lambda = grid, standardize = T) ## alpha = 0 gives a model for ridge regression, in contrast to alpha = 1 representing Lasso model
set.seed(32)
cv.out = cv.glmnet(X, y, alpha = 0)
bestlam = cv.out$lambda.min
bestlam
ridge.coef = predict(ridge.mod, type = "coefficients", s = bestlam)[1:ncol(dat.train),]
b0 = as.numeric(ridge.coef)
b0
n1 = nrow(pop1)
n2 = nrow(pop2)
pop1$class = matrix(1, n1, 1) # We make provision for the intercept because b0 includes it
pop2$class = matrix(1, n2, 1)
X1_mean = colMeans(pop1)
X2_mean = colMeans(pop2)
X.av = as.matrix((X1_mean + X2_mean)/2)
B = as.matrix(b0)
```r
m1 = nrow(pop1.test)
m2 = nrow(pop2.test)
test.new = rbind(pop1.test, pop2.test)
test.new$class = matrix(1, (m1 + m2), 1)
test_A = test.new[1:m1, ]
test_B = test.new[(m1 + 1):(m1 + m2), ]
CpopA = 0
yestA = numeric(0)
for (m in 1:nrow(test_A)){
yestA[m] = t(B) %*% (matrix(as.numeric(test_A[m,])) - X.av)
if (yestA[m] > 0){
  CpopA = CpopA + 1
}
}

CpopA
CpopB = 0
yestB = numeric(0)
for (k in 1:nrow(test_B)){
yestB[k] = t(B) %*% (matrix(as.numeric(test_B[k,]))-X.av)
if (yestB[k] < 0){
  CpopB = CpopB + 1
}
}

CpopB
RRDA.error = 1-((CpopA+CpopB)/(m1 + m2))
RRDA.error
```
D.4 Regularized FDA

The difference between the RFDA here, and the one in C.3 is that in this Section, we demonstrate how to use the glmnet package to obtain the best value of $\lambda$, as we saw in LaDA and RRDA. This discovery is important to us because until now, the grid search procedure we adopted involved fitting a model with a given value of $\lambda$ in the grid. Thereafter, we compute the error rate, and repeat the same process with another value of $\lambda$, until we exhaust all of them in the grid. At the end, the model with the smallest error rate is used, and the corresponding $\lambda$ is considered optimal based on the dataset used. Such method is time intensive, in the sense that a lot of time is spent in the process.

```r
load(file.choose())
t1 = Sys.time()
dat.train = rbind(pop1, pop2)
dat.test = rbind(pop1.test, pop2.test)
## We use the glmnet package to find the best Lambda as in LaDA and RRDA, thereafter, the rest of the procedure applies
grid=10^seq(10,-3,length=200)
library(glmnet)
X = as.matrix(dat.train[,1:-1])
y = as.vector(dat.train[,1])
set.seed(32)
```
cv.out = cv.glmnet(X, y, alpha = 0)
bestlam = cv.out$lambda.min
bestlam
pop1 = pop1[, -1]
pop2 = pop2[, -1]
pop1.test = pop1.test[, -1]
pop2.test = pop2.test[, -1]
n1 = nrow(pop1)
n2 = nrow(pop2)

X.1 = colMeans(pop1)
X.2 = colMeans(pop2)
S = ((n1-1)*var(pop1) + (n2-1)*var(pop2))/(n1+n2-2)
S.p = solve(S + bestlam*diag(ncol(pop1))) ## bestlam takes the value for Lambda here
b.hat = t(X.1 - X.2)%*%S.p
m = 1/2 * (b.hat %*% matrix((X.1 + X.2)))
# pop1.correct
class1.C = 0
disA = numeric(0)
for (i in 1 : nrow(pop1.test)){
disA[i] = b.hat %*% matrix(as.numeric(pop1.test[i, ]))
if(disA[i] > m) {
class1.C = class1.C + 1
}
}
D.5  Repeated Measures ANOVA

dat = read.csv(file.choose(), sep="","", header=T)
dat[1:4,]
attach(dat)
qqnorm(FDA, main = "Normal Q-Q Plot for FDA", lwd=2)
qqline(FDA, lwd=2)
qqnorm(LaDA, main = "Normal Q-Q Plot for LaDA", lwd=2)
```r
qqline(LaDA, lwd=2)
qqnorm(RRDA, main = "Normal Q-Q Plot for RRDA", lwd=2)
qqline(RRDA, lwd=2)

apply(dat[,2:4],2,shapiro.test)
data = stack(dat)
bartlett.test(values~ind, data)
data$Dataset = rep(rownames(dat), 3)
data$Dataset = factor(data$Dataset)

colnames(data) = c("Values", "Classifiers", "Datasets")
aov.out = aov(Values ~ Classifiers + Error(Datasets/Classifiers), data=data)
summary(aov.out)
```
Bibliography


BIBLIOGRAPHY


Hsu, Chih-Wei et al. (2003). “A Practical Guide to Support Vector Classification”.


Kim, Eric (2013). *Everything You Wanted to Know about the Kernel Trick*. URL: 
http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html.

Kurita, Takio et al. (2002). “A Modification of Kernel-Based Fisher Discriminant 

Wiley & Sons.


Li, Yongmin et al. (2003). “Recognising Trajectories of Facial Identities using 
1086.

ics.uci.edu/ml.

Liu, J Jay et al. (2005). “Automatic Masking in Multivariate Image Analysis using 
Support Vector Machines”. *Chemometrics and Intelligent Laboratory Systems 
79.1*, pp. 42–54.

Liu, Qingshan et al. (2004). “Improving Kernel Fisher Discriminant Analysis for 
Face Recognition”. *IEEE Transactions on Circuits and Systems for Video Tech-

Intrinsic Structures in Data*. [Online; accessed 4-May-2016]. URL: http:// 


BIBLIOGRAPHY


BIBLIOGRAPHY


