

Nearest Hypersphere Classification: A Comparison with Other Classification Techniques

by

Stephan van der Westhuizen



*Thesis presented in partial fulfilment of the requirements for the degree of
Master of Commerce in the Faculty of Economic and Management Sciences at
Stellenbosch University*

Supervisor: Dr. M.M.C. Lamont

December 2014

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own work, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature:

Date:

CS van der Westhuizen

ABSTRACT

Classification is a widely used statistical procedure to classify objects into two or more classes according to some rule which is based on the input variables. Examples of such techniques are Linear and Quadratic Discriminant Analysis (LDA and QDA). However, classification of objects with these methods can get complicated when the number of input variables in the data become too large ($n \ll p$), when the assumption of normality is no longer met or when classes are not linearly separable. Vapnik *et al.* (1995) introduced the Support Vector Machine (SVM), a kernel-based technique, which can perform classification in cases where LDA and QDA are not valid. SVM makes use of an optimal separating hyperplane and a kernel function to derive a rule which can be used for classifying objects. Another kernel-based technique was proposed by Tax and Duin (1999) where a hypersphere is used for domain description of a single class. The idea of a hypersphere for a single class can be easily extended to classification when dealing with multiple classes by just classifying objects to the nearest hypersphere.

Although the theory of hyperspheres is well developed, not much research has gone into using hyperspheres for classification and the performance thereof compared to other classification techniques. In this thesis we will give an overview of Nearest Hypersphere Classification (NHC) as well as provide further insight regarding the performance of NHC compared to other classification techniques (LDA, QDA and SVM) under different simulation configurations.

We begin with a literature study, where the theory of the classification techniques LDA, QDA, SVM and NHC will be dealt with. In the discussion of each technique, applications in the statistical software R will also be provided. An extensive simulation study is carried out to compare the performance of LDA, QDA, SVM and NHC for the two-class case. Various data scenarios will be considered in the simulation study. This will give further insight in terms of which classification technique performs better under the different data scenarios. Finally, the thesis ends with the comparison of these techniques on real-world data.

OPSOMMING

Klassifikasie is 'n statistiese metode wat gebruik word om objekte in twee of meer klasse te klassifiseer gebaseer op 'n reël wat gebou is op die onafhanklike veranderlikes. Voorbeelde van hierdie metodes sluit in Lineêre en Kwadratiese Diskriminant Analise (LDA en KDA). Wanneer die aantal onafhanklike veranderlikes in 'n datastel te veel raak, die aanname van normaliteit nie meer geld nie of die klasse nie meer lineêr skeibaar is nie, raak die toepassing van metodes soos LDA en KDA egter te moeilik. Vapnik *et al.* (1995) het 'n kern gebaseerde metode bekendgestel, die Steun Vektor Masjien (SVM), wat wel vir klassifisering gebruik kan word in situasies waar metodes soos LDA en KDA misluk. SVM maak gebruik van 'n optimale skeibare hipervlak en 'n kern funksie om 'n reël af te lei wat gebruik kan word om objekte te klassifiseer. 'n Ander kern gebaseerde tegniek is voorgestel deur Tax and Duin (1999) waar 'n hipersfeer gebruik kan word om 'n *gebied beskrywing* op te stel vir 'n datastel met net een klas. Dié idee van 'n enkele klas wat beskryf kan word deur 'n hipersfeer, kan maklik uitgebrei word na 'n multi-klas klassifikasie probleem. Dit kan gedoen word deur slegs die objekte te klassifiseer na die naaste hipersfeer.

Alhoewel die teorie van hipersfere goed ontwikkel is, is daar egter nog nie baie navorsing gedoen rondom die gebruik van hipersfere vir klassifikasie nie. Daar is ook nog nie baie gekyk na die prestasie van hipersfere in vergelyking met ander klassifikasie tegnieke nie. In hierdie tesis gaan ons 'n oorsig gee van Naaste Hipersfeer Klassifikasie (NHK) asook verdere insig in terme van die prestasie van NHK in vergelyking met ander klassifikasie tegnieke (LDA, KDA en SVM) onder sekere simulasië konfigurasies.

Ons gaan begin met 'n literatuurstudie, waar die teorie van die klassifikasie tegnieke LDA, KDA, SVM en NHK behandel gaan word. Vir elke tegniek gaan toepassings in die statistiese sagteware R ook gewys word. 'n Omvattende simulasië studie word uitgevoer om die prestasie van die tegnieke LDA, KDA, SVM en NHK te vergelyk. Die vergelyking word gedoen vir situasies waar die data slegs twee klasse het. 'n Verskeidenheid van data situasies gaan ook ondersoek word om verdere insig te toon in terme van wanneer watter tegniek die beste vaar. Die tesis gaan afsluit deur die genoemde tegnieke toe te pas op praktiese datastelle.

Aan my hemelse Vader.

Dankie dat U my gered het.

Mag die tesis U naam verheerlik.

ERKENNINGS

Ek wil graag die volgende persone bedank:

- Dankie aan my ouers, Stephan en Max, wie in my glo en wie vir my 'n wêreld van geleenthede geskep het. Ek is ongelooflik lief vir julle.
- My studieleier, Dr. Morne Lamont, baie dankie vir jou leiding en hulp met die skryf van hierdie tesis. Baie dankie vir jou geduld en dat jou deur altyd oop was vir my. Ek waardeer dit meer as wat jy dink.
- Alma en Hildegard, my laaste jare by Stellenbosch Universiteit sou nie moontlik gewees het sonder julle twee nie. Baie dankie dat julle altyd daar was vir my.
- My opregte dank word ook aan die *National Research Foundation* uitgespreek, wie sonder die tesis ook nie moontlik sou wees nie.

CONTENTS

CHAPTER 1

INTRODUCTION	1
1.1 Problem Statement	1
1.2 Scope of the Study	2
1.3 Contribution of the Study	2
1.4 Chapter Outline	3

CHAPTER 2

LINEAR AND QUADRATIC DISCRIMINANT ANALYSIS	5
2.1 Introduction	5
2.2 An Optimal Classification Model	6
2.3 Linear Discriminant Analysis	7
2.3.1 The Two-Class Case	7
2.3.2 The Multi-Class Case	10
2.4.1 The Two-Class Case	11
2.4.2 The Multi-Class Case	12
2.5.1 Software	13
2.5.2 Data	13
2.5.3 Performing LDA and QDA in R	16
2.6 Conclusion	22

CHAPTER 3

SUPPORT VECTOR MACHINES	23
3.1 Introduction	23
3.2 Support Vector Machines	23
3.2.1 The Linearly Separable Case	25
3.2.2 The Linearly Non-Separable Case	30
3.3 Nonlinear Support Vector Machines	32
3.3.1 Nonlinear Transformations and the Kernel Trick	32
3.3.2 Properties of the Kernel Function	33
3.3.3 Examples of Kernels	34
3.3.4 Classification in Feature Space	36

3.3.5 The Multi-Class Case.....	37
3.4 Performing SVM with R.....	37
3.4.1 The Kernlab Package.....	37
3.4.2 Application in R.....	38
3.5 Conclusion.....	42
CHAPTER 4	
CLASSIFICATION WITH HYPERSPHERES.....	43
4.1 Introduction.....	43
4.2 The Hypersphere.....	44
4.2.1 The Hard-Margin Solution.....	44
4.2.2 The Soft-Margin Solution.....	48
4.2.3 Applications of the Smallest Enclosing Hyperspheres in R.....	50
4.3 Nearest Hypersphere Classification.....	61
4.3.1 Theory of Nearest Hypersphere Classification.....	61
4.3.2 Application of NHC in R.....	64
4.3.3 Finding the Optimal Parameter through Cross-Validation.....	69
4.3.4 An Example of Performing NHC with Three Classes.....	70
4.4 Aspects of Nearest Hypersphere Classification.....	71
4.5 Conclusion.....	72
CHAPTER 5	
SIMULATION STUDIES AND REAL-WORLD DATA APPLICATIONS.....	73
5.1 Introduction.....	73
5.2 Simulation studies.....	74
5.2.1 Measuring Classification Performance.....	74
5.2.2 Generating the data.....	75
5.2.3 Simulation Configurations.....	78
5.2.4 Discussion of the simulation results.....	83
5.3 Real-world data applications.....	88
5.3.1 The Data.....	88
5.3.2 Classification results.....	91
5.4 Conclusions.....	92
CHAPTER 6	
CONCLUSION.....	93
6.1 Summary of Findings.....	93

6.2 Future Research Recommendations.....	94
REFERENCES	96
APPENDIX	
CODE IN R.....	99
A. Main Simulation Program.....	99
B. Generating Normal Data.....	100
C. Generating Lognormal Data.....	101
D. Linear Discriminant Analysis	102
E. Quadratic Discriminant Analysis.....	104
F. Support Vector Machine	106
G. Nearest Hypersphere Classification.....	108

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Classification is a widely used statistical technique that classifies objects into two or more classes based on a rule that has been derived from the input variables. Many such techniques already exist today in traditional Statistics, such as Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). However, these traditional techniques are based on the assumption that the distribution of the data must be normal and therefore cannot be used effectively in many data sets that are generated today. Other problems that are faced by traditional statistical classification techniques are the dimensionality dilemma, that is, when the number of input variables exceeds the number of objects (observations). The data may also not be linearly separable which will be a great difficulty for traditional statistical classifications techniques.

Vapnik *et al.* (1995) introduced the Support Vector Machine (SVM) that can be used in data scenarios when traditional classification techniques such as LDA and QDA fail. SVM make use of an optimal separating hyperplane and a kernel function to derive a rule that is used to classify objects. This classification rule is not dependent on the distribution of the data. Another kernel-based technique was proposed by Tax and Duin (1999) where a single hypersphere can be used to obtain a domain description of a data set. This scenario can easily be extended to the multi-class case by classifying objects via the usage of hyperspheres to the nearest enclosing hypersphere. This is called Nearest Hypersphere Classification (NHC).

Although the theory on hyperspheres is well developed not much research has gone into using hyperspheres for classification and its performance relative to other classification techniques. In this thesis we will address this problem. Firstly, we will introduce LDA and QDA which can be considered as the basis for statistical classification theory. We will then introduce SVM and the kernel trick where the latter is considered vital for any kernel-based classification technique.

We will conclude this thesis with a simulation study and with real-world data applications. The simulation study and the real-world data applications will be used to compare LDA,

QDA, SVM and NHC. The simulation study will incorporate different data configurations to test which classification techniques perform better under the different data configurations. The real-world data application will also assess the performance of the classification techniques.

1.2 Scope of the Study

The objectives of this study are:

1. To provide an introduction to the field of statistical classification by looking at LDA, QDA, SVM and NHC. This will be done by means of a literature study.
2. To carry out a simulation study to assess the performance of LDA, QDA, SVM and NHC under different data configurations. The simulation study will cover several different data scenarios.
3. A real-world data application will also be presented which will also assess the performances of the proposed techniques.

1.3 Contribution of the Study

The contribution of this study in the field of classification can be summarized as follows:

1. Assessing the performance of NHC relative to other classification techniques has not been thoroughly researched. The main purpose of this study is to contribute and aid in this shortcoming.
2. The application of hyperspheres in multi-class classification has not received significant attention in the literature. It is one of the purposes of this study to address this problem.
3. This study will impact fields where statistical classification is frequently used.

1.4 Chapter Outline

In Chapter 2 LDA and QDA will be discussed as outlined in Johnson and Wichern (2007). Section 2.2 will discuss the optimal classification model and what the necessary features of such a model should include. In Section 2.3 we discuss the theory on LDA for both the two-class case and the multi-class case. The theory on QDA for the two-class case and the multi-class case will be discussed in Section 2.4. The application of LDA and QDA in the statistical software R will be discussed in Section 2.5. This section will also introduce two data sets that will be used as examples throughout the thesis. Section 2.6 will look at the conclusions and will also introduce kernel-based machine learning methods that will be discussed in Chapter 3 and Chapter 4.

Chapter 3 will deal with SVM and we will look at both the linearly separable and linearly non-separable cases. SVM will be discussed as outlined in Izenman (2008). In Section 3.1 the SVM methodology will be introduced. Section 3.2 will look at the theory on linear SVM in the light of two cases, the linearly separable case and linearly non-separable case. Non-linear SVM will be discussed in Section 3.3 where we will also discuss the kernel trick, kernel function, properties of the kernel function, as well as examples of the kernel function. In Section 3.4 the application of SVM in R will be dealt with where the R function `ksvm()` will be used. Section 3.5 will look at the conclusions of the chapter.

In Chapter 4 we will introduce hyperspheres and NHC. Theory on hyperspheres and NHC is discussed as outlined in Tax and Duin (1999), Tax (2001), Shawe-Taylor and Christianini (2004), as well as in Lamont (2008). The theory on hyperspheres will be discussed in Section 4.2. In this section we look at two possible solutions that can be used to construct a hypersphere, i.e. the hard-margin solution and the soft-margin solution. The former is also known as the Smallest Enclosing Hypersphere (SEH). The theory and applications of the SEH in R will be dealt with in Section 4.2. In Section 4.2.2 we will briefly discuss the theory on the soft-margin solution. In Section 4.3 the theory on NHC will be discussed. An illustration of NHC in R will also be shown in this section. This section will also look at cross-validation as a means of estimating the optimal parameter of the NHC. Section 4.4 will discuss some of the aspects of NHC whereas Section 4.5 will look at some conclusions.

A simulation study will be used to assess the performance of the techniques described in the thesis and the results will be discussed in Chapter 5. Section 5.2 will look at various ways to

quantify classification performance. Generating the data for the simulation study is also discussed in Section 5.2. The results for the simulation study are discussed at the end of Section 5.2. In Section 5.3, a real-world data application will also be used to assess the performance of the classification techniques LDA, QDA, SVM and NHC.

The final chapter will discuss the conclusion of this study. Future research opportunities will also be identified.

CHAPTER 2

LINEAR AND QUADRATIC DISCRIMINANT ANALYSIS

2.1 Introduction

Discriminant analysis and classification are very important areas of research in Statistics. Sir Ronald Fisher, who is considered as one of the fathers of Statistics, used the idea of classification of objects in his paper (Fisher, 1936), which is probably the earliest application of such an analysis in Statistics. In this paper, Fisher derived a linear function based on four measurements to separate and classify three Iris species. It is from this study where the very popular Iris data set made its debut in the statistical community. Other classification techniques such as Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are also well-known classification techniques that were founded in the twentieth century (Johnson and Wichern, 2007). LDA and QDA are both built on the assumption that the data come from normal populations. Even though the solutions to Fisher's method and LDA are quite similar, Fisher did not make the assumption of normally distributed data.

The aim of this chapter is to review the theory on LDA and QDA for the two-class and the multi-class cases. These methods have become quite popular among Statisticians. In Section 2.2 the aspects of an optimal classification model will be discussed, this will be followed by a detailed discussion of LDA in Section 2.3 for the two-class and multi-class cases. A similar discussion will follow in Section 2.4 for QDA. The derivation of the classification rules for both methods will also be shown. Illustrations of these methods as well as their implementation in the R programming language will receive attention in Section 2.5. Throughout this chapter (as well as later chapters) we make use of the Iris data set for practical illustrations. As previously mentioned, this data set consists of three Iris species (see Figure 2.1). Another data set that will be used is the Haemophilia data set. See Section 2.5.2 for the descriptions of the data.



Figure 2.1: The three Iris species (Setosa, Versicolor and Virginica)

2.2 An Optimal Classification Model

A good classification model should result in few misclassifications. However, for a classification model to be considered optimal, it must have certain key characteristics, that will distinguish it from any other classification model. Two of these characteristics include defined probabilities of correctly classifying an object into a certain class and costs of misclassification. A classification model that ignores these key characteristics may result in serious problems (Johnson and Wichern, 2007).

In some situations it may be that one or more populations are larger in size than other populations in relative terms. There is a higher probability of classifying new objects that belong to a larger population to a particular class than to classify objects that belong to a smaller population to their respective class. An optimal classification rule should therefore take these probabilities of classification into account. Therefore, if an object belongs to a small population and it is to be correctly classified, the data must show overwhelming proof.

Let p_1 be the prior probability of classifying an object into the first class, Π_1 , and let p_2 be the prior probability of classifying an object into the second class, Π_2 . The conditional probability of classifying an object as Π_2 when, in fact, it belongs to Π_1 is $P(2|1)$. Similarly, the conditional probability of classifying an object as Π_1 when it belongs to Π_2 , is $P(1|2)$. Then the overall probabilities of correctly and incorrectly classifying objects can be derived as the product of the prior and conditional probabilities. That is,

$$\begin{aligned}
 P(\text{object correctly classified as } \Pi_1) &= P(1|1)p_1 \\
 P(\text{object misclassified as } \Pi_1) &= P(1|2)p_2 \\
 P(\text{object correctly classified as } \Pi_2) &= P(2|2)p_2 \\
 P(\text{object misclassified as } \Pi_2) &= P(2|1)p_1.
 \end{aligned} \tag{2.1}$$

Classification should also incorporate costs of misclassification. Suppose the probability of classifying an object into Π_1 when it belongs to Π_2 , is very small, then the cost of misclassification could be very extreme. For example, suppose a patient must be classified either as having a certain illness or not. If the illness is very uncommon the probability of the patient having the illness may be very low, but misclassifying the patient as healthy when in fact the patient does have the illness could lead to high cost of misclassification, that is,

possibly death. Below is the cost matrix which summarizes the different costs of misclassification. The cost of misclassifying an object belonging to Π_1 is $c(2|1)$ and the cost of misclassifying an object belonging to Π_2 is $c(1|2)$.

Table 2.1: The Cost matrix showing the costs of misclassification

Cost Matrix		Classify as:	
		Π_1	Π_2
True Population:	Π_1	0	$c(2 1)$
	Π_2	$c(1 2)$	0

The expected cost of misclassification (ECM) is a measurement that can be used to determine the average cost of misclassification. An optimal classification model should have an ECM as small as possible. The ECM is calculated by taking the off-diagonal entries of the cost matrix and multiplying them with their probabilities of occurrence:

$$ECM = c(2|1)P(2|1)p_1 + c(1|2)P(1|2)p_2. \quad (2.2)$$

In Sections 2.3 and 2.4 we will show the derivations of LDA and QDA that will minimize the ECM.

2.3 Linear Discriminant Analysis

This section is based on the theory as outlined in Johnson and Wichern (2007).

2.3.1 The Two-Class Case

Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be two normal densities with mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ respectively. The densities of $\mathbf{X}_i' = [X_{i1}, X_{i2}, \dots, X_{ip}]$ for populations Π_1 and Π_2 are given by

$$f_i(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \right] \quad (2.3)$$

for $i = 1, 2$.

Minimizing the ECM is equivalent to using the ratio

$$\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \frac{\frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_1|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) \right]}{\frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}_2|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right]} \geq \left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right)$$

which can be simplified as follows for LDA.

LDA makes the assumption that the covariance matrices are equal, that is, $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$. This assumption of equal covariance matrices leads to the ratio that minimizes the ECM to become

$$\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \frac{\frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) \right]}{\frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right]} \geq \left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right).$$

Suppose that the population parameters $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}$ are known, then cancelling out the terms $(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}$, the minimum ECM classification regions become

$$\text{Region 1: } \left\{ \mathbf{x}: \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right] \geq \left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right) \right\}$$

$$\text{Region 2: } \left\{ \mathbf{x}: \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right] < \left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right) \right\} \quad (2.4)$$

Since the quantities in (2.4) are nonnegative, the natural logarithms can be taken. This is shown below:

$$\begin{aligned} & \ln \left(\exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right] \right) \\ &= -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \\ &= \boldsymbol{\mu}_1' \boldsymbol{\Sigma}^{-1} \mathbf{x} - \boldsymbol{\mu}_2' \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}_1' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2). \end{aligned}$$

The two regions then become

$$\begin{aligned}
 \text{Region 1: } & \left\{ \mathbf{x}: (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) \geq \ln \left[\left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right) \right] \right\} \\
 \text{Region 2: } & \left\{ \mathbf{x}: (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) < \ln \left[\left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right) \right] \right\}. \quad (2.5)
 \end{aligned}$$

Let \mathbf{x}_0 be a new object that needs to be classified. The classification rule can then be written as:

Allocate \mathbf{x}_0 to Π_1 if

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} \mathbf{x}_0 - \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) \geq \ln \left[\left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right) \right],$$

otherwise allocate \mathbf{x}_0 to Π_2 . (2.6)

The population parameters may not be known and then it is necessary to replace the parameters by their plugged-in estimates. Replacing the parameters $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}$, by their corresponding statistics $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$ and \mathbf{S}_{pooled} , gives the following sample classification rule:

Allocate \mathbf{x}_0 to Π_1 if

$$(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)' \mathbf{S}_{pooled}^{-1} \mathbf{x}_0 - \frac{1}{2} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)' \mathbf{S}_{pooled}^{-1} (\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) \geq \ln \left[\left(\frac{c(1|2)}{c(2|1)} \right) \left(\frac{p_2}{p_1} \right) \right],$$

otherwise allocate \mathbf{x}_0 to Π_2 (2.7)

The pooled covariance matrix can be calculated by using the expression

$$\mathbf{S}_{pooled} = \left[\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1)} \right] \mathbf{S}_1 + \left[\frac{n_2 - 1}{(n_1 - 1) + (n_2 - 1)} \right] \mathbf{S}_2$$

where $\mathbf{S}_i = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)'$ and where n_i is the respective sample sizes for $i = 1, 2$. The sample means can be written as $\bar{\mathbf{x}}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{x}_{ij}$ for $i = 1, 2$.

Let the discriminant function for LDA be denoted by $\hat{f}_{LDA}(\mathbf{x})$. Then, consider the case when $c(1|2) = c(2|1)$ and $p_1 = p_2$ so that the ratio that minimizes the ECM is equal to

$$\left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right) = 1.$$

Taking the natural logarithm gives $\ln(1) = 0$. The discriminant function can then be written as

$$\hat{f}_{LDA}(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle + b \quad (2.8)$$

with $\mathbf{a}' = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)' \mathbf{S}_{pooled}^{-1}$ and

$$b = -\frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)' \mathbf{S}_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2).$$

An alternative way of writing the classification rule is by using the sign of the discriminant function. If \mathbf{x}_0 is a new object that needs to be classified then:

Allocate \mathbf{x}_0 to Π_1 if

$$\text{sign}(\hat{f}_{LDA}(\mathbf{x}_0)) \geq 0,$$

otherwise, allocate \mathbf{x}_0 to Π_2 .

(2.9)

2.3.2 The Multi-Class Case

Suppose we have g classes in our data set. In the multi-class case LDA can be performed by calculating scores for each of the g classes in the data set (Johnson and Wichern, 2007). These scores are called linear discriminant scores and are calculated as

$$d_i(\mathbf{x}) = \boldsymbol{\mu}_i' \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}_i' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i + \ln p_i, \quad (2.10)$$

for $i = 1, 2, \dots, g$. By replacing the parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with their sample counterparts $(\bar{\mathbf{x}}, \mathbf{S}_{pooled})$ we can estimate the sample linear discriminant score by

$$\hat{d}_i(\mathbf{x}) = \bar{\mathbf{x}}_i' \mathbf{S}_{pooled}^{-1} \mathbf{x} - \frac{1}{2} \bar{\mathbf{x}}_i' \mathbf{S}_{pooled}^{-1} \bar{\mathbf{x}}_i + \ln p_i, \quad (2.11)$$

for $i = 1, 2, \dots, g$ and the pooled covariance matrix is calculated by using the expression

$$\mathbf{S}_{pooled} = \frac{1}{n_1 + \dots + n_g - g} \left((n_1 - 1) \mathbf{S}_1 + \dots + (n_g - 1) \mathbf{S}_g \right).$$

The classification rule for multi-class LDA is given next. For a new object \mathbf{x}_0 :

Allocate \mathbf{x}_0 to Π_i if

$$\hat{d}_i(\mathbf{x}_0) = \max_j [\hat{d}_1(\mathbf{x}_0), \dots, \hat{d}_g(\mathbf{x}_0)] \quad (2.12)$$

for $i = 1, 2, \dots, g$. We see that this rule assigns \mathbf{x}_0 to the nearest class or population.

2.4 Quadratic Discriminant Analysis

2.4.1 The Two-Class Case

When the assumption of equal covariance matrices is no longer met, the framework in Section 2.3 can be used to obtain a classification rule for Quadratic Discriminant Analysis (QDA). The theory on QDA as outlined by Johnson and Wichern (2007) is described here. Recall the two normal densities $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ having mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$. In the case of QDA we assume that the covariance matrices are unequal, i.e. $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. The classification regions using (2.4) can be shown to be:

$$\begin{aligned} \text{Region 1: } & \left\{ \mathbf{x} : -\frac{1}{2} \mathbf{x}' (\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}) \mathbf{x} + (\boldsymbol{\mu}_1' \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2' \boldsymbol{\Sigma}_2^{-1}) \mathbf{x} - k \geq \left[\frac{c(1|2)}{c(2|1)} \left(\frac{p_2}{p_1} \right) \right] \right\} \\ \text{Region 2: } & \left\{ \mathbf{x} : -\frac{1}{2} \mathbf{x}' (\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}) \mathbf{x} + (\boldsymbol{\mu}_1' \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2' \boldsymbol{\Sigma}_2^{-1}) \mathbf{x} - k < \left[\frac{c(1|2)}{c(2|1)} \left(\frac{p_2}{p_1} \right) \right] \right\} \\ & \text{with } k = \frac{1}{2} \ln \left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|} \right) + \frac{1}{2} (\boldsymbol{\mu}_1' \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2' \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2). \end{aligned} \quad (2.13)$$

Again, let \mathbf{x}_0 be a new object that needs to be classified. The quadratic classification rule can then be written as:

Allocate \mathbf{x}_0 to Π_1 if

$$-\frac{1}{2} \mathbf{x}_0' (\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}) \mathbf{x}_0 + (\boldsymbol{\mu}_1' \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2' \boldsymbol{\Sigma}_2^{-1}) \mathbf{x}_0 - k \geq \ln \left[\frac{c(1|2)}{c(2|1)} \left(\frac{p_2}{p_1} \right) \right]$$

where $k = \frac{1}{2} \ln \left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|} \right) + \frac{1}{2} (\boldsymbol{\mu}_1' \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2' \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2)$,

otherwise allocate \mathbf{x}_0 to Π_2 . (2.14)

When the population parameters ($\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$) are unknown, we can substitute them with the sample statistics ($\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \mathbf{S}_1$ and \mathbf{S}_2). The sample classification rule then becomes the following:

Allocate \boldsymbol{x}_0 to Π_1 if

$$-\frac{1}{2}\boldsymbol{x}'_0(\mathbf{S}_1^{-1} - \mathbf{S}_2^{-1})\boldsymbol{x}_0 + (\bar{\boldsymbol{x}}'_1\mathbf{S}_1^{-1} - \bar{\boldsymbol{x}}'_2\mathbf{S}_2^{-1})\boldsymbol{x}_0 - k \geq \ln \left[\frac{c(1|2)}{c(2|1)} \left(\frac{p_2}{p_1} \right) \right]$$

where $k = \frac{1}{2} \ln \left(\frac{|\mathbf{S}_1|}{|\mathbf{S}_2|} \right) + \frac{1}{2} (\bar{\boldsymbol{x}}'_1\mathbf{S}_1^{-1}\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}'_2\mathbf{S}_2^{-1}\bar{\boldsymbol{x}}_2)$,

otherwise allocate \boldsymbol{x}_0 to Π_2 . (2.15)

Consider again the case when $c(1|2) = c(2|1)$ and $p_1 = p_2$. Then, the quadratic discriminant function can be written as

$$\hat{f}_{QDA}(\boldsymbol{x}) = -0.5\boldsymbol{x}'(\mathbf{S}_1^{-1} - \mathbf{S}_2^{-1})\boldsymbol{x} + (\bar{\boldsymbol{x}}'_1\mathbf{S}_1^{-1} - \bar{\boldsymbol{x}}'_2\mathbf{S}_2^{-1})\boldsymbol{x} - k. \quad (2.16)$$

So, when a new object \boldsymbol{x}_0 needs to be classified we can use the rule:

Allocate \boldsymbol{x}_0 to Π_1 if

$$\text{sign}(\hat{f}_{QDA}(\boldsymbol{x}_0)) \geq 0,$$

otherwise, allocate \boldsymbol{x}_0 to Π_2 . (2.17)

2.4.2 The Multi-Class Case

The quadratic discriminant score for the i^{th} class as illustrated in Johnson and Wichern (2007) is given by

$$d_i^Q(\boldsymbol{x}) = -\frac{1}{2} \ln |\boldsymbol{\Sigma}_i| - \frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) + \ln p_i \quad (2.18)$$

where $i = 1, 2, \dots, g$. The quadratic score $d_i^Q(\boldsymbol{x})$ is composed of contributions from the generalized variance $|\boldsymbol{\Sigma}_i|$, the prior probability p_i , and the square of the Mahalanobis distance from \boldsymbol{x} to the population mean $\boldsymbol{\mu}_i$. The allocation rule is given by:

Allocate \mathbf{x}_0 to Π_i if

$$d_i^Q(\mathbf{x}) = \max_j [d_1^Q(\mathbf{x}), \dots, d_g^Q(\mathbf{x})] \quad (2.19)$$

for $i = 1, 2, \dots, g$.

According to Johnson and Wichern (2007) the first two terms are the same for $d_1^Q(\mathbf{x}), d_2^Q(\mathbf{x}), \dots, d_g^Q(\mathbf{x})$, and consequently, they can be ignored for purposes of allocation.

Both LDA and QDA have a computational advantage when it comes to estimating the parameters and classifying objects. This is because of the simple nature of both techniques.

2.5 Application in R

2.5.1 Software

Applying LDA or QDA to a classification problem is not difficult. A large amount of statistical software is available which can be used in a user friendly environment to perform either LDA or QDA. The R function `lda()` will be used to perform LDA and the function `qda()` will be used to perform QDA. These two functions are located within the R package, MASS. The package, created by Brian Ripley and Bill Venables (2002), consists of the functions and data sets to support their textbook, *Modern Applied Statistics with S* (4th edition, 2002). MASS is a well established package and frequently used by R users.

2.5.2 Data

The first data set that will be used is the well known Iris data set. As mentioned in Section 2.1, the Iris data set owes its popularity to Sir Ronald Fisher who used it to publish his findings on the multiple measurements on taxonomic problems (Fisher, 1936). Fisher used four measurements, *Sepal Length*, *Sepal Width*, *Petal Length* and *Petal Width*, to derive a rule which will discriminate between three Iris species, *Setosa*, *Versicolor* and *Virginica*. Data on these three species were collected by observing fifty plants for each species. In Figure 2.2, a matrix of scatter plots of the Iris data set is given. *Setosa* is shown as the red points,

Versicolor as the blue points and *Virginica* as the green points. It can be seen that *Setosa* is well separated from the other two species of Irises while there is some overlap between *Versicolor* and *Virginica* especially when only observing the sepal length and sepal width measurements.

Another data set that will be used throughout this thesis is the Haemophilia data set. In a study conducted by B.N. Bouma (1975), the detection of potential Haemophilia A carriers was researched. The study consisted of taking blood samples from two groups of women based on two measurements,

$$X_1 = \log_{10}(AHF \text{ activity}) \text{ and } X_2 = \log_{10}(AHF_{like} \text{ antigen}).$$

The first group of $n_1 = 30$ women, consisting of women who did not carry the Haemophilia gene, was called the normal group. The second group of $n_2 = 45$ women, consisting of women who were known to be Haemophilia carriers, was called the obligatory group. The 75 pairs of observations for the two groups of women are shown in Figure 2.5 with X_1 displayed on the horizontal axis and X_2 displayed on the vertical axis. The normal group is shown as the red points and the obligatory group is shown as the blue points. It is clear from Figure 2.5 that the groups of women overlap.

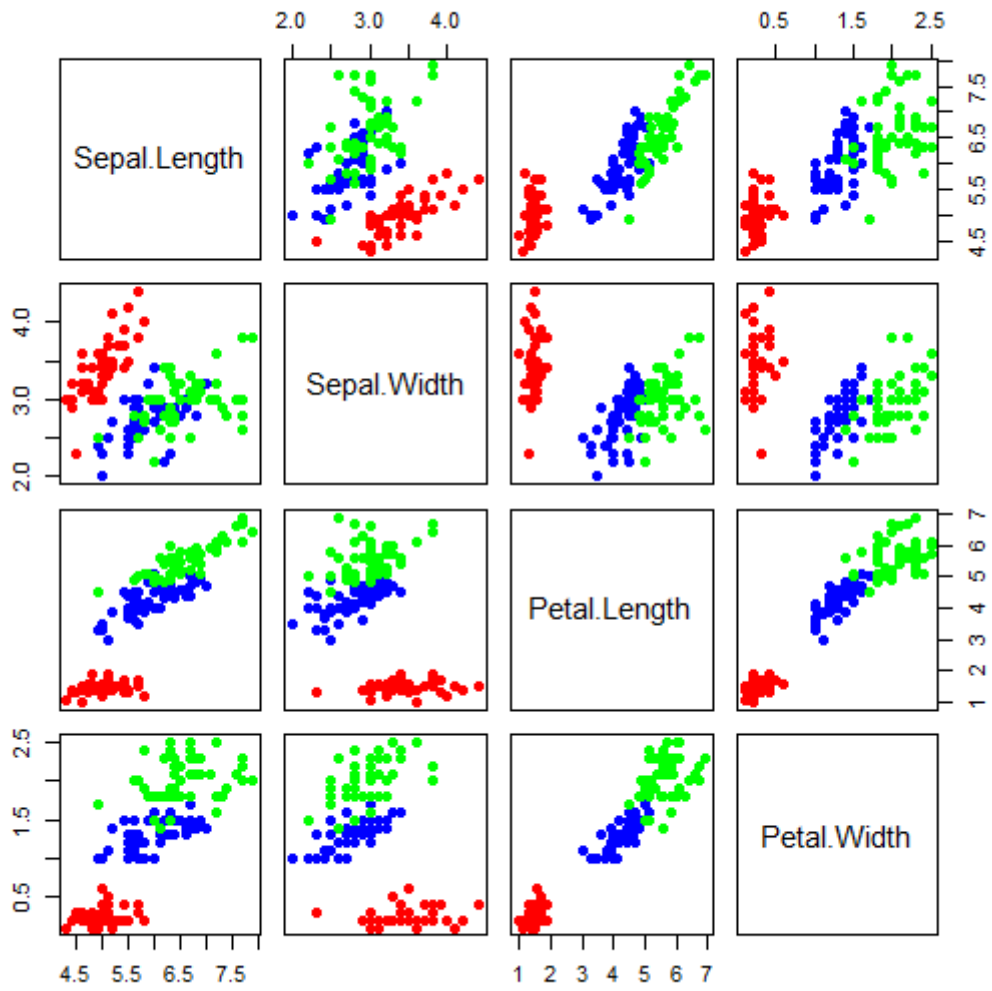


Figure 2.2: A matrix of scatter plots of the Iris data

2.5.3 Performing LDA and QDA in R

The two data sets that were introduced in Section 2.5.2 will now be used to illustrate the classification process with LDA and QDA. However, to keep in line with the two-class theory of Section 2.3, and Section 2.4, only the classes *Versicolor* and *Virginica* of the Iris data set will be used.

Below is the command that can be executed in R to perform LDA. This is done with the Iris data.

```
>library(MASS)
>lda(Species~.,iris.data, prior=c(1,1)/2)
```

First, the package MASS has to be loaded in order to use the `lda()` function. The first argument of the algorithm contains the model that is to be used for prediction. *Species* is the response variable and is modelled by all the remaining input variables, that is, *Sepal Length*, *Sepal Width*, *Petal Length* and *Petal Width*. The second argument specifies the data set to be used and the third argument requires the prior probabilities. Since only two classes are used where both have equal numbers of objects we will assume that their prior probabilities are the same.

To perform a classification with the Iris data, a learning set and a test set will be constructed. The learning set will be used to build the LDA model and the test set will be used to test the performance of the model. The test set is constructed by taking a random sample of the objects from the data. In this example, 15 objects were randomly selected from the two classes *Versicolor* and *Virginica* respectively. The remaining 70 objects are used for the learning set. The classes of the test set can then be forecast with the function `predict()`. This is shown below.

```
>irisindex1<-sample(1:50,15) #index of objects from Versicolor
>irisindex2<-sample(51:100,15)#index of objects from Virginica
>iris.learn <-iris.data[-c(irisindex1,irisindex2),] #learning set
>iris.test <-iris.data[c(irisindex1,irisindex2),] #test set
>model.lda <-lda(Species~.,iris.learn) #fitting the LDA model
>predict(model.lda,iris.test)$class #predicting with the LDA model
```

The classification regions for LDA are presented in Figure 2.3, where *Sepal Length* is shown on the horizontal axis and *Sepal Width* is shown on the vertical axis. The *Versicolor* class is presented as the blue points and *Virginica* as the green points. It can be seen that the two regions are well defined by the decision rule which is represented by the dashed line.

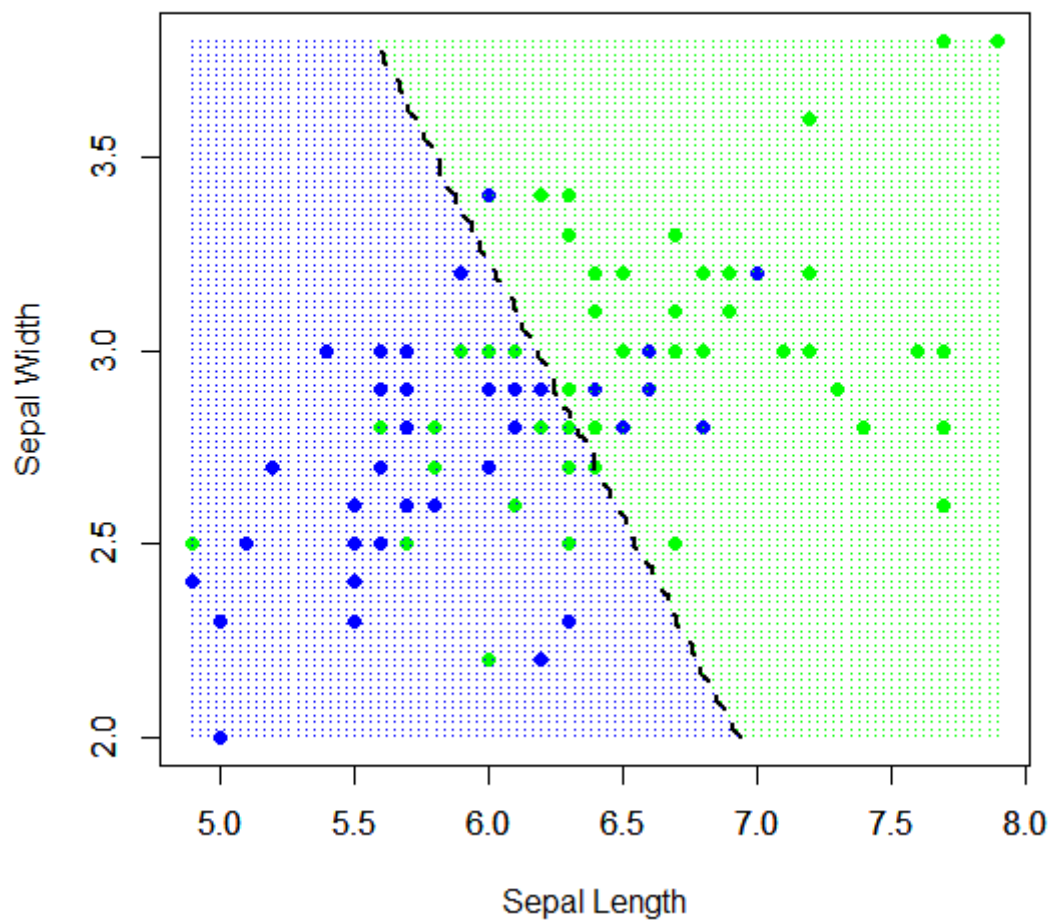


Figure 2.3: LDA classification of the Iris data set for the *Versicolor* and *Virginica* classes.

A measurement that will be used to measure the performance of a classification technique is the test error. The test error calculates the fraction of objects in the test set that were misclassified (n_{iM}) to the number of objects in the test set as follows:

$$\text{test error} = \frac{n_{1M} + n_{2M}}{n_1 + n_2}.$$

The average test error (over 100 repetitions) obtained for the LDA classification of the Iris data in our example was 0.0346. Similarly, QDA can be used to perform a classification with the Iris data, also only with the *Versicolor* and *Virginica* classes, with the functions `qda()` and `predict()`. The steps are shown below.

```
>model.qda <-qda(Species~.,iris.learn) #fitting the QDA model
>predict(model.qda,iris.test)$class #predicting with the QDA model
```

Again, the last line in the code predicts the classes of the objects in the test set. The classification regions for QDA are shown graphically in Figure 2.4. *Sepal Length* is shown on the horizontal axis and *Sepal Width* on the vertical axis. *Versicolor* is shown as the blue points and *Virginica* is shown as the green points. The average test error for QDA, also over 100 repetitions, is 0.0356.

LDA and QDA were also executed with the Haemophilia data. Similar coding, which is shown below, was used to construct a learning set and a test set from the Haemophilia data to perform the classification procedure.

```
>haemindex1<-sample(1:30,9)#index to select objects from group 1
>haemindex2<-sample(31:75,14)#index to select objects from group 2
>haem.learn<-haemo.data[-c(haemindex1,haemindex2),] #learning set
>haem.test<-haemo.data[c(haemindex1,haemindex2),] #test set
>haem.model.lda <-lda(Group~.,haem.learn) #fitting the LDA model
>predict(haem.model.lda,haem.test)$class #predicting with the model
```

The test set consists of 23 objects that were randomly drawn from the original data. The remaining 52 objects are used for the construction of the learning set. The resulting LDA model is shown graphically in Figure 2.5. The average test error over 100 repetitions obtained was 0.1495. In Figure 2.6 the QDA model is shown graphically. An average test error of

0.1517 was obtained for QDA. The code for developing the QDA model is similar to that of the Iris model.

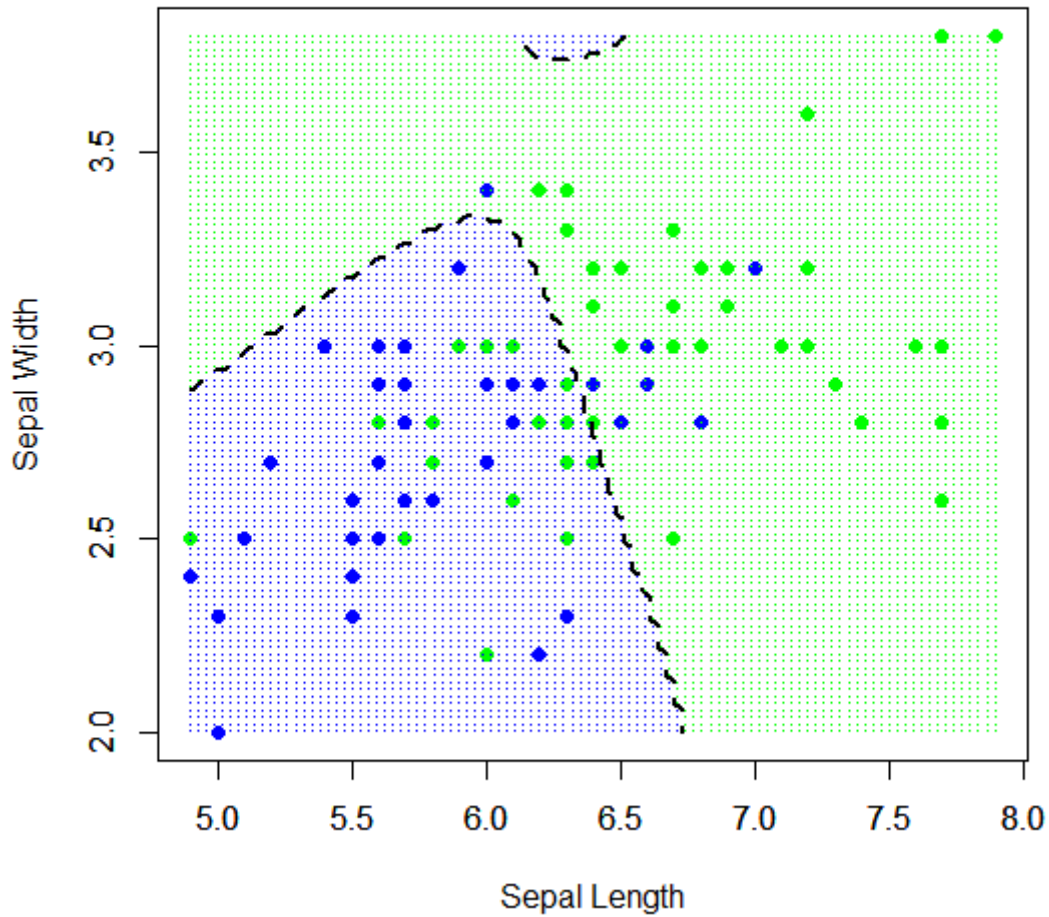


Figure 2.4: QDA classification regions of the Iris data set for the *Versicolor* and *Virginica* classes.

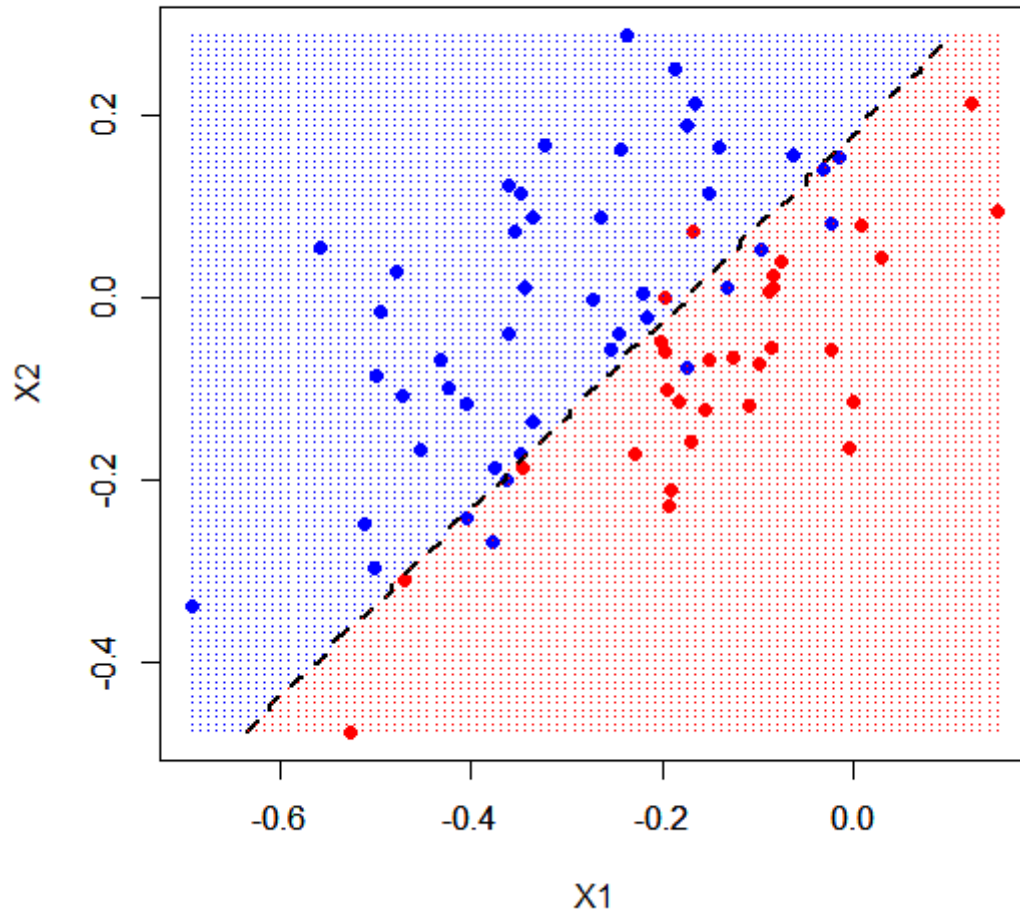


Figure 2.5: LDA classification regions for the Haemophilia data

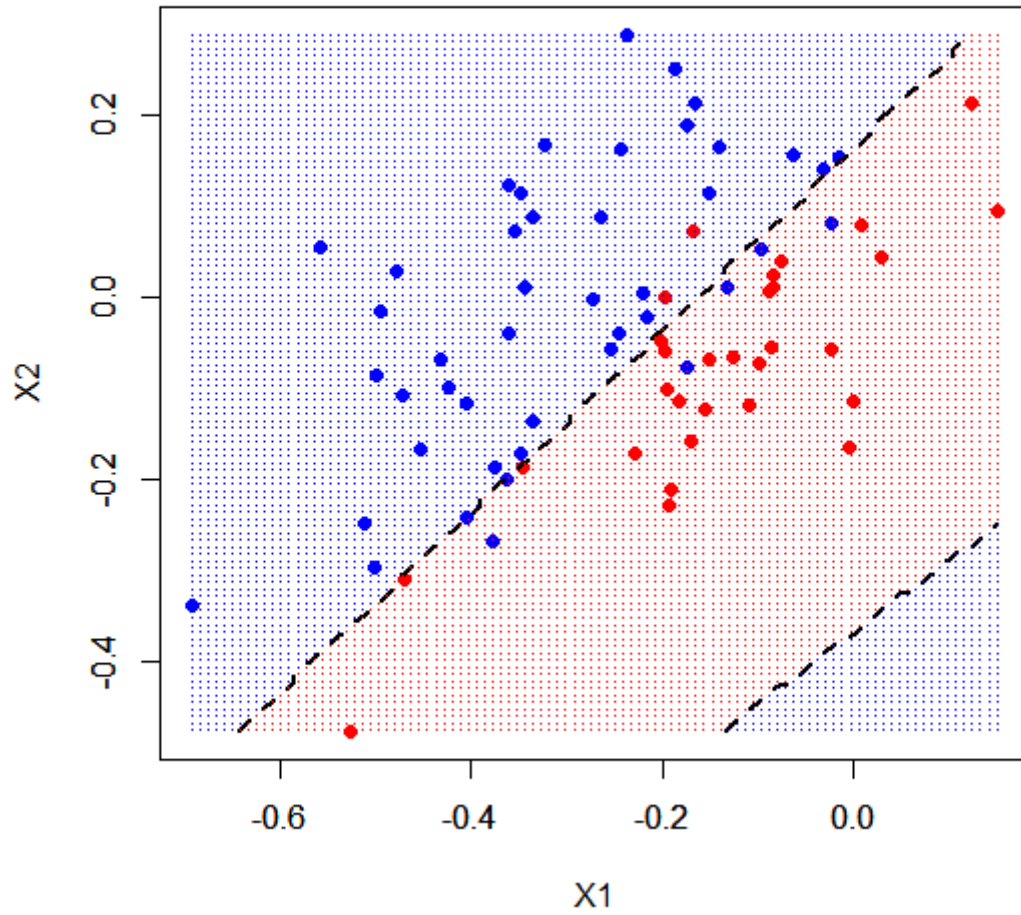


Figure 2.6: QDA classification regions for the Haemophilia data

2.6 Conclusion

In Section 2.2 the characteristics of an optimal classification model were discussed. These characteristics are: incorporating prior probabilities into the model and taking into account the costs of misclassifying objects. We have seen that the ECM rule can be used to estimate the average cost of misclassification. In Section 2.3, LDA was introduced and it was seen that LDA requires the assumption of normality to be met as well as the assumption of equal covariance matrices. LDA extends easily to the multi-class case by means of introducing discriminant scores where an object is classified to the class with the maximum score. We then looked at QDA in Section 2.4 which also has the underlying assumption of normality, but does not require the covariance matrices to be equal. The extension to the multi-class case worked similar to LDA.

We saw that some of the advantages of LDA and QDA include their computation speed since their parameters are easily estimated and the fact that both techniques are easily adjustable for the multi-class case. Some drawbacks of these techniques are that sometimes the assumption of normality may not be met and both techniques may be sensitive to outliers. Another disadvantage is that LDA and QDA may give computational problems when the number of variables exceeds the number of objects (Johnson and Wichern, 2007).

The classification techniques are executed in Section 2.5 with the R programming language. The Iris data set and the Haemophilia data set are introduced in this section and are used in the analysis. The average test error is used to measure the performance of the classification rules and we saw that LDA achieved a test error of 0.0346 and 0.1495 for the Iris data and Haemophilia data sets respectively. QDA achieved a test error of 0.0356 for the Iris data and 0.1517 for the Haemophilia data. We saw that both techniques performed equally well with the Iris data, while LDA outperformed QDA with the Haemophilia data. The test errors from the Haemophilia data for both techniques are relatively large. This could be the result of the big overlap between the classes of both data sets.

In the next chapter we will introduce a methodology which may give a computational advantage when the number of variables exceeds the number of objects, or when the assumption of normality is not met.

CHAPTER 3

SUPPORT VECTOR MACHINES

3.1 Introduction

The Support Vector Machine (SVM) methodology emerged in the field of machine learning and was proposed by Vladimir Vapnik in the 1990's. Initially, Vapnik proposed a *maximal margin classifier* that can be optimised to discriminate between two or more classes and hence be used for classification. It was only later when Vapnik introduced the term Support Vector (SV) by which it is known today (Vapnik, 1995). SVMs are currently of great interest especially to applied scientists in machine learning, data mining and bioinformatics. SVMs have also been successfully applied to classification problems. Some of these examples include handwritten digit recognition, text categorization, cancer classification, protein secondary-structure prediction and cloud classification using satellite-radiance profiles (Izenman, 2008).

The SVM can be divided into the linear SVM and the nonlinear SVM. The linear SVM will be looked at in Section 3.2. In Section 3.3, nonlinear SVMs will be looked at with a focus on nonlinear transformations and the kernel trick. The properties of kernel functions will be given attention in Section 3.3.2 and examples of kernel functions will be shown in Section 3.3.3. In Section 3.3.4 a discriminant function will be given for the SVM. We will also refer briefly to the multi-class SVM in Section 3.3.5. Finally, the implementation of the SVM in \mathbb{R} will be dealt with in Section 3.4, while concluding remarks will be given in Section 3.5. The theory on the SVM that will be dealt with in this chapter will be discussed as outlined in Izenman (2008).

3.2 Support Vector Machines

The SVM for a two-class classification problem is obtained by maximizing a margin between the two classes. This margin is defined as the distance between two hyperplanes which are determined by the support vectors. This is shown schematically in Figure 3.1. The support vectors are the points that lie on the two hyperplanes, H_1 and H_2 , and are defined as those data points that form the shortest distance between themselves and the dashed line. The

dashed line is known as the separating hyperplane which is defined as the hyperplane which separates the points of the two classes without error. Thus, it is clear from Figure 3.1 that the aim is to find a separating hyperplane such that the distances between the closest two data points on either side are maximised. Such a hyperplane is called an optimal separating hyperplane.

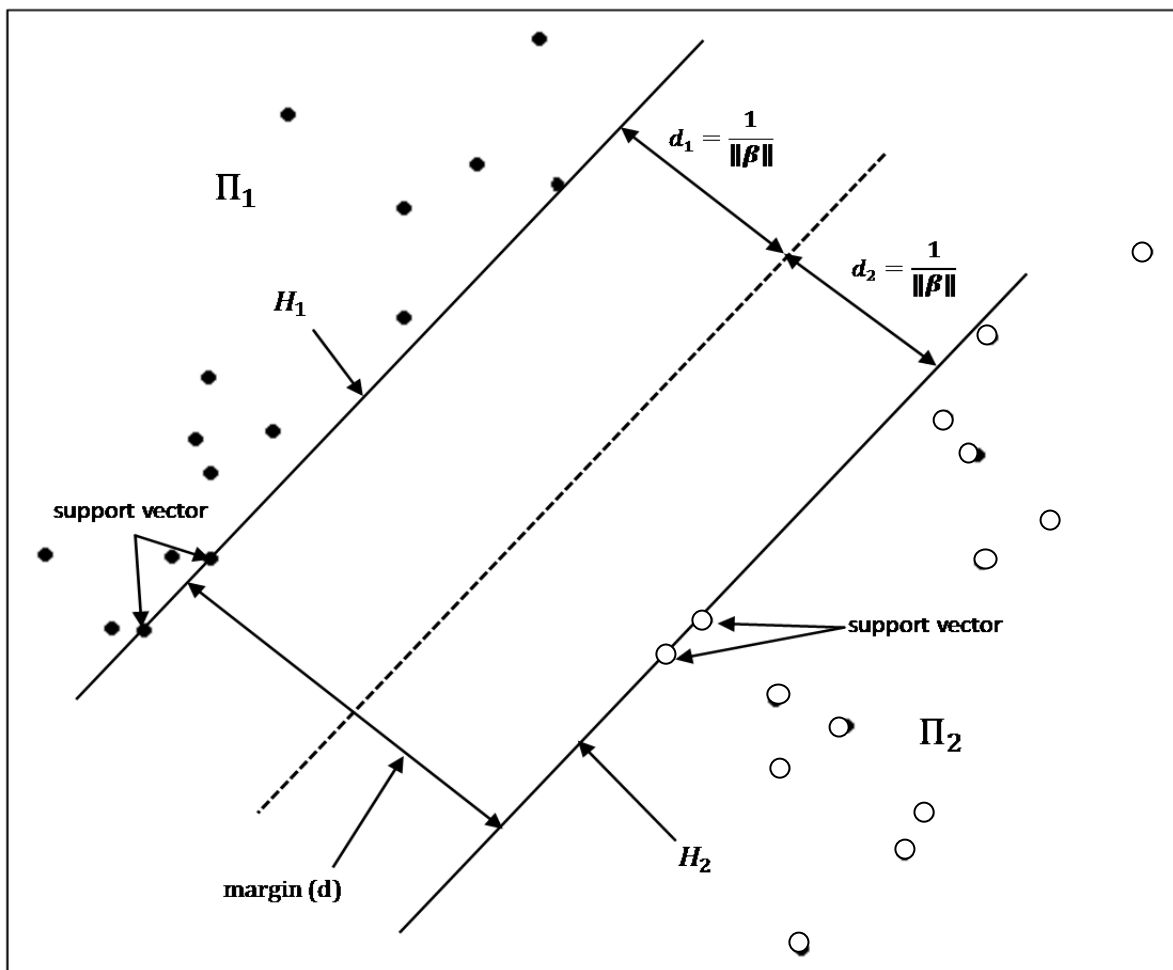


Figure 3.1: A schematic illustration of a linear SVM

3.2.1 The Linearly Separable Case

Let $y_i \in \{-1, +1\}$ be the variable representing the two classes, Π_1 and Π_2 , where Π_1 is represented by -1 and Π_2 is represented by $+1$. Also, let \mathbf{x}' be the $(1 \times p)$ data vector from a $(n \times p)$ data matrix \mathbf{X} , where n is the number of objects and p is the number of variables. Suppose that in a two-class classification problem, the classes Π_1 and Π_2 can be separated by a hyperplane,

$$\mathbf{x}: f(\mathbf{x}) = \beta_0 + \mathbf{x}'\boldsymbol{\beta} = 0 \quad (3.1)$$

where $\boldsymbol{\beta}$ is known as the weight vector and β_0 as the bias. When all the points in the data set can be successfully separated into the two classes, Π_1 and Π_2 , the hyperplane is called a separating hyperplane. There can be an endless number of such separating hyperplanes, therefore the optimal separating hyperplane is sought.

Given a separating hyperplane, let d_1 be the distance from the separating hyperplane to the nearest data point belonging to Π_1 , and let d_2 be the distance from the separating hyperplane to the nearest data point belonging to Π_2 . The distance that is defined by $d = d_1 + d_2$ is called the margin of the separating hyperplane. When the respective distances d_1 and d_2 are maximized, the separating hyperplane in such a case is called the optimal separating hyperplane. This is illustrated by Figure 3.1.

In the linearly separable case in the context of two classes, there exist β_0 and $\boldsymbol{\beta}$ such that

$$\begin{aligned} \beta_0 + \mathbf{x}'_i\boldsymbol{\beta} &\geq -1, \text{ if } y_i = -1, \\ \beta_0 + \mathbf{x}'_i\boldsymbol{\beta} &\leq +1, \text{ if } y_i = +1, \end{aligned} \quad (3.2)$$

for all i . If there are data vectors in the learning set such that the equalities in (3.2) hold, then these data vectors lie on the hyperplanes $\{\mathbf{x}: (\beta_0 - 1) + \mathbf{x}'\boldsymbol{\beta} = 0\}$ and $\{\mathbf{x}: (\beta_0 + 1) + \mathbf{x}'\boldsymbol{\beta} = 0\}$ which are denoted by H_1 and H_2 in Figure 3.1. Points that lie on either one of these two hyperplanes are called support vectors (SV) and are denoted by \mathbf{x}_1 and \mathbf{x}_2 . When \mathbf{x}_1 lies on H_1 and \mathbf{x}_2 lies on H_2 , it suggests that

$$\beta_0 + \mathbf{x}'_1\boldsymbol{\beta} = -1, \beta_0 + \mathbf{x}'_2\boldsymbol{\beta} = +1. \quad (3.3)$$

The difference between these two equations is $\mathbf{x}'_2\boldsymbol{\beta} - \mathbf{x}'_1\boldsymbol{\beta} = 2$, and their sum is $\beta_0 = -\frac{1}{2}(\mathbf{x}'_2\boldsymbol{\beta} + \mathbf{x}'_1\boldsymbol{\beta})$. The perpendicular distances from the hyperplane $\beta_0 + \mathbf{x}'\boldsymbol{\beta} = 0$ to the points \mathbf{x}_1 and \mathbf{x}_2 can be obtained as

$$d_1 = \frac{|\beta_0 + \mathbf{x}'_1\boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} = \frac{1}{\|\boldsymbol{\beta}\|}, \quad d_2 = \frac{|\beta_0 + \mathbf{x}'_2\boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} = \frac{1}{\|\boldsymbol{\beta}\|}. \quad (3.4)$$

Therefore, the margin of the separating hyperplane is $d = 2/\|\boldsymbol{\beta}\|$. The inequalities in (3.2) can be combined into a single set which can be written as,

$$y_i(\beta_0 + \mathbf{x}'_i\boldsymbol{\beta}) \geq +1, \quad i = 1, 2, \dots, n. \quad (3.5)$$

From (3.3) it is clear that if \mathbf{x}_i is a SV with respect to the hyperplane in (3.1), then its margin equals 1, that is, when

$$y_i(\beta_0 + \mathbf{x}'_i\boldsymbol{\beta}) = 1. \quad (3.6)$$

The goal now is to find the optimal separating hyperplane which maximizes the margin, $d = 2/\|\boldsymbol{\beta}\|$, subject to the conditions in (3.5). In other words, the goal is to find β_0 and $\boldsymbol{\beta}$ which will

$$\min \left[\frac{1}{2} \|\boldsymbol{\beta}\|^2 \right],$$

$$\text{subject to: } y_i(\beta_0 + \mathbf{x}'_i\boldsymbol{\beta}) \geq 1, \quad i = 1, 2, \dots, n. \quad (3.7)$$

This is called a convex optimization problem, minimizing a quadratic function subject to linear constraints. The convex nature of the optimization problem ensures that there is a global minimum without any local minima. Lagrangian multipliers are introduced by multiplying the constraints by positive Lagrangian multipliers. The primal function is then formed by subtracting each product from the objective function (3.7),

$$F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{\beta}\|^2 - \sum_{i=1}^n \alpha_i [y_i(\beta_0 + \mathbf{x}'_i\boldsymbol{\beta}) - 1],$$

$$\text{with } \alpha_i \geq 0, \quad i = 1, 2, \dots, n. \quad (3.8)$$

Note that $\boldsymbol{\alpha}$ is the n -vector of Lagrangian coefficients. The goal is to minimize F with respect to the primal variables β_0 and $\boldsymbol{\beta}$, and then to maximize the resulting minimum- F with respect to the dual variables $\boldsymbol{\alpha}$.

Izenman (2008) shows that the Karush-Kuhn-Tucker (Karush, 1939; Kuhn and Tucker, 1951) conditions give necessary conditions to the solution of a constrained optimization problem (3.8). For the primal problem, β_0 , $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ have to satisfy:

$$\frac{\partial F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})}{\partial \beta_0} = - \sum_{i=1}^n \alpha_i y_i = 0, \quad (3.9)$$

$$\frac{\partial F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = \mathbf{0}, \quad (3.10)$$

$$y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) - 1 \geq 0, \quad (3.11)$$

$$\alpha_i \geq 0, \quad (3.12)$$

$$\alpha_i [y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) - 1] = 0, \text{ for } i = 1, 2, \dots, n. \quad (3.13)$$

Solving Equations (3.9) and (3.10) give the results

$$\sum_{i=1}^n \alpha_i y_i = 0, \quad (3.14)$$

$$\boldsymbol{\beta} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i. \quad (3.15)$$

Substituting (3.14) and (3.15) into (3.8) results in the minimum value of $F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})$:

$$\begin{aligned} F_D(\boldsymbol{\alpha}) &= \frac{1}{2} \|\boldsymbol{\beta}^*\|^2 - \sum_{i=1}^n \alpha_i [y_i(\beta_0^* + \mathbf{x}'_i \boldsymbol{\beta}^*) - 1] \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}'_i \mathbf{x}_j) - \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}'_i \mathbf{x}_j) + \sum_{i=1}^n \alpha_i \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}'_i \mathbf{x}_j). \end{aligned} \quad (3.16)$$

Expression (3.16) is known as the dual functional of the optimization problem.

The vectors of Lagrangian multipliers $\boldsymbol{\alpha}$ are found by maximizing the dual function (3.16) subject to the constraints (3.12) and (3.14). This can be written in matrix notation as follows:

$$\begin{aligned} \max_{\alpha} \left[F_D(\alpha) = \mathbf{1}'_n \alpha - \frac{1}{2} \alpha' \mathbf{H} \alpha \right] \\ \text{subject to: } \alpha_i \geq 0, \alpha' \mathbf{y} = \mathbf{0}, \end{aligned} \quad (3.17)$$

where $\mathbf{y} = (y_1, y_2, \dots, y_n)'$ and $\mathbf{H} = [H_{ij}]$ is a square $(n \times n)$ matrix with $H_{ij} = y_i y_j (\mathbf{x}'_i \mathbf{x}_j)$ and $i = 1, 2, \dots, n$. Let $\hat{\alpha}$ denote the solution to this problem, then

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i \quad (3.18)$$

yields the optimal weight vector for $\boldsymbol{\beta}$. Whenever $\hat{\alpha}_i > 0$, then $y_i(\beta_0^* + \mathbf{x}'_i \boldsymbol{\beta}^*) = 1$ and this set of objects \mathbf{x}_i are the support vectors. Only such objects are considered in finding $\hat{\boldsymbol{\beta}}$. Objects for which $\hat{\alpha}_i = 0$, are not considered as support vectors. Let sv be the subset of indices that identify the support vectors, then (3.18) can be rewritten as

$$\hat{\boldsymbol{\beta}} = \sum_{i \in sv} \hat{\alpha}_i y_i \mathbf{x}_i. \quad (3.19)$$

Therefore, $\hat{\boldsymbol{\beta}}$ is a linear function only of the support vectors \mathbf{x}_i , $i \in sv$. According to Izenman (2008), in practice, the number of support vectors is relatively small compared to the sample size. However, the support vectors carry all the information necessary to determine the optimal separating hyperplane.

Since the optimal bias $\hat{\beta}_0$ is not determined directly from the optimization solution, it can be estimated by solving (3.13) for each support vector and then averaging the results. The estimated bias of the optimal hyperplane is then given by

$$\hat{\beta}_0 = \frac{1}{|sv|} \sum_{i \in sv} \left(\frac{1 - \mathbf{x}'_i \hat{\boldsymbol{\beta}}}{y_i} \right), \quad (3.20)$$

where $|sv|$ is the number of support vectors. The estimated optimal hyperplane can thus be written as

$$\{\mathbf{x}: \hat{\beta}_0 + \mathbf{x}' \hat{\boldsymbol{\beta}} = 0\} \text{ or } \{\mathbf{x}: \hat{\beta}_0 + \sum_{i \in sv} \hat{\alpha}_i y_i (\mathbf{x}' \mathbf{x}_i) = 0\}$$

As previously stated only support vectors are relevant in computing the optimal separating hyperplane which means that objects that are not by definition support vectors are irrelevant for solving of the optimization problem.

Let $\hat{f}_{SVM}(\mathbf{x})$ represent the discriminant function for SVM:

$$\begin{aligned}\hat{f}_{SVM}(\mathbf{x}) &= \hat{\beta}_0 + \mathbf{x}'\hat{\boldsymbol{\beta}} \\ &= \hat{\beta}_0 + \sum_{i \in sv} \hat{\alpha}_i y_i (\mathbf{x}'\mathbf{x}_i).\end{aligned}\quad (3.21)$$

Then the classification rule for SVMs is as follows:

Allocate \mathbf{x}_0 to Π_1 if

$$\text{sign}(\hat{f}_{SVM}(\mathbf{x}_0)) \geq 0, \quad (3.22)$$

otherwise classify \mathbf{x}_0 to Π_2 .

If $j \in sv$, then, from (3.21),

$$y_j \hat{f}_{SVM}(\mathbf{x}) = y_j \hat{\beta}_0 + \sum_{i \in sv} \hat{\alpha}_i y_i y_j (\mathbf{x}'\mathbf{x}_i) = 1. \quad (3.23)$$

Thus, the squared norm of the weight vector $\hat{\boldsymbol{\beta}}$ of the optimal hyperplane is

$$\begin{aligned}\|\hat{\boldsymbol{\beta}}\|^2 &= \sum_{i \in sv} \sum_{j \in sv} \hat{\alpha}_i \hat{\alpha}_j y_i y_j (\mathbf{x}'_i \mathbf{x}_j) \\ &= \sum_{j \in sv} \hat{\alpha}_j y_j \sum_{i \in sv} \hat{\alpha}_i y_i (\mathbf{x}'_i \mathbf{x}_j) \\ &= \sum_{j \in sv} \hat{\alpha}_j (1 - y_j \hat{\beta}_0) \\ &= \sum_{j \in sv} \hat{\alpha}_j.\end{aligned}\quad (3.24)$$

It follows from (3.24) that the optimal hyperplane has a maximum margin $2/\|\hat{\boldsymbol{\beta}}\|$, where

$$\frac{1}{\|\hat{\boldsymbol{\beta}}\|} = \left(\sum_{j \in sv} \hat{\alpha}_j \right)^{-1/2}. \quad (3.25)$$

3.2.2 The Linearly Non-Separable Case

In practice, a classification rule such as the optimal separating hyperplane will not always lead to 100% correct classification of objects into their correct classes. There may be some overlap of points between the classes and this will result in some misclassification of points. A reason for the overlap could be the presence of high variances in the classes. As a result, one or more of the constraints mentioned in Section 3.2.1 may be violated. To overcome this problem a more flexible formulation of the problem can be attained which will lead to the so-called soft-margin solution. Vapnik (1995) introduced a nonnegative slack variable to solve the linearly non-separable case.

The nonnegative slack variable, $\xi_i \geq 0$, is associated with each object (\mathbf{x}_i, y_i) in the data. Using the slack variable, the constraints (3.7) now become $y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) + \xi_i \geq 1$ for $i = 1, 2, \dots, n$. Data points that obey this constraint have $\xi_i = 0$. The classifier now has to find the optimal hyperplane that controls both the margin, $2/\|\boldsymbol{\beta}\|$, and some computationally simple function of the slack variables. The *soft-margin optimization problem* is to find $\beta_0, \boldsymbol{\beta}$, and ξ_i to

$$\min \left[\frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^n \xi_i \right],$$

$$\text{subject to: } \xi_i \geq 0, \quad y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) \geq 1 - \xi_i, \quad i = 1, 2, \dots, n, \quad (3.26)$$

where $C > 0$ is a regularization parameter which takes the form of a tuning constant that controls the size of the slack variables and balances the two terms in the function to be minimized.

The primal function $F_p = F_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\eta})$ for the non-separable case is then written as

$$F_p = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) - (1 - \xi_i)] - \sum_{i=1}^n \eta_i \xi_i, \quad (3.27)$$

with $\alpha_i \geq 0$ and $\eta_i \geq 0$ as the Lagrange multipliers. Differentiating F_p with respect to $\beta_0, \boldsymbol{\beta}$, and ξ_i leads to the following results:

$$\frac{\partial F_p}{\partial \beta_0} = - \sum_{i=1}^n \alpha_i y_i, \quad (3.28)$$

$$\frac{\partial F_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \quad (3.29)$$

$$\frac{\partial F_P}{\partial \xi_i} = C - \alpha_i - \eta_i. \quad (3.30)$$

Setting these equations equal to zero and solving them leads to the following results,

$$\sum_{i=1}^n \alpha_i y_i = 0, \quad \boldsymbol{\beta}^* = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \quad \alpha_i = C - \eta_i. \quad (3.31)$$

The dual function of the non-separable case is given by

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}'_i \mathbf{x}_j). \quad (3.32)$$

From the constraints $C - \alpha_i - \eta_i = 0$ and $\eta_i \geq 0$, it follows that $0 \leq \alpha_i \leq C$. Using the Karush-Kuhn-Tucker (Karush, 1939; Kuhn and Tucker, 1951) conditions give the following:

$$y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) - (1 - \xi_i) \geq 0, \quad (3.33)$$

$$\xi_i \geq 0, \quad (3.34)$$

$$\alpha_i \geq 0, \quad (3.35)$$

$$\eta_i \geq 0, \quad (3.36)$$

$$\alpha_i [y_i(\beta_0 + \mathbf{x}'_i \boldsymbol{\beta}) - (1 - \xi_i)] = 0, \quad (3.37)$$

$$\xi_i(\alpha_i - C) = 0, \quad (3.38)$$

for $i = 1, 2, \dots, n$. The dual maximization problem can be written as follows: we have to find $\boldsymbol{\alpha}$ to

$$\max_{\boldsymbol{\alpha}} \left[F_D(\boldsymbol{\alpha}) = \mathbf{1}'_n \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}' \mathbf{H} \boldsymbol{\alpha} \right],$$

$$\text{subject to: } \boldsymbol{\alpha}' \mathbf{y} = 0, \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq C \mathbf{1}_n. \quad (3.39)$$

The only difference between the optimization problem here and that for the linearly separable case (3.17), is that the Lagrangian coefficients α_i , $i = 1, 2, \dots, n$, are each bounded above by C .

If $\hat{\alpha}$ solves this optimization problem, then

$$\hat{\beta} = \sum_{i \in sv} \hat{\alpha}_i y_i \mathbf{x}_i \quad (3.40)$$

results in the optimal weight vector, where the set sv of support vectors consists of objects which satisfy the equality in the constraint (3.33).

3.3 Nonlinear Support Vector Machines

A data set can be of the nature such that the use of an ordinary linear classifier would not be appropriate. In this section, the linear support vector machine will be extended to the nonlinear case.

3.3.1 Nonlinear Transformations and the Kernel Trick

Suppose we have an input space and each object in input space, $\mathbf{x}_i \in \mathcal{R}^p$, is transformed using some nonlinear mapping function $\Phi: \mathcal{R}^p \rightarrow \mathcal{F}$. The nonlinear mapping function Φ is called the feature map and the space \mathcal{F} is called the feature space where the dimension of the feature space may be very high or even infinite. Assume that \mathcal{F} is a space of real-valued functions on \mathcal{R} with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$.

Suppose we have a sample $\{\mathbf{x}_i, y_i\}$, where $y_i \in \{-1, +1\}$. We can transform this sample using the nonlinear mapping function Φ . The transformed sample is then $\{\Phi(\mathbf{x}_i), y_i\}$. If $\Phi(\mathbf{x}_i)$ is substituted for \mathbf{x}_i in the development of the linear SVM, then data would only enter the optimization problem by way of the inner products $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = \Phi(\mathbf{x}_i)' \Phi(\mathbf{x}_j)$. However, when using nonlinear transformations in such a way, a computational problem arises when computing inner products. The nonlinear SVM works by finding an optimal separating hyperplane in high-dimensional feature space \mathcal{F} . However, the construction of this hyperplane is very difficult because of the possible extremely high dimensionality. The kernel trick provides a solution to this problem and it was Vapnik (1995) who was first to apply the kernel trick to SVM.

The kernel is a function for computing inner products of the form $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ in feature space \mathcal{F} . The trick is that instead of computing these inner products in \mathcal{F} , rather compute them using a nonlinear kernel function, $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ in input space, which helps to speed up computations.

3.3.2 Properties of the Kernel Function

A *kernel* k is a function $k: \mathfrak{R}^r \times \mathfrak{R}^r \rightarrow \mathfrak{R}$ such that, for all $\mathbf{x}, \mathbf{y} \in \mathfrak{R}^p$,

$$k(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle. \quad (3.41)$$

The kernel function is used to compute inner-products in feature space by using only the original sample data. That is, the inner product $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$ is replaced by the kernel function $k(\mathbf{x}, \mathbf{y})$. The choice of which kernel function to use implicitly determines the mapping function, Φ , as well as the feature space, \mathcal{F} . The advantage of using kernels as inner products is that for a given kernel function k , the need to know the explicit form of Φ is unnecessary.

It is required that the kernel function be symmetric, $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$, and satisfy the inequality, $[k(\mathbf{x}, \mathbf{y})]^2 \leq k(\mathbf{x}, \mathbf{x})k(\mathbf{y}, \mathbf{y})$, derived from the Cauchy-Schwarz inequality. If $k(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x} \in \mathfrak{R}^p$, this implies that $\|\Phi(\mathbf{x})\|_{\mathcal{F}} = 1$. A kernel k is said to have the *reproducing property* if the kernel has the property that it corresponds to an inner product in a high dimensional space, that is for any $f \in \mathcal{F}$,

$$\langle f(\cdot), k(\mathbf{x}, \cdot) \rangle = f(\mathbf{x}). \quad (3.42)$$

If k has this property, it is a reproducing kernel. In particular, if $f(\cdot) = k(\mathbf{x}, \cdot)$, then,

$$\langle k(\mathbf{x}, \cdot), k(\mathbf{y}, \cdot) \rangle = k(\mathbf{x}, \mathbf{y}). \quad (3.43)$$

Let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ be a set of n points in \mathfrak{R}^p . Then the $(n \times n)$ -matrix $\mathbf{K} = (k_{ij})$, where $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $i = 1, 2, \dots, n$, is called the Gram matrix of k with respect to $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$. If the Gram matrix \mathbf{K} satisfies $\mathbf{u}'\mathbf{K}\mathbf{u} \geq 0$, for any non-zero n -vector \mathbf{u} , then it is said to be nonnegative-definite with nonnegative eigenvalues. In this case k is a nonnegative-definite kernel or a Mercer kernel (Mercer, 1909). If k is a specific Mercer kernel on $\mathfrak{R}^p \times \mathfrak{R}^p$, then a

unique space \mathcal{F}_k can be constructed of real-valued functions for which k is its reproducing kernel. \mathcal{F}_k is called the *reproducing kernel space*.

3.3.3 Examples of Kernels

The following table lists a few examples of popular kernel functions, $k(\mathbf{x}, \mathbf{y})$, found in the machine learning literature.

Table 3.1: Examples of kernel functions

Name	Kernel Function
d^{th} degree Polynomial	$k(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^d$
Gaussian	$k(\mathbf{x}, \mathbf{y}) = \exp\{-\gamma\ \mathbf{x} - \mathbf{y}\ ^2\}$
Laplacian	$k(\mathbf{x}, \mathbf{y}) = \exp\{-\gamma\ \mathbf{x} - \mathbf{y}\ \}$
Sigmoid	$k(\mathbf{x}, \mathbf{y}) = \tanh(a\langle \mathbf{x}, \mathbf{y} \rangle + b)$

The first kernel listed is the d^{th} degree Polynomial kernel function and it only has one parameter, d . The parameter is an integer and if $d = 1$, the feature map reduces to the linear kernel. The second kernel listed is the Gaussian kernel function with parameter γ . Other authors write the Gaussian kernel as $k(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right\}$ which is similar to our notation, with $\gamma = \frac{1}{2\sigma^2}$. This parameter in the Gaussian kernel is a scaling parameter which will also be discussed in more depth in Chapter 4. The Sigmoid kernel has two parameters, $a > 0$ and $b > 0$.

The kernel functions listed in the table are all Mercer kernels and it is possible to show that they correspond to inner products in \mathcal{F} . The following illustrates that the d^{th} degree polynomial and the Gaussian kernel are inner products in a high-dimensional space. These two examples are also shown in Lamont (2008).

The Gaussian kernel is given by the function

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}.$$

Suppose the case $x \in \mathfrak{R}^1$ is used. Then, using the Taylor series expansion, $e^t = \sum_{i=0}^{\infty} \frac{t^i}{i!}$,

the Gaussian kernel can be written as

$$\begin{aligned}
 k(x_i, x_j) &= \exp(-\gamma \|x_i - x_j\|^2) \\
 &= \exp(-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2) \\
 &= \exp(-\gamma x_i^2 - \gamma x_j^2) \exp(2\gamma x_i x_j) \\
 &= \exp(-\gamma x_i^2 - \gamma x_j^2) \left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \dots \right) \\
 &= \exp(-\gamma x_i^2 - \gamma x_j^2) \\
 &\quad \times \left(1 + \sqrt{\frac{2\gamma}{1!}} x_i \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i \sqrt{\frac{(2\gamma)^2}{2!}} x_j + \sqrt{\frac{(2\gamma)^3}{3!}} x_i \sqrt{\frac{(2\gamma)^3}{3!}} x_j + \dots \right) \\
 &= \langle \Phi(x_i), \Phi(x_j) \rangle.
 \end{aligned}$$

Now it can be seen that

$$\Phi(x)' = \exp(-\gamma x^2) \left(1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \dots \right)$$

is a mapping function that corresponds to a nonlinear transformation of the input space. In the same way, $\mathbf{x} \in \mathfrak{R}^p$ can be proved.

The d^{th} degree polynomial kernel is given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d.$$

This kernel function corresponds to the following inner product:

$$\begin{aligned}
 k(\mathbf{x}_i, \mathbf{x}_j) &= \left(\sum_{i,j=1}^p x_i x_j \right)^d \\
 &= \left(\sum_{i_1, j_1=1}^p x_{i_1} x_{j_1} \right) \left(\sum_{i_2, j_2=1}^p x_{i_2} x_{j_2} \right) \dots \left(\sum_{i_d, j_d=1}^p x_{i_d} x_{j_d} \right) \\
 &= \sum_{i_1, j_1=1}^p \sum_{i_2, j_2=1}^p \dots \sum_{i_d, j_d=1}^p (x_{i_1} x_{i_2} \dots x_{i_d}) (x_{j_1} x_{j_2} \dots x_{j_d}) \\
 &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle.
 \end{aligned}$$

In practice it is not always clear which kernel to use if no information is available in the literature. The Gaussian kernel function is a good kernel to start with since it only has one parameter that needs to be estimated and it provides flexible solutions. However, it is also important to know how to estimate the unknown parameters of a kernel function. In Section 4.3.3 in Chapter 4 we will address this issue.

3.3.4 Classification in Feature Space

Assume that the objects in the data are linearly separable in feature space that corresponds to a kernel function k . The dual optimization problem is then to find $\boldsymbol{\alpha}$ and β_0 , to

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \left[F_D(\boldsymbol{\alpha}) = \mathbf{1}'_n \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}' \mathbf{H} \boldsymbol{\alpha} \right], \\ \text{subject to: } \boldsymbol{\alpha} \geq 0, \boldsymbol{\alpha}' \mathbf{y} = 0, \end{aligned} \quad (3.44)$$

where $\mathbf{y} = (y_1, y_2, \dots, y_n)'$, $\mathbf{H} = [H_{ij}]$, and $H_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j k_{ij}$, $i, j = 1, 2, \dots, n$.

Suppose, $\hat{\boldsymbol{\alpha}}$ and $\hat{\beta}_0$ solve this problem, then

$$\mathbf{x}: \hat{\beta}_0 + \sum_{i \in \mathcal{S}^v} \hat{\alpha}_i y_i k(\mathbf{x}, \mathbf{x}_i) = 0 \quad (3.45)$$

is the optimal separating hyperplane in feature space corresponding to the kernel k . The discriminant function for SVM becomes:

$$\hat{f}_{SVM}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i \in \mathcal{S}^v} \hat{\alpha}_i y_i k(\mathbf{x}, \mathbf{x}_i). \quad (3.46)$$

Then, suppose that a new object, \mathbf{x}_0 has to be classified either into Π_1 or Π_2 , the classification rule for the SVM can now be written as:

Allocate \mathbf{x}_0 to Π_1 if

$$\text{sign}\{\hat{f}_{SVM}(\mathbf{x}_0)\} > 0,$$

otherwise allocate \mathbf{x}_0 to Π_2 .

(3.47)

3.3.5 The Multi-Class Case

To construct a multi-class SVM classifier, we need to consider all g classes, $\Pi_1, \Pi_2, \dots, \Pi_g$ simultaneously, and the classifier has to reduce to the binary SVM classifier if $g = 2$. The multi-class case for the nonlinear SVM falls outside the scope of this thesis, however, for a detailed discussion regarding the construction of a multi-class SVM classifier refer to Izenman (2008). We show the allocation rule for a multi-class SVM, as shown in Izenman, below:

Allocate \mathbf{x}_0 to Π_k if

$$\hat{f}_{SVM,k}(\mathbf{x}_0) = \max\{\hat{\beta}_{0,k} + \sum_l \hat{\beta}_{l,k} k(\mathbf{x}_l, \mathbf{x})\},$$

otherwise allocate \mathbf{x}_0 to Π_2 .

(3.48)

3.4 Performing SVM with R

3.4.1 The Kernlab Package

Even though the SVM is a recent development, latest versions of statistical software do include built-in programmes that are more than capable to perform SVM classifications, regressions and anomaly detection. Classification with the SVM will be carried out with the R package `kernlab`. The package provides the user with kernel functionality accompanied with other kernel based utility functions and kernel based algorithms (Hornik, Karatzoglou, Smola, 2004). In this section, the function `ksvm()` will be used to perform a SVM classification. The following is the `ksvm()` function with its basic arguments:

```
>ksvm(x, data = NULL, ..., subset, na.action = na.omit, scaled =
      TRUE)
```

or

```
>ksvm(x, y = NULL, scaled = TRUE, type = NULL,
      kernel = "rbfdot", kpar = "automatic",
      C = 1, nu = 0.2, epsilon = 0.1, prob.model = FALSE,
      class.weights = NULL, cross = 0, fit = TRUE, cache = 40,
      tol = 0.001, shrinking = TRUE, ...,
      subset, na.action = na.omit) #C is the cost parameter C
```

The function `ksvm()` requires several arguments that are necessary for a two-class classification. In Table 3.2 the vital arguments of the function are given accompanied with a short explanation for each argument.

Table 3.2: List of arguments for the `ksvm()` function in the `kernlab` R package

Arguments for <code>ksvm()</code>	Explanation
kernel	Specifying the kernel function to be used.
kpar	Setting the kernel parameter. If “automatic” is chosen, R will automatically choose the appropriate parameter to use.
C	Set the cost parameter.
cross	If an integer is specified a k-fold cross validation will be performed.
prob.model	If set to TRUE, R will build a model based on class probabilities.
type	Specify whether classification, regression or novelty detection should be performed.

3.4.2 Application in R

We will first illustrate the SVM classification procedure with the Iris data where the following code can be executed in R to perform the SVM classification. First, it is necessary to load the R package `kernlab` which is then followed by building the SVM model. The model is then constructed using the `ksvm()` function.

```
>library(kernlab)
>iris.SVM.model <- ksvm(iris.learn[, 1] ~ ., data = iris.learn[, -
  1],kernel = "rbfdot", kpar = "automatic", C = 1, cross =
  3,prob.model = T, type = "C-svc")
```

The same data as in Chapter 2 is analysed here. The Gaussian kernel is used (`rbfdot`), with the parameter γ estimated via cross-validation. Based on the model above, the classes of the test set can then be forecast with the function `predict()`. The test set was the same random sample of 30 objects as in Chapter 2.

```
>pred.class <- predict(iris.SVM.model, iris.test[, -1])
```

The classification results can be seen graphically in Figure 3.2 with *Sepal Length* displayed on the horizontal axis and *Sepal Width* on the vertical axis. The blue points represent the *Versicolor* class while the green points represent the *Virginica* class. The decision rule is shown as the dashed line. The average test error was estimated at 0.0513. The Iris data is therefore very well separated by the implementation of SVM. Comparing the results to LDA and QDA in Chapter 2, it can be seen that SVM error is slightly higher than the LDA and QDA test errors (0.0346 and 0.0356).

In Figure 3.3 the SVM classification results are shown graphically for the Haemophilia data. The red region represents the classification region for the first group of women, those who do not carry the Haemophilia gene, while the blue region represents the classification region for the second group of women, those who do carry the gene. The average test error was estimated at 0.1482. The SVM error is now slightly better than LDA and QDA, which achieved test errors of 0.1495 and 0.1517 respectively.

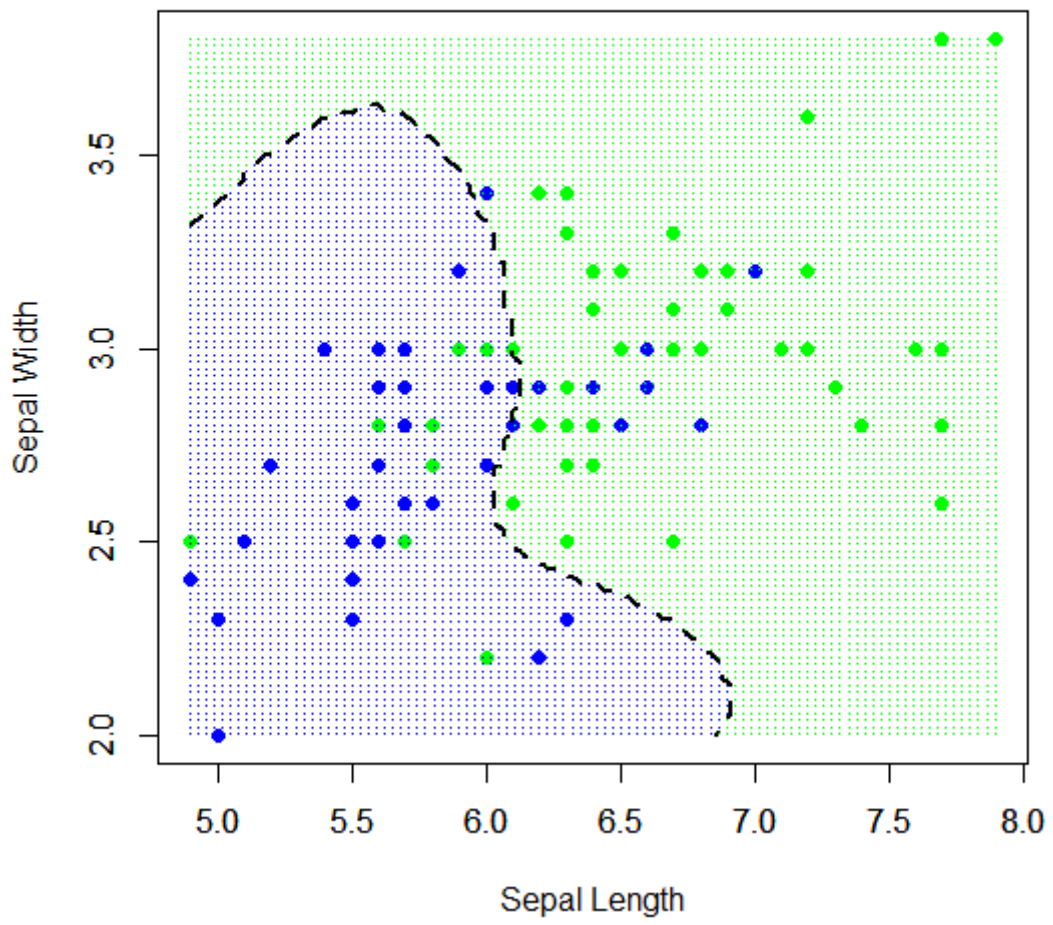


Figure 3.2: SVM classification with the Iris data

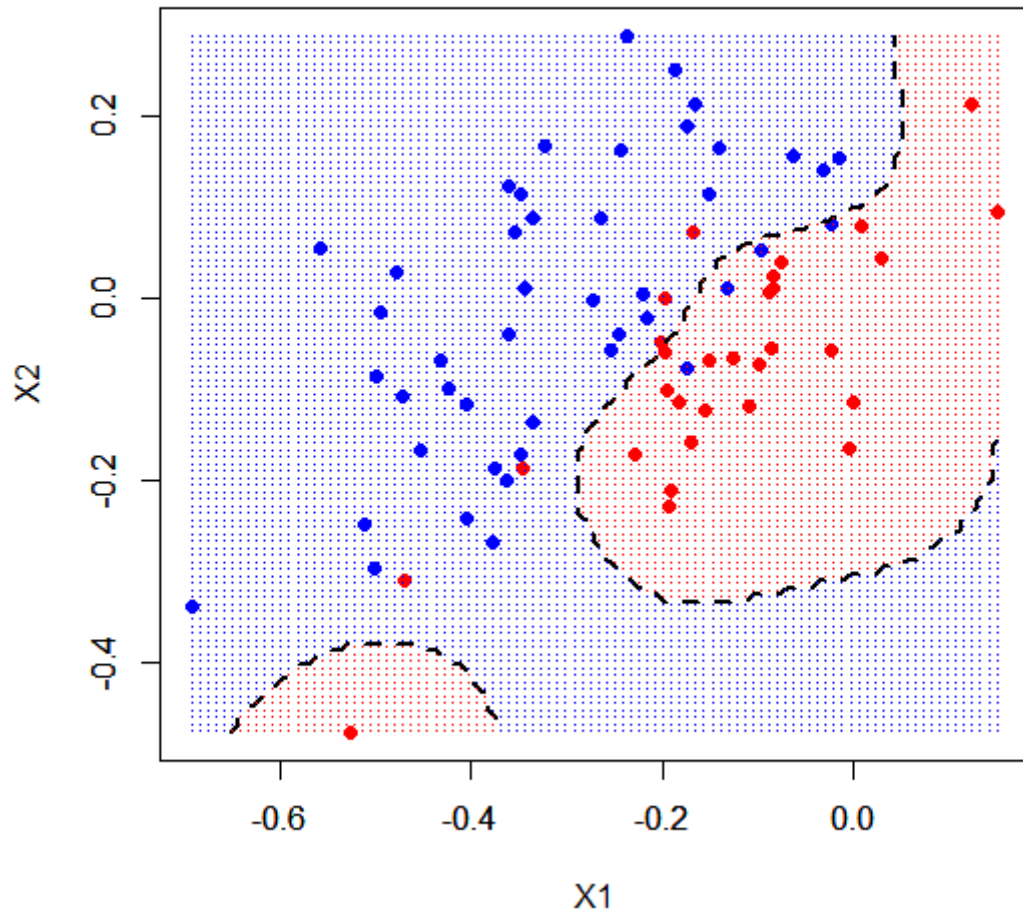


Figure 3.3: A SVM classification with the Haemophilia data

3.5 Conclusion

In this chapter we saw that the SVM may be very useful when the distribution of the data is not known. The SVM does not require the assumption of normality like LDA and QDA, therefore it can be applied to a wider range of classification problems. In Section 3.2 the linear and nonlinear separable cases of linear SVMs were discussed. This section also provided the foundation that was needed to extend the SVM to the nonlinear SVM. The nonlinear SVM was discussed in Section 3.3 which covered nonlinear transformations, the kernel trick and the kernel function. We saw that the nonlinear SVM had a computational problem when it tried to calculate the inner products in feature space. This is why the kernel trick was introduced since it allows us to calculate inner products in feature space by means of a nonlinear kernel function. However, SVMs are still computationally costly.

In Section 3.3.3 examples of kernel functions were listed together with their corresponding parameters. Choosing the appropriate kernel function can be difficult, however literature suggests that the Gaussian kernel function is a good kernel to start with. Estimating the parameters of the kernel function can also be a complex task when the kernel function has more than one parameter. In Section 3.3.5 a very brief discussion was given of the extension of SVM to the multi-class case.

In Section 3.4 an overview was given of the SVM functions in the R language and in Section 3.4.2 applications were done in R with the Iris and Haemophilia data sets. We saw that SVM achieved an average test error of 0.0513 with the Iris data and 0.1482 with the Haemophilia data. In both instances SVM performed similar to LDA and QDA. In the next chapter we will introduce another kernel based classification procedure which has similar characteristics as the SVM.

CHAPTER 4

CLASSIFICATION WITH HYPERSPHERES

4.1 Introduction

In the previous chapters it was seen that classification is often performed by classifying an object into one of two (or more) classes. One example looked at was when a patient has to be classified as either a carrier of a certain gene or not such as in the case of the Haemophilia data. The other example was flowers which have to be classified as belonging to one of three flower species. A less well known classification problem also exists, and this is where there is only one class which objects can be classified into. This is known in the machine learning field as a domain description problem or one-class classification. In domain description the assignment is not to discriminate between classes of objects, but to give a description of a set of objects similar to a confidence region. The description of a set of points is also called a support region.

Some support region estimation methods already exist. However, these methods usually assume that the data have some underlying probability distribution. In this chapter hyperspheres are introduced as a method for estimating support regions and it will be seen that hyperspheres do not require a known probability distribution of a data set. Hyperspheres are not only used for support region estimation, but can also be used for multi-class classification. The method of estimating support regions using hyperspheres was first introduced by David Tax and Robert Duin (1999) and was inspired by Vladimir Vapnik (1995).

In this chapter, two techniques that implement hyperspheres for classification will be viewed. The Smallest Enclosing Hypersphere (SEH), which can be used for support region estimation and one-class classification, and Nearest Hypersphere Classification (NHC), which can be used for multi-class classification. The theory of using hyperspheres for classification will be discussed in Section 4.2 while the applications of SEH in \mathbb{R} will be discussed in Section 4.2.3. In Section 4.3, the theory on NHC and the applications of NHC in \mathbb{R} will be discussed. We also look at parameter estimation through cross-validation using a grid search in Section 4.3.3. This will be followed by a short summary regarding aspects of hyperspheres in Section 4.4 and a conclusion in Section 4.5.

4.2 The Hypersphere

The theory on hyperspheres is discussed in Tax and Duin (1999), Tax (2001), Shawe-Taylor and Cristianini (2004) as well as in Lamont (2008). References to these sources will be made throughout the remainder of the chapter. Two solutions for the hypersphere will be discussed, the hard-margin solution which is also called the Smallest Enclosing Hypersphere and the soft-margin solution or the ν -soft hypersphere. The hard-margin solution will result in a support region that will cover an area including all objects whereas the soft-margin solution will result in a support region that will cover an area including objects that belong to a certain class, but also allows for outliers or objects that do not belong to that class to fall outside the support region. The latter is known as outlier detection where certain objects differ significantly from the rest.

4.2.1 The Hard-Margin Solution

Consider a single group of objects, $\Phi(\mathbf{x}_i), i = 1, \dots, n$, in feature space, \mathcal{F} . A hypersphere fitted around these objects which is large enough to include all the objects, but also the smallest possible hypersphere (by the radius) is called the SEH. Such a hypersphere can be defined by a centre (\mathbf{c}) and a radius ($r = \|\Phi(\mathbf{x}_i) - \mathbf{c}\|$), where $\Phi(\mathbf{x}_i)$ is the point furthest away from the centre, but on the surface of the hypersphere.

For a given data set we can find the centre of the sphere as follows. Let \mathbf{c}^* be defined by

$$\mathbf{c}^* = \arg \min_{\mathbf{c}} \left[\max_{1 \leq i \leq n} \|\Phi(\mathbf{x}_i) - \mathbf{c}\| \right], \quad (4.1)$$

where \mathbf{c}^* is the centre of the hypersphere which has the smallest possible radius. The mapping function, Φ , is unknown and therefore finding \mathbf{c}^* in (4.1) is impossible. Tax and Duin (1999) gives a possible solution to this problem. They argue that constructing the hypersphere in feature space is equivalent to solving the quadratic optimization problem shown below:

$$\begin{aligned} & \min_{\mathbf{c}, r} [r^2] \\ & \text{subject to: } \|\Phi(\mathbf{x}_i) - \mathbf{c}\|^2 \leq r^2, \text{ for } i = 1, 2, \dots, n, \end{aligned} \quad (4.2)$$

where r is the radius of the hypersphere. It should be noted that minimizing r^2 is equivalent to minimizing r . By introducing Lagrangian multipliers ($\alpha_i \geq 0$), the optimization problem above can be solved by defining a primal function:

$$\begin{aligned}
 F_P(\mathbf{c}, r, \boldsymbol{\alpha}) &= r^2 + \sum_{i=1}^n \alpha_i [\|\Phi(\mathbf{x}_i) - \mathbf{c}\|^2 - r^2] \\
 &= r^2 + \sum_{i=1}^n \alpha_i [\langle \Phi(\mathbf{x}_i) - \mathbf{c}, \Phi(\mathbf{x}_i) - \mathbf{c} \rangle - r^2] \\
 &= r^2 + \sum_{i=1}^n \alpha_i [\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2\langle \mathbf{c}, \Phi(\mathbf{x}_i) \rangle + \langle \mathbf{c}, \mathbf{c} \rangle] - r^2 \sum_{i=1}^n \alpha_i.
 \end{aligned} \tag{4.3}$$

This function is then minimized with respect to the primal variables, \mathbf{c} and r , and maximized with respect to the Lagrangian multipliers, α_i .

Taking partial derivatives with respect to \mathbf{c} and r and setting them equal to zero, gives

$$\frac{\partial F_P}{\partial \mathbf{c}} = 2 \sum_{i=1}^n \alpha_i (\Phi(\mathbf{x}_i) - \mathbf{c}) = \mathbf{0}, \tag{4.4}$$

and

$$\frac{\partial F_P}{\partial r} = 2r \left(1 - \sum_{i=1}^n \alpha_i \right) = 0. \tag{4.5}$$

From (4.5) it can be seen that $\sum_{i=1}^n \alpha_i = 1$, and therefore from (4.4) we obtain

$$\mathbf{c} = \sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i). \tag{4.6}$$

In Figure 4.1 the hard-margin solution is schematically illustrated in feature space. The hypersphere is shown as the dashed line along with its centre \mathbf{c} and radius r . All the objects $\Phi(\mathbf{x})$ are inside the hypersphere. One object is on the surface of the hypersphere and is indicated by $\Phi(\mathbf{x})^*$. We will later see that the point(s) that lie on the surface of the hypersphere are also called support vectors.

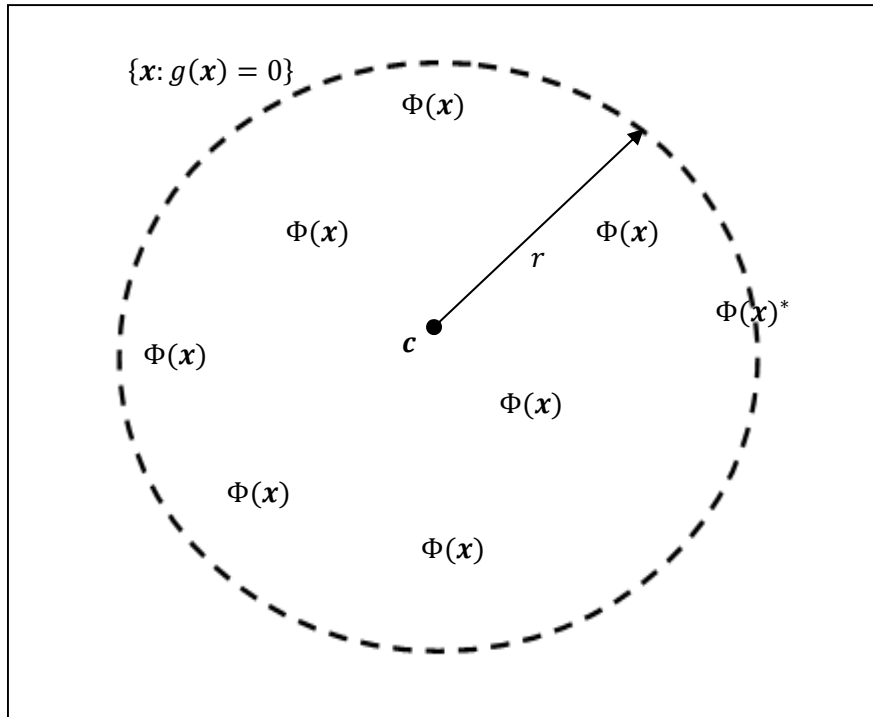


Figure 4.1: Schematic illustration of the hard-margin hypersphere in feature space

Substituting these results into the primal function gives the dual formulation of the function:

$$\begin{aligned}
 F_D(\mathbf{c}, r, \boldsymbol{\alpha}) &= r^2 + \sum_{i=1}^n \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2 \sum_{j=1}^n \alpha_j \langle \sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\
 &\quad + \sum_{i=1}^n \alpha_i \langle \sum_{k=1}^n \alpha_k \Phi(\mathbf{x}_k), \sum_{j=1}^n \alpha_j \Phi(\mathbf{x}_j) \rangle - r^2 \sum_{i=1}^n \alpha_i \\
 &= r^2 + \sum_{i=1}^n \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2 \sum_{j=1}^n \alpha_j \langle \sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\
 &\quad + \langle \sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i), \sum_{j=1}^n \alpha_j \Phi(\mathbf{x}_j) \rangle - r^2 \\
 &= \sum_{i=1}^n \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2 \sum_{i,j=1}^n \alpha_i \alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle + \sum_{i,j=1}^n \alpha_i \alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\
 &= \sum_{i=1}^n \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - \sum_{i,j=1}^n \alpha_i \alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle. \tag{4.7}
 \end{aligned}$$

By replacing the inner products in (4.7) with kernel functions, the optimal Lagrangian multipliers, α_i , can be determined by solving:

$$\begin{aligned} & \max_{\alpha} \left[\sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right] \\ & \text{subject to: } \sum_{i=1}^n \alpha_i = 1 \text{ and } \alpha_i \geq 0 \text{ for } i = 1, 2, \dots, n. \end{aligned} \quad (4.8)$$

The solution for the optimal values $(\alpha_1^*, \alpha_2^*, \dots, \alpha_n^*)$ can be found by using a quadratic programming solver. Once the values $(\alpha_1^*, \alpha_2^*, \dots, \alpha_n^*)$ are solved, the radius and the centre can be determined.

It can be shown, using the Karush-Kuhn-Tucker conditions (Karush, 1939; Kuhn and Tucker, 1951), that only the objects that lie on the surface of the hypersphere have non-zero optimal values, that is, $\alpha_i^* > 0$. The remaining objects lying within the sphere have $\alpha_i^* = 0$. Only the objects with non-zero α_i^* are needed in the construction of the hypersphere and these objects are called the support vectors. Therefore, using any of the support vectors denoted by \mathbf{x}_i , the radius can be calculated as

$$\begin{aligned} r &= \|\Phi(\mathbf{x}_i) - \mathbf{c}\| \\ &= \left\| \Phi(\mathbf{x}_i) - \sum_{j=1}^n \alpha_j^* \Phi(\mathbf{x}_j) \right\| \\ &= \sqrt{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2 \sum_{j=1}^n \alpha_j^* \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle + \sum_{k,j=1}^n \alpha_k^* \alpha_j^* \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_j) \rangle}. \end{aligned} \quad (4.9)$$

We have seen in Chapter 3 that the inner products can be replaced by kernel functions. Therefore, by substituting the inner products with kernel functions, Equation (4.9) becomes

$$r^* = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) - 2 \sum_{j=1}^n \alpha_j^* k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{k,j=1}^n \alpha_k^* \alpha_j^* k(\mathbf{x}_k, \mathbf{x}_j)}. \quad (4.10)$$

Also, by replacing α_i with their corresponding optimal values, α_i^* , the centre of the hypersphere can be written as

$$\mathbf{c}^* = \sum_{i=1}^n \alpha_i^* \Phi(\mathbf{x}_i). \quad (4.11)$$

In Lamont (2008) it is mentioned that the SEH in feature space corresponds to a support region in input space. Therefore, radius (r^*), centre (\mathbf{c}^*), and any support vector (\mathbf{x}_i) for a given data set, can be used to construct a support region $g(\mathbf{x})$ in input space, i.e.

$$\{\mathbf{x} \in \mathfrak{R}^p: g(\mathbf{x}) \leq 0\}, \quad (4.12)$$

where

$$\begin{aligned} g(\mathbf{x}) &= \|\Phi(\mathbf{x}) - \mathbf{c}^*\|^2 - r^{*2} \\ &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle - 2 \sum_{j=1}^n \alpha_j^* \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_j) \rangle + \sum_{k,j=1}^n \alpha_k^* \alpha_j^* \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_j) \rangle - r^{*2} \\ &= k(\mathbf{x}, \mathbf{x}) - 2 \sum_{j=1}^n \alpha_j^* k(\mathbf{x}, \mathbf{x}_j) + \sum_{k,j=1}^n \alpha_k^* \alpha_j^* k(\mathbf{x}_k, \mathbf{x}_j) - r^{*2} \\ &= k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}_i, \mathbf{x}_i) + 2 \sum_{j=1}^n \alpha_j^* [k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}, \mathbf{x}_j)]. \end{aligned} \quad (4.13)$$

This solution is known as the hard-margin solution or SEH. Figures 4.3 and 4.5 contain examples of support regions obtained by using the SEH.

4.2.2 The Soft-Margin Solution

We can extend the hard-margin solution to the soft-margin solution by implementing a slack variable, ξ_i (Tax and Duin, 1999). The usage of a slack variable was introduced in Chapter 3 in Section 3.2.2. A hypersphere estimated with the soft-margin solution can be obtained by solving the quadratic optimization problem:

$$\begin{aligned} &\min_{c,r} \left[r^2 + C \sum_{i=1}^n \xi_i \right] \\ &\text{subject to: } \|\Phi(\mathbf{x}_i) - \mathbf{c}\|^2 \leq r^2 + \xi_i, \text{ and } \xi_i \geq 0, i = 1, 2, \dots, n. \end{aligned} \quad (4.14)$$

The slack variable, $\xi_i \geq 0$, is included in the formulation to penalize outlying cases and the parameter $C > 1/n$ controls the trade-off between the volume of the hypersphere and the number of outliers.

Formulating this problem (4.14) as a Lagrangian and replacing all inner products in this formulation with a kernel function leads to the following optimization problem:

$$\begin{aligned} & \max_{\alpha} \left[\sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right] \\ & \text{subject to: } \sum_{i=1}^n \alpha_i = 1 \text{ and } 0 \leq \alpha_i \leq C \text{ for } i = 1, 2, \dots, n. \end{aligned} \quad (4.15)$$

The optimal values, α_i^* , can be found by using a quadratic programming solver. The support vectors are the \mathbf{x}_i 's for which $i \in \{i: 0 < \alpha_i^* \leq C\}$.

The support region for the soft-margin solution can then be estimated by

$$\{\mathbf{x} \in \mathfrak{R}^p: g(\mathbf{x}) \leq 0\}, \quad (4.16)$$

where

$$g(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}_i, \mathbf{x}_i) + 2 \sum_{j=1}^n \alpha_j^* [k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}, \mathbf{x}_j)] \quad (4.17)$$

and \mathbf{x}_i is any support vector and \mathbf{x}_j is the j^{th} data object.

The soft-margin solution is accompanied by an outlier detector which can be defined for any object \mathbf{x}_i satisfying the inequality in:

$$\begin{aligned} \Psi(\mathbf{x}_i) &= I[\|\Phi(\mathbf{x}_i) - \mathbf{c}\|^2 > r^{*2}] \\ &= I \left[k(\mathbf{x}_i, \mathbf{x}_i) - 2 \sum_{j=1}^n \alpha_j^* k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{k,j=1}^n \alpha_k^* \alpha_j^* k(\mathbf{x}_k, \mathbf{x}_j) > r^{*2} \right] \end{aligned} \quad (4.18)$$

where $\Psi(\mathbf{x})$ is an index function and outputs 1 for outliers and 0 otherwise for $i = 1, \dots, n$.

In Figure 4.2 the soft-margin solution is illustrated schematically in feature space. The hypersphere is shown by the dashed line and the outliers, each accompanied by a corresponding slack variable ξ_i , is shown by $\Phi(\mathbf{x})$.

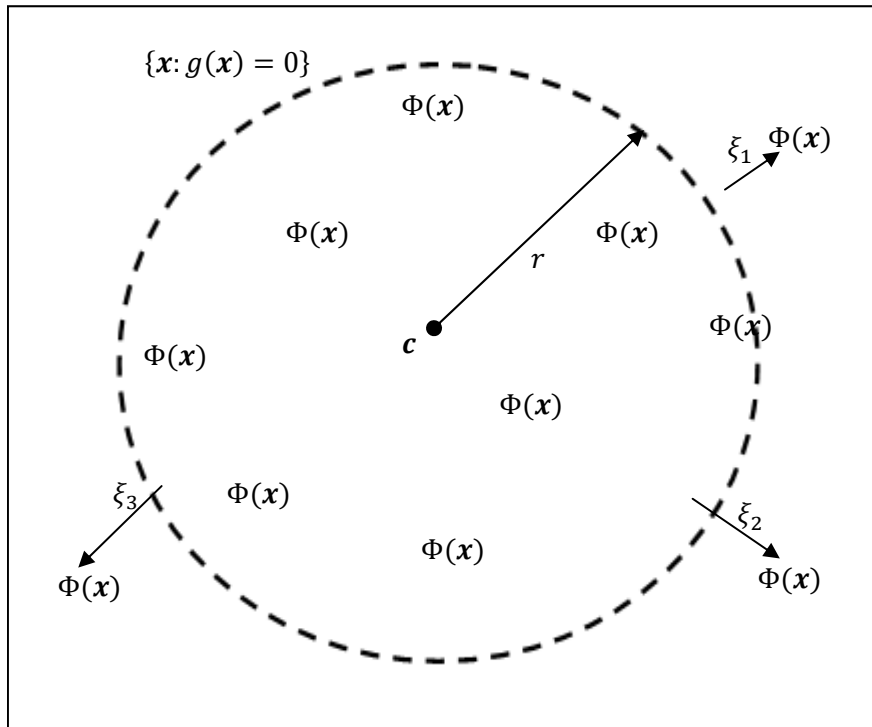


Figure 4.2: Schematic illustration of the soft-margin hypersphere in feature space

4.2.3 Applications of the Smallest Enclosing Hyperspheres in R

4.2.3.1 Software

Code was written in R for this chapter to perform SEH. In Chapter 3 the R package `kernlab` was introduced and will also be used in this section to aid in the application of the SEH. Two main functions that will be used from the `kernlab` package are the functions `rbfdot()`, which is a kernel function, and `ipop()`, which is a quadratic optimization solver.

The `rbfdot()` function, which is the Gaussian kernel function, is one of many kernel functions in `kernlab`. Other examples of kernel functions in `kernlab` include the d^{th} degree polynomial, Laplacian kernels, etc. These are listed in Table 3.1. The SEH will be illustrated with the Gaussian kernel function due to its popularity and less complicated nature. The kernel function is displayed again below,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\} \quad i, j = 1, \dots, n.$$

The optimal Lagrangian multiplier, α_i^* , can be estimated with the `ipop()` function. The function is executed in R as follows:

```
>ipop(c, H, A, b, l, u, r)
```

This function is used to solve the quadratic optimization programming problem of the form

$$\min \left[\mathbf{c}'\mathbf{x} + \frac{1}{2}\mathbf{x}'\mathbf{H}\mathbf{x} \right]$$

$$\text{subject to: } \mathbf{b} \leq \mathbf{A}\mathbf{x} \leq \mathbf{b} + \mathbf{r} \quad \text{and} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},$$

where \mathbf{b} and \mathbf{r} are both unit vectors defining the constraints, and \mathbf{l} and \mathbf{u} are the lower and upper bound vectors. \mathbf{A} is a matrix which defines the constraints under which the function is optimized. In the context of the SEH, the following will be used:

- \mathbf{c} contains elements $k(\mathbf{x}_i, \mathbf{x}_i)$ for $\{i = 1, 2, \dots, n\}$, that is a $n \times 1$ vector consisting of ones if the Gaussian kernel function is used.
- \mathbf{H} contains elements $k(\mathbf{x}_i, \mathbf{x}_j)$ for $\{i, j = 1, 2, \dots, n\}$ and is also known as the $n \times n$ Gram matrix.
- \mathbf{b} will be a vector consisting of the value of one, and \mathbf{r} will be a vector consisting of the value of zero. \mathbf{A} will be a vector consisting of ones.
- \mathbf{u} will be a vector of ones and \mathbf{l} will be a vector of zeros.

4.2.3.2 Example using simulated data

Data generation

Three bivariate data sets were constructed to illustrate the support region from the SEH. The data sets of size $n = 100$ were generated as follows:

- a) The first data set was generated from a standard normal distribution,

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right).$$

- b) The second data set was generated from a normal distribution with parameters

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix} \right).$$

c) The third data set was generated from a lognormal distribution with parameters

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \log N \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right).$$

The R code for generating these three data sets is given in the appendix.

Coding in R

To find the optimal values of α_i^* , several steps are required to construct all the necessary arguments for `ipop()` that was mentioned in Section 4.2.3.1. The function below was written in order to determine the optimal values α_i^* , as well as to determine the value of r^{*2} :

```
function(data, gamma)
{
  data <- as.matrix(data)
  n <- nrow(data)
  p <- ncol(data)

  kernelrbf <- rbfdot(sigma=gamma)          ## gamma is the parameter
  Gram <- kernelMatrix(rbfdot(gamma), data) ##gamma is the parameter
  kii <- matrix(diag(Gram), nrow=n)        ##kii represents  $k(\mathbf{x}_i, \mathbf{x}_i)$ 
  A <- t(matrix(1, nrow=n))
  b <- 1
  r <- 0
  u <- matrix(1, nrow=n)
  l <- matrix(0, nrow=n)
  alphas <- primal(ipop(c=kii, H=Gram, A, b, l, u, r))
  alphas.vec <- as.matrix(alphas, ncol=1)
```

```

ind <- zapsmall(alphas)>0.00001

sv <- data[ind,][1,]

nr.sv <- nrow(data[ind,])

svmat <- matrix(rep(sv,n),byrow=T,ncol=p)

k.sv.j <- diag(kernelMatrix(kernelrbf,data,svmat))

      rstar<-sqrt(kernelrbf(sv,sv)-2*t(alphas.vec)%*%k.sv.j+
      t(alphas.vec)%*%Gram%*%alphas.vec)  ##rstar represents r*

list(Gram=Gram,alphas=alphas.vec,index=ind,sv=sv,rstar=rstar,
      nr.sv=nr.sv)
}

```

The `primal()` function is used to extract the primal solution for the optimal values $(\alpha_1^*, \alpha_2^*, \dots, \alpha_n^*)$ from the `ipop()` function.

The code was executed for the three data sets above. The parameter $\gamma = 0.1$ was used for all three data sets. The resulting support regions are shown in Figure 4.3. The aim of these plots is to show the flexibility of the SEH when it comes to estimating the support regions. The support regions follow the distributions of the data sets very well except for the lognormal distribution. A different value for γ should be used in this case which will change the shape of the region. All the objects in each plot either fall inside the support region or on the boundary.

The points on the boundaries represent the objects with $\alpha_i^* > 0$. These points are the support vectors mentioned in Section 4.2.1. Figure 4.4 contains index plots of the support vectors for each of the three data sets. The spikes represent the objects with $\alpha_i^* > 0$, and the number of support vectors is clear from this. It can be seen that the standard normal data has 7 support vectors, the normal data has 5 support vectors and the lognormal data has 13 support vectors. Only the objects which are support vectors are needed in the estimation of the support regions. The more the distribution of the data deviates from a circle or ellipse, the more support vectors are needed to describe the region.

Choosing the appropriate kernel function along with its parameters to estimate support regions is vital, since it affects the shape of the support region and the number of support vectors. In this section the Gaussian kernel function performed well. A unique relationship exists between the parameter of the Gaussian kernel, γ , and the number of support vectors. This will be investigated in Section 4.2.3.3.

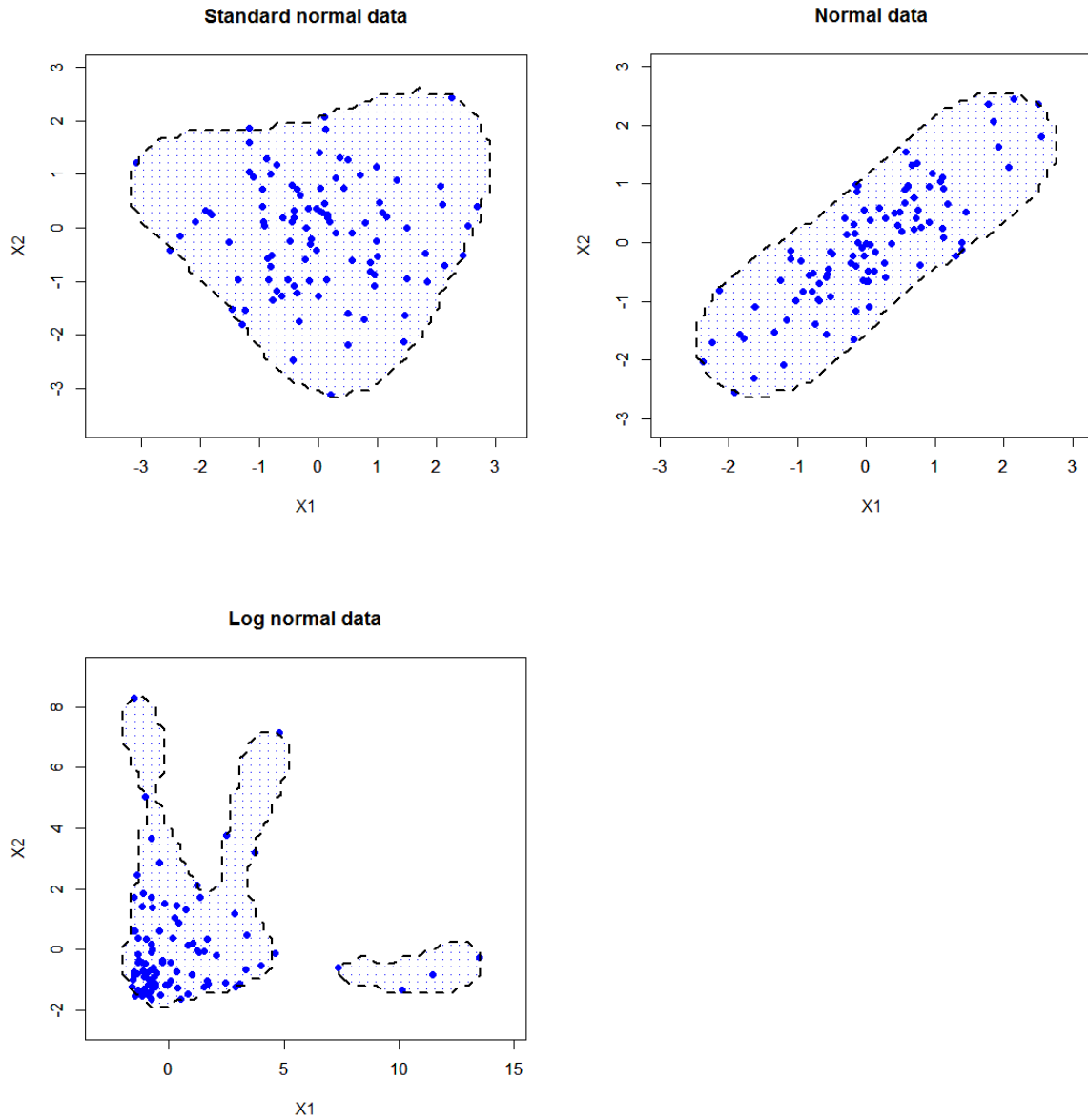


Figure 4.3: Three illustrative examples of the support regions in input space which correspond to smallest enclosing hyperspheres in feature space. The Gaussian kernel function was used with $\gamma = 0.1$ for each data set.

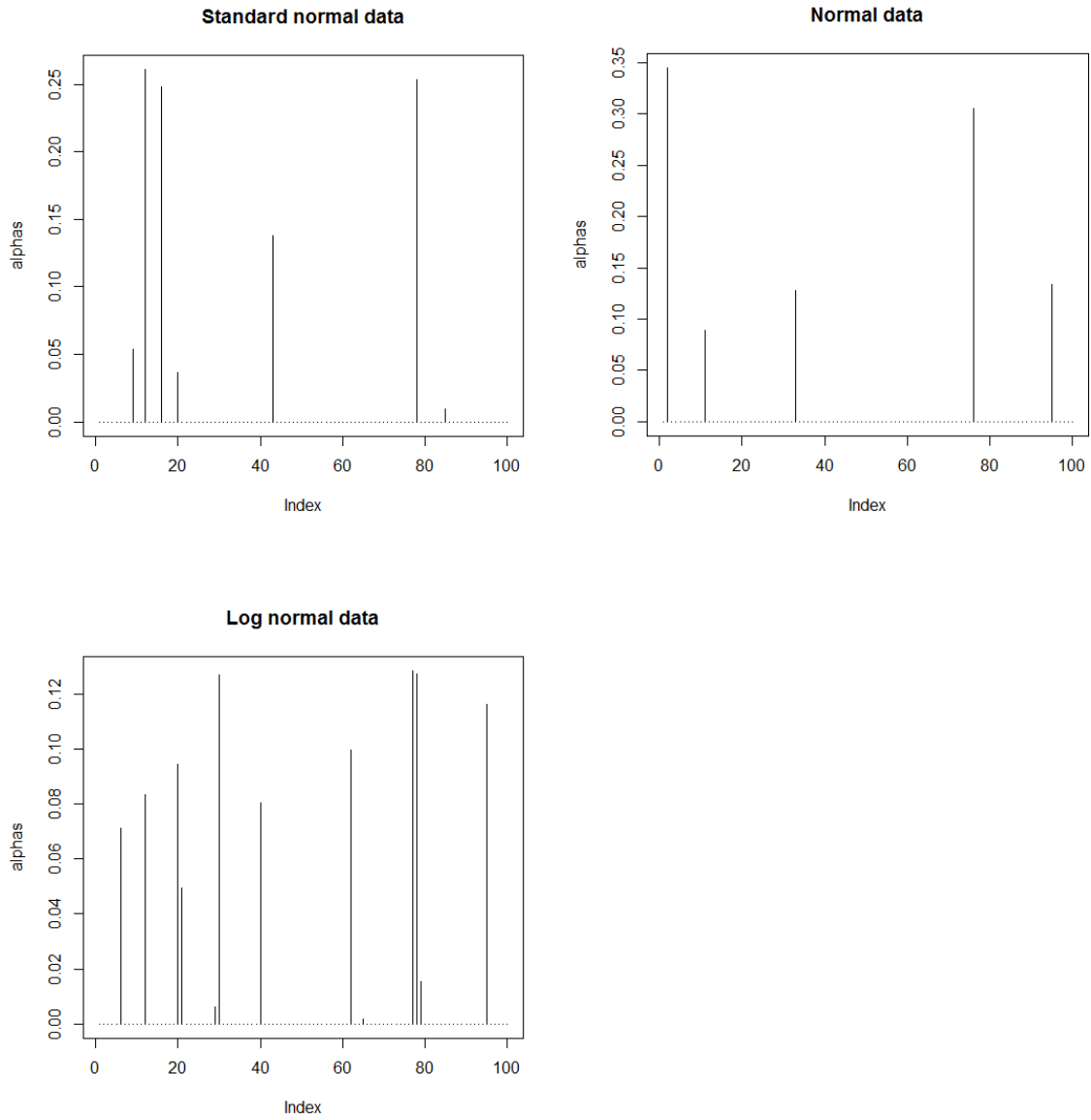


Figure 4.4: Index plots of the support vectors of the three data sets in Figure 4.3. The standard normal data has 7 support vectors, the normal data has 5 support vectors and the lognormal data has 13 support vectors.

4.2.3.3 The Relationship between the Number of Support Vectors and γ

In this section it will be demonstrated how the number of support vectors can be manipulated by changing the parameter γ for the Gaussian kernel. In Figure 4.3 the support regions were obtained by using $\gamma = 0.1$ and only a few support vectors were necessary to construct these support regions. To illustrate how the number of support vectors changes with changing the value for γ , the three data sets that were introduced in Section 4.2.3.2 will again be used, but now using $\gamma = 2$. The new support regions are shown in Figure 4.5. There is a significant change of shape of the support regions. The support regions for $\gamma = 2$ are not as smooth as the ones when $\gamma = 0.1$ was used. There is also a dramatic increase in the number of the support vectors. This can be seen in Figure 4.6 where the index plots are shown for the values α_i^* . For the standard normal data the number of support vectors increased from 7 to 52, for the normal data it increased from 5 to 35 and for the lognormal data it increased from 13 to 55.

In Figure 4.5 for the lognormal data it seems that certain points fall outside the support region, but by our definition of the SEH, no object can fall outside the boundaries of the support region. The boundaries for these points are just not visible. However, these points may be potential outliers and this can be determined by using the soft-margin solution hypersphere which makes provision for outlier detection (see Section 4.2.2).

It is thus clear that the number of support vectors increases when the value for γ is increased. The question now is, to what extent does γ have to be increased for all the objects in the data set to be considered as support vectors? This question is addressed in Figure 4.7. In Figure 4.7 the fraction of objects that are support vectors is plotted against a sequence of values for γ . We would expect that as the value for γ increases the fraction will approach one. For all three data sets it can be seen that the fraction of support vectors increases rapidly until γ reaches a value around 20. After this, the fraction stabilizes and at a very slow rate increases to 1. Even for $\gamma = 100$ the fraction is only 0.98 for the lognormal data. Only at $\gamma = 243$ did the fraction achieve a value of 1. Choosing the appropriate γ value for the SEH is still an open question. However, it is clear from our discussion that a smaller value may be preferred since it may give a better description of the support region.

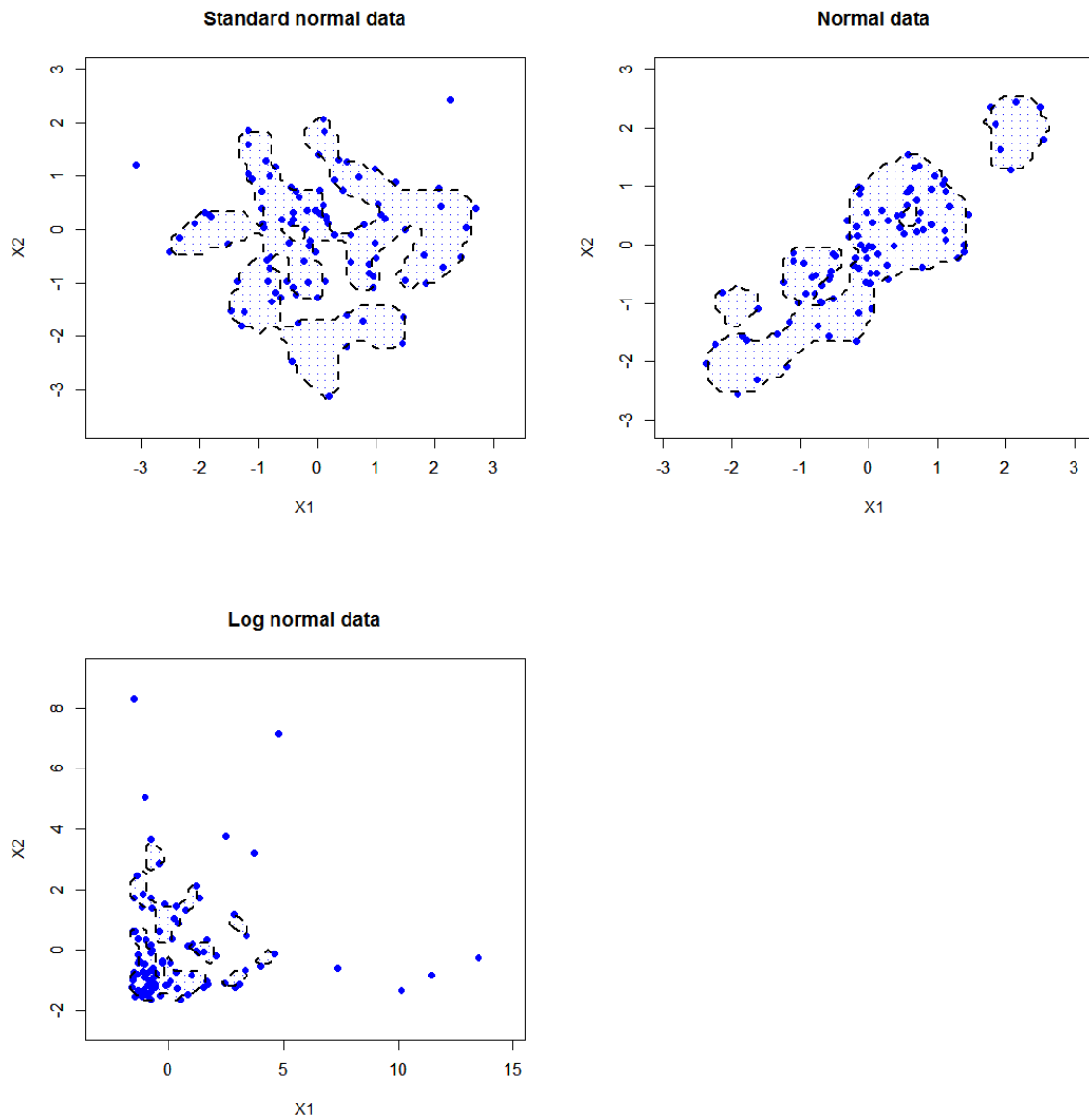


Figure 4.5: Three illustrative examples of the support regions in input space which correspond to enclosing hyperspheres in feature space. The Gaussian kernel function was used with $\gamma = 2$ for each data set.

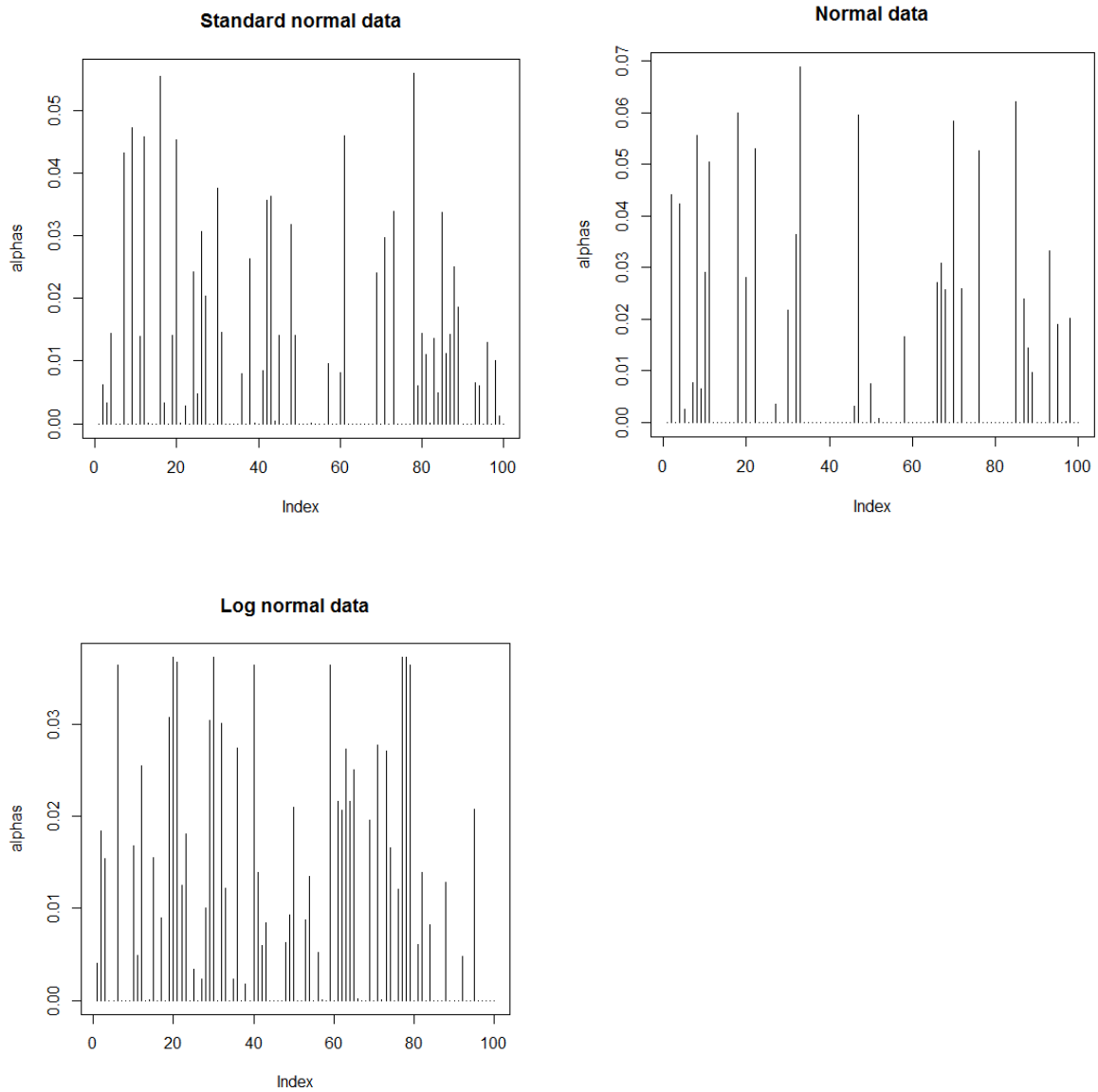


Figure 4.6: Index plots of the support vectors of the three data sets in Figure 4.5. The standard normal data has 52 support vectors, the normal data has 35 support vectors and the lognormal data has 55 support vectors.

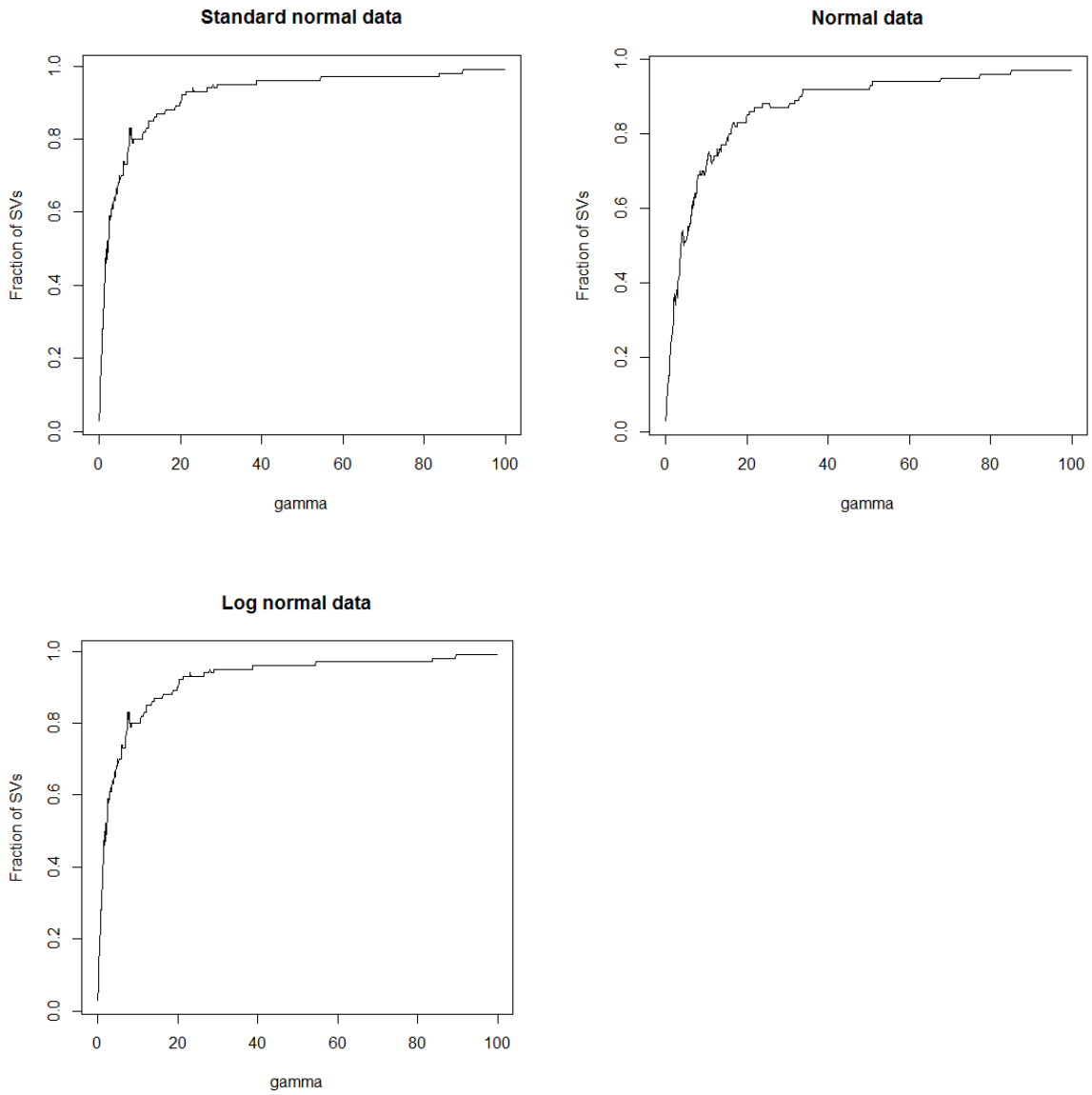


Figure 4.7: Relationship between the fraction of objects that are support vectors and the value for γ .

4.3 Nearest Hypersphere Classification

4.3.1 Theory of Nearest Hypersphere Classification

In Chapter 3 it was explained how the SVM uses an optimal separating hyperplane that maximizes the margin to perform classification. In this section the aim is to construct hyperspheres with minimum radii for sets of objects in effect to maximize the distance to other hyperspheres. This method is called Nearest Hypersphere Classification (NHC). The distinction between SVM and NHC is illustrated in Figure 4.8 for the two-class case. Both classes have to be fitted with a hypersphere, where c_j and r_j are the respective centres and radii for $j = 1, 2$. The first class has a larger variance compared to the second class. The result is that the hypersphere for the first class will have a larger radius than the hypersphere for the second class. The optimal separating hyperplane is shown as the dotted line between the classes and will be used to classify objects when using SVM. However, when using NHC for classification, NHC works by classifying a new object into the class with the smallest distance between the object and the centre of a nearby hypersphere. These distances can be calculated with dissimilarity functions.

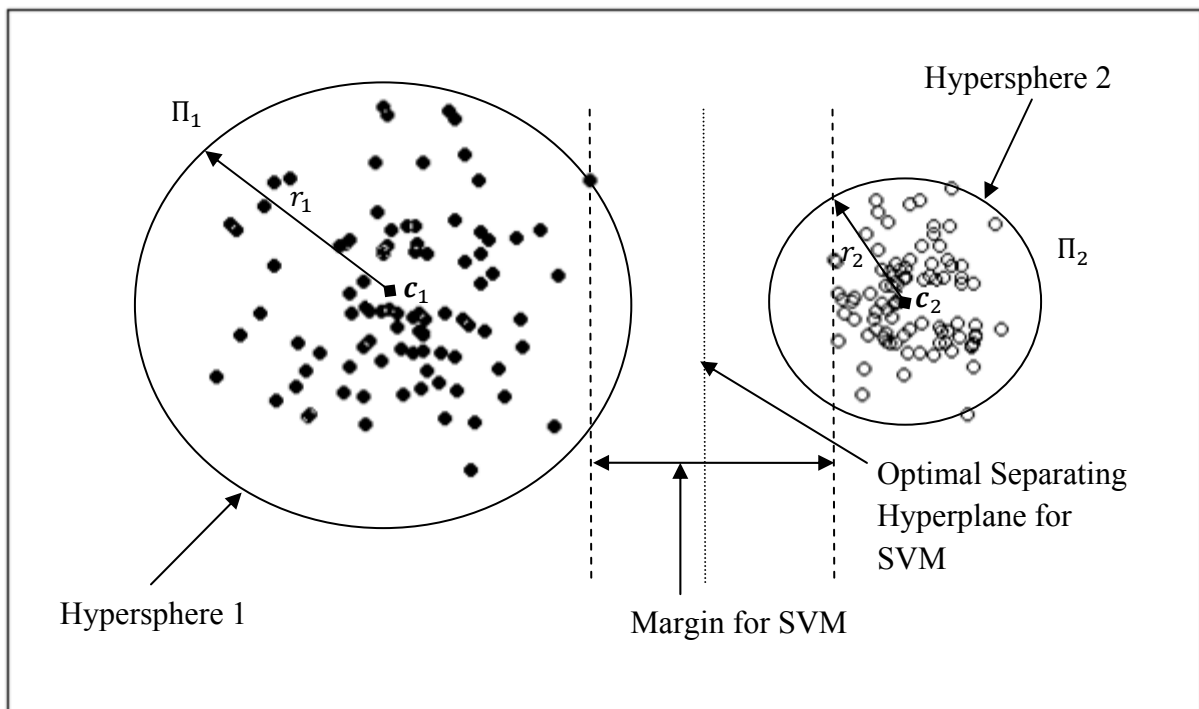


Figure 4.8: Illustrating the difference between NHC and SVM

In general the NHC can be explained as follows. Let $\Pi_1, \Pi_2, \dots, \Pi_g$ represent g classes. Denote a set of indices corresponding to the objects from Π_j by $J_j = \{1, 2, \dots, n_j\}$. Let $\delta_j(\mathbf{x})$ denote the dissimilarity function which will be used for the distance between an object and the centre of a hypersphere for the j th class. The following is an example of such a function:

$$\delta_j(\mathbf{x}) = \|\Phi(\mathbf{x}) - \mathbf{c}_j^*\|, \quad j = 1, \dots, g. \quad (4.19)$$

Since there might be classes with different variances, the following can be used as a dissimilarity function,

$$\delta_j(\mathbf{x}) = \frac{\|\Phi(\mathbf{x}) - \mathbf{c}_j^*\|}{r_j^*}, \quad (4.20)$$

where r_j^* compensates for the difference in variances between the classes. Another example is $\delta_j(\mathbf{x}) = \frac{\|\Phi(\mathbf{x}) - \mathbf{c}_j^*\|}{r_j^{*\lambda}}$, with $\lambda \geq 0$. Hao *et al.* (2009) give more examples of dissimilarity functions that can be used for computing the distances.

For the multi-class classification problem let the following be the dissimilarity function:

$$\begin{aligned} \delta_j^2(\mathbf{x}) &= \frac{\|\Phi(\mathbf{x}) - \mathbf{c}_j^*\|^2}{r_j^{*2}} \\ &= \frac{1}{r_j^{*2}} \left\{ k(\mathbf{x}, \mathbf{x}) - 2 \sum_{k \in J_j} \alpha_k^* k(\mathbf{x}, \mathbf{x}_k) + \sum_{i, k \in J_j} \alpha_i^* \alpha_k^* k(\mathbf{x}_i, \mathbf{x}_k) \right\} \text{ for } j = 1, \dots, g. \end{aligned} \quad (4.21)$$

The classification rule for NHC is then very straightforward. Suppose we have a new object \mathbf{x}_0 that has to be classified. Then,

allocate \mathbf{x}_0 into Π_j for which

$$\delta_j^2(\mathbf{x}_0) = \min_{1 \leq i \leq g} [\delta_i^2(\mathbf{x}_0)]. \quad (4.22)$$

For a two-class classification problem, the NHC classification rule can be simplified as follows:

Classify \mathbf{x}_0 into Π_2 if

$$\text{sign}\{\hat{f}_{NHC}(\mathbf{x}_0)\} > 0,$$

otherwise classify \mathbf{x}_0 into Π_1 .

(4.23)

The function $\hat{f}_{NHC}(\mathbf{x}_0)$ can be written as

$$\begin{aligned} \hat{f}_{NHC}(\mathbf{x}_0) &= \frac{\|\Phi(\mathbf{x}_0) - \mathbf{c}_1^*\|^2}{r_1^{*2}} - \frac{\|\Phi(\mathbf{x}_0) - \mathbf{c}_2^*\|^2}{r_2^{*2}} \\ &= \frac{1}{r_1^{*2}} \left\{ k(\mathbf{x}_0, \mathbf{x}_0) - 2 \sum_{k \in J_1} \alpha_k^* k(\mathbf{x}_0, \mathbf{x}_k) + \sum_{i, k \in J_1} \alpha_i^* \alpha_k^* k(\mathbf{x}_i, \mathbf{x}_k) \right\} \\ &\quad - \frac{1}{r_2^{*2}} \left\{ k(\mathbf{x}_0, \mathbf{x}_0) - 2 \sum_{k \in J_2} \alpha_k^* k(\mathbf{x}_0, \mathbf{x}_k) + \sum_{i, k \in J_2} \alpha_i^* \alpha_k^* k(\mathbf{x}_i, \mathbf{x}_k) \right\}, \end{aligned} \quad (4.24)$$

$$\text{and } r_j^{*2} = k(\mathbf{x}_i, \mathbf{x}_i) - 2 \sum_{i, k \in J_1} \alpha_k^* k(\mathbf{x}_i, \mathbf{x}_k) + \sum_{i, k \in J_2} \alpha_i^* \alpha_k^* k(\mathbf{x}_i, \mathbf{x}_k) \quad (4.25)$$

for $j = 1, 2$, with \mathbf{x}_i any support vector from class j .

4.3.2 Application of NHC in R

NHC works by first fitting a separate hypersphere to each of the classes in a data set. Secondly, the distances between the objects that have to be classified and the centres of all the hyperspheres are calculated. An object is then classified to the class which has the smallest distance. The following R code was written to illustrate NHC for a two-class case.

```
function(learn.data, test.data, gamma) {
  kernelrbf <- rbfdot(sigma=gamma)

  #Determining groups and sizes
  test.data2 <- test.data

  test.data <- as.matrix(test.data)[, -1]

  g1 <- as.matrix(learn.data[learn.data[, 1]==-1, ])[, -1]
  g2 <- as.matrix(learn.data[learn.data[, 1]==+1, ])[, -1]

  n1 <- nrow(g1)
  n2 <- nrow(g2)
  nt <- nrow(test.data)

  #Determining the radius, alphas and Gram matrix for each group
  g1.calc <- calc.Hyp(g1, gamma=gamma)
  g2.calc <- calc.Hyp(g2, gamma=gamma)

  g1.Gram <- g1.calc$Gram
  g2.Gram <- g2.calc$Gram

  g1.alphas <- g1.calc$alphas
  g2.alphas <- g2.calc$alphas

  g1.r <- g1.calc$rstar
  g2.r <- g2.calc$rstar
}
```

```

#Determining the distances between test data and the two groups
and determining the signs

test.kii <-matrix(diag(kernelMatrix(kernelrbf,test.data)),nrow=nt)

#Group 1 distances with test data

g1.test.kj <- kernelMatrix(kernelrbf,test.data,g1)

g1.test.dist <- matrix(0,nrow=nt)

for(i in 1:nt){

g1.test.dist[i,] <- (test.kii[i,]-2*t(g1.alphas)%*%
g1.test.kj[i,] + t(g1.alphas)%*%g1.Gram%*%g1.alphas)/g1.r^2

}

#Group 2 distances with test data

g2.test.kj <- kernelMatrix(kernelrbf,test.data,g2)

g2.test.dist <- matrix(0,nrow=nt)

for(i in 1:nt){

g2.test.dist[i,] <- (test.kii[i,] -2*t(g2.alphas) %*%
g2.test.kj[i,]+t(g2.alphas)%*%g2.Gram%*%g2.alphas)/g2.r^2

}

#Determining the signs for test data and classifying them
accordingly

sign.test.vec <- matrix(0,nrow=nt)

for(i in 1:nt){

sign.test.vec[i,] <- sign(g1.test.dist[i,] - g2.test.dist[i,])

}

class <- ifelse(sign.test.vec<0,-1,1)

APER <- (nt-sum(class==test.data2[,1]))/nt

```

```
list(Classification=data.frame(Classified=ifelse(class==-1, "Group
1", "Group 2"), Actual.Membership=ifelse(test.data2[,1]==-1, "Group
1", "Group 2")), APER=APER)
}
```

NHC was performed on the *Versicolor* and *Virginica* classes of the Iris data with $\gamma = 0.1$. The results are shown in Figure 4.9. The species *Versicolor* is shown as the blue points and *Virginica* is shown as the green points. The dashed line represents the decision rule. NHC resulted in an average error rate of 0.0546 for the Iris data. In Figure 4.10 the results are shown for classification of the Haemophilia data set using a value of $\gamma = 8.3$. This resulted in an average error rate 0.2082. The obligatory group is represented by the blue points and the normal group is represented as the red points.

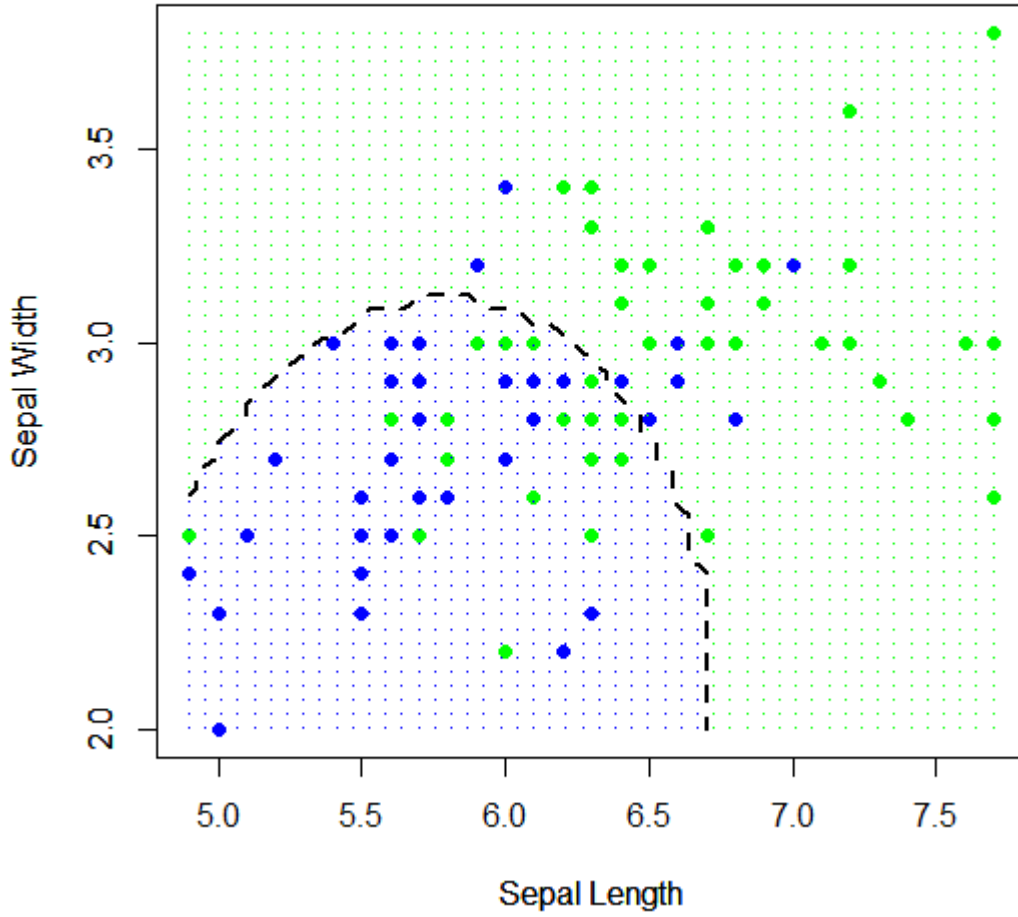


Figure 4.9: NHC classification regions for species *Versicolor* and *Virginica* ($\gamma = 0.1$).

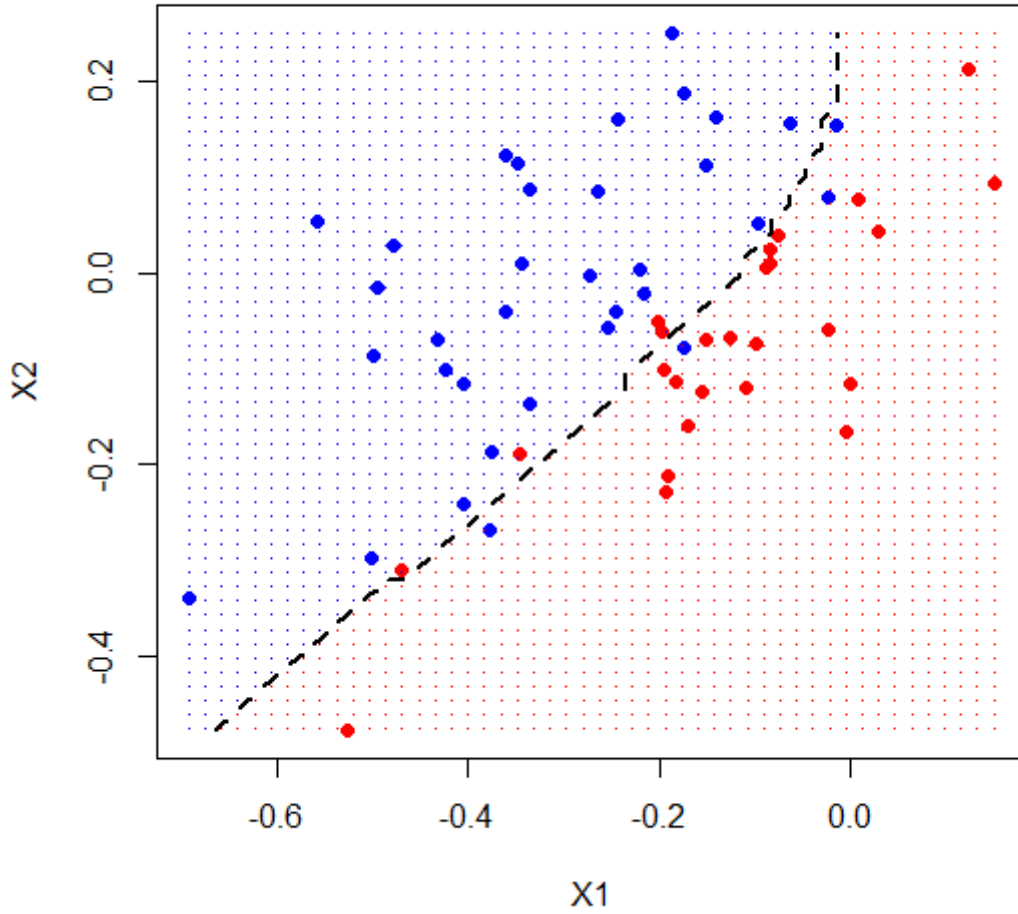


Figure 4.10: NHC classification regions for the Haemophilia data set ($\gamma = 8.3$).

4.3.3 Finding the Optimal Parameter through Cross-Validation

The parameters that are used to construct the hyperspheres in this chapter as well as the SVM in Chapter 3 are the optimal parameters. Optimal parameters are the preferred parameters that generally give the best classification results. In this section it will be shown how to estimate the optimal parameter for the Gaussian kernel function. The aim is to find the optimal γ so that we can most accurately predict the test data. Chang and Lin (2010) suggest using cross-validation to estimate the optimal value of γ for the Gaussian kernel function.

Cross-validation works by separating the data set into several parts, where one will be used for testing the classification model and the others for building the model. One version of this is called V -fold cross-validation. In V -fold cross-validation, the data set is divided into V subsets of approximately equal size. One of the V subsets is then used as a testing set while the remaining $V - 1$ subsets are used as learning sets. The error rate is then estimated by taking the average of the resulting V error rates which resulted from classifying all the V subsets of data. To find the optimal γ , Chang and Lin (2010) suggest using a grid-search and by using cross-validation. A sequence of values is constructed for γ and the one with the best cross-validation accuracy is chosen as the estimated optimal parameter. An example of the range of the sequence suggested by Chang and Lin is 2^{-15} to 2^{15} . The optimal parameters that were obtained for the Iris and Haemophilia data sets in this way were $\gamma = 0.1$ and $\gamma = 8.3$ respectively.

Using a grid-search to find the optimal parameter can be very time consuming especially when the data set is large. In such a case a coarse grid-search can be used. This is done by identifying a region on the grid where one might expect the best results will be delivered and then performing a finer grid-search in that region. However, there is one problem on how to identify that better region on the grid. Lamont (2008) provides a possible solution. According to Lamont the value $\gamma = 1/p$ generally leads to good results, where p is the number of variables in the data set. Therefore, we can choose this value as an initial starting place for our coarse grid-search.

4.3.4 An Example of Performing NHC with Three Classes

So far we have only executed the NHC with two classes. In this section we will perform NHC with the Iris data, including all three species, *Setosa*, *Versicolor* and *Virginica*. We use the initial starting value of $\gamma = 1/p = 1/4 = 0.25$ for the Gaussian kernel parameter. The results for this classification are shown in Figure 4.11. The decision rule is shown as the dashed line while the support region for *Setosa* is represented by the red region, *Versicolor* by the blue region and *Virginica* by the green region. The classification achieved an error rate of 10%.

We can now perform a coarse grid-search around the value of $\gamma = 0.25$ and try to improve on the error rate. The coarse grid-search resulted in an optimal parameter value of $\gamma = 0.1441$. We can now use this value to perform our final classification. The final resulting error rate is 2.22% which is a good improvement on the initial error rate of 10%.

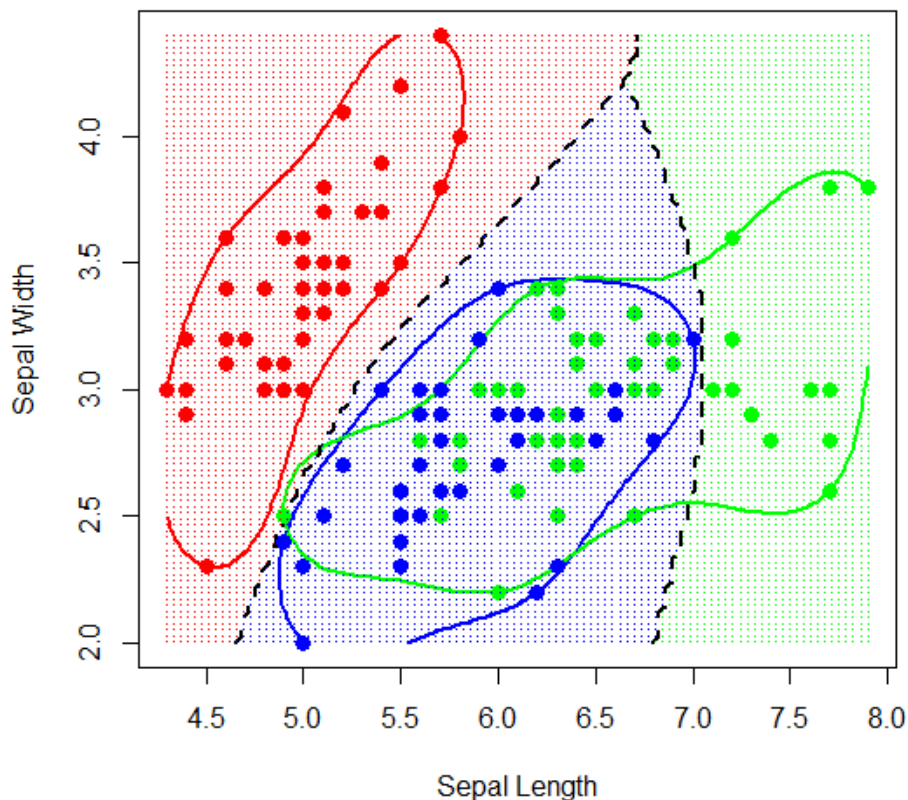


Figure 4.11: NHC with the Iris data set ($\gamma = 0.25$). The classification regions accompanied by the support regions are shown for each species.

4.4 Aspects of Nearest Hypersphere Classification

NHC works well compared to the other classification techniques discussed in the previous chapters. In Table 4.1 the classification results are summarised for LDA, QDA, SVM and NHC for the Iris data set and the Haemophilia data set. With the Iris data set, NHC achieved a test error of 0.0546 while SVM achieved a test error of 0.0513, LDA achieved 0.0346 and QDA achieved a test error of 0.0356. For the Haemophilia data set the SVM achieved a test error of 0.1482 outperforming NHC, LDA and QDA.

Table 4.1: Summary of the average error rates (over 100 repetitions) of LDA, QDA, SVM and NHC for the Iris and Haemophilia data sets

Classification Technique	Error Rate	
	Iris (<i>Versicolor</i> and <i>Virginica</i>)	Haemophilia
LDA	0.0346	0.1495
QDA	0.0356	0.1517
SVM	0.0513	0.1482
NHC	0.0546	0.2082

NHC is a non-parametric technique and it requires no assumption regarding the distribution of the underlying data. NHC is also a non-linear classifier which can be seen in Figure 4.9 and Figure 4.10. The NHC can also be easily extended to the multi-class case (see Figure 4.11). This is done by fitting hyperspheres to all the classes and calculating the distances between the centres of the hyperspheres and the objects that have to be classified. Another aspect of NHC is that only the support vectors are used to construct the hypersphere which gives NHC a computational advantage. It has also been suggested that NHC can be used for data sets where $p \gg n$ (Lamont, 2008). This can be especially helpful in studies where additional observations are very expensive.

4.5 Conclusion

In this chapter the hypersphere was introduced as a method for estimating the support region of a set of objects and also as a method for classifying objects. It was mentioned that the SEH derived from the hard-margin solution is used for estimating the support region. The SEH is a kernel-based method which means that a certain kernel function is used as a transformation function. The SEH was illustrated with the Gaussian kernel function in Section 4.2.3. The Gaussian kernel requires only one parameter (γ) which needs to be estimated. Three data sets were generated with different distributions, the standard normal, normal and lognormal, to illustrate the application of the SEH with the Gaussian kernel function for values 0.1 and 2 for γ . It was seen in Section 4.2.3.3 that for $\gamma = 0.1$ the support regions were smoother than for $\gamma = 2$ and also a smaller number of support vectors were needed to estimate the support region when $\gamma = 0.1$. In Section 4.2.3.4 the relationship between γ and the number of support vectors needed for estimation was investigated. It was found that for values up to 20 for γ the number of support vectors dramatically increased after which it stabilized. We also looked at the soft-margin solution of the hypersphere. It was seen that this hypersphere can be used for identifying outliers and this is known as anomaly detection in the machine learning field.

NHC was also introduced in this chapter as a method for classifying objects. The technique requires one parameter that needs to be estimated if the Gaussian kernel function is used. Cross-validation can be used to calculate the test error rate which is used to measure the accuracy of the classification model. It was seen that a grid search can be performed via cross-validation to estimate the optimal parameter of NHC. However, it was also seen that it could be time consuming to find the optimal parameter. Lamont (2008) suggests using $\gamma = 1/p$ as a starting value when seeking the optimal parameter. A coarse grid-search can then be performed around this initial value to help speed up computations.

In Section 4.4 we summarised aspects of NHC and also gave a brief comparison of LDA, QDA, SVM and NHC by evaluating the classification techniques by comparing the different error rates that were obtained by each technique for the Iris and Haemophilia data sets. In the next chapter the performance of NHC will be compared to that of LDA, QDA and SVM through extensive simulation studies and real-world data applications. We will look at different data scenarios and conclude under each scenario which classification technique is the best to employ.

CHAPTER 5

SIMULATION STUDIES AND REAL-WORLD DATA APPLICATIONS

5.1 Introduction

It is of utmost importance that a classification technique produces good classification results. This is especially true when the cost of misclassification is high, for example, a patient might be classified as being healthy when in fact he or she could be terminally ill. The cost of misclassification in this example could possibly be death. To assess the performance of the classification techniques in this thesis, a Monte Carlo simulation study will be used. Simulation studies give additional insight not only into the performance of a technique, but also under which circumstances a technique might perform better or worse. In this chapter, NHC will be compared with LDA, QDA and SVM by means of a simulation study to give additional insight into the data scenarios where the different techniques perform better or worse. Not only will the classification techniques be compared by means of a simulation study, but also through a real-world data application where we will look at real life classification problems and test whether LDA, QDA, SVM or NHC give the best results.

In Section 5.2.1 we will discuss the measures that are used to assess the performance within a simulation study. This will be followed by a brief explanation of the data that will be generated, that is the normal data and lognormal data, in Section 5.2.2. The relationship between the parameters of the two distributions will also be discussed in this section. In Section 5.2.3 a discussion will be given regarding the different data configurations which will be used for the generation of the data. In Section 5.2.4 the results of the simulation study will be given and discussed. The real-world data applications will be dealt with in Section 5.3, where two data sets will be used for classification by implementing LDA, QDA, SVM and NHC. Finally, a brief conclusion will be given in Section 5.4.

5.2 Simulation studies

5.2.1 Measuring Classification Performance

In this section the test error will be used for quantifying the classification performance of a classification model for the two-class case only. The test error will be used with Monte Carlo simulation studies while with the real-world data application the cross-validation method will be used. For the Monte Carlo simulation study a learning data set and a testing data set will be generated independently from the same distribution. For the real-world data application, cross-validation will be used to split the data set into a learning data set and a test data set.

5.2.1.1 Test Error

Assume that the learning data pairs (\mathbf{x}, y) , where $y \in \{\pm 1\}$ represents the response and $\mathbf{x} \in \mathcal{R}^p$ the input variables, are drawn independently from a joint distribution. Also, assume that a test data set of size n_T is drawn from the same distribution. The classifier is built on the learning data and then applied to the test data to calculate the test error. The test error, which is a measure of the classification performance, is defined by

$$R_{test} = \frac{1}{n_T} \sum_{i=1}^{n_T} I[\varphi(\mathbf{x}_i) \neq y_i], \quad (5.1)$$

where $\{(\mathbf{x}_i, y_i), i = 1, \dots, n_T\}$ is the test data and $\varphi(\mathbf{x}_i)$ is the predicted response from a classifier for \mathbf{x}_i . A low test error is desirable for a classification model.

5.2.1.2 Cross-Validation

Cross-validation is a well known technique for measuring the performance of a classifier when no test data is available. Such techniques include the leave-one-out, the V -fold cross-validation and the Monte Carlo cross-validation. Leave-one-out is the case where one object is left out and the classifier is built on the remaining objects. V -fold cross-validation works by randomly dividing the data into V non-overlapping groups of equal size. One group is then removed from the data and the classifier is built on the remaining $V - 1$ groups. The classifier is then used to predict the responses of the omitted group to estimate the cross-

validation error. This process is repeated V times, each time omitting a different group. The cross-validation error is then calculated by averaging the resulting V prediction errors. Monte Carlo cross-validation is where the data is randomly split into two parts, learning and test data. The classifier is built on the learning data and its performance is evaluated on the test data. This process is repeated multiple (say 100) times. In this thesis Monte Carlo cross-validation will be used.

5.2.2 Generating the data

In this chapter the training and test data sets for the simulation studies will be generated from multivariate normal and lognormal distributions. These distributions are specifically chosen to represent data scenarios where the normality assumption is both valid and invalid. The lognormal distribution will represent data where the normality assumption is not valid. When parametric techniques (LDA and QDA) are compared to non-parametric techniques (SVM and NHC), it is important to make the distinction between normal and non-normal data, because of the underlying assumption of normality that LDA and QDA require. For a more detailed outline regarding the different configurations of data that will be generated, refer to Section 5.3. In the following subsections the normal and lognormal distributions will be briefly explained as well as the relationship between them.

5.2.2.1 The Normal Distribution

The univariate normal distribution has a density function given by

$$f(x) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], -\infty < x < \infty, \quad (5.2)$$

where μ is the mean and σ is the standard deviation. This can be extended to the multivariate case which has a density function given by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \quad (5.3)$$

where $\mathbf{x}' = (x_1, x_2, \dots, x_p)$ is the observed vector of \mathbf{X} and $-\infty < x_i < \infty$, $i = 1, 2, \dots, p$. The density function is denoted by $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix. It is well known that the normal distribution is symmetrical around $\boldsymbol{\mu}$ and that it is bell shaped. In Figure 5.1, an example of a univariate normal density function with parameters $\mu = 1$ and $\sigma = 0.7$ is presented.

5.2.2.2 The Lognormal Distribution

In the univariate case, a stochastic variable $Y = e^X$ has a lognormal distribution if its natural logarithm, $X = \log(Y)$ has a normal distribution. The univariate lognormal distribution has a density function given by

$$f(y) = \frac{1}{(2\pi)^{1/2}\sigma y} \exp\left[-\frac{1}{2}\left(\frac{\log(y) - \mu}{\sigma}\right)^2\right], 0 < y < \infty, \quad (5.4)$$

where y is the observed value of Y . Let the lognormal distribution be denoted by $LN(m, s^2)$. The lognormal distribution has the properties that it is positively skewed and the variable varies from zero to infinity. It can be shown that a relationship exists between the parameters (μ, σ^2) of the normal distribution and the parameters (m, s^2) of the lognormal distribution. This relationship is explained by Johnson's translation system and will be discussed in Section 5.2.2.3.

The multivariate lognormal distribution will be denoted by $LN_p(\mathbf{m}, \mathbf{S})$ where \mathbf{m} is the $p \times 1$ mean vector and \mathbf{S} is the $p \times p$ covariance matrix and $\mathbf{Y}' = (Y_1, Y_2, \dots, Y_p)$ represents the p -dimensional random vector of lognormal variables. Figure 5.2 shows an example of a univariate lognormal density function with parameters $s = 0.3$ and $m = 0.5$.

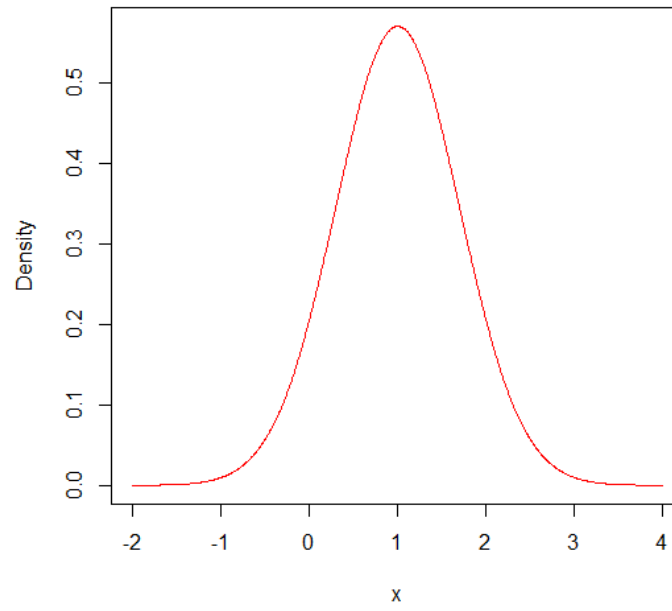


Figure 5.1: Normal density function

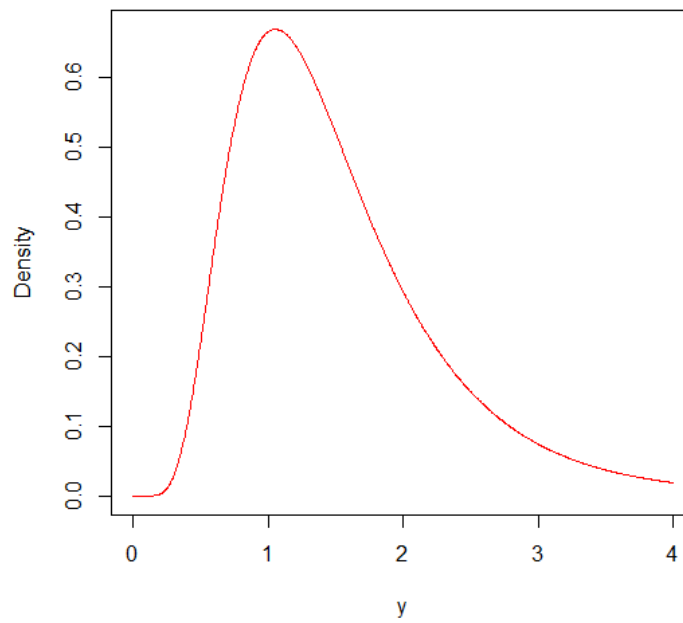


Figure 5.2: Lognormal density function

5.2.2.3 Johnson's Translation System

The relationship that exists between the parameters of the normal and lognormal distributions is explained in Aitchison and Brown (1957). This relationship is given by

$$\mu = \log\left(\frac{m^2}{\sqrt{s^2+m^2}}\right) \text{ and } \sigma^2 = \log\left[\left(\frac{s}{m}\right)^2 + 1\right].$$

Taking the exponential values, the relationship can also be given by

$$m = e^{\frac{2\mu+\sigma^2}{2}} \text{ and } s^2 = e^{2\mu+\sigma^2}(e^{\sigma^2} - 1).$$

For the simulation study normal data will be generated from the multivariate normal distribution. The lognormal data for the simulation studies will be generated from a multivariate normal distribution and then by using Johnson's translation system the normal data will be transformed into lognormal data. See Aitchison and Brown (1957), Johnson (1982) and Lamont (2008) for more information on Johnson's translation system and using it to generate data from a multivariate lognormal distribution. The R programmes used to generate the normal and lognormal data are given in the Appendix.

5.2.3 Simulation Configurations

Different configurations were constructed to represent different scenarios. In Section 5.2 it was stated that data will be generated from the multivariate normal and lognormal distributions. Within each of these two distributions different data scenarios will be considered. The first scenario will be location differences between classes and the second scenario will be dispersion differences between classes. Furthermore, we will look at the effect of correlations as well as the effect of different learning sample sizes, on the classification results. Tables 5.1 and 5.2 summarize the data scenarios that will be considered and in Table 5.3 the 16 configurations that will be generated are listed. In Figures 5.3 and 5.4, graphical representations of the normal and lognormal data are given, for both data scenarios. Figure 5.3 shows the normal and lognormal data for unequal locations and equal dispersions and Figure 5.4 shows the normal and lognormal data for equal locations and unequal

dispersions. In both figures the normal data is shown as the blue circles and the lognormal data as the red crosses.

Table 5.1: The different data scenarios that will be investigated in the simulation study as well as the parameters for the normal and lognormal distributions that will be used. These parameters are used when $\mu_1 \neq \mu_2$ and $\Sigma_1 = \Sigma_2$.

	<u>Normal data</u>	<u>Lognormal data</u>
1. Mean vector	$\mu_1 = \mathbf{0}; \mu_2 = 0.6\mathbf{1}$	$\mu_1 = \mathbf{0}; \mu_2 = \mathbf{1}$
2. Covariance matrix	$\Sigma_1 = \Sigma_2 = \Sigma^*$	$\Sigma_1 = \Sigma_2 = \Sigma^*$
3. Correlation	$\rho = 0; \rho = 0.7$	$\rho = 0; \rho = 0.7$
4. Learning sample sizes	$n_1 = n_2 = 50$	$n_1 = n_2 = 50$
	$n_1 = 25; n_2 = 75$	$n_1 = 25; n_2 = 75$

Σ^* is a $p \times p$ matrix with diagonal elements 1, and off-diagonal elements ρ . Note that $p = 7$ will be used as the number of variables in the simulation study.

Table 5.2: The different data scenarios that will be investigated in the simulation study as well as the parameters for the normal and lognormal distributions that will be used. These parameters are used when $\mu_1 = \mu_2$ and $\Sigma_1 \neq \Sigma_2$.

	<u>Normal data</u>	<u>Lognormal data</u>
1. Mean vector	$\mu_1 = \mu_2 = \mathbf{0}$	$\mu_1 = \mu_2 = \mathbf{0}$
2. Covariance matrix	$\Sigma_1 = \Sigma^*; \Sigma_2 = 4^2\Sigma^*$	$\Sigma_1 = \Sigma^*; \Sigma_2 = 4^2\Sigma^*$
3. Correlation	$\rho = 0; \rho = 0.7$	$\rho = 0; \rho = 0.7$
4. Learning sample sizes	$n_1 = n_2 = 50$	$n_1 = n_2 = 50$
	$n_1 = 25; n_2 = 75$	$n_1 = 25; n_2 = 75$

Σ^* is a $p \times p$ matrix with diagonal elements 1, and off-diagonal elements ρ . Note that $p = 7$ will be used as the number of variables in the simulation study.

Table 5.3: The 16 configurations for the generated data

Configurations	Distribution	Sample Size	Location; Dispersion	Correlation
1	Normal	equal	Unequal and equal	0
2	Normal	equal	Unequal and equal	0.7
3	Normal	equal	Equal and unequal	0
4	Normal	equal	Equal and unequal	0.7
5	Normal	unequal	Unequal and equal	0
6	Normal	unequal	Unequal and equal	0.7
7	Normal	unequal	Equal and unequal	0
8	Normal	unequal	Equal and unequal	0.7
9	Lognormal	equal	Unequal and equal	0
10	Lognormal	equal	Unequal and equal	0.7
11	Lognormal	equal	Equal and unequal	0
12	Lognormal	equal	Equal and unequal	0.7
13	Lognormal	unequal	Unequal and equal	0
14	Lognormal	unequal	Unequal and equal	0.7
15	Lognormal	unequal	Equal and unequal	0
16	Lognormal	unequal	Equal and unequal	0.7

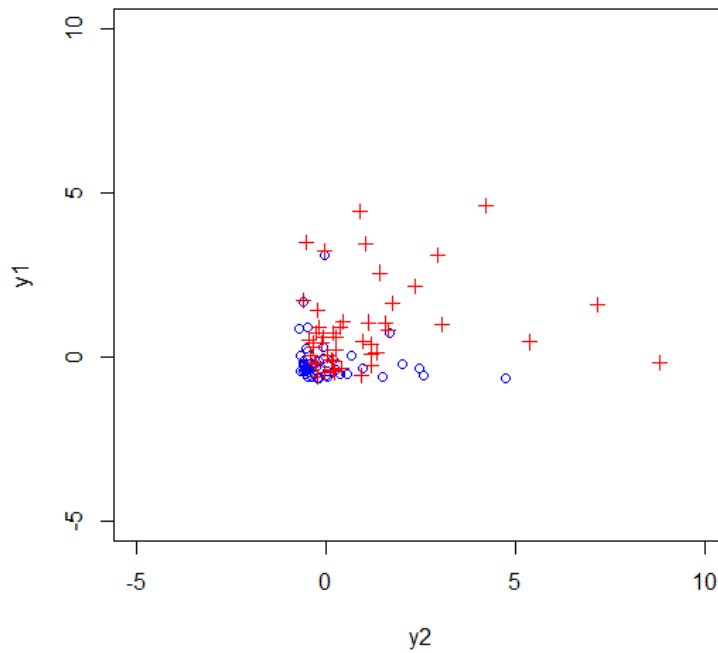
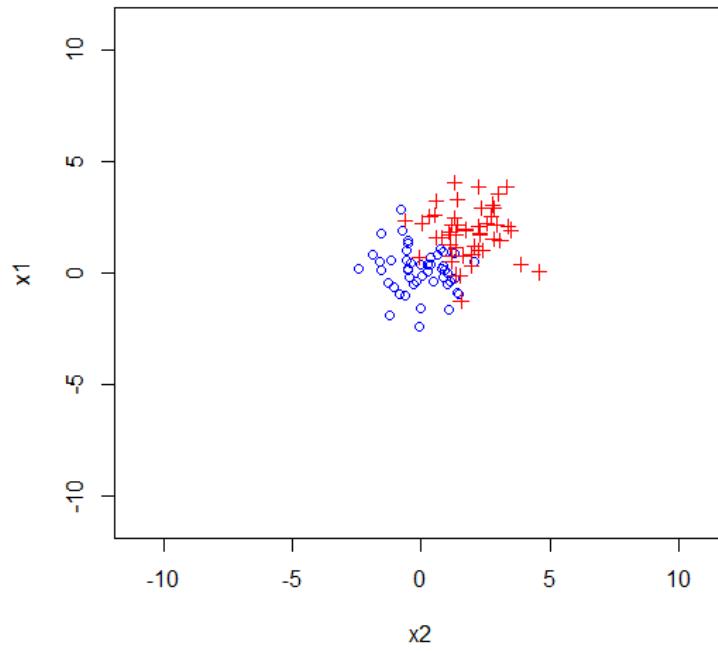


Figure 5.3: Examples of the normal and lognormal data with unequal locations and equal dispersions.

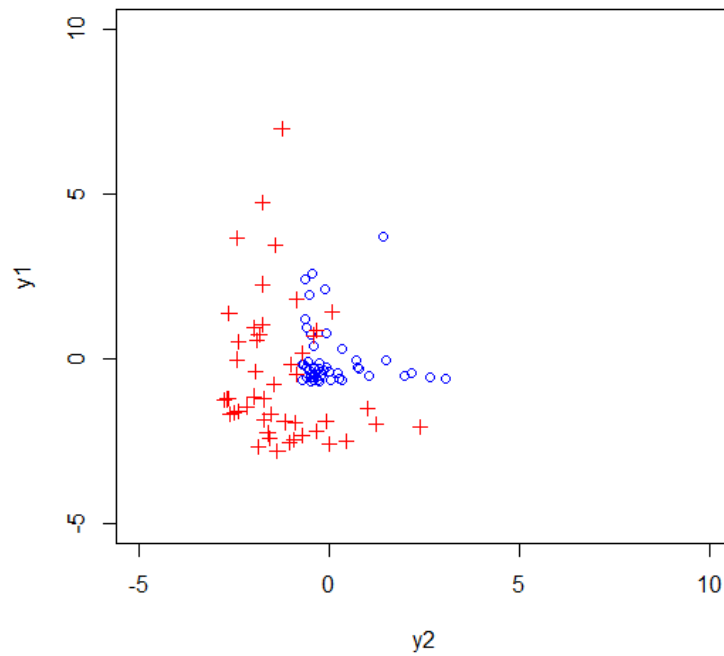
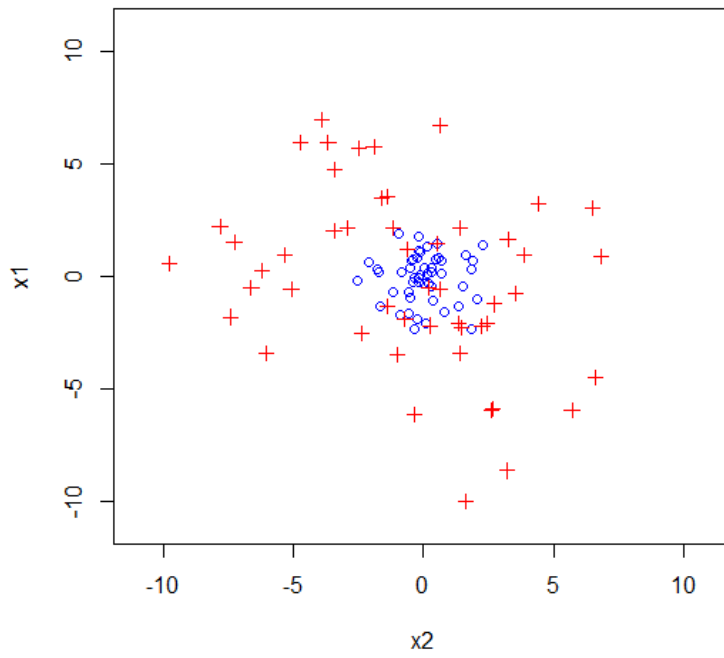


Figure 5.4: Examples of the normal and lognormal data with equal locations and unequal dispersions.

5.2.4 Discussion of the simulation results

The results for the simulation study for the normal data are given in Table 5.4 and for the lognormal data in Table 5.5. Each table lists the classification techniques that have been discussed in this thesis. The Gaussian kernel (see Table 3.1) was used in all the studies. This kernel function only requires γ to be estimated. For SVM, the Gaussian kernel was used with γ obtained via cross-validation. SVM was also performed with the cost parameter of $C = 1$. For NHC, the value $\gamma = 1/p$ was used as well as the γ obtained via cross-validation. Also, the dissimilarity function (4.20) was used for NHC.

The results for each configuration were obtained by taking the mean of the errors of 100 simulations. The standard deviations of the errors were also calculated and are shown in brackets. For each configuration in both tables the classification technique that performed best is highlighted as **bold face** figures.

Some of the results are also shown graphically in box-and-whisker plots in Figures 5.5- 5.8. A box-and-whisker plot is an effective graphical tool to compare multiple distributions on the same axis scale. These plots provide an easily interpretable view of how the means of the distributions are located and their relative distances to each other, as well as give additional insight into the dispersions of the distributions and therefore how they compare in spread relative to each other.

In Table 5.4 it can be seen that the best classifiers were mostly LDA and QDA. This is because of the underlying assumption of normality for both LDA and QDA. Therefore, it is expected that LDA and QDA would perform better with normal data. For the dispersion difference cases (3, 4, 7, and 8), QDA performs better than LDA, because LDA assumes that the covariance matrices of the two classes are equal. Figures 5.5 and 5.6 also give the same conclusions regarding the performance of LDA and QDA relative to each other. In Figure 5.5, configuration 1 is displayed and it can be seen that LDA outperforms the other techniques while in Figure 5.6 (configuration 3) QDA and SVM perform the best. NHC performs relatively well compared to the other techniques when inspecting the values of Table 5.4. This conclusion can also especially be seen in Figure 5.5. It can also be noted that all the techniques have test errors approximately between 0.2 and 0.3.

When comparing the 4th and the 5th columns of Table 5.4, (the normal data configurations) it can be noted that NHC with the optimal γ has lower error rates than NHC with $\gamma = 1/7$, when the optimal γ is less than the initial starting value of $\gamma = 1/7$. The opposite is true for the lognormal data in Table 5.5. In Table 5.5, NHC performed with the optimal γ has lower error rates than NHC with $\gamma = 1/7$ when the optimal γ is greater than the initial starting value of $\gamma = 1/7$. This is true for all the configurations except for configurations 15 and 16. Therefore, it seems that one could expect that the optimal γ should be less than $1/p$ when the data is normally distributed and larger than $1/p$ when the data is lognormally distributed.

In Table 5.5 it can be seen that SVM performs the best overall for the lognormal data. The parametric techniques LDA and QDA do not perform as well as with the normal data. However, in Figure 5.7 LDA performs relatively well compared to the other techniques. In Figure 5.8, both LDA and QDA are outperformed by the SVM and the NHC with the optimal γ . Overall, it seems that NHC with the optimal γ performs better with the lognormal data than with the normal data. Taking the overall averages, NHC scores an average of 0.31831 for the normal data and 0.17139 for the lognormal data.

Almost all the techniques performed worse for unequal learning sample sizes. This can be seen when comparing configurations 1 - 4 with configurations 5 - 8 of Table 5.4 and configurations 9 - 12 with configurations 13 - 16 of Table 5.5. For example, when comparing LDA from the 3rd configuration to the 7th configuration there is a 15.5% increase in the average test error. Taking another example, in Table 5.4 there is a 37.7% increase in the test error of SVM from the 10th configuration to the 14th configuration.

Finally, when interpreting the results of the configurations with $\rho = 0.7$ with the configurations with $\rho = 0$, we see that the errors are very similar in Table 5.4. The opposite is true for Table 5.5: the results are worse when the correlation between the variables increases for the lognormal data. Therefore, one can conclude that classification results will remain almost unchanged for normal data when the correlation between variables increases, but not for lognormal data. There is generally an increase in the error rates when correlations are increased for the lognormal data.

Table 5.4: Simulation results for LDA, QDA, SVM and NHC with the normal data.

Configuration	LDA	QDA	SVM	NHC (1/p)	NHC (opt)	Optimal Gamma
1	0.23160 (0.01427)	0.26791 (0.02009)	0.24276 (0.01539)	0.26314 (0.02047)	0.28267 (0.02127)	<i>0.3501</i>
2	0.23143 (0.01595)	0.26741 (0.01834)	0.24235 (0.01516)	0.26223 (0.02271)	0.25682 (0.02409)	<i>0.0601</i>
3	0.41938 (0.02456)	0.01832 (0.00546)	0.01307 (0.00398)	0.16030 (0.02371)	0.14373 (0.02391)	<i>0.1101</i>
4	0.41584 (0.02832)	0.01707 (0.00527)	0.01361 (0.00404)	0.15560 (0.02379)	0.11436 (0.02290)	<i>0.0701</i>
5	0.23262 (0.01495)	0.29525 (0.02969)	0.33957 (0.03652)	0.34265 (0.03280)	0.43331 (0.01936)	<i>0.3201</i>
6	0.23642 (0.01573)	0.29717 (0.03055)	0.34210 (0.03967)	0.34362 (0.03778)	0.31564 (0.03918)	<i>0.0901</i>
7	0.48442 (0.02515)	0.10745 (0.03266)	0.49774 (0.00602)	0.32982 (0.02740)	0.49998 (0.00010)	2
8	0.48495 (0.02758)	0.10849 (0.03444)	0.49779 (0.00797)	0.32416 (0.02803)	0.49998 (0.00010)	2

Table 5.5: Simulation results for LDA, QDA, SVM and NHC with the lognormal data

Configuration	LDA	QDA	SVM	NHC (1/p)	NHC (opt)	Optimal Gamma
9	0.06081 (0.00747)	0.09000 (0.01968)	0.04538 (0.01043)	0.12361 (0.03587)	0.07740 (0.02181)	<i>0.2201</i>
10	0.19046 (0.04532)	0.25769 (0.08865)	0.10441 (0.01171)	0.38240 (0.06251)	0.12021 (0.01987)	1.7
11	0.37691 (0.04994)	0.08624 (0.01127)	0.04786 (0.00974)	0.12214 (0.02404)	0.15129 (0.07932)	<i>0.0301</i>
12	0.34137 (0.06578)	0.15486 (0.04210)	0.06637 (0.01723)	0.22246 (0.08455)	0.13761 (0.01763)	<i>1.1101</i>
13	0.05948 (0.00727)	0.11753 (0.02958)	0.08501 (0.01884)	0.14013 (0.03792)	0.12743 (0.02785)	<i>0.2501</i>
14	0.17513 (0.04203)	0.25848 (0.07247)	0.14372 (0.01951)	0.32034 (0.07835)	0.17274 (0.03566)	<i>0.9101</i>
15	0.40468 (0.05389)	0.10657 (0.02082)	0.04812 (0.00838)	0.20202 (0.03443)	0.31887 (0.02161)	<i>0.6201</i>
16	0.38359 (0.07662)	0.14867 (0.02577)	0.07250 (0.01518)	0.23548 (0.08271)	0.26557 (0.02006)	<i>1.9901</i>

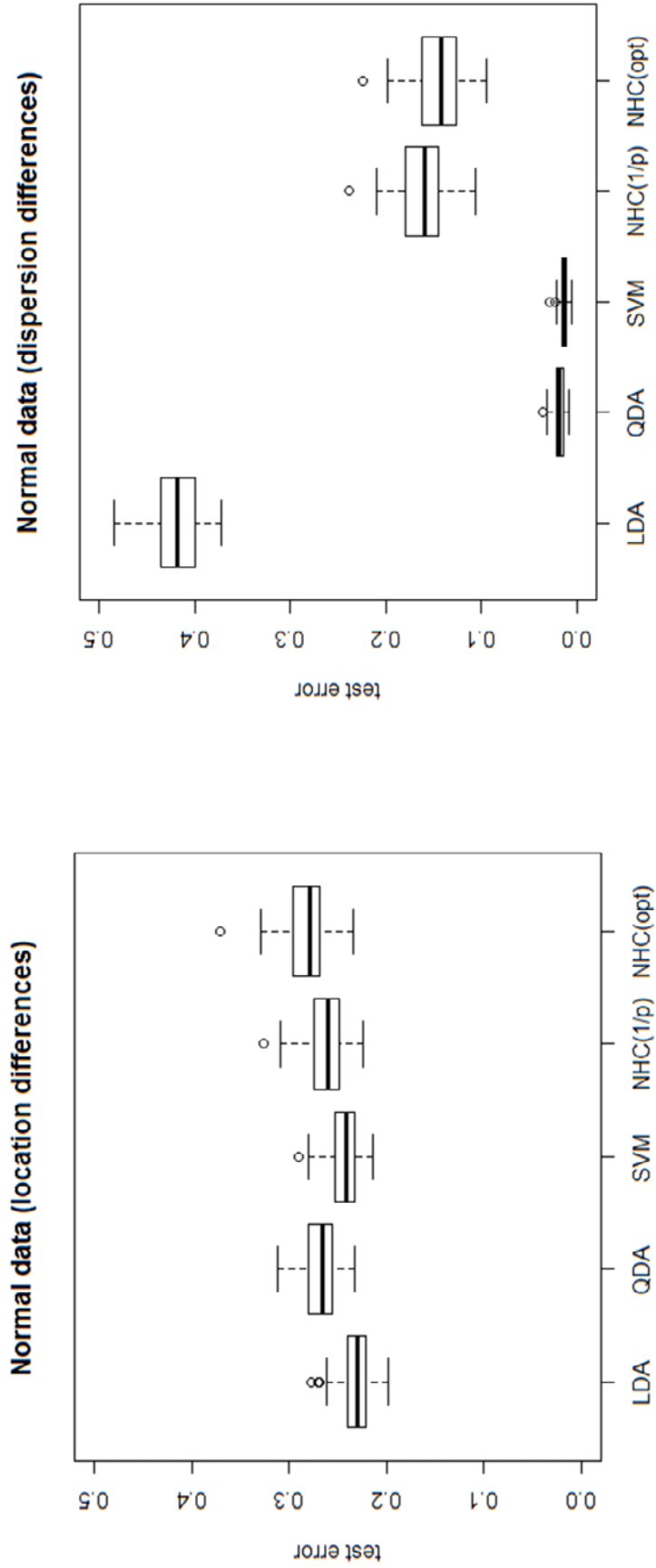


Figure 5.5: Boxplot showing the classification results for LDA, QDA, SVM and NHC. Configuration 1.

Figure 5.6: Boxplot showing the classification results for LDA, QDA, SVM and NHC. Configuration 3.

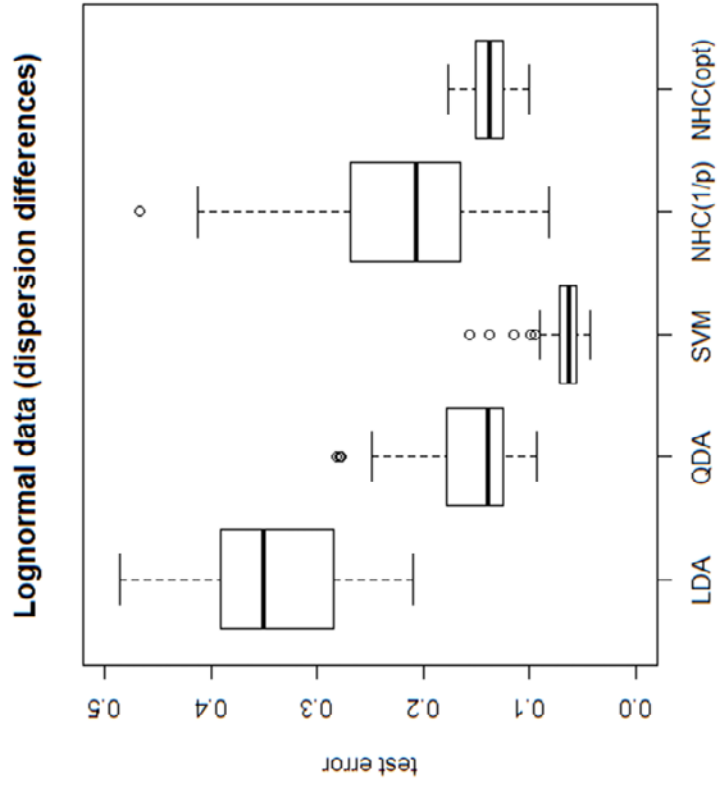


Figure 5.8: Boxplot showing the classification results for LDA, QDA, SVM and NHC. Configuration 12.

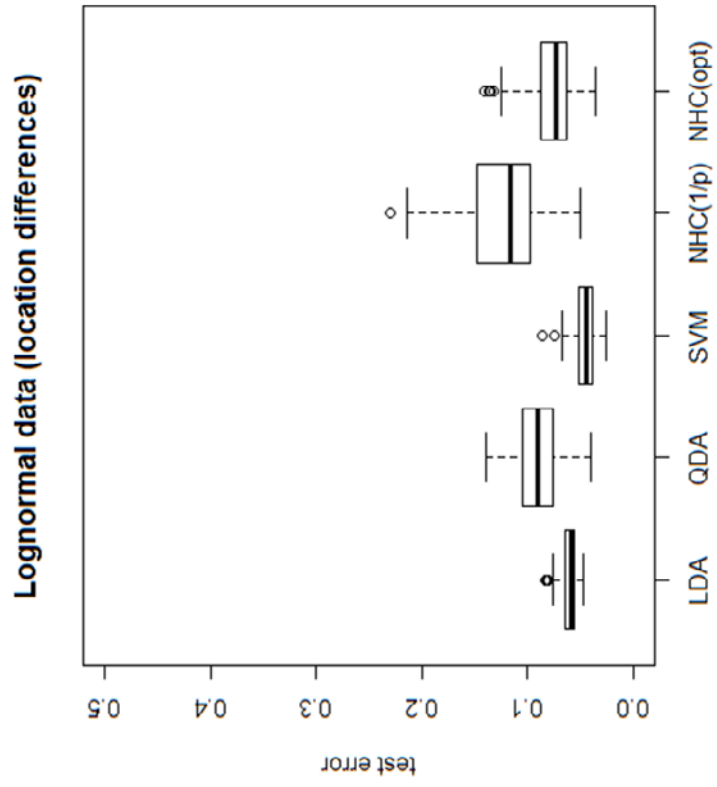


Figure 5.7: Boxplot showing the classification results for LDA, QDA, SVM and NHC. Configuration 9.

5.3 Real-world data applications

In this section LDA, QDA, SVM and NHC will be compared by applying the techniques to real-world data. The data sets that will be used will be discussed in Section 5.3.1 followed by a short analysis of each in Section 5.3.2.

5.3.1 The Data

The data sets that will be used for analyses are the *South African Heart Disease* data (cf. Hastie *et al.*, 2009, & Rousseauw *et al.*, 1983) and the *Red Wine Quality* data (cf. Cortez *et al.*, 2009). Both these data sets consist of two classes.

5.3.1.1 South African Heart Disease data

The *South African Heart disease* data set is adapted from a study by Rousseauw *et al.* (1983) and consists of 9 variables and 462 objects (males) which are grouped into two classes, heart disease present and heart disease not present, with sizes $n_1 = 160$ and $n_2 = 302$. The aim of the study is to predict the presence of coronary heart disease in males in a high-risk region of the Western Cape in South Africa. The input variables are listed below:

systolic blood pressure (sbp), cumulative tobacco in kilogram (tobacco), low density lipoprotein cholesterol (ldl), adiposity, type-A behaviour (typea), obesity, current alcohol consumption (alcohol) and age.

A matrix plot of these variables is given in Figure 5.9. The blue points represent the group of males without the heart disease and the red points represent the males with the heart disease.

5.3.1.2 Red Wine Quality data

This data set was constructed to test the quality of Portuguese red wine (Cortez *et al.*, 2008). Experts rated the wine on a scale from 0 (very bad) to 10 (very good). The data set consists of 1599 objects and 11 input variables and a response variable. The data was divided into two classes, bad wine and good wine, according to the rated values of the wine. Red wine with rated values 0 to 5 was grouped into the bad class while the wines with rated values of 6 to 10 were grouped into the good class. There are 744 wines in the first class and 855 wines in the second class. The input variables for the study are listed below:

Fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates and alcohol level.

A matrix scatter plot of these variables is given in Figure 5.10. The blue points represent the bad wine and the red points represent the good wine.

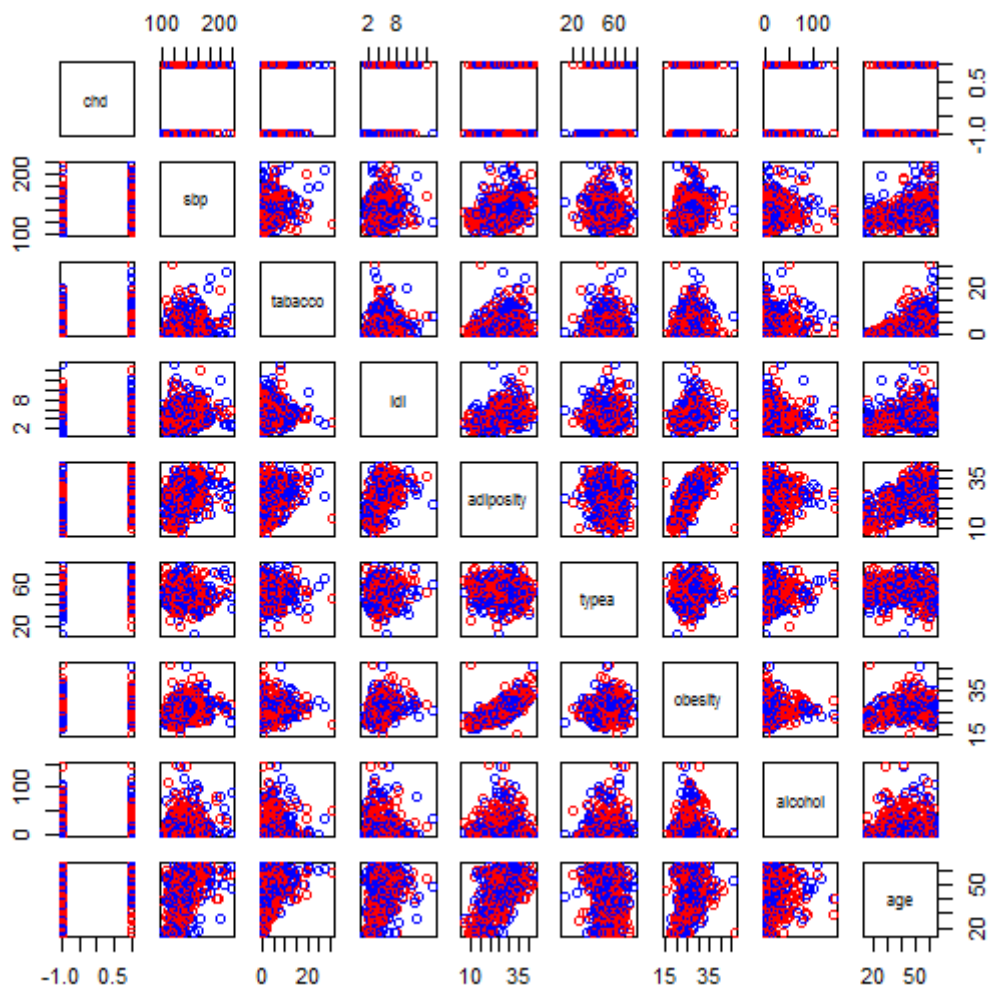


Figure 5.9: Matrix plot of the South African heart disease data.

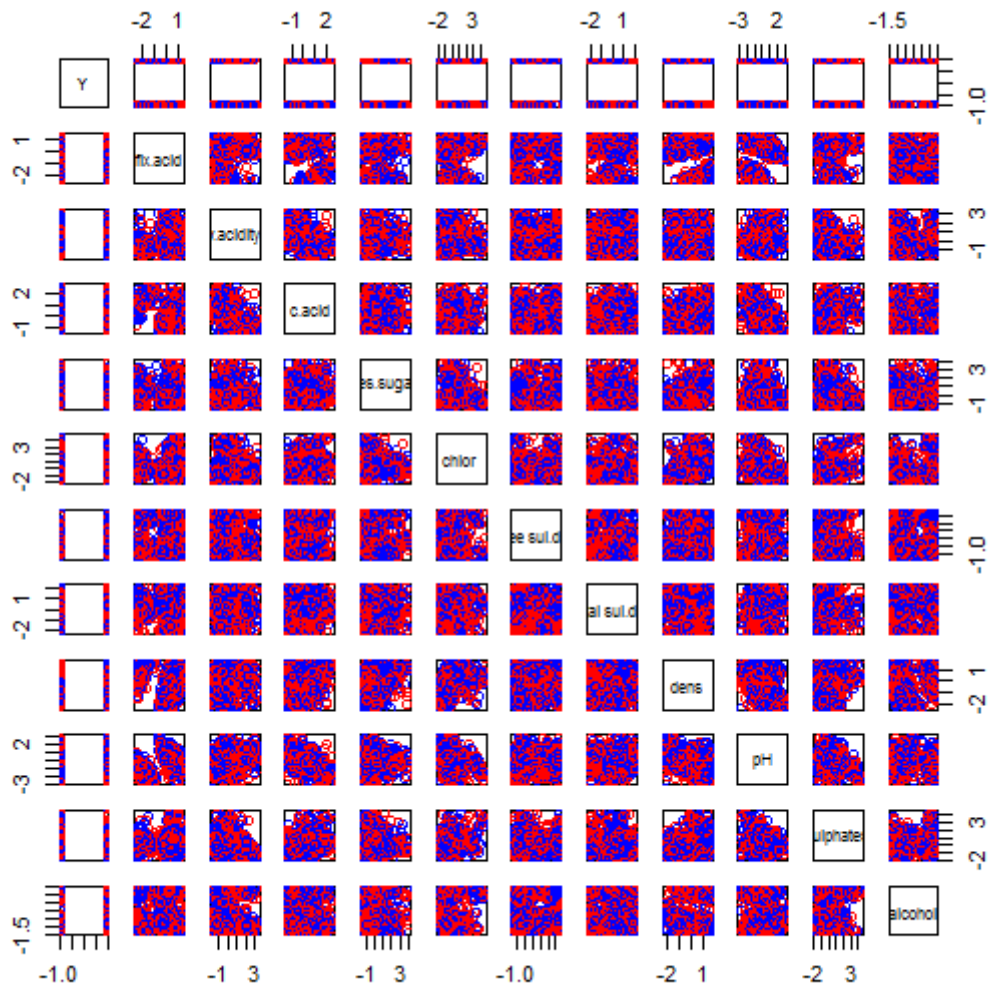


Figure 5.10: Matrix plot of the Red Wine quality data.

5.3.2 Classification results

Before applying LDA, QDA, SVM and NHC, each input variable in the two data sets will be standardized to have zero mean and unit variance across classes. The input variables for the data sets are standardized using the means and standard deviations obtained from the data sets.

5.3.2.1 South African Heart Disease data

The results of the classifications of LDA, QDA, SVM and NHC for the *South African Heart Disease* data are given in Table 5.6. The cross-validation error rate was estimated by taking the average of 200 simulation runs. The standard deviations are shown in brackets. We set the cost parameter of the SVM to $C = 1$. The optimal γ as well as the value $\gamma = 1/p = 1/8$ were used for the NHC. Using the methodology that was discussed in Section 4.3.3, resulted in an optimal parameter of $\gamma = 2$.

It can be seen that SVM produced the best results which are highlighted in **bold face** figures. However, LDA, QDA and NHC with $\gamma = 1/8$ are not far behind. Both parametric techniques outperformed NHC. NHC with $\gamma = 1/8$ gave better results than the NHC with $\gamma = 2$.

Table 5.6: Classification results for the South African Heart Disease data

	LDA	QDA	SVM	NHC(1/p)	NHC(opt)	Gamma
mean	0.31939	0.32222	0.31713	0.32617	0.35687	2
sd	(0.03846)	(0.04193)	(0.03859)	(0.03839)	(0.03829)	

5.3.2.2 Red Wine Quality data

In Table 5.7, the classification results for LDA, QDA, SVM, and NHC with the optimal $\gamma = 1$ as well as the NHC with $\gamma = 1/11$ are shown for the *Red Wine Quality* data. The NHC with the optimal γ gave the best results, but by a small margin compared to the SVM. SVM gave the second best results followed by QDA, LDA and NHC with $\gamma = 1/11$.

Table 5.7: Classification results for the Red Wine Quality data

	LDA	QDA	SVM	NHC(1/p)	NHC(opt)	Gamma
mean	0.30031	0.28393	0.25990	0.32318	0.25892	1
sd	0.01980	0.01882	0.02047	0.02105	0.02593	

5.4 Conclusions

In this chapter we compared the classification techniques LDA, QDA, SVM and NHC, that were discussed in this thesis, by means of a simulation study. In Section 5.2.1, we discussed how to quantify the performance of a classification technique and looked specifically at the test error and the cross-validation error. In Section 5.2.3, the different configurations that were generated were shown. The simulations were done on data that was generated from the normal distribution and from the lognormal distribution. To simulate the lognormal data we used Johnson's translation system which was discussed in Section 5.2.2.3. We compiled 16 different data configurations, 8 from the normal data and 8 from the lognormal data, to represent different data scenarios. The scenarios we looked at were equal and unequal locations, equal and unequal dispersions, equal and unequal sample sizes, and correlated and uncorrelated data.

The classification results were shown in Tables 5.4 and 5.5, which showed the means and standard deviations of the errors obtained. Some results were also shown graphically in box-and-whisker plots. LDA and QDA performed generally the best with the normal data and SVM and NHC performed the best with the lognormal data. A conclusion was also drawn regarding the parameter γ of NHC when performed on normal and lognormal data. It was seen that for the lognormal data, the optimal γ generally gives better results than $\gamma = 1/p$ when the optimal $\gamma > 1/p$. The opposite seemed to be true for the normal data.

The techniques were also compared by means of real-world data applications. We looked at the *South African Heart Disease* data and the *Red Wine Quality* data. SVM performed the best with the heart disease data and NHC performed with the optimal γ performed the best with the wine data.

CHAPTER 6

CONCLUSION

The main purpose of this study was to investigate the performance of NHC compared to other classification techniques, viz. LDA, QDA and SVM. The comparison was done by means of a simulation study where various data configurations were considered. Two real-world data applications were also used to compare the mentioned techniques.

6.1 Summary of Findings

In Chapter 2 an overview of an optimal classification model was given. We also looked at LDA and QDA in terms of theory and practical applications. We saw that both techniques gave relatively good results when applied on the Iris and Haemophilia data sets. However, as the theory suggests, LDA and QDA may not give optimal results when the assumption of normality is not met and when the number of variables in the data becomes too large ($p \gg n$).

In Chapter 3 we introduced the SVM which is a technique that originated in the field of machine learning. It was seen that SVM can be used in a linear or nonlinear fashion. The nonlinear SVM is a kernel-based technique which means that it uses kernel functions to derive a rule which can be used to classify objects into classes. SVM is a nonparametric technique; it does not require the distribution of the data to be known which gives it a great competitive advantage to parametric techniques such as LDA and QDA. SVM was also applied on the Iris and the Haemophilia data sets and in both cases it performed superior to LDA and QDA.

In Chapter 4 we introduced classification with hyperspheres. We looked at how a hypersphere is constructed via the hard-margin solution and the soft-margin solution. A hypersphere constructed by using the hard-margin solution, known as the Smallest Enclosing Hypersphere, is a hypersphere that must contain all the objects of a class. A hypersphere constructed with the soft-margin solution may exclude some objects that differ from the rest of the objects by some characteristic. This is why this technique may be considered as an anomaly detection technique.

Chapter 4 also looked specifically at NHC. NHC is also a kernel-based technique and it works by fitting each class in the data set with its own hard-margin hypersphere and then classifying new objects to the class with the smallest distance between the object and the hypersphere. NHC was performed with the Iris data set and gave the best results compared to the other classification techniques. With the Haemophilia data set NHC gave a similar result as the SVM.

In Chapter 5 we looked at a simulation study to compare the performance of LDA, QDA, SVM and NHC. The different data configurations looked at normal and lognormal data, equal and unequal locations, equal and unequal dispersions, equal and unequal sample sizes and correlated and uncorrelated data. For each configuration a test error was obtained for each technique which was used to compare the different techniques. NHC was performed with two values for its parameter. The one was the optimal parameter and the other one was $\gamma = 1/p$.

With the normally distributed data LDA and QDA did the best. In the cases when the dispersion for the two classes was equal, LDA outperformed QDA. The opposite is true for when the classes have unequal dispersions. In the case of the lognormal data SVM did the best. NHC did relatively well with the lognormal data, but not so with the normal data. There is still scope to improve the performance of NHC. This can possibly be achieved by looking into alternative kernel functions or dissimilarity functions when implementing NHC.

6.2 Future Research Recommendations

1. Different kernel functions can be considered when researching the performance of the NHC relative to other techniques. Only the Gaussian kernel function was used in this thesis.

2. The dissimilarity function $\delta_j(\mathbf{x}) = \frac{\|\Phi(\mathbf{x}) - \mathbf{c}_j^*\|}{r_j^*}$ was used in this study. Considering other dissimilarity functions when implementing NHC might be helpful in improving the performance of NHC.

3. Hyperspheres constructed by using the soft-margin solution are considered as an anomaly detection method. However, research is needed to compare the relative performance of this technique to other machine learning anomaly detection techniques.

REFERENCES

Aitchison, J. & Brown, J.A.C., 1957. "The Lognormal Distribution." *The Economic Journal* 67(268). 713-715.

Boser, B.E., Guyon, I.M. & Vapnik, V.N., 1992. "A Training Algorithm for Optimal Margin Classifiers." *In Proceedings of the Fifth Annual Workshop of Computational Learning Theory* 5. 144-152.

Bouma, B.N., 1975. "Evaluation of the Detection Rate of Haemophilia Carriers." *Statistical Methods for Clinical Decision Making* 7(2). 339-350.

Chang, C. & Lin, C., 2010. "A Practical Guide to Support Vector Classification." Department of Computer Science. National Taiwan University.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. & Reis, J., 2009. "Modelling Wine Preferences by Data Mining from Physicochemical Properties." *Decision Support Systems* 47(4). 547-553.

Fisher, R.A., 1936. "The Use of Multiple Measurements in Taxonomic Problems." *Ann. Eugenics* 7. 179-188.

Hao, P.Y., Chiang, J.H. & Lin, Y-H., 2009. "A New Maximal-Margin Spherical-Structured Multi-Class Support Vector Machine." *Appl. Intell.* 30. 98-111.

Hastie, T., Tibshirani, R. & Friedman, R., 2009. *The Elements of Statistical Learning*. 2nd edition. New York: Springer.

Hornik, K., Karatzoglou, A. & Smola A., 2004. “Kernlab – An S4 Package for Kernel Methods in R.” *Journal of Statistical Software* 11(9). 1-20.

Izenman, A.J., 2008. *Modern Multivariate Statistical Techniques*. New York: Springer.

Johnson, M.E., Ramberg, J.S. & Wang, C., 1982. “The Johnson Translation System in Monte Carlo Studies.” *Communications in Statistics: Simulation and Computation* 11(5). 521-527.

Johnson, R.A. & Wichern, D.W., 2007. *Applied Multivariate Statistical Analysis*. 6th edition. New Jersey: Prentice Hall.

Karush, W., 1939. “Minima of functions of several variables with inequalities as side constraints.” Masters thesis, Department of Mathematics. University of Chicago.

Kuhn, H.W. & Tucker, A.W., 1951. “Nonlinear programming.” 2nd *Berkeley Symposium on Mathematical Statistics and Probabilistics*. 481 – 492.

Lamont, M.M.C., 2008. “Assessing the Influence of Observations on the Generalization Performance of the Kernel Fisher Discriminant Classifier.” PhD thesis, Department of Statistics and Actuarial Sciences. University of Stellenbosch.

Mercer, J., 1909. “Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations.” *Philosophical Transactions of the Royal Society* 209(441-458). 415-446.

Ripley, B.D. & Venables, W.N., 2002. *Modern Applied Statistics with S-plus*. 4th edition. New York: Springer.

Rousseauw, J., du Plessis, J., Benade, A., Jordaan, P., Kotze, J. & Ferreira, J., 1983. "Coronary risk factor screening in three rural communities." *South African Medical Journal* 64. 430–436.

Shawe-Taylor, J. & Cristianini, N., 2004. *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press.

Tax, D.M.J. & Duin, R.P.W., 1999. "Support Vector Domain Description." *Pattern Recognition Letters* 20. 1191-1199.

Tax, D.M.J., 2001. "One-class classification." PhD thesis, Technischen Universität Berlin.

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

APPENDIX

CODE IN R

A. Main Simulation Program

```
function(distr,n1,n2,p,r,mu,var1,var2,gamma,N=100)
{
  #p is the number of variables
  #r is the correlation between variables
  #mu contains the means for the two classes
  #var1 and var2 is the variances for the two classes
  #N is the number of simulations that are to be done
  #k is a proportion determining the amount of data in the test set

  auto <- 1/p;out <- matrix(0,nrow=N,ncol=5)
  for(i in 1:N){
    train <- distr(n1,n2,p,r,mu,var1,var2)
    test <- distr(n1=1000,n2=1000,p,r,mu,var1,var2)
    lda.pe <- LDA(train,test,draw.regions=F)$APER
    qda.pe <- QDA(train,test,draw.regions=F)$APER
    svm.pe <- my.SVM(train,test,draw.regions=F)$APER
    nec.pe.auto <- NHC(train,test,gamma=auto)$APER
    nec.pe <- NHC(train,test,gamma)$APER
    out[i,] <- c(lda.pe,qda.pe,svm.pe,nec.pe.auto,nec.pe)}
  colnames(out) <- c("LDA","QDA","SVM","NHC|gamma=1/p","NHC|gamma")
  ave <- apply(out,2,mean);sds <- apply(out,2,sd)
  results <- rbind(ave,sds);colnames(results) <- colnames(out)
  list(Output=out,Stats=results)
}
```

B. Generating Normal Data

```
function(n1,n2,p,r,mu,var1=1,var2)
{
  y.vec <- matrix(c(rep(-1,n1),rep(1,n2)),ncol=1)
  mu1 <- matrix(mu[1],p,ncol=1)
  mu2 <- matrix(mu[2],p,ncol=1)
  sigma1 <- matrix(r/var1,p,p)
  diag(sigma1) <- var1
  sigma2 <- matrix(r/var2,p,p)
  diag(sigma2) <- var2
  class1 <- mvrnorm(n1,mu1,sigma1)
  class2 <- mvrnorm(n2,mu2,sigma2)
  data <- cbind(y.vec,rbind(class1,class2))
  return(data)
}
```

C. Generating Lognormal Data

```
function (n1,n2,p,r,mu,var1,var2)
{
  y.vec <- matrix(c(rep(-1,n1),rep(1,n2)),ncol=1)
  E <- exp(1)
  LAM <- sqrt(1/(E*(E-1)))
  EP <- -1*sqrt(1/(E-1))
  cov1 <- var1*log(r*(E-1)+1)
  cov2 <- var2*log(r*(E-1)+1)
  sigma1 <- matrix(cov1,p,p)
  sigma2 <- matrix(cov2,p,p)
  diag(sigma1) <- var1
  diag(sigma2) <- var2
  mu1 <- matrix(mu[1],p,ncol=1)
  mu2 <- matrix(mu[2],p,ncol=1)
  class1 <-
    sqrt(var1)*(LAM*exp((1/sqrt(var1))*mvrnorm(n1,mu1,sigma1))+EP)
  class2 <-
    sqrt(var2)*(LAM*exp((1/sqrt(var2))*mvrnorm(n2,mu2,sigma2))+EP)
  data <- cbind(y.vec,rbind(class1,class2))
  return(data)
}
```

D. Linear Discriminant Analysis

```
function (learn.data,test.data,draw.regions=T)
{
  #provision to make sure data are matrices and determining group
  #sizes

  learn.data <- as.matrix(learn.data)

  test.data <- as.matrix(test.data)

  g1 <- as.matrix(learn.data[learn.data[,1]==-1,],[-1])
  g2 <- as.matrix(learn.data[learn.data[,1]==+1,],[-1])

  n1 <- nrow(g1)
  n2 <- nrow(g2)

  n <- n1+ n2

  #calculating stats on the learn set
  xbar1 <- matrix(apply(g1,2,mean))
  xbar2 <- matrix(apply(g2,2,mean))

  s1 <- var(g1)
  s2 <- var(g2)

  s.pooled <- s1*((n1-1)/(n-2))+s2*((n2-1)/(n-2))
  sinv <- solve(s.pooled)

  b <- 0.5 * t(xbar1-xbar2) %*% sinv %*% (xbar1+xbar2)

  #Determining the scores for the test data and classifying them
  #accordingly

  nt <- nrow(test.data)

  scores <- matrix(0,nrow=nt)

  for(i in 1:nt){
    scores[i,] <- (t(xbar1-xbar2) %*% sinv %*% test.data[i,-1])-b
  }

  clas <- ifelse(scores > 0,"Group.1","Group.2")
}
```

```
#Plotting the data if draw.regions has been specified
if(draw.regions==T){
LDA.graphics(data=learn.data[,1:3],draw.test=T,test.data=
test.data[,1:3],legend=F)
}
else{}
#Calculating the apparent error rate
clas <-
data.frame(Classified=clas,Actual.membership=ifelse(test.data[,1]
== -1,"Group.1","Group.2"))
APER <- (nrow(test.data)-sum(ifelse(scores>0,1,1)==test.data[,1]))
      /nrow(test.data)
list(Classification=clas,APER=APER)
}
```


E. Quadratic Discriminant Analysis

```
function (learn.data,test.data,draw.regions=T)
{
  #provision to make sure data are matrices and determining group
  #sizes

  learn.data <- as.matrix(learn.data)

  test.data <- as.matrix(test.data)

  g1 <- as.matrix(learn.data[learn.data[,1]==-1,],[-1])
  g2 <- as.matrix(learn.data[learn.data[,1]==+1,],[-1])

  n1 <- nrow(g1)
  n2 <- nrow(g2)

  n <- n1+ n2

  #calculating stats on the learn set
  xbar1 <- matrix(apply(g1,2,mean))
  xbar2 <- matrix(apply(g2,2,mean))

  s1 <- var(g1)
  s2 <- var(g2)

  sinv1 <- solve(s1)
  sinv2 <- solve(s2)

  k <- 0.5*log(det(s1)/det(s2))+0.5*(t(xbar1)%*%sinv1%*%xbar1-
    t(xbar2)%*%sinv2%*%xbar2)

  #Determining scores for test data and classifying them to a group
  nt<-nrow(test.data[,])
  scores<-matrix(rep(0,nt),ncol=1)

  for(i in 1:nt){
    scores[i,]<--0.5*t(test.data[i,-1])%*%(sinv1-
    sinv2)%*%test.data[i,-1]+(t(xbar1)%*%sinv1-
    t(xbar2)%*%sinv2)%*%test.data[i,-1]-k }
}
```

```
clas<-ifelse(scores>0,"Group.1","Group.2")

#Plotting the regions if draw.region has been specified
if(draw.regions==T){
QDA.graphics(data=learn.data[,1:3],draw.test=T,test.data=
test.data[,1:3],legend=F)
}
else {
}

#Calculating the apparent error rate
clas <- data.frame(Classified=clas,Actual.membership=
  ifelse(test.data[,1]==-1,"Group.1","Group.2"))
APER <- (nrow(test.data)-sum(ifelse(scores>0,1,1)==test.data[,1]))
  /nrow(test.data)

list(Classification=clas,APER=APER)
}
```

F. Support Vector Machine

```

function (learn.data, test.data, kernel = "rbfdot", kpar =
"automatic", C = 1, cross = 3, prob.model = T, draw.regions = T)
{
  #provision to make sure data are in matrix format
  learn.data <- as.matrix(learn.data)
  test.data <- as.matrix(test.data)
  d.data <- rbind(learn.data,test.data)
  g1 <- as.matrix(learn.data[learn.data[,1]==-1,],[-1])
  g2 <- as.matrix(learn.data[learn.data[,1]==+1,],[-1])
  n1 <- nrow(g1)
  n2 <- nrow(g2)
  n <- n1+ n2

  #Loading the necessary package and fitting the svm model as well
  #as determining the scores
  library(kernlab)
  out <- ksvm(learn.data[, 1] ~ ., data = learn.data[, -1],
             kernel = kernel, kpar = kpar, C = C, cross = cross,
             prob.model = prob.model, type = "C-svc")
  scores <- predict(out, test.data[, -1])

  #if draw.regions are specified the graphics are given otherwise no
  #graphical output will be given
  if (draw.regions == T) {
    SVM.graphics(data=d.data[,1:3],kernel=kernel,kpar=kpar,C=C,
cross=cross,prob.model=prob.model,draw.test=T,test.data=
test.data[,1:3],legend=F)
  }
  else {
  }
}

```

```
#provision is being made for the test data in terms of the class
#variable

#the classification output is also given

clas <- ifelse(test.data[,1]==-1,"Group 1","Group 2")

class <- data.frame(Classified=ifelse(scores>0,"Group 2",
    "Group 1"),Actual.membership=clas)

APER <- (nrow(test.data)-sum(ifelse(scores>0,1,1)==test.data[,1]))
    /nrow(test.data)

list(Model.Output = out,Classification = class, APER = APER)
}
```

G. Nearest Hypersphere Classification

```
function(learn.data,test.data,gamma)
{
  kernelrbf <- rbfdot(sigma=gamma)
  #Determining groups and sizes
  test.data2 <- test.data
  test.data <- as.matrix(test.data)[,-1]
  g1 <- as.matrix(learn.data[learn.data[,1]==-1,])[,-1]
  g2 <- as.matrix(learn.data[learn.data[,1]==+1,])[,-1]
  n1 <- nrow(g1)
  n2 <- nrow(g2)
  nt <- nrow(test.data)
  #Determining the radius, alphas and Gram matrix for each group
  g1.calc <- calc.Hyp(g1,gamma=gamma)
  g2.calc <- calc.Hyp(g2,gamma=gamma)
  g1.Gram <- g1.calc$Gram
  g2.Gram <- g2.calc$Gram
  g1.alphas <- g1.calc$alphas
  g2.alphas <- g2.calc$alphas
  g1.r <- g1.calc$rstar
  g2.r <- g2.calc$rstar
  #Determining the distances between test data and the two groups
  #and determining the signs
  test.kii <- matrix(diag(kernelMatrix(kernelrbf,test.data))
                    ,nrow=nt)
  #Group 1 distances with test data
  g1.test.kj <- kernelMatrix(kernelrbf,test.data,g1)
```

```

g1.test.dist <- matrix(0,nrow=nt)
for(i in 1:nt){
g1.test.dist[i,] <- (test.kii[i,] -2*t(g1.alphas)**g1.test.kj[i,]
                    +t(g1.alphas)**g1.Gram**g1.alphas)/g1.r^2
}
#Group 2 distances with test data
g2.test.kj <- kernelMatrix(kernelrbf,test.data,g2)
g2.test.dist <- matrix(0,nrow=nt)
for(i in 1:nt){
g2.test.dist[i,] <- (test.kii[i,] -2*t(g2.alphas)**g2.test.kj[i,]
                    +t(g2.alphas)**g2.Gram**g2.alphas)/g2.r^2
}
#Determining the signs for test data and classifying them
#accordingly
sign.test.vec <- matrix(0,nrow=nt)
for(i in 1:nt){
    sign.test.vec[i,] <- sign(g1.test.dist[i,] - g2.test.dist[i,])
}
class <- ifelse(sign.test.vec<0,-1,1)
APER <- (nt-sum(class==test.data2[,1]))/nt
list(Classification=data.frame(Classified=ifelse(class==1,
    "Group 1","Group 2"),
    Actual.Membership=ifelse(test.data2[,1]==-1,"Group 1",
    "Group 2")),APER=APER)
}

```