ASSESSING THE INFLUENCE OF OBSERVATIONS ON THE GENERALIZATION PERFORMANCE OF THE KERNEL FISHER DISCRIMINANT CLASSIFIER

by

Morné Michael Connell Lamont

Dissertation presented for the degree of
Doctor of Philosophy
at Stellenbosch University

Promoter: Prof. Nelmarie Louw
Co-promoter: Prof. Sarel Steel

December, 2008

DECLARATION

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 1 November 2008

OPSOMMING

Kern Fisher diskrimimant analise (KFDA) is ’n kernfunksie-gebaseerde tegniek wat gebruik kan word om waarnemings van onbekende oorsprong te klassifiseer in voorafgedefinieerde groepe. KFDA kan basies beskou word as ’n nie-linieêre uitbreiding van Fisher se linieêre diskriminant analise (FLDA). In hierdie tesis verduidelik ons hoe FLDA veralgemeen kan word na KFDA. Twee tegnieke wat verwant is aan KFDA word ook bespreek. Ons fokus is op binêre klassifikasie.

Die invloed van atipiese punte in ’n diskriminant analise is al deur verskeie navorsers ondersoek. In hierdie tesis ondersoek ons die invloed van atipiese punte op sekere aspekte van KFDA. Een belangrike aspek hier is die gedrag van die veralgemeningsfout van die KFD klassifiseerder. Verskeie ander aspekte word ook ondersoek met die doel om kriteria te ontwikkel wat punte identifiseer wat ’n nadelige effek op die KFD veralgemeningsfout het. Die ondersoek word gedoen deur middel van ’n Monte Carlo simulasie studie.

Die afvoer van KFDA kan ook gebruik word om aposteriori-waarskynlikhede te verkry. In hierdie tesis bespreek ons twee benaderings hiervoor. Twee klassifiseerders, wat gebruik maak van hierdie waarskynlikhede om waarnemings te klassifiseer, word ook afgelei en hul veralgemeningsfoute word dan vergelyk met dié van die oorspronklike KFD klassifiseerder.

Die hoofdoel van die tesis is om kriteria te ontwikkel wat punte identifiseer wat ’n nadelige effek op die KFD veralgemeningsfout het. Nege sulke kriteria word in die tesis voorgestel. Die meriete van die kriteria word ook getoets in ’n Monte Carlo simulasie studie asook op praktiese datastelle. Om die kriteria op ’n een-vir-een-weglating basis te evalueer, het ’n berekeningsuitdaging tot gevolg, veral vir groot datastelle. In hierdie tesis maak ons ’n voorstel dat die kleinste insluitende hipersfeer as ’n filter gebruik kan word om die hoeveelheid berekeninge te verminder. Die effektiwiteit van die filter word ook getoets in ’n Monte Carlo simulasie studie, sowel as op praktiese datastelle.

SUMMARY

Kernel Fisher discriminant analysis (KFDA) is a kernel-based technique that can be used to classify observations of unknown origin into predefined groups. Basically, KFDA can be viewed as a non-linear extension of Fisher's linear discriminant analysis (FLDA). In this thesis we give a detailed explanation how FLDA is generalized to obtain KFDA. We also discuss two methods that are related to KFDA. Our focus is on binary classification.

The influence of atypical cases in discriminant analysis has been investigated by many researchers. In this thesis we investigate the influence of atypical cases on certain aspects of KFDA. One important aspect of interest is the generalization performance of the KFD classifier. Several other aspects are also investigated with the aim of developing criteria that can be used to identify cases that are detrimental to the KFD generalization performance. The investigation is done via a Monte Carlo simulation study.

The output of KFDA can also be used to obtain the posterior probabilities of belonging to the two classes. In this thesis we discuss two approaches to estimate posterior probabilities in KFDA. Two new KFD classifiers are also derived which use these probabilities to classify observations, and their performance is compared to that of the original KFD classifier.

The main objective of this thesis is to develop criteria which can be used to identify cases that are detrimental to the KFD generalization performance. Nine such criteria are proposed and their merit investigated in a Monte Carlo simulation study as well as on real-world data sets.

Evaluating the criteria on a leave-one-out basis poses a computational challenge, especially for large data sets. In this thesis we also propose using the smallest enclosing hypersphere as a filter, to reduce the amount of computations. The effectiveness of the filter is tested in a Monte Carlo simulation study as well as on real-world data sets.

FOR THE GLORY OF JESUS CHRIST

ACKNOWLEDGEMENTS

I would like to express my heartfelt appreciation to:

- God, for His amazing grace to come this far in my career. My heart is eternally greatful to You Father for the favour and blessings You have bestowed on my life here on earth. I pray that the fruit of my life will bring glory to Your name.
- my promoters proff. Nelmarie Louw and Sarel Steel. Thank you for your guidance, support and encouragement throughout this study. You have played a significant role in shaping my career and thinking in Statistics. Thank you for the many sacrifices and contributions to my career as a Statistician.
- my parents, Connell and Thelma. Thank you for the love and support through my many difficult sacrifices. Thank you for teaching me good values and for encouraging me to be a blessing to this world.
- my brother, sisters and their families. Thank you also for your love and support.
- my friends. Thank you for your support and for your interest in my studies. I especially want to thank Oom Jer, Ruwaida, Kiddo and Liezl who's door was always open for me when my frustrations were tearing my soul apart.
- my colleagues at Stellenbosch University department of Statistics, especially prof. Tertius de Wet, for encouraging me with my studies.
- my employer, Stellenbosch University, for the opportunity to do my PhD for free and the privilege to teach Statistics to many students.
- Shofar Christian Church. You have become the place where I can grow spiritually. Being part of the body of Jesus Christ means everything to me and I thank you for assisting me in my spiritual pilgrimage.
- From my childhood until now, many people have been a blessing to me. I love you and pray that God will bless you abundantly. Stand tall and do it with class …

Luke 9:48 *… for he that is least among you all, the same shall be great*

CONTENTS

CHAPTER 1

INTRODUCTION

"The best thing about being a statistician is that you get to play in everyone's backyard."

–JohnTukey

1.1 Statistical discrimination and classification

Discriminant analysis is a multivariate statistical technique concerned with separating distinct groups of entities by using certain measurements (input variables) taken on the observations. The goal of a discriminant analysis is to describe either graphically or algebraically, the differential features of observations from several known populations. A set of "discriminants" is found which will enable us to separate the populations as much as possible. Classification on the other hand deals with the allocation of new entities (with unknown group-membership) to previously defined groups. Suppose we have a set of entities belonging to two or more populations. A rule can be obtained from the data that can be used to assign a new entity to any of the known populations. The common name in statistical literature that is used for both discrimination and classification is discriminant analysis.

Several discrimination techniques have been developed over the years. Some of the well-known techniques in the statistical literature include Fisher's linear discriminant analysis, linear and quadratic discriminant analysis, logistic regression, *k*-nearest neighbours, canonical discriminant analysis and kernel density discriminant analysis. Other discrimination techniques from the machine learning literature that have gained popularity in statistics are classification trees, artificial neural networks, support vector machines and kernel Fisher discriminant analysis. Good references for the techniques mentioned above include Hastie *et al.* (2001), Breiman *et al.* (1984) and Johnson and Wichern (1992).

The following are some examples of the application of a discriminant analysis:

- A biologist could record different variables of similar types (species) of flowers, and then perform a discriminant analysis to determine the set of variables that allows for the best separation between the species. Using these variables a classifier can be obtained that can be used to classify flowers (of which the species are unknown) into any one of the groups using only these variables.
- A financial researcher in a bank may use financial information to classify a loan applicant as high risk or low risk.
- A medical researcher may record different variables relating to patients' backgrounds in order to learn which variables best predict whether a patient is likely to recover completely, partially, or not at all, from a certain medical condition, *e.g.* type of cancer or maybe a heart attack.
- An educational researcher could use the high school marks of students to predict whether a student will be successful or unsuccessful at university.

Given the many discrimination techniques, there is no single one that is always superior to the others. There are a few important factors that influence the choice of a discriminant technique. Such factors are the area of research (biology, finance, medicine, *etc*.), the interpretability of the discriminant results, flexibility of the technique (non-linearity), type of input variables (numerical, categorical or mixed) and robustness to outliers.

1.2 Kernel-based techniques

An interesting development has taken place in multivariate statistical analysis over the past two decades. A class of kernel-based techniques has been developed, where ordinary linear multivariate techniques such as principal component analysis, cluster analysis and discriminant analysis are transformed into powerful non-linear techniques by using kernel functions. Kernel-based techniques which are not extensions of existing linear statistical methods have also been developed.

Probably the most well-known kernel-based technique is the support vector machine (SVM) (*cf.* Boser *et al.*, 1992). Since the introduction of the SVM, many other kernel-based techniques have been developed, which include kernel principal component analysis (*cf.* Shawe-Taylor and Cristianini, 2004), kernel-based clustering (*cf.* Girolami, 2002), kernel canonical correlation analysis (*cf.* Kuss and Graepel, 2003), the relevance vector machine (*cf.* Tipping, 2000), one-class classification techniques (*cf.* Tax and Duin, 1999), least squares support vector machines (*cf.* Suykens and Vandewalle, 1999) and kernel Fisher discriminant analysis (KFDA) (Mika *et al.*, 1999). The research in this thesis will focus mainly on KFDA as introduced by Mika *et al.* (1999) and Mika (2002).

Basically, KFDA can be seen as a non-linear generalization of Fisher's linear discriminant analysis (FLDA) proposed by Fisher (1936). The idea of FLDA is to obtain a linear discriminant function by maximizing the ratio of the between-class to within-class variation obtained from the training data. Such a function discriminates between the classes in the training data and from this function a linear classifier, that can be used to classify observations, can be derived. KFDA is performed by applying FLDA in a high dimensional feature space. First, the training data are mapped from input space into this high dimensional feature space via a non-linear mapping function. In feature space a linear discriminant function is obtained by maximizing the ratio of the between-class to within-class variation obtained from the mapped data. Performing explicit calculations in feature space is impossible since the dimension of this space is very high or even infinite. An important ingredient, in order to perform the calculations in feature space, is the so-called kernel function. Using an appropriate kernel function, a non-linear classifier can be constructed in input space. In this way, KFDA becomes a non-linear version of FLDA with the kernel function playing an important role in obtaining the non-linear classifier.

Since its introduction, KFDA has been successfully applied to practical problems with a generalization performance similar (often superior) to other discrimination techniques (*cf.* Mika *et al.*, 1999; Louw and Steel, 2005). In Chapter 2 we explain in more detail how the KFD classifier is obtained.

1.3 The influence of observations and aims of the thesis

The influence of observations on various aspects of discriminant and regression techniques has been studied by many researchers. In regression analysis the influence of single observations has been studied by Cook (1977), Pregibon (1981), Léger and Altman (1993) and Steel and Uys (2006). Similar studies have been done in discriminant analysis by Campbell (1978), Fung (1995a, 1995b, 1995c, 1996), Johnson (1987), Critchley and Vitiello (1991) and many others. The aim of these studies was to assess the influence of observations on certain aspects of the model as well as to develop criteria which can be used to identify influential cases. The following are reasons why the identification of influential cases as well as the investigation of such cases are important:

- From a statistical modelling perspective, influential cases affect the estimation of model parameters, model selection as well as the generalization performance of the model.
- From a practical perspective, influential cases could represent outliers (atypical cases), human errors, experimental errors or even a shift in the distribution of the data.

Aims of the thesis:

The research in this thesis is quite similar to the research in the references given above, in the sense that we will study the influence of observations in KFDA.

- The first aim is to study the effect of atypical cases on various geometrical aspects of KFDA which include the generalization performance of the KFD classifier. The generalization performance is a measure of the classifier’s ability to classify observations of unknown group-membership into a known population. The classification error rate will be used to measure the generalization performance of the KFD classifier. A low error rate corresponds to a good classification performance and vice versa.

- The motivation for this study is to develop criteria which can be used to identify cases that have a detrimental effect (negative) on the KFD generalization performance, *i.e.* increases the error rate. Several such criteria are proposed and its effectiveness is evaluated in an extensive Monte Carlo simulation study as well as on real-world data sets.
- One of the disadvantages of calculating criteria on a leave-one-out approach is that the analysis becomes computationally prohibitive in large data sets. In this thesis we therefore also propose a filtering or pre-screening procedure with which a subset of observations can be obtained from the training data. By evaluating only the subset of observations in each criterion, computations can be reduced considerably, and the proposed criteria can be applied to large data sets more effectively. The effectiveness of the filter will be evaluated in a Monte Carlo simulation study and on real-world data sets.

1.4 Issues concerning the Monte Carlo simulation studies

As already mentioned above, all the investigations in this thesis are done via simulation studies. The selection of the configurations in each study are arbitrary but were selected to ensure a large number of variations in our investigations. However, the following should be taken into count when interpreting the simulation studies.

- The configurations of the simulation studies in Section 3.7 and 5.4 are exactly the same.
- The configurations for the study in Section 6.6 differ slightly from these two simulations studies. In Sections 3.7 and 5.4 small and mixed samples are used, while in Section 6.6 these samples are replaced with large samples. This alteration is

necessary since Chapter 6 tackles the computational problem encountered with large samples. The simulation designs in Sections 3.7, 5.4 and 6.6 are consistent and therefore comparable.

- The investigations in Chapter 4 do not form part of the main aim of the thesis. The concluding results of this chapter are merely used in further Chapters. The simulation studies in Sections 4.2.2 and 4.4.2 should be seen as independent investigations and therefore do not have the same designs as was used in Sections 3.7, 5.4 and 6.6.

1.5 Outline of the thesis

In Chapter 2 we start with an in-depth explanation of how the KFD classifier is obtained. We first review two-class FLDA and then show that KFDA is a non-linear generalization of FLDA. We also review some of the definitions and mathematics needed to perform calculations needed for KFDA and other kernel-based methods. A quadratic programming formulation of KFDA is also described. This chapter ends with a short review of other discrimination techniques related to KFDA. Two related techniques, the SVM and regularized kernel Fisher discriminant analysis, are then explained. Multi-class KFDA is also discussed and some of the advantages and disadvantages of KFDA are mentioned.

Chapter 3 contains an extensive Monte Carlo simulation study of the effect of atypical cases on aspects of KFDA. We start by defining these aspects, which include the training error, test error, average distance of misclassified training cases to the decision boundary, average margin, Rayleigh coefficient, between-class to within-class variance ratio, the alignment and the length of vector $\mathbf{v}$ (which represents a direction of discrimination). This is followed by a discussion of the distributions from which the data will be generated for the simulation studies as well as how the atypical cases are generated. The estimation of the hyperparameters in KFDA is also addressed in this chapter.

In Chapter 4 we explain how posterior probabilities can be estimated by using the KFDA output. Two methods are discussed. The first method makes use of the so-called projections and the assumption that these projections (for each class) follow a normal distribution. We perform a Monte Carlo simulation study to evaluate this normality assumption and show how Bayes' rule can be used to estimate posterior probabilities using these projections. For the second method, a logistic regression is performed on the discriminant scores. The estimated posterior probabilities are then obtained from the logistic regression output. In this chapter we also compare the generalization performance of three KFD classifiers in a Monte Carlo simulation study. The posterior probabilities discussed in this chapter will also be used in Chapter 5 to develop another criterion to identify influential cases.

Chapter 5 focuses on the identification of cases that have a detrimental effect on the KFD generalization performance. In this chapter we start with a review of the literature on outlier diagnostics that are calculated on a leave-one-out basis. Nine criteria, each calculated using a leave-one-out approach, are proposed for the detection of cases that have a detrimental effect on the KFD generalization performance. The performance of each criterion is evaluated in an extensive Monte Carlo simulation study as well as on real-world data sets.

In Chapter 6 we address the computational challenge that is faced when calculating each criterion on a leave-one-out basis. We propose using a smallest enclosing hypersphere in feature space as a filter to obtain a subset of observations which are evaluated on a leave-one-out basis. The smallest enclosing hypersphere is obtained for each class separately. In this chapter we discuss the theory of the smallest enclosing hypersphere and explain how we propose using this procedure to filter the training data. We then evaluate the proposal by means of a Monte Carlo simulation study as well as on real-world data sets.

Chapter 7 is a general summary of the contributions made in this thesis.

1.6 Notation

The following list of symbols is used throughout the thesis

Symbol	Explanation
n, p, N	Number of observations in training data, number of variables in input space and number of variables in feature space
Π	A population
$\mathtt{m}, \boldsymbol{\Sigma}$	Mean vector and covariance matrix
$T\ , J_j$	Training data and index set for class j
$\mathbf{x}, \mathrm{y}$	An observation in input space and its label
$\mathrm{y} \in \{\pm 1\}$	Positive and negative labels used to indicate the two classes
$f(\mathbf{x})$	Linear discriminant function
$\varphi(\mathbf{x});\ sign\{f(\mathbf{x})\}$	A classifier
$\Re$	The set of real numbers
$\langle \mathbf{v}, \mathbf{x} \rangle$	Inner product between vectors
$X\ , F$	Input space and feature space
$\bar{\mathbf{x}}_j\ , \mathbf{S}_\mathbf{B}\ , \mathbf{S}_\mathrm{w}$	Mean vector, between-class and within-class scatter matrices in input space
$\tilde{\mathbf{v}} = (\tilde{\mathrm{v}}_1, \tilde{\mathrm{v}}_2, ..., \tilde{\mathrm{v}}_p)$	Optimal values of the weight vector in the FLDA classifier
Δ^2	Squared Mahalanobis distance
$\Phi : X \to F$	Mapping function, mapping from input space to feature space
$\Phi(\mathbf{x})$	Mapped data
$k(\mathbf{x}, \mathbf{z})$	The kernel function
$\bar{\mathbf{x}}_j^\Phi\ , \mathbf{S}_\mathbf{B}^\Phi\ , \mathbf{S}_\mathrm{w}^\Phi$	Mean vector, between-class and within-class scatter matrices in feature space
$\lVert\mathbf{x}\rVert = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$	Euclidean norm or length of a vector
$\gamma\ , \lambda\ , C$	Hyperparameters of a kernel-based technique
$\tilde{\boldsymbol{\alpha}} = (\tilde{\alpha}_1, \tilde{\alpha}_2, ..., \tilde{\alpha}_n)$	Optimal values of the weight vector in the KFDA and SVM

	classifiers
$\mathbf{G}$	Gram (kernel) matrix
$\mathbf{M}$, $\mathbf{N}$	Kernelized between-class and within-class scatter matrices
$\mathbf{N}_\lambda$	Regularized within-class scatter matrix
$\mathbf{I}$, $\mathbf{1}_{n_j}$	Identity matrix and unit matrix
$\tilde{a}$, $\tilde{b}$, $\tilde{c}$, $\tilde{d}$	Optimal thresholds for FLDA, KFDA, SVM and regKFDA respectively
$2/\lVert\mathbf{w}\rVert$	Margin for SVM
ξ	Slack variable used in the SVM
$\min[.]$; $\max[.]$	An optimization problem
L_P, L_D	Primal and dual Lagrangians
$\tilde{\mathrm{b}} = (\tilde{\beta}_1, \tilde{\beta}_2, ..., \tilde{\beta}_n)$	Optimal values of the weight vector in regKFDA classifier
$I[.]$	An indicator function
R_{test}, R_{emp}	Test and training error rate
$\bar{d}$	Average distance of misclassified training cases to the KFDA decision boundary
$\bar{m}$	Average margin
r	Rayleigh coefficient
v	Between-class to within-class variance ratio
a	Alignment
q_i	Projections in feature space
$\hat{\mathrm{p}}(\mathrm{y} = +1 \mid \mathrm{x})$	Estimated posterior probability
δ	Discriminant score in feature space
$\bar{f}$	Average squared difference between the discriminant scores
$\bar{h}$	Average squared difference between 1 and the estimated posterior probability
$\alpha_1^*, \alpha_2^*, ..., \alpha_n^*$	Optimal values for the parameters of the hypersphere

CHAPTER 2

KERNEL FISHER DISCRIMINANT ANALYSIS

"To isolate mathematics from the practical demands of the sciences is to invite the sterility of a cow shut away from the bulls."

– Pafnuty Chebyshev

2.1 A framework for two-class classification

In this section a framework for two-class classification is given and the notation that will be used throughout the remainder of the thesis is introduced. Consider two populations represented by Π_1, with mean vector $\boldsymbol{\mu}_1$ and covariance matrix $\boldsymbol{\Sigma}_1$, and Π_2 with mean vector $\boldsymbol{\mu}_2$ and covariance matrix $\boldsymbol{\Sigma}_2$, respectively. We observe a binary response variable $\mathrm{Y} \in \{\pm 1\}$, together with p input variables $\mathrm{X}_1, \mathrm{X}_2, ..., \mathrm{X}_p$. These variables are measured for $n_1 + n_2 = n$ observations, with the first n_1 observations from Π_1 (corresponding to $\mathrm{Y} = +1$) and the remaining n_2 observations from Π_2 (corresponding to $\mathrm{Y} = -1$). We denote the set of indices corresponding to the observations from Π_1 by $J_1 = \{1, 2, ..., n_1\}$, and the set corresponding to the observations from Π_2 by $J_2 = \{n_1 + 1, n_1 + 2, ..., n\}$. The resulting training data will be denoted by $T = \{(\mathbf{x}_i, \mathrm{y}_i), i = 1, ..., n\}$, where $\mathbf{x}_i \in \Re^p$ represents the observed input variables and $\mathrm{y}_i \in \{\pm 1\}$ represents the observed response for the *i*-th sample case. The objective is to use T to construct a function, $f(\mathbf{x})$, called a discriminant function, so that $sign\{f(\mathbf{x})\}$ can be used to assign a new observation with observed values for the input variables in the vector $\mathbf{x}$ to one of the two populations. We refer to $sign\{f(\mathbf{x})\}$ as a classifier.

One way to construct $f(\mathbf{x})$ is by using a linear function, *i.e.*

$$f(\mathbf{x}) = \langle \mathrm{n}, \mathbf{x} \rangle + \beta . \tag{2.1}$$

In (2.1) the constant β and the vector n are unknown parameters which have to be estimated from the training data, while $\langle \mathrm{n}, \mathbf{x} \rangle$ denotes the inner product between n and $\mathbf{x}$. Some of the advantages of using (2.1) are the following:

- it is a simple linear function and there are only $p+1$ parameters that have to be estimated;
- the parameter values are easy to interpret;
- it often produces good classification results because of its simplicity and consequent low variance;
- it reduces the risk of over-fitting, especially when p is large relative to n;
- if the training data are generated from normal distributions with a common population covariance matrix, (2.1) can be estimated by using Bayes' rule.

However, a linear function is not always the best function when performing a discriminant analysis. One of the limitations of the linear function is that the resulting decision boundary $\{\mathbf{x} : f(\mathbf{x}) = 0\}$ is linear, which is inadequate for situations where a non-linear decision boundary is required.

In the sections that follow we will discuss several classification techniques that make use of a linear discriminant function. In Section 2.2 the well known Fisher's linear discriminant analysis (FLDA) is reviewed and other related techniques are discussed. In Section 2.3 we start by addressing the inadequacy of the linear decision boundary produced by FLDA. In Section 2.3.1 the basic idea of performing FLDA in a high-dimensional feature space, *i.e.* a non-linearly transformed input space, is explained. The problems encountered when doing this, are also discussed. Section 2.3.2 provides the necessary mathematical tools to overcome these problems. Use of the so-called "kernel trick" is also explained in Section 2.3. In Section 2.4 we discuss kernel Fisher discriminant analysis (KFDA), a non-linear extension of FLDA. By using an appropriate kernel function, KFDA produces a non-linear decision boundary in input space, thus overcoming the inadequacy of the linear decision boundary. In Section 2.5, KFDA is formulated as the solution to a quadratic optimization problem. In Section 2.6 we discuss other kernel based techniques (such as the SVM) related to KFDA. Multi-class KFDA is addressed in Section 2.7 and some of the advantages and disadvantages of KFDA are highlighted in Section 2.8.

2.2 Fisher's linear discriminant analysis

2.2.1 Fisher's linear discriminant function

Let X denote the input space, an Euclidean space, such that the observed vectors $\mathbf{x} \in X \subseteq \Re^p$. The basic idea of FLDA is to find the linear discriminant function

$$f(\mathbf{x}) = \langle \mathbf{v}, \mathbf{x} \rangle + a \tag{2.2}$$

in X, where the parameter vector $\mathbf{v} \in \Re^p$ can be estimated by maximizing the so-called Rayleigh coefficient

$$J(\mathbf{v}) = \frac{\mathbf{v}'\boldsymbol{\Sigma}_{\mathbf{B}}\mathbf{v}}{\mathbf{v}'\boldsymbol{\Sigma}_{\mathbf{w}}\mathbf{v}}. \tag{2.3}$$

The matrices $\boldsymbol{\Sigma}_{\mathbf{B}}$ and $\boldsymbol{\Sigma}_{\mathbf{w}}$ are called the population between-class and within-class scatter matrices respectively. The vector $\mathbf{v}$ that maximizes (2.3) corresponds to the direction along which the classes are maximally separated when the observations are projected onto this direction (*cf.* Figure 2.1). In practice $\boldsymbol{\Sigma}_{\mathbf{B}}$ and $\boldsymbol{\Sigma}_{\mathbf{w}}$ are unknown and are usually estimated from the training data. The sample mean vectors

$$\bar{\mathbf{x}}_1 = \frac{1}{n_1}\sum_{i\in J_1}\mathbf{x}_i \text{ and } \bar{\mathbf{x}}_2 = \frac{1}{n_2}\sum_{i\in J_2}\mathbf{x}_i \tag{2.4}$$

are used as estimates for $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, while $\mathbf{S_B}$ and $\mathbf{S_w}$ are used as estimates for $\boldsymbol{\Sigma}_\mathbf{B}$ and $\boldsymbol{\Sigma}_\mathbf{w}$ respectively. These sample between-class and within-class scatter matrices can be obtained as

$$\mathbf{S_B} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)' \text{ and } \mathbf{S_w} = \sum_{j=1}^{2}\sum_{i\in J_j}(\mathbf{x}_i - \bar{\mathbf{x}}_j)(\mathbf{x}_i - \bar{\mathbf{x}}_j)'. \tag{2.5}$$

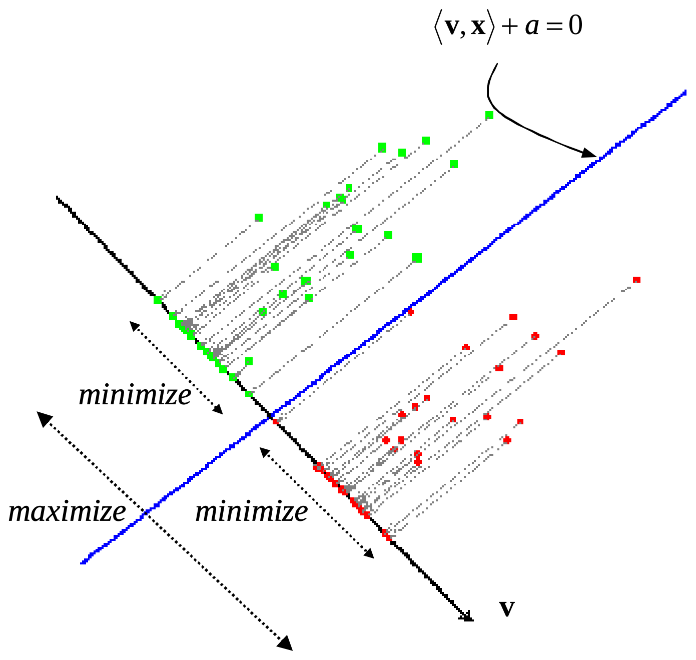


FIGURE 2.1: *Schematic representation of the FLDA classifier in input space. The between/ within class variation ratio is maximized along the direction of discrimination represented by* $\mathbf{v}$. *The solid blue line represents the decision boundary.*

By redefining (2.3) in terms of the sample estimates, the sample Rayleigh coefficient is

$$J(\mathbf{v}) = \frac{\mathbf{v}'\mathbf{S}_{\mathbf{B}}\mathbf{v}}{\mathbf{v}'\mathbf{S}_{\mathbf{w}}\mathbf{v}}. \tag{2.6}$$

To find an estimate of the vector $\mathbf{v}$ in (2.2), we maximize (2.6). The estimate of $\mathbf{v}$ will be denoted by $\tilde{\mathbf{v}}$ and its components $(\tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_p)$ can be obtained by making use of the following maximization lemma which is an extension of the Cauchy-Schwarz inequality.

MAXIMIZATION LEMMA:
For any positive definite matrix **B** and vector **d**, it can be shown that

$$\max_{\mathbf{y}\neq 0}\left[\frac{\mathbf{y}'(\mathbf{d}\mathbf{d}')\mathbf{y}}{\mathbf{y}'\mathbf{B}\mathbf{y}}\right] = \mathbf{d}'\mathbf{B}^{-1}\mathbf{d}, \tag{2.7}$$

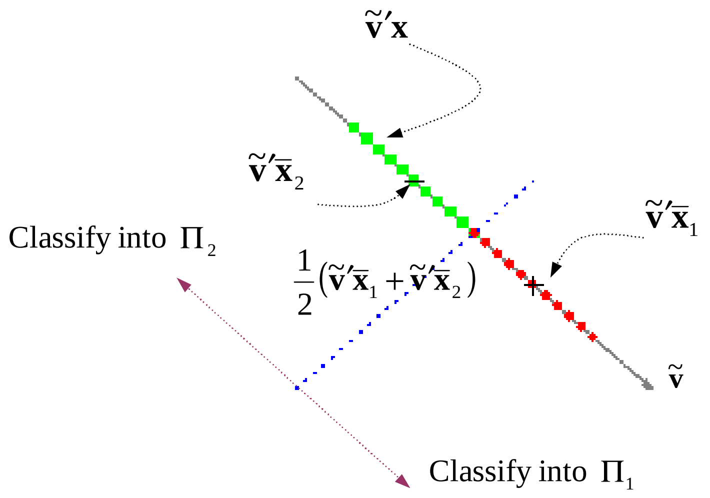
with maximum attained when $\mathbf{y} = c\mathbf{B}^{-1}\mathbf{d}$ for any constant $c \neq 0$.

Proof: See Johnson and Wichern (1992, *p*.65). ■

By replacing $\mathbf{B}^{-1}$ and $\mathbf{d}$ in (2.7) with $\mathbf{S}_{\mathbf{w}}^{-1}$ and $(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$, we obtain the optimal **v** as

$$\tilde{\mathbf{v}} = c\mathbf{S}_{\mathbf{w}}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2). \tag{2.8}$$

The constant c does not change the direction of discrimination, but merely influences the length of $\tilde{\mathbf{v}}$. We therefore choose $c = 1$. The scalar $a \in \Re$ in (2.2) is known as the intercept or threshold. To obtain an estimate of this threshold we will use the following procedure, which is also illustrated in Figure 2.2. Firstly, project the mean vectors, $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$, onto the direction $\tilde{\mathbf{v}}$. Secondly, calculate the midpoint between these projected



$\tilde{\mathbf{v}}'\mathbf{x}$
$\tilde{\mathbf{v}}'\bar{\mathbf{x}}_2$
$\tilde{\mathbf{v}}'\bar{\mathbf{x}}_1$
Classify into Π_2
$\frac{1}{2}(\tilde{\mathbf{v}}'\bar{\mathbf{x}}_1+\tilde{\mathbf{v}}'\bar{\mathbf{x}}_2)$
$\tilde{\mathbf{v}}$
Classify into Π_1

FIGURE 2.2: *Schematic illustration of how the threshold, a, is obtained for FLDA. The observations in each class, as well as their means* $(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2)$ *are projected onto the line* $\tilde{\mathbf{v}}$. *The dotted blue line represents the midpoint between class means, which is then used to obtain the threshold.*

means as follows:

$$\frac{1}{2}(\tilde{\mathbf{v}}'\bar{\mathbf{x}}_1+\tilde{\mathbf{v}}'\bar{\mathbf{x}}_2)=\frac{1}{2}\tilde{\mathbf{v}}'(\bar{\mathbf{x}}_1+\bar{\mathbf{x}}_2)$$

$$=\frac{1}{2}(\bar{\mathbf{x}}_1-\bar{\mathbf{x}}_2)'\mathbf{S}_{\mathrm{w}}^{-1}(\bar{\mathbf{x}}_1+\bar{\mathbf{x}}_2). \tag{2.9}$$

It is often the case that classes do not have the same number of observations. To take into consideration the difference in the sample sizes, we add the factor $log\left({}^{n_2}\!/_{n_1}\right)$ to the midpoint. This factor is zero when $n_1 = n_2$, positive when $n_1 < n_2$ and negative when $n_1 > n_2$. The notion of using this factor in FLDA comes from the normal linear discriminant rule which is derived for two normal populations with equal covariance matrices and unequal prior probabilities (*cf.* Section 2.2.2).

Thus, expression (2.9) becomes $\frac{1}{2}\left(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2\right)' \mathbf{S}_{\mathbf{w}}^{-1}\left(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2\right) + log\left({}^{n_2}\!/_{n_1}\right)$, which we then use to classify $\mathbf{x}$ into Π_1 if

$$\tilde{\mathbf{v}}'\mathbf{x} \geq \frac{1}{2}\left(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2\right)' \mathbf{S}_{\mathbf{w}}^{-1}\left(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2\right) + log\left({}^{n_2}\!/_{n_1}\right), \qquad (2.10)$$

and else into Π_2. Using the estimated discriminant function, $f(\mathbf{x}) = \langle \tilde{\mathbf{v}}, \mathbf{x} \rangle + \tilde{a}$, the classifier above can be reconstructed as

$$\varphi(\mathbf{x}) = sign\{\langle \tilde{\mathbf{v}}, \mathbf{x} \rangle + \tilde{a}\}, \qquad (2.11)$$

with

$$\tilde{a} = -\frac{1}{2}\left(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2\right)' \mathbf{S}_{\mathbf{w}}^{-1}\left(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2\right) - log\left({}^{n_2}\!/_{n_1}\right). \qquad (2.12)$$

If $\varphi(\mathbf{x})$ is positive, the observation $\mathbf{x}$ is assigned to the $+1$ class, otherwise to the -1 class.

2.2.2 Methods related to FLDA

Two-class FLDA is known to be related to several other methods. We briefly mention a few of these methods in this section.

It is possible to prove that there is a connection between FLDA and least squares regression (*cf.* Mika, 2002, and Herbrich, 2001). In linear regression problems we estimate the linear function $\hat{y} = \langle \boldsymbol{\beta}, \mathbf{x} \rangle + \beta_0$, such that the difference between the predicted values $\hat{y}$, and the observed values y is small. This is done by minimizing the sum of squared errors (SSE) defined as $\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ with respect to $\boldsymbol{\beta}$ and β_0. Let $\mathbf{X}$ denote the training data without the response variable and let $\mathbf{y}$ denote the response vector. In matrix form we can write the SSE as

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \left\| [\mathbf{X} \quad \mathbf{1}] \begin{bmatrix} b \\ \beta_0 \end{bmatrix} - \mathbf{y} \right\|^2 .$$

It can be shown that the vector b which minimizes the SSE is equal to $\hat{b} = \omega \mathbf{S}_w^{-1} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$, which lies in the same direction as the discriminant vector given by (2.8). The symbol ω represents a scalar.

The leading eigenvector of the matrix $\mathbf{S}_w^{-1}\mathbf{S}_\mathbf{B}$ is also related to (2.8). It is possible to prove that the eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_\mathbf{B}$, which corresponds to its largest (non-zero) eigenvalue, also lies in the same direction as (2.8) (*cf.* Härdle and Simar, 2003; Mika, 2002, *p.*34). Since we can write $\mathbf{S}_\mathbf{B} = \mathbf{dd}'$, where $\mathbf{d} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$, it follows that $\mathbf{S}_\mathbf{B}$ has a rank of one. $\mathbf{S}_w^{-1}\mathbf{S}_\mathbf{B}$ only has one non-zero eigenvalue[1] and the corresponding eigenvector can be obtained as $\mathbf{S}_w^{-1}\mathbf{d} = \mathbf{S}_\mathbf{w}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$, which is the same as (2.8) with $c = 1$.

Another connection exists between FLDA and the well-known normal linear discriminant analysis (LDA) (*cf.* Johnson and Wichern, 1992; Anderson, 2003). Assume that the training data are generated from two multivariate normal distributions with equal covariance matrices, *i.e.* with density functions

[1] Here we use the result, rank($\mathbf{AB}$) $\leq$ min{rank($\mathbf{A}$), rank($\mathbf{B}$)}

$$f_j(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\left|\mathbf{\Sigma}_j\right|^{1/2}} exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_j\right)'\mathbf{\Sigma}_j^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_j\right)\right\}, \text{ for } j\text{=1, 2,}$$

and $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}$. By using Bayes' rule we obtain the following posterior probabilities of class membership for case $\mathbf{x}$,

$$\mathrm{p}(j\,|\,\mathbf{x}) = \pi_j f_j(\mathbf{x}) \Big/ \sum_{k=1}^{2} \pi_k f_k(\mathbf{x}), \text{ for } j\text{=1, 2.}$$

Here π_j, j =1, 2, represents the prior probabilities of class membership which is often estimated from the data as $n_j/(n_1+n_2)$. If $\mathrm{p}(j=1\,|\,\mathbf{x}) > \mathrm{p}(j=2\,|\,\mathbf{x})$, then we classify $\mathbf{x}$ into Π_1, otherwise we classify $\mathbf{x}$ into Π_2. From these posterior probabilities, the following equivalent classification rule is obtained:

Classify $\mathbf{x}$ into Π_1 if

$$-\frac{1}{2}(\mathbf{x}-\mathtt{m}_1)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathtt{m}_1)+\frac{1}{2}(\mathbf{x}-\mathtt{m}_2)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathtt{m}_2)-log\left(\pi_2/\pi_1\right)\geq 0,$$

and into Π_2 otherwise. By replacing the population parameters $\mathtt{m}_j$, $\mathbf{\Sigma}$ and π_j with their sample estimates $\bar{\mathbf{x}}_j$, $\mathbf{S}_{\text{pooled}} = \mathbf{S}_{\text{w}}/(n-2)$ and $n_j/(n_1+n_2)$ respectively, the following sample classification rule can be constructed (*cf*. Johnson and Wichern, 1992):

Classify $\mathbf{x}$ into Π_1 if

$$\left\langle \mathbf{S}^{-1}_{\text{pooled}}(\bar{\mathbf{x}}_1-\bar{\mathbf{x}}_2),\mathbf{x}\right\rangle \geq \frac{1}{2}(\bar{\mathbf{x}}_1-\bar{\mathbf{x}}_2)'\mathbf{S}^{-1}_{\text{pooled}}(\bar{\mathbf{x}}_1+\bar{\mathbf{x}}_2)+log\left(n_2/n_1\right), \tag{2.13}$$

and into Π_2 otherwise. The correspondence between this inequality and the classifier (2.10) is evident.

Finally, in Härdle and Simar (2003) a relationship between FLDA and the so-called maximum likelihood rule is shown. Assuming that we have two normal populations with equal covariance matrices, the likelihood for population j is

$$L_j(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\}, \text{ for } j=1, 2.$$

It is clear that the likelihood $L_j(\mathbf{x})$ will be maximized when the Mahalanobis distance between $\mathbf{x}$ and $\boldsymbol{\mu}_j$, $\Delta_j^2 = (\mathbf{x}-\boldsymbol{\mu}_j)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)$, is minimized. This results in the following maximum likelihood classification rule:

Classify $\mathbf{x}$ into Π_1 when $L_1(\mathbf{x}) > L_2(\mathbf{x})$, otherwise it is classified into Π_2. This is equivalent to classifying $\mathbf{x}$ into Π_1 when

$$(\mathbf{x}-\boldsymbol{\mu}_1)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) < (\mathbf{x}-\boldsymbol{\mu}_2)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2),$$

else into Π_2. The inequality above can now be written as

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

and by replacing the population parameters $\boldsymbol{\mu}_j$ and $\mathbf{\Sigma}$ with the sample estimates $\bar{\mathbf{x}}_j$ and $\mathbf{S}_{\text{pooled}}$, the classification rule becomes

$$\left\langle \mathbf{S}_{\text{pooled}}^{-1}(\bar{\mathbf{x}}_1-\bar{\mathbf{x}}_2), \mathbf{x}\right\rangle \geq \frac{1}{2}(\bar{\mathbf{x}}_1-\bar{\mathbf{x}}_2)'\mathbf{S}_{\text{pooled}}^{-1}(\bar{\mathbf{x}}_1+\bar{\mathbf{x}}_2), \tag{2.14}$$

which closely resembles the classification rule defined in (2.10).

2.3 Extending FLDA to feature space

2.3.1 The basic idea

Although FLDA is a very useful and powerful discriminant technique, it certainly has some limitations. As mentioned in Section 2.1, using a linear discriminant function restricts us to linear decision boundaries. There are often cases when classes are not well separated by a linear decision boundary. In such cases a non-linear decision boundary is needed and the result is often a considerably improved classification performance. There are several classification techniques that produce non-linear decision boundaries, for example *k*-nearest neighbours, classification trees, artificial neural networks and quadratic discriminant analysis (*cf.* Hastie *et al*., 2001). These techniques do not fit into the framework of the linear discriminant function and the advantages mentioned in Section 2.1 are therefore not applicable. However, Mika *et al.* (1999) proposed a technique which transforms FLDA into a powerful non-linear discriminant technique. They perform FLDA in a high-dimensional space, known as the feature space and call this technique kernel Fisher discriminant analysis (KFDA). It is derived as follows.

Let Φ be a non-linear mapping function and $F \subseteq \Re^N$ $(N \leq \infty)$ be some high-dimensional feature space. By applying the mapping $\Phi : X \rightarrow F$, the training data $\{\mathbf{x}_i, i = 1,...,n\}$ in input space are mapped into feature space, where FLDA is then performed on the transformed training data $\{\Phi(\mathbf{x}_i), i = 1,...,n\}$. This yields a linear decision boundary in F which, because of the non-linear transformation Φ, corresponds to a non-linear decision boundary in X. Figure 2.3 is an illustration of this basic idea.

The linear discriminant function in F is

$$f(\mathbf{x}) = \langle \mathbf{v}, \Phi(\mathbf{x}) \rangle + b\,. \qquad (2.15)$$

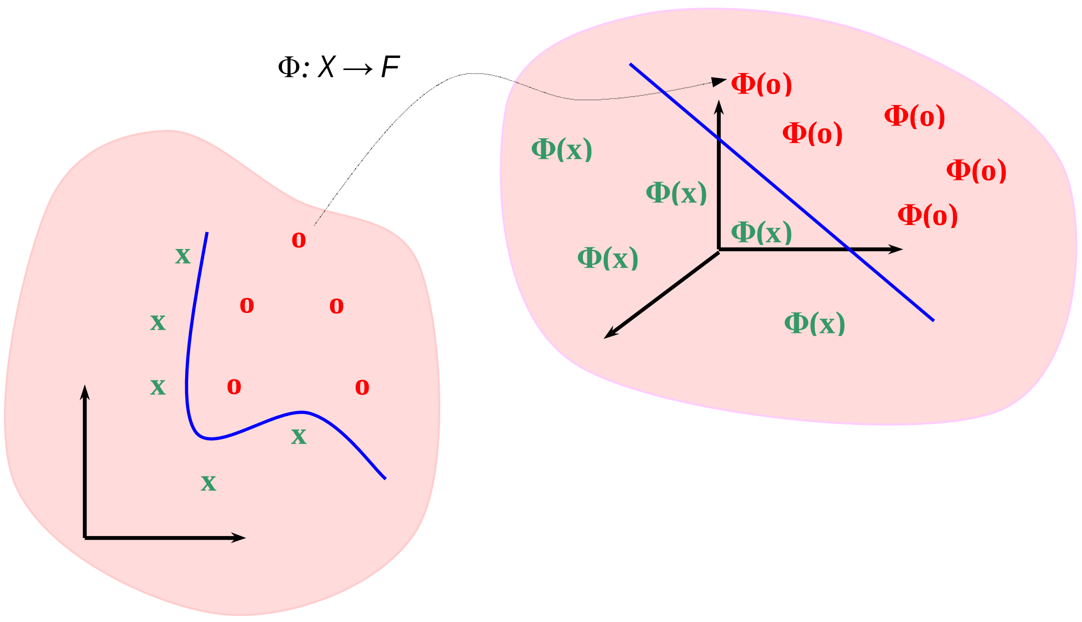


FIGURE 2.3: *Schematic illustration of the mapping* $\Phi : \Re^2 \to \Re^3$. *The non-linear decision boundary in a low dimensional input space corresponds to a linear decision boundary in a higher-dimensional feature space.*

The parameter vector $\mathbf{v} \in \Re^N$ is obtained by maximizing the Rayleigh coefficient in feature space, *i.e.* maximizing

$$J(\mathbf{v}) = \frac{\mathbf{v}'\mathbf{S}_{\mathbf{B}}^{\Phi}\mathbf{v}}{\mathbf{v}'\mathbf{S}_{\mathbf{w}}^{\Phi}\mathbf{v}}, \tag{2.16}$$

where $\mathbf{S}_{\mathbf{B}}^{\Phi}$ and $\mathbf{S}_{\mathbf{w}}^{\Phi}$ respectively correspond to the sample between-class and within-class scatter matrices in F. Using the transformed data these matrices can be expressed as

$$\mathbf{S}_{\mathbf{B}}^{\Phi} = \left(\overline{\mathbf{x}}_1^{\Phi} - \overline{\mathbf{x}}_2^{\Phi}\right)\left(\overline{\mathbf{x}}_1^{\Phi} - \overline{\mathbf{x}}_2^{\Phi}\right)' \text{ and } \mathbf{S}_{\mathbf{w}}^{\Phi} = \sum_{j=1}^{2}\sum_{i \in J_j}\left(\Phi(\mathbf{x}_i) - \overline{\mathbf{x}}_j^{\Phi}\right)\left(\Phi(\mathbf{x}_i) - \overline{\mathbf{x}}_j^{\Phi}\right)', \tag{2.17}$$

with

$$\bar{\mathbf{x}}_j^{\Phi} = \frac{1}{n_j} \sum_{i \in J_j} \Phi(\mathbf{x}_i) \text{ for } j=1, 2. \tag{2.18}$$

The construction of (2.15) raises the following questions:

- Which non-linear transformation should be used when mapping the data into feature space?
- How large is the dimension of the feature space, and if the dimension of this space is very large or infinite, how will the calculation of (2.16), (2.17) and (2.18) be performed?

To overcome these potential computational problems, kernel functions are used (*cf.* Section 2.3.2). These functions possess special properties derived from mathematical results from the theory of reproducing kernel Hilbert spaces and Mercer's theorem. As will be shown in Section 2.4, these results are essential in order to perform calculations in F. The results and definitions which are discussed in the next section are therefore important preliminaries for the derivation of the classifier in KFDA.

2.3.2 Feature space mathematics

In this section we consider the basic mathematical definitions, theorems and properties needed to do calculations in feature space. The most essential ingredient in the derivation of the classifier in KFDA is the kernel function. Literature dealing with the theory and applications of kernel functions include Kolmogorov (1941), Aronszajn (1950), and Aizerman *et al.* (1964). More recent literature on kernel functions are Boser *et al.* (1992), Shawe-Taylor and Cristianini (2004), Herbrich (2001), and Schölkopf and Smola (2002). Use of the kernel function in statistical classification problems was introduced by Boser *et al.* (1992) when they proposed the Support Vector Machine (*cf.* Section 2.6.2). Mika *et al.* (1999) kernelized FLDA by following a similar idea.

The kernel function is defined as follows.

DEFINITION 2.1 (Kernel function):

For a given feature mapping Φ from finite input space X, a kernel function is the inner product function

$$k(\mathbf{x},\mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$$

for all $\mathbf{x}, \mathbf{z} \in X$. ■

If the algorithm which we apply to construct a classifier in feature space has the property that the feature space vectors $\Phi(\mathbf{x})$ appear in the algorithm only in the form of inner products, we substitute the kernel $k(\mathbf{x},\mathbf{z})$ for $\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$. This is known as the "kernel trick" and it obviates explicit specification of the mapping function $\Phi(.)$ as well as calculations in feature space. In practice this is implemented by specifying and using a kernel without ever considering the implied mapping or the space which it induces. With regard to specifying a kernel, a special class of kernel functions that satisfy conditions defined by Mercer's theorem, will be used.

MERCER'S THEOREM:

Assume that the input space $X \subseteq \Re^p$ is a finite measurable space. Suppose the kernel function k is a real-valued, continuous, symmetric function, such that the integral operator T_k given by

$$(T_k f)(\mathbf{x}) = \int_X k(\mathbf{x},\mathbf{z}) f(\mathbf{z}) d\mathbf{z}$$

is positive semi-definite, *i.e.* for all f

$$\int_X \int_X k(\mathbf{x},\mathbf{z}) f(\mathbf{z}) f(\mathbf{x}) d\mathbf{z} d\mathbf{x} \geq 0 .$$

Let ψ_i be the normalized orthogonal eigenfunction of T_k associated with eigenvalues $\lambda_i \geq 0$. Then it can proved that

$$\begin{aligned} k(\mathbf{x},\mathbf{z}) &= \sum_{i=1}^{N} \sqrt{\lambda_i}\,\psi_i(\mathbf{x})\sqrt{\lambda_i}\,\psi_i(\mathbf{z}) \\ &= \sum_{i=1}^{N} \Phi_i(\mathbf{x})\Phi_i(\mathbf{z}) = \langle \Phi(\mathbf{x}),\Phi(\mathbf{z})\rangle \end{aligned} \tag{2.19}$$

for all $\mathbf{x},\mathbf{z} \in X$, where $N \leq \infty$ is the number of positive eigenvalues.

Proof: See Shawe-Taylor and Cristianini (2004, *p*.64); Mercer (1909).

■

To satisfy Mercer's conditions, a kernel function k should be symmetric, continuous and positive semi-definite. Table 2.1 contains examples of such kernel functions given in the literature (*cf*. Boser *et al*., 1992; Vapnik, 1995, and Herbrich, 2001). Each kernel function in Table 2.1 corresponds to a different number of dimensions for the feature space, as depicted in the last column. This will be illustrated in Section 2.3.3 with two examples. For each function we also have to estimate certain unknown hyperparameters, as shown in the middle column of Table 2.1. These parameters are generally estimated via cross-validation from the training data. This will be discussed further in Section 3.5. Another question that arises is why Mercer kernels are used? A Mercer kernel is also called a reproducing kernel and has the property that it corresponds to an inner product in a high-dimensional space, which is also a reproducing kernel Hilbert space (RKHS) (*cf*. Schölkopf, 1997, and Hamers, 2004). Thus the feature space in which we will perform FLDA is actually a Hilbert space[2]. Essentially, a Hilbert space can be seen as an infinite dimensional Euclidean space (*cf*. Jordan, 2004). A RKHS is defined in Mika (2002) as follows.

[2] For an introduction to Hilbert spaces see Debnath and Mikusiński (1999)

TABLE 2.1: Examples of kernel functions satisfying Mercer's conditions.

Name	Kernel function	Dim(F)
m-th degree polynomial	$k(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z} \rangle^{m}, \quad m \in N^{+}$	$\binom{n+m-1}{m}$
Complete polynomial	$k(\mathbf{x},\mathbf{z}) = (\langle \mathbf{x},\mathbf{z} \rangle + d)^{m}, \quad d \in \Re^{+},\ m \in N^{+}$	$\binom{n+m}{m}$
Gaussian	$k(\mathbf{x},\mathbf{z}) = exp\left(-\gamma \lVert \mathbf{x}-\mathbf{z} \rVert^{2}\right), \quad \gamma \in \Re^{+}$	∞
Mahalanobis	$k(\mathbf{x},\mathbf{z}) = exp\left(-(\mathbf{x}-\mathbf{z})' \boldsymbol{\Sigma}(\mathbf{x}-\mathbf{z})\right),$ $\boldsymbol{\Sigma} = \text{diag}\left(\sigma_1^{-2},\sigma_2^{-2},....,\sigma_p^{-2}\right),\ \left(\sigma_1^{-2},\sigma_2^{-2},....,\sigma_p^{-2}\right) \in \Re^{+}$	∞
Inverse multi-quadric	$k(\mathbf{x},\mathbf{z}) = \dfrac{1}{\sqrt{\lVert \mathbf{x}-\mathbf{z} \rVert^{2} + d^{2}}}, \quad d \in \Re^{+}$	∞
Sigmoidal	$k(\mathbf{x},\mathbf{z}) = tanh(\kappa \langle \mathbf{x},\mathbf{z} \rangle + \Theta), \quad \kappa, \Theta \in \Re^{+}$	∞

DEFINITION 2.2 (Reproducing Kernel Hilbert Space [RKHS]):

Let X be an input space and H a Hilbert space of functions $f : X \to \Re$. Then H is called a RKHS endowed with an inner product if there exists a function $k : X \times X \to \Re$ with the properties that k spans the whole of H and has the reproducing property, *i.e.* $\langle f, k(.,\mathbf{x}) \rangle = f(\mathbf{x})$ for all $f \in H$, in particular $\langle k(.,\mathbf{x}), k(.,\mathbf{z}) \rangle = k(\mathbf{x},\mathbf{z})$.

■

A Hilbert space constructed by using such a reproducing kernel is called a RKHS. We will now explain the connection between the mapping Φ obtained from (2.19) and the RKHS (*cf.* Wabha, 1973; Schölkopf, 1997). The following explanation is taken from Schölkopf (1997). Assuming that k is a Mercer kernel defined by (2.19), it is possible to construct an inner product such that k becomes a reproducing kernel for a Hilbert space of linear functions of the form

$$f(\mathbf{x}) = \sum_{j=1}^{\infty} a_j k(\mathbf{x}, \mathbf{x}_j) = \sum_{j=1}^{\infty} a_j \sum_{i=1}^{N} \sqrt{\lambda_i}\psi_i(\mathbf{x})\sqrt{\lambda_i}\psi_i(\mathbf{x}_j).$$

Furthermore, in Schölkopf (1997) it is shown that the above can be written as

$$f(\mathbf{x}) = \sum_{l=1}^{N} \alpha_l \sqrt{\lambda_l}\psi_l(\mathbf{x}), \tag{2.20}$$

where $\alpha_l = \sqrt{\lambda_l}\sum_{i=1}^{\infty} a_i \psi_l(\mathbf{x}_i)$. Suppose we have the function $g(\mathbf{z}) = \sum_{j=1}^{N} \beta_j \sqrt{\lambda_j}\psi_j(\mathbf{z})$, we then see that the structure of (2.19) is similar to the RKHS inner product,

$$\langle f, g \rangle = \sum_{l=1}^{N}\sum_{j=1}^{N} \alpha_l \beta_j \sqrt{\lambda_l}\psi_l(\mathbf{x})\sqrt{\lambda_j}\psi_j(\mathbf{z}).$$

This therefore implies that F becomes a Hilbert space when induced by a Mercer kernel.

To conclude our discussion on feature space mathematics, note that the following three constructions of a RKHS are equivalent:

- specifying a specific RKHS
- specifying a mapping $\Phi(.)$
- specifying a Mercer kernel $k(.,.)$

The last construction of a RKHS is preferred in all kernel based algorithms, since it makes calculations much easier. Finally, we overcome the questions raised in Section 2.3.1 by using a Mercer kernel to do implicit calculations in F.

2.3.3 Kernel functions and inner products

The following illustrative examples show that the kernel function corresponds to an inner product in a high-dimensional space. As shown in the previous section, Mercer kernels can be used to construct a RKHS which we denote by F. The kernels in Table 2.1 are all Mercer kernels and it is possible to show that they correspond to inner products in F. We will consider only the Gaussian and the *m*-th degree polynomial kernels and prove that they are indeed inner products in a high-dimensional space.

Example 1:

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 we have the simple case of $\mathrm{x} \in \Re^1$. Using the Taylor series expansion, $e^t = \sum_{i=0}^{\infty} \frac{t^i}{i!}$, the Gaussian kernel can now be written as

$$\begin{aligned}
k(\mathrm{x}_i, \mathrm{x}_j) &= exp\left(-\gamma\|\mathrm{x}_i - \mathrm{x}_j\|^2\right) \\
&= exp\left(-\gamma \mathrm{x}_i^2 + 2\gamma \mathrm{x}_i \mathrm{x}_j - \gamma \mathrm{x}_j^2\right) \\
&= exp\left(-\gamma \mathrm{x}_i^2 - \gamma \mathrm{x}_j^2\right) exp\left(2\gamma \mathrm{x}_i \mathrm{x}_j\right) \\
&= exp\left(-\gamma \mathrm{x}_i^2 - \gamma \mathrm{x}_j^2\right)\left(1 + \frac{2\gamma \mathrm{x}_i \mathrm{x}_j}{1!} + \frac{(2\gamma \mathrm{x}_i \mathrm{x}_j)^2}{2!} + \frac{(2\gamma \mathrm{x}_i \mathrm{x}_j)^3}{3!} + \ldots\right) \\
&= exp\left(-\gamma \mathrm{x}_i^2 - \gamma \mathrm{x}_j^2\right) \times \\
&\qquad \left(1.1 + \sqrt{\frac{2\gamma}{1!}}\mathrm{x}_i \sqrt{\frac{2\gamma}{1!}}\mathrm{x}_j + \sqrt{\frac{(2\gamma)^2}{2!}}\mathrm{x}_i^2 \sqrt{\frac{(2\gamma)^2}{2!}}\mathrm{x}_j^2 + \sqrt{\frac{(2\gamma)^3}{3!}}\mathrm{x}_i^3 \sqrt{\frac{(2\gamma)^3}{3!}}\mathrm{x}_j^3 + \ldots\right) \\
&= \left\langle \Phi(\mathrm{x}_i), \Phi(\mathrm{x}_j) \right\rangle.
\end{aligned}$$

From this we see that

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

is a mapping function that corresponds to a non-linear transformation of X. Similarly, we can prove the same for $\mathbf{x} \in \Re^p$. Thus working with the Gaussian kernel allows us to calculate inner products in $F \subseteq \Re^N$, where $N = \infty$. ■

Example 2:

The *m*-th degree polynomial kernel, given by

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

corresponds to the following inner product (*cf*. Schölkopf, 1997):

$$\begin{aligned} k(\mathbf{x}_i, \mathbf{x}_j) &= \left(\sum_{i,\, j=1}^{p} \mathrm{x}_i \mathrm{x}_j \right)^m \\ &= \left(\sum_{i_1,\, j_1=1}^{p} \mathrm{x}_{i_1} \mathrm{x}_{j_1} \right) \cdots \left(\sum_{i_m,\, j_m=1}^{p} \mathrm{x}_{i_m} \mathrm{x}_{j_m} \right) \\ &= \sum_{i_1,\, j_1=1}^{p} \cdots \sum_{i_m,\, j_m=1}^{p} \left(\mathrm{x}_{i_1} \ldots \mathrm{x}_{i_m}\right)\left(\mathrm{x}_{j_1} \ldots \mathrm{x}_{j_m}\right) \\ &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle. \end{aligned}$$

Again we see that the kernel function used in this example can be used to evaluate inner products in $F \subseteq \Re^N$. Using this kernel function we can show that the dimension of the feature space is equal to $N = (n+m-1)!/m!(n-1)!$. Since the dimension is dependent on both the degree of the polynomial and the sample size, N becomes very large when working with large samples. ■

2.4 Kernel Fisher discriminant analysis

Recall the discriminant function (2.15) and the Rayleigh coefficient (2.16). We will now proceed with the derivation of KFDA by using the mathematical results in Section 2.3. From the theory of RKHS we can write $\mathbf{v} \in F$ as a linear expansion of the mapped data (*cf.* Mika *et al.*, 1999), *viz.*

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i), \text{ with } \alpha_i \in \Re. \tag{2.21}$$

This implies that (2.15) can be written as

$$\begin{aligned} f(\mathbf{x}) &= \left\langle \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \right\rangle + b \\ &= \sum_{i=1}^{n} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + b \\ &= \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b. \end{aligned} \tag{2.22}$$

Since the inner product can be calculated by using a kernel function, implementing (2.22) requires values for $\alpha_1, \alpha_2, ..., \alpha_n$ and b. By using the expansion (2.21) and mean vector (2.18) we can write

$$\mathbf{v}'\overline{\mathbf{x}}_j^{\Phi} = \frac{1}{n_j} \sum_{i=1}^{n} \sum_{k \in J_j} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k) \rangle$$

$$
\begin{aligned}
&= \frac{1}{n_j}\sum_{i=1}^{n}\sum_{k\in J_j}\alpha_i k(\mathbf{x}_i,\mathbf{x}_k) \\
&= \sum_{i=1}^{n}\alpha_i \frac{1}{n_j}\sum_{k\in J_j} k(\mathbf{x}_i,\mathbf{x}_k) \\
&= \boldsymbol{\alpha}'\mathbf{m}_j\text{, for } j\text{=1, 2.} \qquad (2.23)
\end{aligned}
$$

The $n\times 1$ vector $\mathbf{m}_j$ is

$$
\mathbf{m}_j = \frac{1}{n_j}\begin{bmatrix} \sum_{k\in J_j} k(\mathbf{x}_1,\mathbf{x}_k) \\ \sum_{k\in J_j} k(\mathbf{x}_2,\mathbf{x}_k) \\ \vdots \\ \sum_{k\in J_j} k(\mathbf{x}_n,\mathbf{x}_k) \end{bmatrix}\text{, for } j\text{=1, 2,}
$$

where each element in the vector represents the sum of the inner products (which is replaced by the kernel function) between each case in the entire training data set and each case in class *j* only, divided by the number of cases in class *j*. The numerator in (2.16) can now be written as

$$
\begin{aligned}
\mathbf{v}'\mathbf{S}_{\mathbf{B}}^{\Phi}\mathbf{v} &= \mathbf{v}'\left(\overline{\mathbf{x}}_1^{\Phi}-\overline{\mathbf{x}}_2^{\Phi}\right)\left(\overline{\mathbf{x}}_1^{\Phi}-\overline{\mathbf{x}}_2^{\Phi}\right)'\mathbf{v} \\
&= \left(\mathbf{v}'\overline{\mathbf{x}}_1^{\Phi}-\mathbf{v}'\overline{\mathbf{x}}_2^{\Phi}\right)\left(\left(\overline{\mathbf{x}}_1^{\Phi}\right)'\mathbf{v}-\left(\overline{\mathbf{x}}_2^{\Phi}\right)'\mathbf{v}\right) \\
&= \left(\boldsymbol{\alpha}'\mathbf{m}_1-\boldsymbol{\alpha}'\mathbf{m}_2\right)\left(\mathbf{m}_1'\boldsymbol{\alpha}-\mathbf{m}_2'\boldsymbol{\alpha}\right) \\
&= \boldsymbol{\alpha}'\underbrace{\left(\mathbf{m}_1-\mathbf{m}_2\right)\left(\mathbf{m}_1-\mathbf{m}_2\right)'}_{\mathbf{M}}\boldsymbol{\alpha} \\
&= \boldsymbol{\alpha}'\mathbf{M}\boldsymbol{\alpha}\,. \qquad (2.24)
\end{aligned}
$$

Similarly, an expression for the denominator in (2.16) can be obtained as

$$
\begin{aligned}
\mathbf{v}'\mathbf{S}_{\mathbf{w}}^{\Phi}\mathbf{v} &= \mathbf{v}'\left[\sum_{j=1}^{2}\sum_{k\in J_j}\left(\Phi(\mathbf{x}_k)-\overline{\mathbf{x}}_j^{\Phi}\right)\left(\Phi(\mathbf{x}_k)-\overline{\mathbf{x}}_j^{\Phi}\right)'\right]\mathbf{v} \\
&= \mathbf{v}'\left[\sum_{j=1}^{2}\sum_{k\in J_j}\Phi(\mathbf{x}_k)\Phi(\mathbf{x}_k)' - \sum_{j=1}^{2} n_j\left(\overline{\mathbf{x}}_j^{\Phi}\right)\left(\overline{\mathbf{x}}_j^{\Phi}\right)'\right]\mathbf{v} \\
&= \mathbf{v}'\left[\sum_{j=1}^{2}\sum_{k\in J_j}\Phi(\mathbf{x}_k)\Phi(\mathbf{x}_k)'\right]\mathbf{v} - \mathbf{v}'\left[\sum_{j=1}^{2} n_j\left(\overline{\mathbf{x}}_j^{\Phi}\right)\left(\overline{\mathbf{x}}_j^{\Phi}\right)'\right]\mathbf{v} \\
&= \sum_{j=1}^{2}\sum_{k\in J_j}\langle\mathbf{v},\Phi(\mathbf{x}_k)\rangle\langle\Phi(\mathbf{x}_k),\mathbf{v}\rangle - \sum_{j=1}^{2} n_j\left\langle\mathbf{v},\overline{\mathbf{x}}_j^{\Phi}\right\rangle\left\langle\overline{\mathbf{x}}_j^{\Phi},\mathbf{v}\right\rangle \\
&= \sum_{j=1}^{2}\sum_{k\in J_j}\left\langle\sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_k)\right\rangle\left\langle\Phi(\mathbf{x}_k),\sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i)\right\rangle - \sum_{j=1}^{2} n_j\left\langle\boldsymbol{\alpha},\mathbf{m}_j\right\rangle\left\langle\mathbf{m}_j,\boldsymbol{\alpha}\right\rangle \\
&= \sum_{i=1}^{n}\alpha_i\alpha_i\sum_{j=1}^{2}\sum_{k\in J_j}\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_k)\rangle\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_i)\rangle - \sum_{j=1}^{2} n_j\left\langle\boldsymbol{\alpha},\mathbf{m}_j\right\rangle\left\langle\mathbf{m}_j,\boldsymbol{\alpha}\right\rangle \\
&= \sum_{i=1}^{n}\alpha_i\alpha_i\sum_{j=1}^{2}\sum_{k\in J_j}k(\mathbf{x}_i,\mathbf{x}_k)k(\mathbf{x}_k,\mathbf{x}_i) - \sum_{j=1}^{2} n_j\left\langle\boldsymbol{\alpha},\mathbf{m}_j\right\rangle\left\langle\mathbf{m}_j,\boldsymbol{\alpha}\right\rangle \\
&= \boldsymbol{\alpha}'[\mathbf{GG}']\boldsymbol{\alpha} - \boldsymbol{\alpha}'\left[\sum_{j=1}^{2} n_j\mathbf{m}_j\mathbf{m}_j'\right]\boldsymbol{\alpha} \\
&= \boldsymbol{\alpha}'\underbrace{\left[\mathbf{GG}' - \sum_{j=1}^{2} n_j\mathbf{m}_j\mathbf{m}_j'\right]}_{\mathbf{N}}\boldsymbol{\alpha} \\
&= \boldsymbol{\alpha}'\mathbf{N}\boldsymbol{\alpha}, \qquad (2.25)
\end{aligned}
$$

where the elements of the $n\times n$ matrix $\mathbf{G}$, known as the Gram matrix or the kernel matrix, are the inner products (replaced by the kernel function) of the observations in feature space, *viz.*

$$\mathbf{G} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_n) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & \cdots & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}. \tag{2.26}$$

Note that by using the Gram matrix, we can also write $\mathbf{m}_j$, for j = 1, 2 respectively, as

$$\mathbf{m}_1' = \frac{1}{n_1}\left(\underbrace{1,1,\ldots,1}_{n_1},\underbrace{0,0,\ldots,0}_{n_2}\right)\mathbf{G} \text{ and } \mathbf{m}_2' = \frac{1}{n_2}\left(\underbrace{0,0,\ldots,0}_{n_1},\underbrace{1,1,\ldots,1}_{n_2}\right)\mathbf{G}.$$

The matrix $\mathbf{N}$ can also be obtained by making use of the Gram matrix. Let $\mathbf{G}_j$ be an $n \times n_j$ matrix with elements $k(\mathbf{x}_i,\mathbf{x}_k)$, for $i = 1,\ldots,n$ and $k \in J_j$, *viz*.

$$\mathbf{G}_j = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_{n_j}) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & \cdots & \cdots & k(\mathbf{x}_n,\mathbf{x}_{n_j}) \end{bmatrix}.$$

It can be shown that $\mathbf{GG}' = \sum_{j=1}^{2}\mathbf{G}_j\mathbf{I}\mathbf{G}_j'$, where $\mathbf{I}$ is the $n_j \times n_j$ identity matrix. It can also be shown that $\mathbf{m}_j = \frac{1}{n_j}\mathbf{G}_j\mathbf{1}$ which implies that $\mathbf{m}_j\mathbf{m}_j' = \frac{1}{n_j^2}\mathbf{G}_j(\mathbf{11}')\mathbf{G}_j'$, where $\mathbf{1}$ is a $n_j \times 1$ unit vector. From (2.25),

$$\begin{aligned} \mathbf{N} &= \mathbf{GG}' - \sum_{j=1}^{2} n_j\mathbf{m}_j\mathbf{m}_j' \\ &= \sum_{j=1}^{2}\mathbf{G}_j\mathbf{I}\mathbf{G}_j' - \sum_{j=1}^{2} n_j\frac{1}{n_j^2}\mathbf{G}_j(\mathbf{11}')\mathbf{G}_j' \end{aligned}$$

$$= \sum_{j=1}^{2} \mathbf{G}_j \mathbf{I} \mathbf{G}'_j - \frac{1}{n_j} \mathbf{G}_j (\mathbf{1}\mathbf{1}') \mathbf{G}'_j$$

$$= \sum_{j=1}^{2} \mathbf{G}_j \left(\mathbf{I} - \frac{1}{n_j} \mathbf{J} \right) \mathbf{G}'_j \,,$$

where $\mathbf{J}$ is a $n_j \times n_j$ unit matrix.

Using (2.24) and (2.25), we can rewrite (2.16) in terms of kernel representations as

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}'\mathbf{M}\boldsymbol{\alpha}}{\boldsymbol{\alpha}'\mathbf{N}\boldsymbol{\alpha}}. \tag{2.27}$$

Similar to the maximization of (2.6), we use (2.7) to find the optimal elements $(\tilde{\alpha}_1, \tilde{\alpha}_2, ..., \tilde{\alpha}_n)$ of a that maximizes (2.27) as

$$\tilde{\boldsymbol{\alpha}} = c\mathbf{N}^{-1}(\mathbf{m}_1 - \mathbf{m}_2). \tag{2.28}$$

Again we choose $c = 1$. In (2.27), the matrix $\mathbf{N}$ must be of full rank for $\mathbf{N}^{-1}$ to exist. However, $\mathbf{N}$ is not of full rank and can have a rank of at most $n-2$ (*cf*. Herbrich, 2001). To remedy this, we apply regularization to $\mathbf{N}$. Mika (2002, *p*.45) suggests replacing $\mathbf{N}$ by

$$\mathbf{N}_\lambda = \mathbf{N} + \lambda\mathbf{I} \tag{2.29}$$

or

$$\mathbf{N}_\lambda = \mathbf{N} + \lambda\mathbf{G}\,, \tag{2.30}$$

where the regularization parameter $\lambda > 0$ needs to be specified or estimated from the data. Mika (2002, *p*.46) states that in practice it does not seem to make a difference whether (2.29) or (2.30) is used, provided that careful selection of λ is done. Throughout this thesis, (2.29) will be used and estimating the value of λ will be discussed in Chapter 3. The scalar $b \in \Re$ in (2.22) is obtained analogous to (2.12). The midpoint of the projected means in feature space is

$$\begin{aligned}\frac{1}{2}\mathbf{v}'\left(\overline{\mathbf{x}}_1^{\Phi}+\overline{\mathbf{x}}_2^{\Phi}\right) &= \frac{1}{2}\left(\left\langle \mathbf{v},\overline{\mathbf{x}}_1^{\Phi}\right\rangle+\left\langle \overline{\mathbf{x}}_2^{\Phi},\mathbf{v}\right\rangle\right)\\ &= \frac{1}{2}\left(\left\langle \boldsymbol{\alpha},\mathbf{m}_1\right\rangle+\left\langle \mathbf{m}_2,\boldsymbol{\alpha}\right\rangle\right)\\ &= \frac{1}{2}\boldsymbol{\alpha}'\left(\mathbf{m}_1+\mathbf{m}_2\right).\end{aligned}$$

Using (2.28) and (2.29) in the above, we obtain an estimate for b as

$$\tilde{b} = -\frac{1}{2}\left(\mathbf{m}_1-\mathbf{m}_2\right)'\mathbf{N}_\lambda^{-1}\left(\mathbf{m}_1+\mathbf{m}_2\right)-log\left({}^{n_2}\!/_{n_1}\right), \tag{2.31}$$

similar to (2.12). The kernel Fisher discriminant function (2.22) can now be written as

$$f(\mathbf{x}) = \sum_{i=1}^{n}\tilde{\alpha}_i k\left(\mathbf{x}_i,\mathbf{x}\right)+\tilde{b}\ . \tag{2.32}$$

The classifier for KFDA is obtained as

$$\varphi(\mathbf{x}) = sign\left\{\sum_{i=1}^{n}\tilde{\alpha}_i k\left(\mathbf{x}_i,\mathbf{x}\right)+\tilde{b}\right\} \tag{2.33}$$

and the resulting decision boundary is non-linear in X .

In Figures 2.4 and 2.5 the non-linear decision boundary obtained from KFDA and the linear decision boundary from FLDA are compared. In both figures the training data for each class $(n_1 = n_2 = 25)$ are generated from a mixture of normal distributions as follows: First generate 10 means $\boldsymbol{m}_k, k = 1,...,10$, from a bivariate normal distribution with parameters $\mathrm{m}' = [0 \quad 1]$ and $\boldsymbol{\Sigma} = 2\mathbf{I}$, corresponding to the +1 class (the green squares). Similarly, 10 more means are generated from a bivariate normal distribution with $\mathrm{m}' = [1 \quad 0]$ and $\boldsymbol{\Sigma} = 2\mathbf{I}$, corresponding to the -1 class (the red dots). Then for each class in the training data, we generated n_j (j=1, 2) observations from a bivariate normal distribution as follows: for each observation in the class we picked an $\boldsymbol{m}_k$ at random with probability $1/10$, as mean vector together with covariance matrix $\frac{1}{5}\boldsymbol{\Sigma}$.

In Figure 2.4 a Gaussian kernel with parameter $\gamma = 1/p$ was used to perform KFDA on the training data. This figure clearly illustrates that the FLDA decision boundary (the black dotted straight line) does not separate the two classes very well. The decision boundary obtained from KFDA (the blue broken line) is far more effective in separating the two classes. For the second figure, Figure 2.5, an m-th degree polynomial $(m = 2)$ was used to perform KFDA. Again, the non-linear decision boundary from KFDA separates the two classes far better than the linear decision boundary from FLDA. In both these figures the regularization parameter was fixed at $\lambda = 0.1$.

For a comparison of the classification performance of FLDA and KFDA we refer the reader to Louw and Steel (2005). They investigated the two-class case under various configurations in a simulation study. They report that the classification performance of KFDA is similar to FLDA when classes are more or less separable by a linear decision boundary. However, for the configurations where classes are not separable by a linear decision boundary, KFDA performed much better than FLDA.

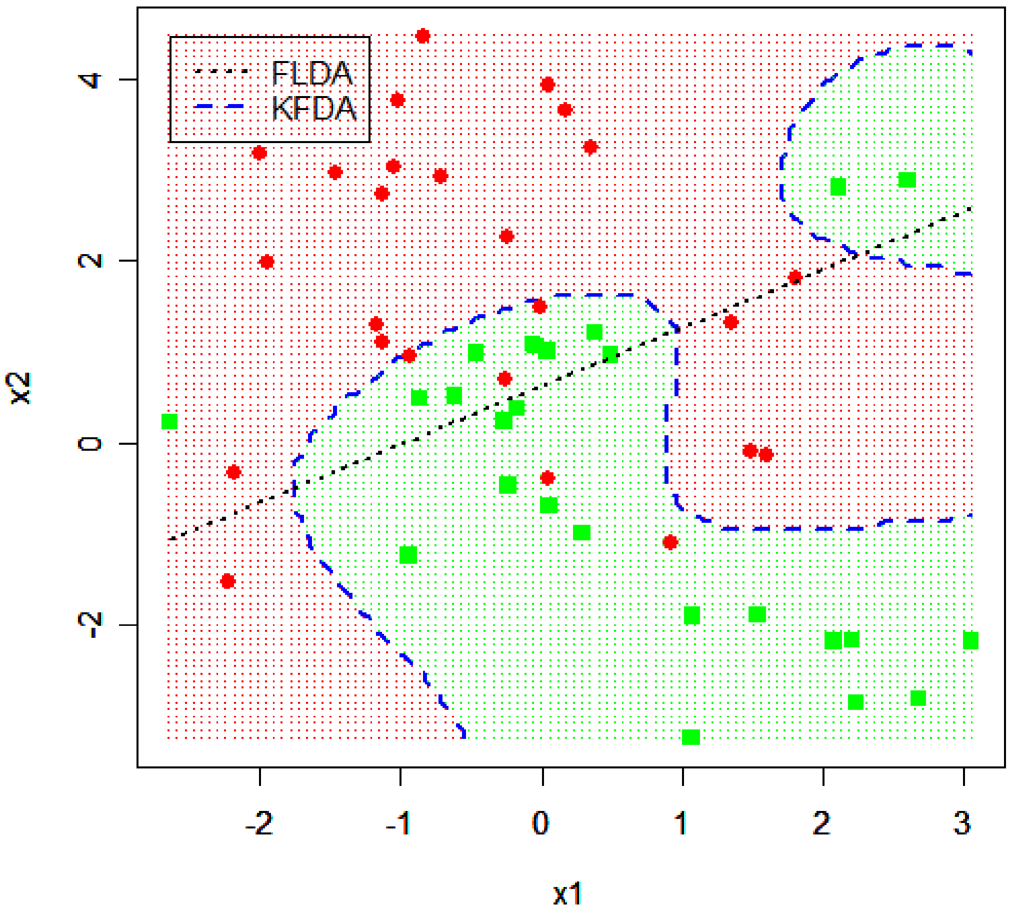


FIGURE 2.4: *Illustration of the decision boundaries of FLDA and KFDA when the Gaussian kernel with* $\gamma = 0.5$ *together with* $\lambda = 0.1$ *is used. The data are simulated from a mixture of normal distributions with the green squares representing the* +1 *class and the red circles the* -1 *class.*

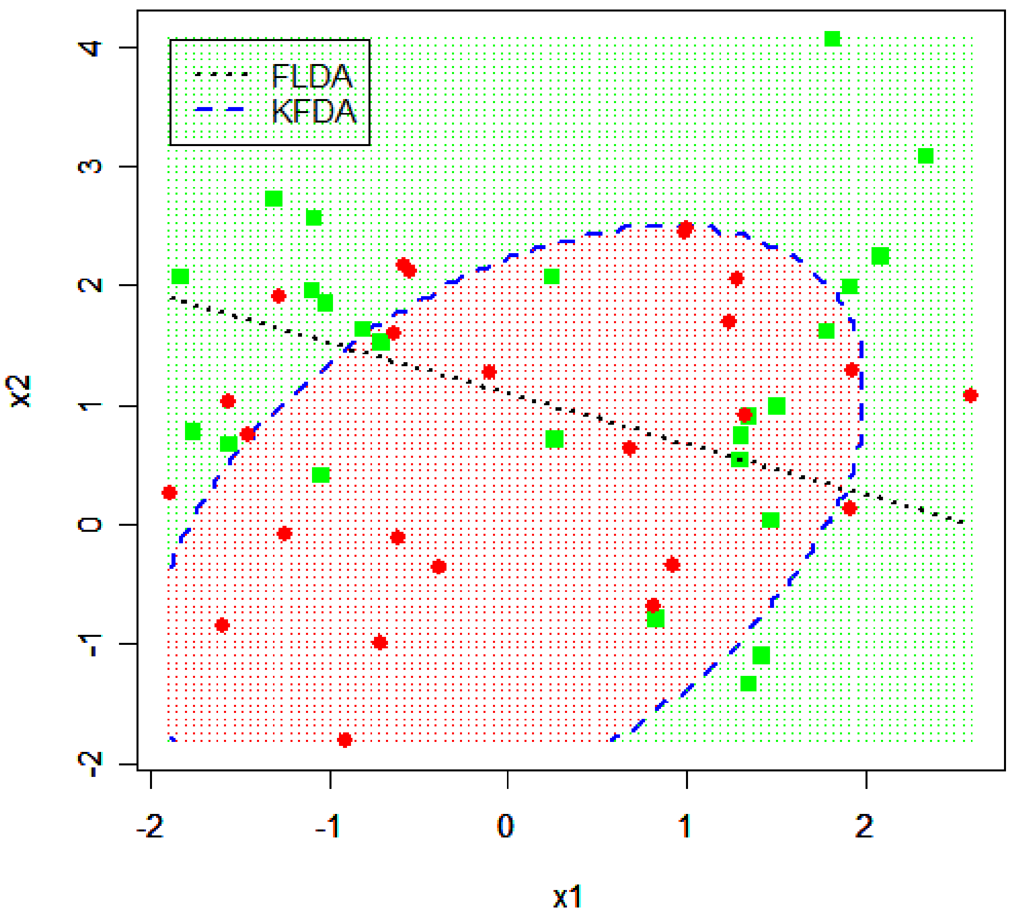


FIGURE 2.5: *Illustration of the decision boundaries of FLDA and KFDA when the 2nd degree polynomial kernel is used together with* $\lambda = 0.1$. *The data are also simulated from a mixture of normal distributions.*

2.5 Quadratic programming formulation of KFDA

In Section 2.2.2 it was shown that FLDA is related to ordinary least squares regression, which can be viewed as solving a quadratic optimization problem. In this section we also cast KFDA into a quadratic optimization framework. The main idea is to illustrate that the optimal values $\tilde{\alpha}_1, \tilde{\alpha}_2, ..., \tilde{\alpha}_n$ and $\tilde{b}$ needed for (2.22) can also be obtained via quadratic programming where we minimize some loss function, $\boldsymbol{\zeta}$, in feature space. Consider the following quadratic optimization problem as stated by Schölkopf and Smola (2002, *p*.459):

$$\min_{\boldsymbol{\alpha}, b, \boldsymbol{\zeta}} \left[\|\boldsymbol{\zeta}\|^2 + C.P(\boldsymbol{\alpha}) \right]$$

$$\textit{subject to}: \; \mathbf{G}\mathrm{a} + \mathbf{1}b = \mathbf{y} + \boldsymbol{\zeta}$$

$$\mathbf{1}'_{+1}\boldsymbol{\zeta} = 0 \text{ and } \mathbf{1}'_{-1}\boldsymbol{\zeta} = 0, \tag{2.34}$$

where $C \in \Re^+$ is a regularization parameter, $P(\mathrm{a})$ is known as a regularization operator with $\mathrm{a} \in \Re^n$, $\mathbf{G}$ is the Gram matrix, $\mathbf{1}$ is a unit vector of length *n*, while

$$\mathbf{1}'_{+1} = \Big(\underbrace{1, 1, ..., 1}_{n_1}, \underbrace{0, 0, ..., 0}_{n_2} \Big) \text{ and } \mathbf{1}'_{-1} = \Big(\underbrace{0, 0, ..., 0}_{n_1}, \underbrace{1, 1, ..., 1}_{n_2} \Big).$$

According to Schölkopf and Smola (2002), it can be shown that the optimization problem (2.34) is equivalent to the optimization of (2.27). The first constraint of (2.34) is the same as

$$\langle \mathbf{v}, \Phi(\mathbf{x}_i) \rangle + b = \mathrm{y}_i + \zeta_i \quad \forall i = 1, ..., n,$$

which, by using (2.21), can be written as $\zeta_i = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b - \mathrm{y}_i$. By minimizing the following function with respect to a and b

$$\sum_{i=1}^{n} \zeta_i^2 = \|\boldsymbol{\zeta}\|^2 = \left\| [\mathbf{G} \quad \mathbf{1}] \begin{bmatrix} \mathrm{a} \\ b \end{bmatrix} - \mathbf{y} \right\|^2 ,$$

the optimal a-vector and the threshold b are obtained. The second constraint of (2.34) ensures that the total loss $\left(\sum_{i=1}^{n} \zeta_i \right)$ in each class is zero. Similar to (2.29), some form of regularization is necessary to perform calculations and therefore we add the term $C.P(\boldsymbol{\alpha})$. The literature provides different functions for $P(\mathrm{a})$, of which $\|\mathrm{a}\|$, $\|\mathrm{a}\|^2$ and $\mathrm{a}'\mathbf{G}\mathrm{a}$ are some examples. The unknown parameter C is generally estimated by means of cross-validation from the training data and plays a similar role as λ in (2.29).

Schölkopf and Smola (2002) state that solving (2.34) for large data sets is impractical. They argue that, for $P(\mathrm{a}) = \|\mathrm{a}\|^2$, (2.34) can be rewritten using the following, more efficient, setup. First, they define

$$\mathbf{a} = \begin{bmatrix} b \\ \mathrm{a} \end{bmatrix} \; \mathbf{c} = \begin{bmatrix} n_1 - n_2 \\ \mathbf{G}'\mathbf{y} \end{bmatrix} \; \mathbf{A}_{+1} = \begin{bmatrix} n_1 \\ \mathbf{G}'\mathbf{1}_{+1} \end{bmatrix} \; \mathbf{A}_{-1} = \begin{bmatrix} n_2 \\ \mathbf{G}'\mathbf{1}_{-1} \end{bmatrix} \; \mathbf{H} = \begin{bmatrix} n & \mathbf{1}'\mathbf{G} \\ \mathbf{G}'\mathbf{1} & \mathbf{G}'\mathbf{G} + C\mathbf{I} \end{bmatrix}, \tag{2.35}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. Secondly, they obtain the formulation

$$\min_{\mathbf{a}} \left[\frac{1}{2} \mathbf{a}'\mathbf{H}\mathbf{a} - \mathbf{c}'\mathbf{a} + \frac{n}{2} \right]$$

$$\textit{subject to}: \; \mathbf{A}'_{+1}\mathbf{a} - n_1 = 0, \; \mathbf{A}'_{-1}\mathbf{a} + n_2 = 0, \tag{2.36}$$

which is equivalent to (2.34). Let λ_{+1} and λ_{-1} denote two positive Lagrange multipliers, then a Lagrangian[3] for (2.36) can be obtained as

$$L_P(\mathbf{a},\lambda_{+1},\lambda_{-1}) = \frac{1}{2}\mathbf{a}'\mathbf{H}\mathbf{a} - \mathbf{c}'\mathbf{a} + \lambda_{+1}(\mathbf{A}'_{+1}\mathbf{a} - n_1) + \lambda_{-1}(\mathbf{A}'_{-1}\mathbf{a} + n_2) + \frac{n}{2}. \tag{2.37}$$

Taking partial derivatives and minimizing (2.37) with respect to $\mathbf{a}$ yields

$$\frac{\partial L_P}{\partial \mathbf{a}} = \mathbf{H}\mathbf{a} - \mathbf{c} + \lambda_{+1}\mathbf{A}_{+1} + \lambda_{-1}\mathbf{A}_{-1} = \mathbf{0},$$

implying that $\mathbf{c} = \mathbf{H}\mathbf{a} + \lambda_{+1}\mathbf{A}_{+1} + \lambda_{-1}\mathbf{A}_{-1}$. By substituting $\mathbf{c}$ into (2.37), the following dual formulation of (2.36) is obtained:

$$\max_{\mathbf{a},\lambda_{+1},\lambda_{-1}} \left[-\frac{1}{2}\mathbf{a}'\mathbf{H}\mathbf{a} - \lambda_{+1}n_1 + \lambda_{-1}n_2 + \frac{n}{2} \right]$$

$$\textit{subject to}\text{: } \mathbf{H}\mathbf{a} - \mathbf{c} + \lambda_{+1}\mathbf{A}_{+1} + \lambda_{-1}\mathbf{A}_{-1} = \mathbf{0}. \tag{2.38}$$

Using the constraint of (2.38), $\mathbf{a}$ is obtained as

$$\mathbf{a} = \mathbf{H}^{-1}(\mathbf{c} - \lambda_{+1}\mathbf{A}_{+1} - \lambda_{-1}\mathbf{A}_{-1}). \tag{2.39}$$

Substituting (2.39) into (2.38) gives the maximization problem (with no constraints)

$$\max_{\lambda_{+1},\lambda_{-1}} \left\{ -\frac{1}{2} \begin{bmatrix} \lambda_{+1} \\ \lambda_{-1} \end{bmatrix}' \begin{bmatrix} \mathbf{A}'_{+1}\mathbf{H}^{-1}\mathbf{A}_{+1} & \mathbf{A}'_{+1}\mathbf{H}^{-1}\mathbf{A}_{-1} \\ \mathbf{A}'_{-1}\mathbf{H}^{-1}\mathbf{A}_{+1} & \mathbf{A}'_{-1}\mathbf{H}^{-1}\mathbf{A}_{-1} \end{bmatrix} \begin{bmatrix} \lambda_{+1} \\ \lambda_{-1} \end{bmatrix} + \begin{bmatrix} \lambda_{+1} \\ \lambda_{-1} \end{bmatrix}' \begin{bmatrix} -n_1 + \mathbf{c}'\mathbf{H}^{-1}\mathbf{A}_{+1} \\ n_2 + \mathbf{c}'\mathbf{H}^{-1}\mathbf{A}_{-1} \end{bmatrix} - \frac{1}{2}\mathbf{c}'\mathbf{H}\mathbf{c} + \frac{n}{2} \right\}$$

[3] In this thesis all the quadratic optimization problems are solved using a Lagrangian formulation. For more detail about how the Lagrangian formulation works, refer to Bertsekas (1995) or Mangasarian (1997).

which can be solved analytically to obtain the values λ_{+1} and λ_{-1}. The optimal vector $\mathbf{a}' = \begin{pmatrix} \tilde{b} & \tilde{\mathrm{a}} \end{pmatrix}$ can be obtained by substituting the solutions of λ_{+1} and λ_{-1} into (2.39). Once we have the vector **a**, we can construct the classifier similar to (2.33).

Note that solving the quadratic optimization problem (2.38) does not necessarily make computations easier than solving the optimization problem defined in (2.27). However, an advantage of using (2.38) is that the matrix $\mathbf{H}^{-1}$ can be approximated in a sparse greedy way which can make computations in (2.38) more feasible than in (2.27), especially when dealing with very large data sets. This sparse greedy approximation is one way in which KFDA can be modified to enhance computations, but since it falls outside the scope of this thesis, we refer the reader to Schölkopf and Smola (2002, *p*.461) and Mika (2002, *p*.53) for a more detailed explanation of this process.

It is clear from the explanations above that KFDA can be formulated as a quadratic optimization problem. An advantage of performing KFDA within this framework is that a sparse greedy approximation to the α_i's can be obtained which can make computations more feasible when the data sets are large. Another advantage of using the quadratic programming formulation is that when the different functions $\|\mathrm{a}\|$ or $\mathrm{a}'\mathbf{G}\mathrm{a}$ are used for $P(\mathrm{a})$, more variants of KFDA can be derived. For examples of such variants the reader is referred to Mika (2002, *pp*.46-53). Note that in this thesis we restrict our focus to KFDA as described in Section 2.4.

2.6 Relationship between KFDA and other kernel based methods

2.6.1 A short overview

Similar to what we have seen for FLDA, many connections exist between KFDA and other classification methods. Shashua (1999) illustrates the relationship between the SVM and KFDA. Shawe-Taylor and Cristianini (2004) show the similarities between regularized kernel Fisher discriminant analysis (regKFDA) and KFDA. Suykens and

Vanderwalle (1999) proposed least squares SVMs, and derived a classifier which is equivalent to the KFD classifier. The relevance vector machine (Tipping, 2000), a Bayesian technique for regression or classification, shows a strong connection to KFDA, as noted in Mika (2002, *p*.62). The next two subsections contain an explanation of the SVM and regKFDA as examples of methods related to KFDA.

2.6.2 Relation to the support vector machine

The SVM was first introduced by Boser *et al.* (1992) and has since become a popular tool for solving statistical classification problems. Similar to KFDA, the SVM finds a linear discriminant function,

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c\,, \tag{2.40}$$

in F where the parameter $\mathbf{w} \in F$ is obtained by maximizing what is known as the margin. A geometrical illustration of the margin is given in Figure 2.6 for the case where the classes are separable. The margin can be obtained as follows: In F, and for a given $\mathbf{w}$ and c, let a $+1$ hyperplane be defined as $\{\mathbf{x} : \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c = +1\}$ and a -1 hyperplane defined as $\{\mathbf{x} : \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c = -1\}$. Then the distance from any point on the $+1$ hyperplane to the decision boundary $\{\mathbf{x} : \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c = 0\}$ is

$$\left| \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c \right| / \|\mathbf{w}\| = 1/\|\mathbf{w}\|\,. \tag{2.41}$$

Similarly, the distance from a point on the -1 hyperplane to the decision boundary is also $1/\|\mathbf{w}\|$. The margin is defined as $2/\|\mathbf{w}\|$, the sum of these two distances. Maximizing the margin, for the case where classes are separable, is equivalent to solving the following quadratic optimization problem,

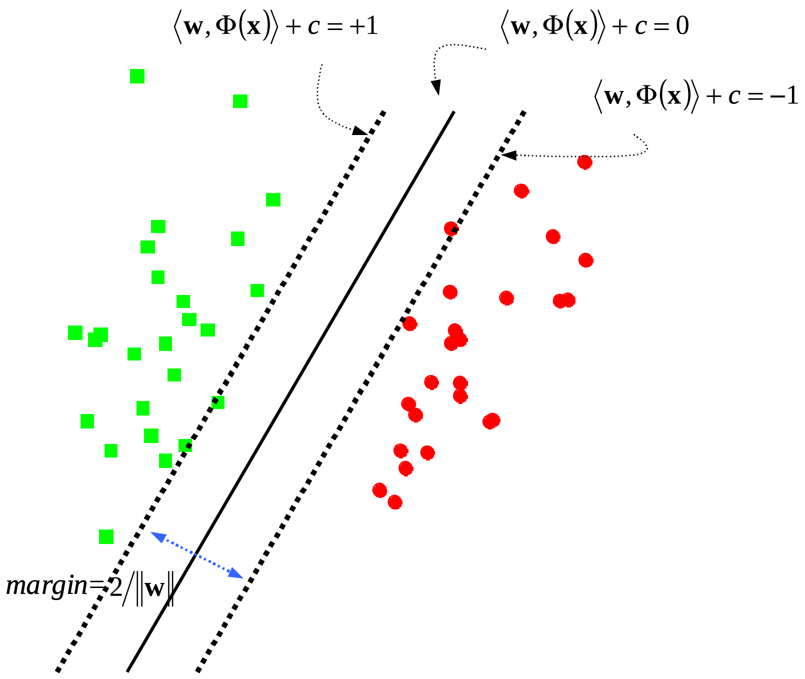

$\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + c = +1$
$\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + c = 0$
$\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + c = -1$
$margin = 2/\|\mathbf{w}\|$

FIGURE 2.6: *Schematic representation of the margin, the hyperplanes and the SVM decision boundary for separable classes in feature space. The solid black line represents the decision boundary obtained by maximizing the margin.*

$$\min_{\mathbf{w},c}\left[\frac{1}{2}\|\mathbf{w}\|^2\right]$$

$$\textit{subject to: } y_i\left(\langle \mathbf{w},\Phi(\mathbf{x}_i)\rangle + c\right) \geq 1, \quad i=1,\ldots,n. \tag{2.42}$$

Introducing Lagrange multipliers $(\alpha_i \geq 0)$, the following primal Lagrangian of (2.42) is obtained:

$$\begin{aligned} L_P(\mathbf{w},c,\mathrm{a}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i\left(\langle \mathbf{w},\Phi(\mathbf{x}_i)\rangle + c\right) - 1\right) \\ &= \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle - \sum_{i=1}^{n}\alpha_i y_i\langle \mathbf{w},\Phi(\mathbf{x}_i)\rangle - \sum_{i=1}^{n}\alpha_i y_i c + \sum_{i=1}^{n}\alpha_i \end{aligned} \tag{2.43}$$

To solve for the unknowns in (2.43) we minimize this function with respect to $\mathbf{w}$ and c, while we maximize it with respect to the α_i's. Taking partial derivatives with respect to $\mathbf{w}$ and c, and setting them equal to zero yields,

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i) = \mathbf{0},$$
$$\frac{\partial L_P}{\partial c} = \sum_{i=1}^{n} \alpha_i y_i = 0. \qquad (2.44)$$

The first equation implies that $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i)$. By substituting the expressions (2.44) into (2.43) the following dual Lagrangian is obtained:

$$L_D(\mathrm{a}) = \frac{1}{2}\left\langle \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i), \sum_{k=1}^{n} \alpha_k y_k \Phi(\mathbf{x}_k) \right\rangle - \sum_{i=1}^{n} \alpha_i y_i \left\langle \sum_{k=1}^{n} \alpha_k y_k \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_i) \right\rangle + \sum_{i=1}^{n} \alpha_i$$
$$= \frac{1}{2}\sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k)\rangle - \sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k)\rangle + \sum_{i=1}^{n} \alpha_i$$
$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k)\rangle. \qquad (2.45)$$

If we replace the inner product in (2.45) with a kernel function, the optimization problem reduces to obtaining the α's, which we get by solving the maximization problem

$$\max_{\mathrm{a}} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k k(\mathbf{x}_i, \mathbf{x}_k) \right]$$

$$\textit{subject to}: \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, \quad i = 1,\dots,n. \qquad (2.46)$$

We will denote the solution to the α's by $\tilde{\alpha}_1, \tilde{\alpha}_2, ..., \tilde{\alpha}_n$. It should be noted that some of the α's in (2.46) are zero. The non-zero α's are associated with the so-called support vectors, *i.e.* any vector $\mathbf{x}_i$ with $i \in M = \{i : \alpha_i > 0\}$. Only these support vectors play a role in obtaining (2.40), which can now be written as

$$\begin{aligned} f(\mathbf{x}) &= \left\langle \sum_{i=1}^{n} \alpha_i \mathrm{y}_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \right\rangle + c \\ &= \sum_{i=1}^{n} \alpha_i \mathrm{y}_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + c, \end{aligned}$$

which becomes

$$f(\mathbf{x}) = \sum_{i \in M} \tilde{\alpha}_i \mathrm{y}_i k(\mathbf{x}_i, \mathbf{x}) + c. \tag{2.47}$$

The support vectors are also used to estimate the scalar $c \in \Re$ in (2.47). This is done by using the Karush-Kuhn-Tucker conditions (*cf.* Karush, 1939; Kuhn and Tucker, 1951) which states that for any support vector $(\mathbf{x}_i)$, the constraint in (2.42) is exactly satisfied, *i.e.*

$$\begin{aligned} \mathrm{y}_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + c) &= \mathrm{y}_i \left(\sum_{k=1}^{n} \tilde{\alpha}_k \mathrm{y}_k k(\mathbf{x}_i, \mathbf{x}_k) + c \right) \\ &= 1 \end{aligned}$$

so that

$$c = \mathrm{y}_i - \sum_{k=1}^{n} \tilde{\alpha}_k \mathrm{y}_k k(\mathbf{x}_i, \mathbf{x}_k).$$

We obtain $\tilde{c}$ by averaging over all the support vectors, *viz.*

$$\tilde{c} = \frac{1}{|M|}\sum_{i\in M}\left(\mathrm{y}_i - \sum_{k=1}^{n}\tilde{\alpha}_k \mathrm{y}_k k(\mathbf{x}_i, \mathbf{x}_k)\right), \tag{2.48}$$

since it yields a numerically stable result (*cf.* Mika, 2002, *p.*20). The SVM classifier is therefore

$$\varphi(\mathbf{x}) = sign\left\{\sum_{i\in M}\tilde{\alpha}_i \mathrm{y}_i k(\mathbf{x}_i, \mathbf{x}) + \tilde{c}\right\}. \tag{2.49}$$

Consider now the case where the classes are overlapping, *i.e.* the non-separable case (*cf.* Figure 2.7). For this case observations lying on the wrong side of the margin are penalized by using so-called slack variables, denoted by ξ. The formulation of (2.42) in the non-separable case becomes

$$\min_{\mathbf{w},c,\xi}\left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right]$$

$$\textit{subject to}\text{: } \mathrm{y}_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + c\right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0,\ i=1,\ldots, n, \tag{2.50}$$

where $C > 0$ is a regularization parameter controlling the trade-off between maximizing the margin and minimizing the training error. Again we solve (2.50) by formulating a Lagrangian. Using Lagrange multipliers $(\alpha_i, \beta_i \geq 0)$ we obtain the following primal Lagrangian of (2.50):

$$\begin{aligned} L_P(\mathbf{w}, c, \xi, \mathrm{a}) &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(\mathrm{y}_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + c\right) - 1 + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i \\ &= \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i \mathrm{y}_i\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle - \sum_{i=1}^{n}\alpha_i \mathrm{y}_i c \\ &\quad + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i, \end{aligned} \tag{2.51}$$

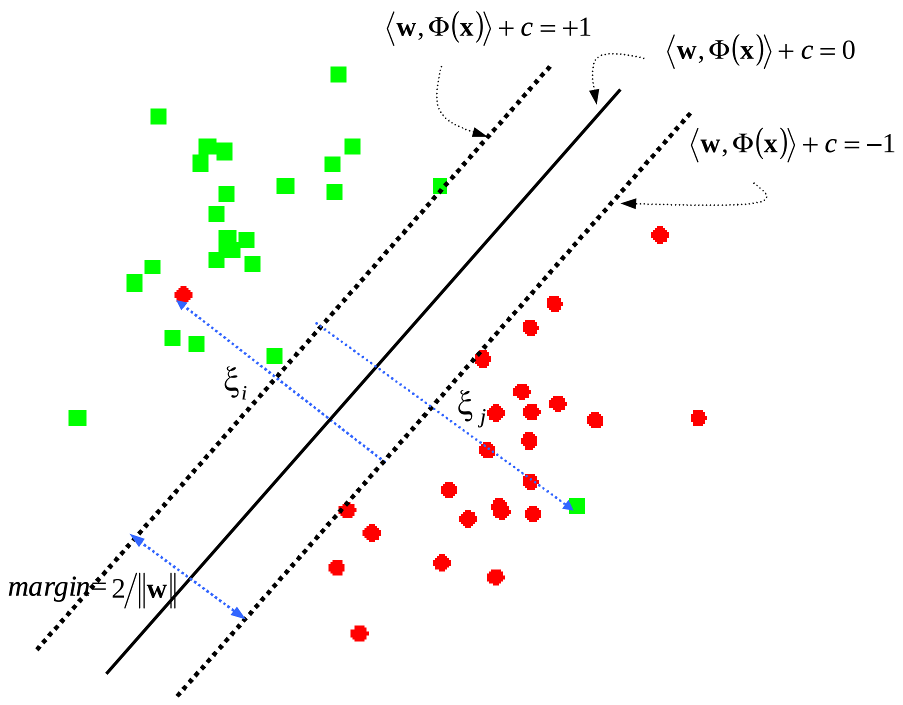

$\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c = +1$
$\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c = 0$
$\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + c = -1$
ξ_i
ξ_j
$margin = 2/\|\mathbf{w}\|$

FIGURE 2.7: *Schematic representation of the SVM classifier for overlapping classes in feature space. The solid black line is the decision boundary obtained by maximizing the margin. The slack variables* $(\xi_i, \forall i = 1,\ldots,n)$ *for observations lying on the wrong side of the margin are positive, otherwise they are zero. In the picture there are two points, marked with arrows that will receive positive slack variables. The rest of the points will receive a zero slack variable.*

which we minimize with respect to **w**, *c* and ξ_i. Taking partial derivatives of (2.51) and setting them to zero, gives

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i) = \mathbf{0},$$

$$\frac{\partial L_P}{\partial c} = \sum_{i=1}^{n} \alpha_i y_i = 0,$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \beta_i = 0. \qquad (2.52)$$

From the above we obtain the equations: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i)$, $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $C = \alpha_i + \beta_i$.

By substituting these equations into (2.51) the following dual Lagrangian is obtained:

$$\begin{aligned} L_D(\mathrm{a}) &= \frac{1}{2}\left\langle \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i), \sum_{k=1}^{n} \alpha_k y_k \Phi(\mathbf{x}_k) \right\rangle - \sum_{i=1}^{n} \alpha_i y_i \left\langle \sum_{k=1}^{n} \alpha_k y_k \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_i) \right\rangle + \sum_{i=1}^{n} \alpha_i \\ &= \frac{1}{2}\sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k) \rangle - \sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_i) \rangle + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k) \rangle . \end{aligned} \qquad (2.53)$$

If we replace the inner product in (2.53) with a kernel function, the problem reduces to obtaining $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_n$, which is found by solving

$$\max_{\mathrm{a}} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,k=1}^{n} \alpha_i \alpha_k y_i y_k k(\mathbf{x}_i, \mathbf{x}_k) \right]$$

$$\textit{subject to}: \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \le \alpha_i \le C,\ i = 1,\ldots,n. \qquad (2.54)$$

The support vectors are now any vector $\mathbf{x}_i$ for which $i \in M = \{i : 0 < \alpha_i \leq C\}$. Similar to the separable case, only these support vectors are used to obtain the threshold (2.48) and the final SVM classifier (2.49).

From the descriptions above the similarity between the KFD classifier (2.33) and the SVM classifier (2.49) is evident. Both KFDA and the SVM yield linear discriminant functions which are derived in feature space by making use of the kernel trick. The main difference between the two techniques is the way in which the parameters for their respective discriminant functions are estimated. For KFDA the optimal α's was obtained by maximizing the Rayleigh coefficient and the threshold *b* was calculated by using all these α's. In the case of the SVM, the optimal α's was obtained by maximizing the margin and the threshold *c* was obtained by using only the $\alpha > 0$ values. The regularization parameter C for the SVM plays a similar regulatory role as the parameter λ in KFDA and both are usually estimated from the data. Similar to KFDA, the decision boundary produced by the SVM classifier is non-linear in input space. Figure 2.8 contains an illustration of the SVM decision boundary in comparison to the KFDA decision boundary.

2.6.3 Relation to regularized kernel Fisher discriminant analysis

A variant of KFDA is given by Shawe-Taylor and Cristianini (2004). The formulation that they present is also a kernel based technique which is similar to KFDA. The following brief explanation is taken from their book. For more detail see Shawe-Taylor and Cristianini (2004, *pp*.133 and 176). Again we want to obtain a linear discriminant function

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + d, \tag{2.55}$$

in F, where the parameter vector $\mathbf{w} \in F$ and the scalar $d \in \Re$ is obtained by maximizing a ratio similar to (2.16). The vector $\mathbf{w}$, which is also a direction of discrimination (*cf.* Shawe-Taylor and Cristianini, 2004, *pp*.133-136), is obtained by maximizing

$$J(\mathbf{w}) = \frac{\mathbf{w}'\mathbf{E}\mathbf{w}}{\mathbf{w}'\mathbf{F}\mathbf{w}}, \tag{2.56}$$

where matrices **E** and **F** are defined as

$$\mathbf{E} = (\mathbf{X}'\mathbf{y})(\mathbf{y}'\mathbf{X}) \text{ and } \mathbf{F} = \left[\frac{n}{2n_1n_2}\mathbf{X}'\mathbf{B}\mathbf{X} + \lambda\mathbf{I}\right] \tag{2.57}$$

respectively. In (2.57), $\mathbf{X}' = [\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2), \mathbf{K}, \Phi(\mathbf{x}_n)]$ represents the mapped data so that $\mathbf{X}\mathbf{X}' = \mathbf{G}$ is the Gram matrix as defined by (2.26). The matrix **E** represents the between-class scatter matrix while $n/2n_1n_2\,\mathbf{X}'\mathbf{B}\mathbf{X}$ represents the within-class scatter matrix, similar to the matrices defined in (2.17). The matrix $\mathbf{F}$ is the regularized version of the within-class scatter matrix making it possible to obtain $\mathbf{F}^{-1}$. The matrix **B**, is defined as $\mathbf{B} = \mathbf{D} - \mathbf{C}^{+} - \mathbf{C}^{-}$, where **D** is a diagonal matrix with diagonal elements

$$\mathbf{D}_{ii} = \begin{cases} 2n_2/n & \text{if } \mathrm{y}_i = +1 \\ 2n_1/n & \text{if } \mathrm{y}_i = -1. \end{cases}$$

The matrices $\mathbf{C}^{+}$ and $\mathbf{C}^{-}$ have the following elements,

$$\mathbf{C}^{+}_{ij} = \begin{cases} 2n_2/(nn_1) & \text{if } \mathrm{y}_i = +1 = \mathrm{y}_j \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i, j = 1, \ldots, n$$

$$\mathbf{C}^{-}_{ij} = \begin{cases} 2n_1/(nn_2) & \text{if } \mathrm{y}_i = -1 = \mathrm{y}_j \\ 0 & \text{otherwise.} \end{cases}$$

The regularization parameter $\lambda > 0$ is either specified by the user or estimated from the training data. It plays a similar role as the regularization parameter in KFDA. Using (2.56) and (2.57), Shawe-Taylor and Cristianini (2004) show that **w** can be expressed as the following linear combination[4] of the training data:

$$\mathbf{w} = \mathbf{X}'\mathrm{b} = \sum_{i=1}^{n} \beta_i \Phi(\mathbf{x}_i). \tag{2.58}$$

Since the mapping Φ is unknown, maximizing (2.56) is impossible, but by using (2.58) a kernelized version of (2.56) can be formulated as follows. The numerator of (2.56) can be written as

$$\begin{aligned} \mathbf{w}'\mathbf{E}\mathbf{w} &= \mathrm{b}'\mathbf{X}[\mathbf{X}'\mathbf{y}\mathbf{y}'\mathbf{X}]\mathbf{X}'\mathrm{b} \\ &= \mathrm{b}'(\mathbf{X}\mathbf{X}')\mathbf{y}\mathbf{y}'(\mathbf{X}\mathbf{X}')\mathrm{b} \\ &= \mathrm{b}'\underbrace{\mathbf{G}\mathbf{y}\mathbf{y}'\mathbf{G}}_{\mathbf{Q}}\mathrm{b} \\ &= \mathrm{b}'\mathbf{Q}\mathrm{b}, \end{aligned}$$

while the denominator becomes

$$\begin{aligned} \mathbf{w}'\mathbf{F}\mathbf{w} &= \mathrm{b}'\mathbf{X}\left[\frac{n}{2n_1n_2}\mathbf{X}'\mathbf{B}\mathbf{X} + \lambda\mathbf{I}\right]\mathbf{X}'\mathrm{b} \\ &= \frac{n}{2n_1n_2}\mathrm{b}'\mathbf{X}[\mathbf{X}'\mathbf{B}\mathbf{X}]\mathbf{X}'\mathrm{b} + \lambda\mathrm{b}'\mathbf{X}\mathbf{X}'\mathrm{b} \\ &= \frac{n}{2n_1n_2}\mathrm{b}'(\mathbf{X}\mathbf{X}')\mathbf{B}(\mathbf{X}\mathbf{X}')\mathrm{b} + \lambda\mathrm{b}'(\mathbf{X}\mathbf{X}')\mathrm{b} \\ &= \frac{n}{2n_1n_2}\mathrm{b}'\mathbf{G}\mathbf{B}\mathbf{G}\mathrm{b} + \lambda\mathrm{b}'\mathbf{G}\mathrm{b} \\ &= \mathrm{b}'\underbrace{\left[\frac{n}{2n_1n_2}\mathbf{G}\mathbf{B}\mathbf{G} + \lambda\mathbf{G}\right]}_{\mathbf{S}}\mathrm{b} \\ &= \mathrm{b}'\mathbf{S}\mathrm{b}. \end{aligned}$$

[4] Note that (2.58) and (2.21) are similar expansions.

Thus, the ratio (2.56) becomes

$$J(\mathrm{b}) = \frac{\mathrm{b}'\mathbf{Q}\mathrm{b}}{\mathrm{b}'\mathbf{S}\mathrm{b}}, \tag{2.59}$$

where

$$\mathbf{Q} = \mathbf{Gyy}'\mathbf{G} \text{ and } \mathbf{S} = \left[\frac{n}{2n_1n_2}\mathbf{GBG} + \lambda\mathbf{G}\right]. \tag{2.60}$$

Calculating (2.60) is easy since we use the kernel function to obtain the Gram matrix. Thus, the problem is reduced to finding the vector b. Using the maximization lemma (2.7) we obtain the optimal vector $\tilde{\boldsymbol{\beta}}$ as

$$\begin{aligned}\tilde{\mathrm{b}} &= \left[\frac{n}{2n_1n_2}\mathbf{GBG} + \lambda\mathbf{G}\right]^{-1}\mathbf{Gy} \\ &= \left[\frac{n}{2n_1n_2}\mathbf{BG} + \lambda\mathbf{I}\right]^{-1}\mathbf{y} \qquad \text{(because } \mathbf{G} \text{ is non-singular)}\end{aligned}$$

and since the constant $n/2n_1n_2$ only influences the length of the vector and not the direction, we will use

$$\tilde{\mathrm{b}} = (\mathbf{BG} + \lambda\mathbf{I})^{-1}\mathbf{y} \tag{2.61}$$

to obtain the discriminant function. The elements of $\tilde{\mathrm{b}}$ will be denoted by $\tilde{\beta}_1, \tilde{\beta}_2, ..., \tilde{\beta}_n$. Using these optimal values and expansion (2.58), the discriminant function (2.55) can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{n} \tilde{\beta}_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + d$$
$$= \sum_{i=1}^{n} \tilde{\beta}_i k(\mathbf{x}_i, \mathbf{x}) + d\,. \qquad (2.62)$$

To obtain an estimate of the scalar d, we need to define the following vectors. Let the mean vector for the $+1$ class be denoted by $\mathrm{m}^+ = \frac{1}{n_1}\mathbf{X}'\mathbf{j}_{+1}$, where $\mathbf{j}_{+1}$ is an $n \times 1$ vector with 1's in the entries corresponding to the $+1$ class and 0's otherwise. Similarly, let $\mathrm{m}^- = \frac{1}{n_2}\mathbf{X}'\mathbf{j}_{-1}$ denote the mean vector of the -1 class with $\mathbf{j}_{-1}$ being a vector with 1's in the entries corresponding to the -1 class and 0's otherwise. Then d is estimated such that $\mathbf{w}'\mathrm{m}^+ - d = d - \mathbf{w}'\mathrm{m}^-$. Using this and (2.58) we can obtain d in (2.62) as

$$d = \frac{1}{2}\mathbf{w}'\left(\mathrm{m}^+ + \mathrm{m}^-\right)$$
$$= \frac{1}{2}\mathrm{b}'\mathbf{X}\left(\frac{1}{n_1}\mathbf{X}'\mathbf{j}_{+1} + \frac{1}{n_2}\mathbf{X}'\mathbf{j}_{-1}\right)$$
$$= \frac{1}{2}\mathrm{b}'\left[\frac{1}{n_1}(\mathbf{X}\mathbf{X}')\mathbf{j}_{+1} + \frac{1}{n_2}(\mathbf{X}\mathbf{X}')\mathbf{j}_{-1}\right]$$
$$= \frac{1}{2}\mathrm{b}'\left[\frac{1}{n_1}\mathbf{G}\mathbf{j}_{+1} + \frac{1}{n_2}\mathbf{G}\mathbf{j}_{-1}\right]$$
$$= \frac{1}{2}\mathrm{b}'\mathbf{G}\left[\frac{1}{n_1}\mathbf{j}_{+1} + \frac{1}{n_2}\mathbf{j}_{-1}\right]. \qquad (2.63)$$

The corresponding classification rule for regKFDA is: classify $\mathbf{x}$ into the $+1$ class if $\mathbf{w}'\Phi(\mathbf{x}) - d > 0$, else into the -1 class. The classifier can therefore be written as

$$\varphi(\mathbf{x}) = sign\left(\sum_{i=1}^{n} \tilde{\beta}_i k(\mathbf{x}_i, \mathbf{x}) + \tilde{d}\right), \qquad (2.64)$$

with

$$\tilde{d} = -\frac{1}{2}\tilde{\mathrm{b}}'\mathbf{G}\left[\frac{1}{n_1}\mathbf{j}_{+1} + \frac{1}{n_2}\mathbf{j}_{-1}\right]. \tag{2.65}$$

It is clear from the description above that KFDA and regKFDA are quite similar. The big difference between the techniques is the following: In the case of KFDA, the regularization is added after the kernelization of (2.16). For regKFDA the regularization is added before the kernelization of (2.56). The classifiers (2.33) and (2.60) have the same form, and the corresponding parameters are obtained by maximizing similar ratios. An example of the non-linear decision boundary produced by regKFDA is shown in Figure 2.8 together with the KFDA and the SVM decision boundaries.

The aim of Figure 2.8 is to illustrate the non-linearity and similarity of the three decision boundaries and not to compare the results of the classifiers. Comparing the three methods is made difficult because of the dependence of each method on their corresponding hyperparameters which we have to optimize. A comparison of the classification performance of these methods will not be done here, since it falls outside the scope of this thesis. For a comparison of the SVM and KFDA classification results, see Mika (2002, *p*.105). In his study he applied these techniques to ten real-world data sets. They report that the SVM and KFDA have a similar classification performance for the different data sets. To our knowledge, no studies comparing the classification performance between KFDA and regKFDA exist in the literature. Since these techniques are very similar, one would expect that they will produce similar results as well.

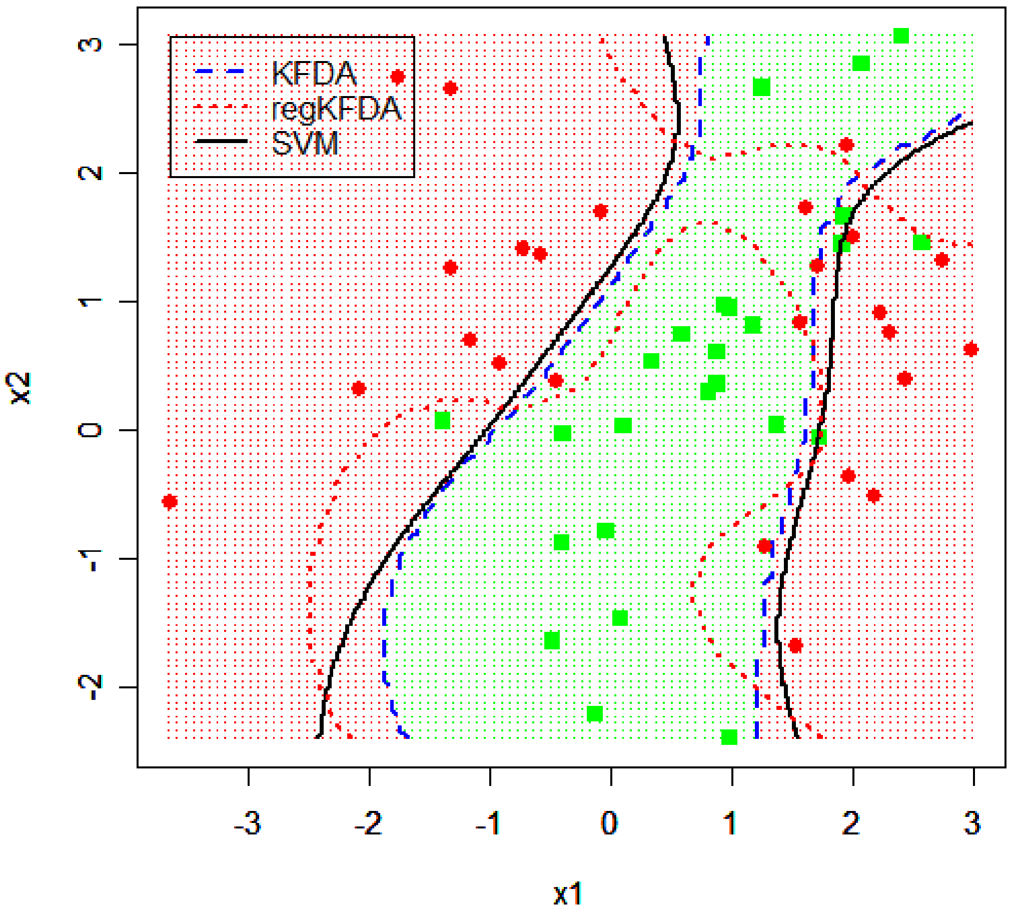


FIGURE 2.8: *Illustrative examples of the KFDA $(\lambda = 0.1)$, SVM $(C = 0.5)$ and regKFDA $(\lambda = 0.1)$ decision boundaries. In all cases the Gaussian kernel was used with $\gamma = 0.5$ while the data are simulated from a mixture of normal distributions. This figure is aimed at illustrating the non-linearity and similarity of the decision boundaries and is not a comparison of the classification performance of the three methods.*

2.7 Multi-class kernel Fisher discriminant analysis

In an SVM setting, several approaches for multi-class classification have been proposed (*cf.* Lee *et al.*, 2001, and Weston, 1999). Two approaches that are often used are the one-against-the-rest and one-against-one respectively. In these cases the multi-class problem is considered as a collection of binary classification problems. In this section we apply these approaches to KFDA.

First consider the one-against-the-rest approach. For g classes in the training data, obtain g binary classifiers constructed using one class, labeled as $+1$, versus the other $(g-1)$ classes combined, labeled as -1. Let

$$f_j(\mathbf{x}) = \sum_{i=1}^{n} \tilde{\alpha}_{ij} k(\mathbf{x}_{ij}, \mathbf{x}) + \tilde{b}_j \ , \ \forall j = 1, \ldots, g$$

be the real-valued output of the discriminant function obtained according to (2.32) for the j-th classifier. Then using this procedure, we classify an observation $\mathbf{x}$ into class j if $f_j(\mathbf{x})$ is the largest of $f_1(\mathbf{x})$, …, $f_g(\mathbf{x})$.

Using the one-against-one approach, the total training data set, containing g classes, is divided into $\upsilon = g!/2!(g-2)!$ pairs (k,j), $k \neq j$, of smaller training data sets, each containing only two classes with class k labeled $+1$ and class j labeled -1. For the υ data sets, υ binary KFD classifiers are obtained according to (2.33). To classify an observation $\mathbf{x}$, a majority voting scheme is used. For example, suppose we have $g = 4$ classes and therefore $\upsilon = 6$ binary classifiers:

$$sign\{f_{12}(\mathbf{x})\},\ sign\{f_{13}(\mathbf{x})\},\ sign\{f_{14}(\mathbf{x})\},\ sign\{f_{23}(\mathbf{x})\},\ sign\{f_{24}(\mathbf{x})\},\ sign\{f_{34}(\mathbf{x})\}$$

corresponding to classes 1-*vs*-2, 1-*vs*-3, 1-*vs*-4, 2-*vs*-3, 2-*vs*-4 and 3-*vs*-4. For these classifiers, observation $\mathbf{x}$ can be classified into class 1 at most 3 times and similarly for classes 2, 3 and 4. Suppose it is classified into class 1 more often than into the other classes, **x** will receive class 1 membership. In general we can say that **x** will be classified into the class with the highest frequency of occurrence. An illustration of the decision boundary produced by using this approach is given in Figure 2.9. The training data, represented by the four classes in the scatter plot, were generated from four different bivariate normal distributions. The one-against-one approach described in the example above was performed on the data using a Gaussian kernel with $\gamma = 0.1$ together with a regularization parameter of $\lambda = 0.1$. The broken line is the decision boundary produced by this approach. The non-linearity in the boundary is due the non-linear kernel used. One of the problems with the one-against-one approach is that the class membership cannot be determined if two or more classes receive the same majority vote. To overcome such problems one can use the real-valued output, $f_{kj}(\mathbf{x})$, instead of the signed output, $sign\{f_{kj}(\mathbf{x})\}$, to do the classification. To illustrate this, suppose class 1 and 2 in the four-class example received the same majority votes $\{1, 1, 4, 2, 2, 3\}$ when the six classifiers were used to classify **x**. In this case we cannot determine the class membership of **x**. However, if we use the values of $f_{kj}(\mathbf{x})$ from which these votes were obtained, say $\{0.534,\ 0.654,\ -0.432,\ 0.221,\ 0.123,\ 0.521\}$, it is evident that the magnitude of the first two values (corresponding to class 1 classification) are much higher than the fourth and the fifth values (corresponding to class 2 classification). Using the magnitude of these values, one would then classify **x** into class 1, since its values are the highest.

The two approaches described in this section are very popular for doing multi-class classification. However, several other ways to generalize two-class KFDA to the multi-class case have been proposed. Since we will restrict attention to the two-class case in this thesis, we refer the reader to Baudat and Anouar (2000), Kuss and Graepel (2003) and Navarrete and del Solar (2002) for detailed discussions on multi-class extensions of KFDA.

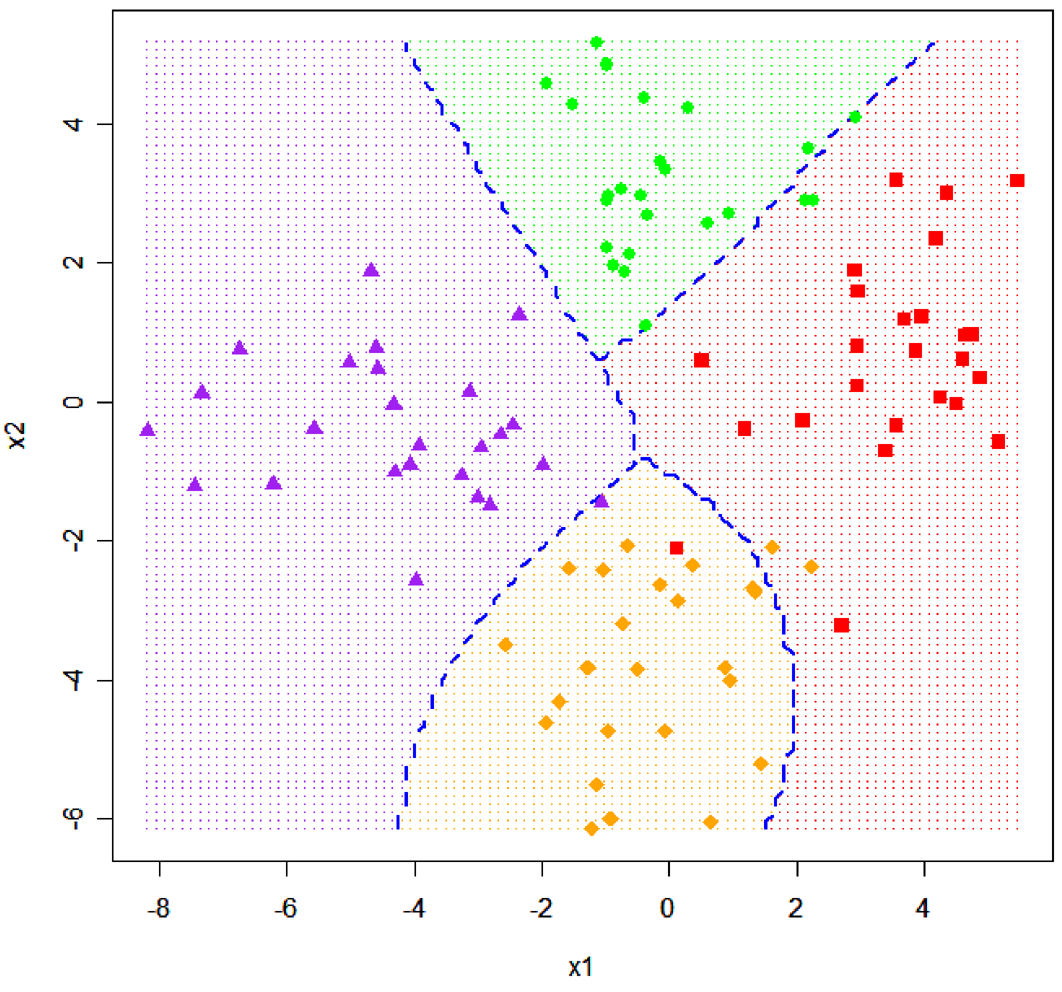


FIGURE 2.9: *Illustrative example of multi-class KFDA decision boundary using the one-against-one approach. The Gaussian kernel was used with* $\gamma = 0.1$ *and* $\lambda = 0.1$. *The data are simulated from four normal distributions with different mean vectors but equal covariance matrices.*

2.8 Summary

In this chapter we have shown that KFDA is a non-linear extension of FLDA. When a linear kernel, $k(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle$, is used, the KFD classifier is equivalent to the FLDA classifier. We illustrated that KFDA is related to the SVM, with the main difference being that the KFD classifier is obtained by maximizing the ratio of the between class-within class variation and the SVM classifier by maximizing the margin in feature space. We have also seen that variants of the KFDA algorithm can be obtained, such as regKFDA described in Section 2.6.

The following are some of the advantages of KFDA:

- KFDA produces a non-linear decision boundary, which is very useful in practice since classes are not always well separated by a linear function.
- Unlike FLDA, KFDA performs well if the populations have unequal covariance matrices (*cf*. Louw and Steel, 2005).
- KFDA can handle data sets where the number of input variables is much larger than the number of observations $(p >> n)$. This is only possible for FLDA with regularization.
- One can calculate posterior probabilities for KFDA by estimating the densities of the projections in feature space (Schölkopf and Smola, 2002; see also Chapter 4).
- The generalization error rate of KFDA compares favourably (and is often superior) to other discrimination techniques such as the SVM and FLDA (see Louw and Steel, 2005, and Mika, 2002, *p*.105).

Some of the disadvantages of KFDA are:

- KFDA is not an "off-the-shelf" procedure. Which kernel function should be used? How do you specify or estimate the hyperparameters such as γ for the Gaussian kernel? How do you specify or estimate the regularization parameter $(\lambda > 0)$ in $\mathbf{N}_\lambda = \mathbf{N} + \lambda\mathbf{I}$? The estimation of these parameters will receive attention in Chapter 3.

- Kernel learning is likely to over/ under–fit if the wrong hyperparameters and/ or kernel function are specified.
- All observations are tunneled through the kernel-bottleneck, making the technique computationally intensive.

KFDA is a fairly new classification tool and there are still many open research questions. However, the application of KFDA in practice has been found to be very successful (*cf.* Mika, 2002).

CHAPTER 3

EMPIRICAL STUDY OF THE EFFECT OF ATYPICAL CASES IN KERNEL FISHER DISCRIMINANT ANALYSIS

"*Mathematics is well and good but nature keeps dragging us around by the nose.*"

–Albert Einstein

3.1 Introduction

It is well-known that atypical cases have a detrimental effect on the generalization performance of the Fisher linear discriminant rule (see for example Critchley and Vitiello, 1991, and Joossens, 2006, in this regard). Since KFDA is a non-linear generalization of FLDA, the question arises whether atypical cases also affect the generalization performance of the KFD classifier. In this chapter we investigate the effect of atypical cases on the generalization performance of the classifier obtained from KFDA. This effect is evaluated via a Monte Carlo simulation study in which the test error rate is estimated. We also investigate the effect of atypical cases on geometrical aspects of KFDA with the aim to develop criteria to identify cases that have a detrimental effect on the generalization performance of the KFD classifier (*cf.* Chapter 5).

This chapter is set out as follows: In Section 3.2 we discuss three basic questions concerning atypical cases and describe the objective of this chapter. Section 3.3 defines several measures that we want to evaluate by means of a simulation study. These measures pertain to the generalization performance and geometrical aspects of KFDA. The data in the simulation study are generated from multivariate normal and lognormal distributions. Section 3.4 describes the data generation process. This section also explains how the training data will be contaminated, *i.e.* how the atypical cases will be inserted into the training data. Section 3.5 deals with hyperparameter estimation in

KFDA. We explain two ways of estimating these parameters. Section 3.6 contains four illustrative examples of atypical cases and their effect on the KFDA decision boundary. A simulation study investigating the effect of atypical cases is described in Section 3.7. A discussion of the simulation results is given in Section 3.7.2. In Section 3.8 we conclude this chapter by summarizing the most important findings of the study.

3.2 Questions concerning atypical cases

Atypical cases in a classification context can generally be defined as observations that are in some sense different from the remaining data in a specific class (*cf*. Figures 3.4 and 3.5 as an illustration of atypical cases). It is known that atypical cases influence model selection, parameter estimation and prediction in many statistical classification models. This normally leads to spurious results which may affect the validity of conclusions. Hence, in a statistical classification problem, the most important questions concerning atypical cases are the following:

- Is the classifier affected by atypical cases?
- If yes, how can we identify the atypical cases?
- How do we deal with such cases?

The main objective of this chapter is to study the effect of atypical cases on the generalization performance, as well as other geometrical aspects of KFDA. As part of this study an extensive Monte Carlo simulation study was performed. The necessary background to carry out this experiment is described in Sections 3.3 to 3.6, while the simulation design and results are reported in Section 3.7.

With regard to the second question, identification of influential cases has been a very active research area in statistics over the last four decades. Identifying such cases is usually a preliminary step in most statistical analyses, because such cases may have both practical and statistical implications. Criteria for the identification of influential cases in statistical models, for example linear and quadratic discriminant analysis, have been proposed in the literature. To our knowledge, no such criteria have been proposed in a KFDA context. The development of criteria to identify influential cases in KFDA is one of the main aims of this thesis and receives attention in Chapter 5.

There are three possible ways to deal with atypical cases. Firstly, by performing a transformation on the input variables the effect of these cases in the analysis can be reduced. The disadvantage of a transformation however, is that important information about the original measurements is lost and we often do not know what transformation to use. Secondly, if these cases are mistakes in the data, they can either be deleted or corrected when possible. Thirdly, some atypical cases are not mistakes and therefore should not be deleted from the data. Such cases can represent a shift in the distribution of the data. Accommodating such observations in models is possible if robust estimates for model parameters can be obtained. Dealing with influential cases in an appropriate manner is an important concern. However, we will focus only on the first two questions in this thesis.

3.3 Aspects of kernel Fisher discriminant analysis

In this section eight measures for quantifying certain aspects of two-class KFDA are discussed. We are specifically interested in studying the behaviour of these measures in the presence as well as the absence of atypical cases, in order to assess the effect of such atypical cases on each measure. The first two measures (the training and the test error) will be used to determine the influence of atypical cases on the classification performance of the KFD classifier. The other six measures are studied specifically with the aim of developing criteria for identification of influential cases (*cf*. Chapter 5). All these measures will be evaluated in a Monte Carlo simulation study as described in Section 3.7.

3.3.1 Training error

The first quantity we consider is the training error (often called the resubstitution error or empirical risk). Quantifying the classification performance of a classifier is very important. The training error gives us an idea of how well the classifier performs when it is applied to the training data. Assume that the training data pairs $(\mathbf{x}, \mathrm{y})$, where $\mathrm{y} \in \{\pm 1\}$ represents the response variable and $\mathbf{x} \in \Re^p$ the input variables, are drawn independently from an unknown joint distribution $\mathrm{p}(\mathbf{x}, \mathrm{y})$. Let $\varphi(\mathbf{x}) = sign\{f(\mathbf{x})\}$ denote the classifier that is estimated from the training data and define the 0-1 loss function

$$\mathbf{l}\{\varphi(\mathbf{x}), \mathrm{y}\} = \begin{cases} 0 & \text{if } \ \mathrm{y} = \varphi(\mathbf{x}) \\ 1 & \text{otherwise.} \end{cases}$$

Then a measure of the classification accuracy of $\varphi(\mathbf{x})$ is its expected misclassification rate

$$\begin{aligned} Err\{\varphi(\mathbf{x}), \mathrm{y}\} &= \mathrm{E}[\mathbf{l}\{\varphi(\mathbf{x}), \mathrm{y}\}] \\ &= \int \mathbf{l}\{\varphi(\mathbf{x}), \mathrm{y}\} d\mathrm{p}(\mathbf{x}, \mathrm{y}). \end{aligned} \tag{3.1}$$

However, since $\mathrm{p}(\mathbf{x}, \mathrm{y})$ is often unknown in practice, (3.1) cannot be calculated directly. The training error of the KFD classifier is an estimate of (3.1) and is defined by

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} I[\varphi(\mathbf{x}_i) \neq \mathrm{y}_i], \tag{3.2}$$

where the classifier $\varphi(\mathbf{x})$ is obtained from the n training data cases using (2.33). The indicator function $I[.]$ is used to count the number of misclassified training data cases. A disadvantage of estimating the KFD classification performance using (3.2) is that this error is optimistic as an estimator. A better estimate of the KFD classification performance is the test error, which is discussed in the next section.

3.3.2 Test error

The test error (often called the generalization error) measures the classification/ generalization performance of a classifier when it is applied to the cases of a new data set drawn from the same joint distribution as the training data. This new data set is known as the test data. The test error of the classifier $\varphi(\mathbf{x})$ is then defined by

$$R_{test} = \frac{1}{n_{\mathrm{T}}} \sum_{i=1}^{n_{\mathrm{T}}} I[\varphi(\mathbf{x}_i) \neq \mathrm{y}_i], \tag{3.3}$$

where $\{(\mathbf{x}_i, \mathrm{y}_i), i = 1, \mathbf{K}, n_{\mathrm{T}}\}$ is the test data set. A low error rate is desirable for a classifier. In the simulation study we will report both the training and the test error. These error rates are calculated using contaminated and uncontaminated (*cf*. Section 3.4.4) training data which will enable us to study the effect of atypical cases on the classification performance of the KFD classifier. Especially the behaviour of the test error is of importance.

3.3.3 Average distance of misclassified training cases to the decision boundary

The measure that we describe in this section is based on the average distance of all the misclassified training cases to the decision boundary produced by KFDA. The distance in feature space from the *l*-th observation in the training data to the decision boundary is given by

$$d_l = \left|\langle \mathbf{v}, \Phi(\mathbf{x}_l)\rangle + b\right| / \|\mathbf{v}\|, \; l = 1,...,n, \tag{3.4}$$

where $\mathbf{v}$ is the KFD weight vector in feature space. Note that this result applies to any classifier that uses a decision boundary which is linear in feature space. The distance (3.4) for KFDA can be written as

$$\begin{aligned}
d_l &= \left|\langle \mathbf{v}, \Phi(\mathbf{x}_l)\rangle + b\right| \Big/ \|\mathbf{v}\| \\
&= \left|\left\langle \sum_{i=1}^{n} \tilde{\alpha}_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_l)\right\rangle + \tilde{b}\right| \Bigg/ \sqrt{\left\langle \sum_{i=1}^{n} \tilde{\alpha}_i \Phi(\mathbf{x}_i), \sum_{l=1}^{n} \tilde{\alpha}_l \Phi(\mathbf{x}_l)\right\rangle} \\
&= \left|\sum_{i=1}^{n} \tilde{\alpha}_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_l)\rangle + \tilde{b}\right| \Bigg/ \sqrt{\sum_{i,l=1}^{n} \tilde{\alpha}_i \tilde{\alpha}_l \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_l)\rangle} \\
&= \left|\sum_{i=1}^{n} \tilde{\alpha}_i k(\mathbf{x}_i, \mathbf{x}_l) + \tilde{b}\right| \Bigg/ \sqrt{\sum_{i,l=1}^{n} \tilde{\alpha}_i \tilde{\alpha}_l k(\mathbf{x}_i, \mathbf{x}_l)} \\
&= \left|\sum_{i=1}^{n} \tilde{\alpha}_i k(\mathbf{x}_i, \mathbf{x}_l) + \tilde{b}\right| \Bigg/ \sqrt{\tilde{a}'\mathbf{G}\tilde{a}}\,, \qquad (3.5)
\end{aligned}$$

where the estimates $\tilde{a}$, $\tilde{b}$ and matrix $\mathbf{G}$ are given by (2.28), (2.31) and (2.26) respectively. Let L index the set of misclassified training observations, and let n_L denote the number of such cases. Then

$$\bar{d} = \frac{1}{n_L} \sum_{l \in L} d_l \qquad (3.6)$$

is the average distance of the misclassified training cases to the KFDA decision boundary. If there are no misclassified cases $\bar{d} = 0$ and if there is little overlap in feature space between classes, $\bar{d}$ will be small. Ideally it is desired that $\bar{d}$ should be zero or very close to zero, since small values imply that the decision boundary succeeds in separating the two classes.

3.3.4 Average margin

The fourth measure makes use of the so-called margin[1]. In feature space, the margin of the *l*-th observation in the training data set is the product of its label, $y_l \in \{\pm 1\}$, and the value of the discriminant function, *viz.*

[1] This margin is defined differently from the margin for the SVM described in Chapter 2.

$$\begin{aligned} m_l &= y_l f(\mathbf{x}_l) \\ &= y_l\left(\langle \mathbf{v}, \Phi(\mathbf{x}_l)\rangle + b\right) \\ &= y_l\left(\sum_{i=1}^{n} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_l)\rangle + b\right). \end{aligned}$$

For KFDA the margin for the *l*-th case can be written in kernel representation as

$$m_l = y_l\left(\sum_{i=1}^{n} \tilde{\alpha}_i k(\mathbf{x}_i,\mathbf{x}_l) + \tilde{b}\right) \tag{3.7}$$

and the average margin is given by

$$\overline{m} = \frac{1}{n}\sum_{l=1}^{n} m_l \ . \tag{3.8}$$

For misclassified training data cases the margin will be negative, whereas correctly classified cases will have a positive margin. Therefore, it is clear that a large average margin is desirable, since it corresponds to a low training error rate.

3.3.5 Rayleigh coefficient

As described in Section 2.4, the KFD classifier is obtained by maximizing the Rayleigh coefficient (2.27). Using the kernel representations described in Section 2.4, the optimal (maximum) value of (2.27) is

$$r = \frac{\tilde{a}'\mathbf{M}\tilde{a}}{\tilde{a}'\mathbf{N}_\lambda \tilde{a}}, \tag{3.9}$$

with $\mathbf{M}$, $\tilde{a}$ and $\mathbf{N}_\lambda$ as defined in (2.24), (2.28) and (2.29) respectively. The effect of atypical cases on the maximized value of the Rayleigh coefficient will be investigated.

3.3.6 Between-class to within-class variance ratio

The mean for the *j*-th class (*j*=1, 2) in feature space was defined in (2.18). Although the mean vector for each of the two classes cannot be evaluated explicitly, the distance between the means can be calculated by using the kernel trick. The squared Euclidean distance between the mean vectors of the two classes, $d\left(\bar{\mathbf{x}}_1^{\Phi}, \bar{\mathbf{x}}_2^{\Phi}\right)$, can be calculated as follows:

$$\begin{aligned}
d\left(\bar{\mathbf{x}}_1^{\Phi}, \bar{\mathbf{x}}_2^{\Phi}\right) &= \left\|\bar{\mathbf{x}}_1^{\Phi} - \bar{\mathbf{x}}_2^{\Phi}\right\|^2 \\
&= \left\langle \bar{\mathbf{x}}_1^{\Phi}, \bar{\mathbf{x}}_1^{\Phi} \right\rangle + \left\langle \bar{\mathbf{x}}_2^{\Phi}, \bar{\mathbf{x}}_2^{\Phi} \right\rangle - 2\left\langle \bar{\mathbf{x}}_1^{\Phi}, \bar{\mathbf{x}}_2^{\Phi} \right\rangle \\
&= \left\langle \frac{1}{n_1}\sum_{k\in J_1}\Phi(\mathbf{x}_k), \frac{1}{n_1}\sum_{l\in J_1}\Phi(\mathbf{x}_l) \right\rangle + \left\langle \frac{1}{n_2}\sum_{k\in J_2}\Phi(\mathbf{x}_k), \frac{1}{n_2}\sum_{l\in J_2}\Phi(\mathbf{x}_l) \right\rangle \\
&\qquad - 2\left\langle \frac{1}{n_1}\sum_{k\in J_1}\Phi(\mathbf{x}_k), \frac{1}{n_2}\sum_{l\in J_2}\Phi(\mathbf{x}_l) \right\rangle \\
&= \frac{1}{n_1^2}\sum_{k\in J_1}\sum_{l\in J_1}\left\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \right\rangle + \frac{1}{n_2^2}\sum_{k\in J_2}\sum_{l\in J_2}\left\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \right\rangle \\
&\qquad - \frac{2}{n_1 n_2}\sum_{k\in J_1}\sum_{l\in J_2}\left\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \right\rangle .
\end{aligned}$$

By replacing the inner products with the kernel function the following is obtained:

$$d\left(\bar{\mathbf{x}}_1^{\Phi}, \bar{\mathbf{x}}_2^{\Phi}\right) = \frac{1}{n_1^2}\sum_{k\in J_1}\sum_{l\in J_1} k(\mathbf{x}_k, \mathbf{x}_l) + \frac{1}{n_2^2}\sum_{k\in J_2}\sum_{l\in J_2} k(\mathbf{x}_k, \mathbf{x}_l) - \frac{2}{n_1 n_2}\sum_{k\in J_1}\sum_{l\in J_2} k(\mathbf{x}_k, \mathbf{x}_l). \qquad (3.10)$$

In a similar manner the average squared deviation in feature space of the observations in the *j*-th class from the mean of that class can be calculated as

$$
\begin{aligned}
s_j^2 &= \frac{1}{n_j}\sum_{k\in J_j}\left\|\Phi(\mathbf{x}_k)-\bar{\mathbf{x}}_j^{\Phi}\right\|^2 \\
&= \frac{1}{n_j}\sum_{k\in J_j}\left[\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_k)\rangle+\langle\bar{\mathbf{x}}_j^{\Phi},\bar{\mathbf{x}}_j^{\Phi}\rangle-2\langle\bar{\mathbf{x}}_j^{\Phi},\Phi(\mathbf{x}_k)\rangle\right] \\
&= \frac{1}{n_j}\left[\sum_{k\in J_j}\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_k)\rangle+\frac{n_j}{n_j^2}\sum_{k\in J_j}\sum_{l\in J_j}\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_l)\rangle-\frac{2}{n_j}\sum_{k\in J_j}\sum_{l\in J_j}\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_l)\rangle\right] \\
&= \frac{1}{n_j}\sum_{k\in J_j}\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_k)\rangle-\frac{1}{n_j^2}\sum_{k\in J_j}\sum_{l\in J_j}\langle\Phi(\mathbf{x}_k),\Phi(\mathbf{x}_l)\rangle, \qquad j=1,2.
\end{aligned}
$$

Again we apply the kernel trick to obtain

$$s_j^2=\frac{1}{n_j}\sum_{k\in J_j}k(\mathbf{x}_k,\mathbf{x}_k)-\frac{1}{n_j^2}\sum_{k\in J_j}\sum_{l\in J_j}k(\mathbf{x}_k,\mathbf{x}_l),\quad j=1,2.$$

Using (3.10) and s_j^2, define the between-class to within-class variance ratio as

$$v=\frac{d\left(\bar{\mathbf{x}}_1^{\Phi},\bar{\mathbf{x}}_2^{\Phi}\right)}{s_1^2+s_2^2}. \tag{3.11}$$

The ratio (3.11) is quite similar to (3.9) in the sense that they both measure a between-within class variance ratio. Note that, unlike the previous measures, (3.11) does not make use of $\tilde{a}$ or $\tilde{b}$ and is therefore independent of the KFDA solution.

3.3.7 Alignment

The concept of kernel target alignment was introduced by Cristianini *et al.* (2002) as a measure of agreement between two Gram matrices. Application of the alignment in our context requires the so-called "ideal" Gram matrix, which is defined by Cristianini *et al.* (2002) as

$$\mathbf{yy}' = \begin{bmatrix} \mathbf{1}_{n_1} & \mathbf{J} \\ \mathbf{J}' & \mathbf{1}_{n_2} \end{bmatrix}, \tag{3.12}$$

where $\mathbf{y}$ is the response vector and therefore $\mathbf{1}_{n_1}$ is an $n_1 \times n_1$ matrix and $\mathbf{1}_{n_2}$ an $n_2 \times n_2$ matrix with all elements equal to 1. The matrix $\mathbf{J}$ is an $n_1 \times n_2$ matrix containing -1 as elements. Since the elements of a Gram matrix may be interpreted as quantities measuring the similarity between observations, it is clear that the ideal Gram matrix reflects "perfect similarity" between cases belonging to the same class, and "perfect dissimilarity" between cases belonging to different classes. The alignment between a given Gram matrix $\mathbf{G}$ and the ideal Gram matrix (3.12) is defined by

$$a = \frac{\langle \mathbf{G}, \mathbf{yy}' \rangle_{\mathrm{F}}}{\sqrt{\langle \mathbf{yy}', \mathbf{yy}' \rangle_{\mathrm{F}} \langle \mathbf{G}, \mathbf{G} \rangle_{\mathrm{F}}}}, \tag{3.13}$$

where $a \in [-1,1]$ and the expressions of the form $\langle .,. \rangle_{\mathrm{F}}$ are known as Frobenius inner products. A Frobenius inner product between pairs of matrices ($\mathbf{A}$ and $\mathbf{B}$) with identical dimensions is $\langle \mathbf{A}, \mathbf{B} \rangle_{\mathrm{F}} = trace(\mathbf{A}'\mathbf{B})$. Note that the quantity (3.13) simplifies to $a = \mathbf{y}'\mathbf{G}\mathbf{y} / n \|\mathbf{G}\|_{\mathrm{F}}$. Cristianini *et al.* (2002) argue that a large value of the alignment is desirable since a large value corresponds to good generalization performance. Like the measure (3.11), the alignment also does not make use of either $\tilde{\alpha}$ or $\tilde{b}$ and is also independent of the KFDA solution.

3.3.8 Length of the vector $\mathbf{v}$

The last measure that will be investigated is the length of the vector $\mathbf{v}$, which can be expressed implicitly (in terms of a kernel representation) as

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{\sum_{i,j=1}^{n} \tilde{\alpha}_i \tilde{\alpha}_j k(\mathbf{x}_i, \mathbf{x}_j)} = \sqrt{\tilde{\boldsymbol{\alpha}}' \mathbf{G} \tilde{\boldsymbol{\alpha}}} \tag{3.14}$$

for KFDA. The expression $\|\mathbf{v}\|$ is also known as the norm of vector **v** and we are interested in studying the behaviour of this length when an atypical case is inserted into the training data.

3.4 Generating the data

Throughout this thesis the training and test data for the simulation studies will be generated from two continuous multivariate distributions: the normal and lognormal distributions. These are examples of a symmetrical distribution and a positively skewed distribution respectively. The following is a brief review of these distributions and the relationship between them.

3.4.1 The normal distribution

The multivariate normal distribution is a generalization of the univariate normal distribution to $p \geq 2$ dimensions. Let W represent a random variable generated from a univariate normal distribution with expected value μ and variance σ^2. The probability density function of such a variable is

$$f(w) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{1}{2}\left(\frac{w-\mu}{\sigma} \right)^2 \right], \quad -\infty < w < \infty, \tag{3.15}$$

where w is the observed value of W. For convenience, such a distribution is often denoted by $\mathrm{N}(\mu,\sigma^2)$. Extending the univariate normal distribution to the multivariate normal distribution is quite straightforward (*cf.* Johnson and Wichern, 1992, *p*.127). Let $\mathbf{W}' = (\mathrm{W}_1, \mathrm{W}_2, ..., \mathrm{W}_p)$ represent a $p \times 1$ random vector generated from a multivariate normal distribution with expected value given by the $p \times 1$ vector $\boldsymbol{\mu}' = (\mu_1, \mu_2, ..., \mu_p)$ and covariance matrix given by the $p \times p$ positive definite matrix $\boldsymbol{\Sigma}$, *viz.*

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \sigma_{p1} & \cdots & \cdots & \sigma_{pp} \end{bmatrix}.$$

The probability density function of such a random vector is

$$f(\mathbf{w}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}, \tag{3.16}$$

where $\mathbf{w}' = (w_1, w_2, ..., w_p)$ is the observed vector of $\mathbf{W}$ and $-\infty < w_i < \infty, i = 1,..., p$. This density function is denoted by $\mathrm{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, analogous to the univariate case. For more detail about the properties of (3.16) see Johnson and Wichern (1992). It is well-known that the bivariate normal density yields a bell-shaped curve which is symmetric around $\boldsymbol{\mu}$. To illustrate the symmetry of this distribution, graphical displays of the bivariate normal density and its contours are given in Figure 3.1. The parameters used to produce these graphs were $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$. In the graph of the contours, the orientation of the ellipse is in line with this positive covariance.

3.4.2 The lognormal distribution

Similar to the normal case above, the multivariate lognormal distribution is a generalization of the univariate lognormal distribution. For the univariate case, a random variable $\mathrm{X} = e^{\mathrm{W}}$ has a lognormal distribution if its natural logarithm, $\mathrm{W} = log(\mathrm{X})$, has a normal distribution. The density function of X is

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{log(x)-\mu}{\sigma}\right)^2\right], \quad 0 \le x < \infty, \tag{3.17}$$

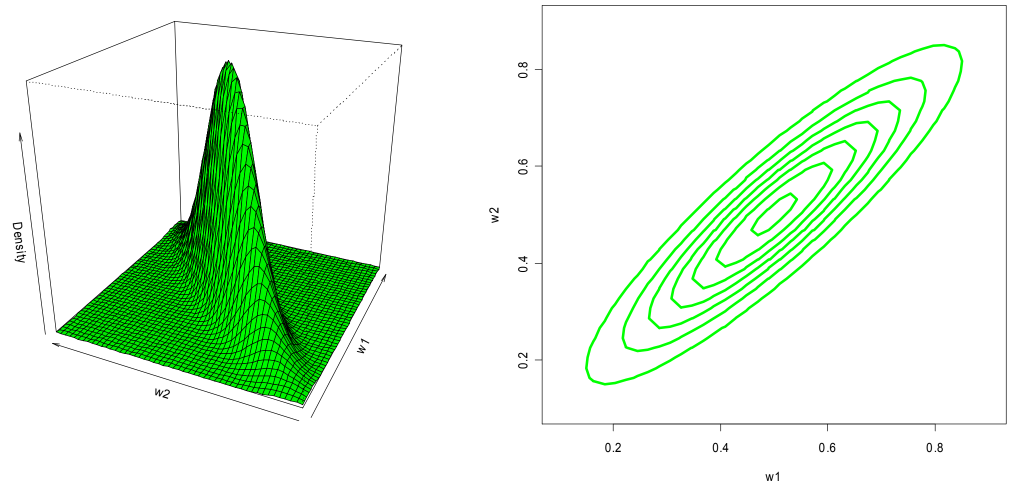


FIGURE 3.1: *The bivariate normal density and normal contours.*

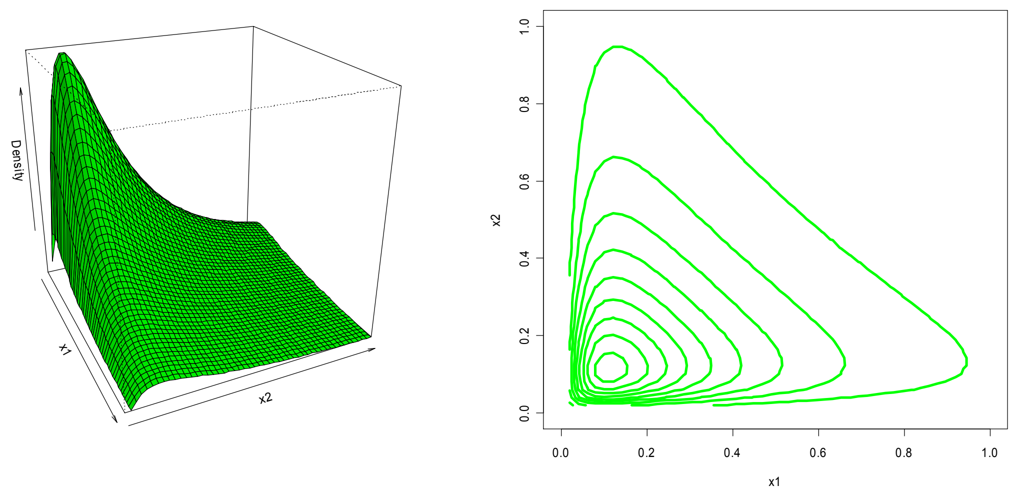


FIGURE 3.2: *The bivariate lognormal density and lognormal contours.*

where x is the observed value of X. For the univariate case the expected value and variance of the lognormal variable will be denoted by m and s^2 respectively. For convenience, we will denote the univariate lognormal distribution by $\mathrm{LN}(\mathrm{m},\mathrm{s}^2)$. The density function (3.17) appears to be quite similar to (3.15). However, a lognormal variable varies between zero and infinity, and its distribution is skewed to the right. It is possible to show that there is a relationship between the parameters (μ,σ^2) of the normal distribution and the parameters of the lognormal distribution $(\mathrm{m},\mathrm{s}^2)$. This relationship will be established in Section 3.4.3.

Similar to the multivariate normal distribution, we can also obtain the multivariate lognormal distribution. Let $\mathbf{X}'=(\mathrm{X}_1,\mathrm{X}_2,...,\mathrm{X}_p)$ represent a *p*-dimensional random vector of lognormal variables with expected value given by the $p\times 1$ vector $\mathbf{m}$ and covariance matrix given by the $p\times p$ matrix $\mathbf{S}$, then $\mathbf{X}$ has a multivariate lognormal distribution. The multivariate lognormal distribution will be denoted by $\mathrm{LN}_p(\mathbf{m},\mathbf{S})$. For all the simulation studies in this thesis we will generate data from a multivariate lognormal distribution using Johnson's translation system (*cf*. Section 3.4.3). This translation exploits the relationship between the normal and lognormal distributions. Graphical displays of the bivariate lognormal density and its contours are given in Figure 3.2, illustrating the skewness of the distribution. These graphs were produced using the bivariate lognormal distribution given in Johnson (1987, *p*.65). The parameters $\mathbf{m}=\begin{bmatrix}0\\0\end{bmatrix}$ and $\mathbf{S}=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$ were used.

3.4.3 Johnson's translation system

Aitchison and Brown (1957) showed the following relationship between the normal parameters (μ,σ^2) and the lognormal parameters $(\mathrm{m},\mathrm{s}^2)$:

$$\mu = log\left(\frac{\mathrm{m}^2}{\sqrt{\mathrm{s}^2+\mathrm{m}^2}}\right) \text{ and } \sigma^2 = log\left[\left(\frac{\mathrm{s}}{\mathrm{m}}\right)^2+1\right], \tag{3.18}$$

hence

$$\mathrm{m} = e^{\frac{2\mu+\sigma^2}{2}} \text{ and } \mathrm{s}^2 = e^{2\mu+\sigma^2}\left(e^{\sigma^2}-1\right). \tag{3.19}$$

In the simulation studies we will be generating data from a multivariate lognormal distribution by using what is known as Johnson's translation system. This system uses the relationship above by first generating normal data and then transforming it to lognormal data. The following is an explanation of the process.

Let $\mathrm{W}_i \sim \mathrm{N}\left(0,\sigma_i^2\right)$ and $\mathrm{cov}\left(\mathrm{W}_i,\mathrm{W}_j\right)=\sigma_{ij}$, for all $i, j = 1,\ldots, p$. Using the notation for the multivariate normal distribution, we therefore have $\mathbf{W} \sim \mathrm{N}_p\left(\mathbf{0},\mathbf{\Sigma}_\mathrm{w}\right)$, where the covariance and correlation matrices are

$$\mathbf{\Sigma}_\mathrm{w} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \sigma_{p1} & \cdots & \cdots & \sigma_p^2 \end{bmatrix} \text{ and } \mathbf{P}_\mathrm{w} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \rho_{p1} & \cdots & \cdots & 1 \end{bmatrix},$$

with the correlation $\rho_{ij} = \sigma_{ij} \Big/ \sqrt{\sigma_i^2\sigma_j^2}$. For simplicity assume equal variances, $\sigma_1^2 = \ldots = \sigma_p^2 = \sigma^2$, and equal covariances, $\sigma_{ij} = \sigma_{ji} = a$. The correlation between all pairs of variables therefore becomes $\rho = a/\sigma^2$. It is possible to obtain multivariate lognormal data, with a common correlation r between variables, by starting from the normal setup. To achieve this, we have to define the covariance a in terms of r. This can be done as follows: let $z_i = w_i/\sigma$ be a transformation of w_i for $i = 1,\ldots, p$, so that $\mathbf{Z} \sim \mathrm{N}_p\left(\mathbf{0},\sigma^{-2}\mathbf{\Sigma}_\mathrm{w}\right)$. From this it follows that the covariance matrix of $\mathbf{Z}$ is equal to the correlation matrix of $\mathbf{Z}$, *viz.* $\mathbf{\Sigma}_{\mathbf{z}} = \sigma^{-2}\mathbf{\Sigma}_{\mathrm{w}} = \mathbf{P}_{\mathbf{z}}$. Using the results in (3.19) it is possible to show that $e^{Z_i} \sim \mathrm{LN}\left(\sqrt{e}, e(e-1)\right)$ and it can be standardized as follows:

$$x_i = \frac{e^{z_i} - \sqrt{e}}{\sqrt{e(e-1)}}, \qquad \forall i = 1,\ldots,p. \tag{3.20}$$

It can also be shown (*cf.* Law and Kelton, 2000) that the covariance between the *i*-th and the *j*-th standardized variable (3.20) is equal to the correlation between the same variables, *i.e.*

$$\mathrm{cov}\left(\mathrm{X}_i,\mathrm{X}_j\right) = \frac{e^{\frac{a}{\sigma^2}} - 1}{e - 1} = \mathrm{corr}\left(\mathrm{X}_i,\mathrm{X}_j\right).$$

Let $r = \mathrm{corr}\left(\mathrm{X}_i,\mathrm{X}_j\right)$, then by a few calculations the following equation defines a in terms of r,

$$a = \sigma^2 log\left[r(e-1)+1\right]. \tag{3.21}$$

Thus for $\mathbf{W} \sim \mathrm{N}_p\left(\mathbf{0}, \mathbf{\Sigma}_{\mathrm{w}}\right)$, with

$$\mathbf{\Sigma}_{\mathrm{w}} = \begin{bmatrix} \sigma^2 & a & \mathbf{L} & a \\ a & \sigma^2 & \mathbf{O} & \mathrm{M} \\ \mathrm{M} & \mathbf{O} & \mathbf{O} & \mathrm{M} \\ a & \mathbf{L} & \mathbf{L} & \sigma^2 \end{bmatrix}$$

and covariance a defined by (3.21), the $p \times 1$ vector

$$\mathbf{X} = \sigma\left(k\mathbf{Y} + \mathbf{g}\right), \tag{3.22}$$

follows a multivariate lognormal distribution. The $p \times 1$ vector $\mathbf{Y}$ in (3.22) is

$$\mathbf{Y}' = \begin{bmatrix} e^{W_1/\sigma} & e^{W_2/\sigma} & \dots & e^{W_p/\sigma} \end{bmatrix},$$

the value of the constant $k = \left(\sqrt{e^2 - e}\right)^{-1}$ and the *i*-th element of the $p \times 1$ vector $\mathbf{g}$ is $g_i = \left(-\sqrt{e-1}\right)^{-1}$ for $i = 1, \dots, p$. The parameters of the lognormal distribution are $\mathbf{m} = \mathbf{0}$ and

$$\mathbf{S} = \begin{bmatrix} \sigma^2 & c & \cdots & c \\ c & \sigma^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ c & \cdots & \cdots & \sigma^2 \end{bmatrix},$$

with c denoting the covariance for the lognormal variables. The corresponding correlation matrix is

$$\mathbf{P_x} = \begin{bmatrix} 1 & r & \cdots & r \\ r & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ r & \cdots & \cdots & 1 \end{bmatrix}.$$

Note that for this example, the variances of the normal and lognormal variables are the same but the covariances for these variables are different for the respective distributions. The transformation (3.22) described above is referred to as Johnson's translation system (*cf.* Johnson, 1987). Figure 3.3 contains scatter plots of normal and lognormal data that were produced using Johnson's translation system with $\sigma^2 = 2$ and $r = 0.5$, and therefore $a = 1.24$, $c = r\sigma^2 = 1$ and $\rho = 0.6201$.

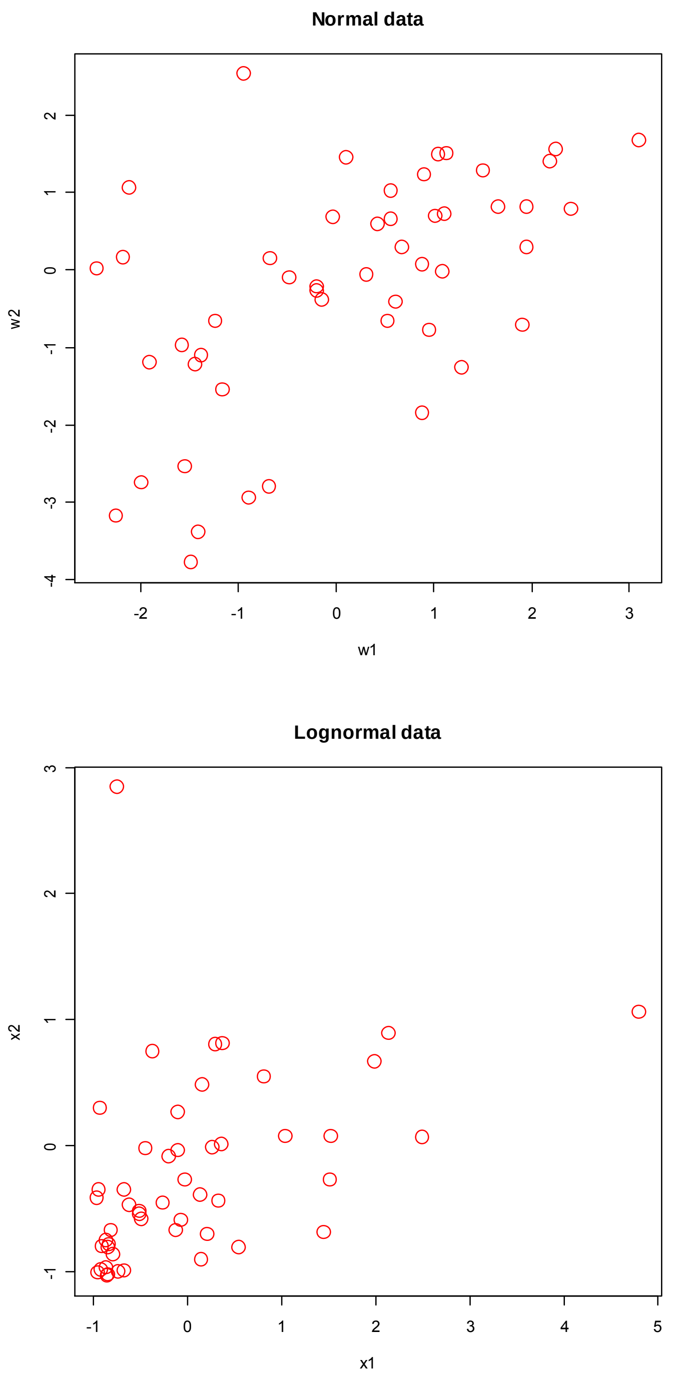


FIGURE 3.3: *An example of normal data which are transformed into lognormal data by using Johnson's translation system. For the normal data we used parameters* $\mathrm{m}' = \begin{bmatrix} 0 & 0 \end{bmatrix}$ *and* $\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1.24 \\ 1.24 & 2 \end{bmatrix}$*, which translates to lognormal data with* $\mathbf{m} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ *and* $\mathbf{S} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

3.4.4 Contamination of the training data

To study the effect of an atypical case on the measures defined in Section 3.3, the training data from class 2 will be contaminated. In the simulations we will study two cases with respect to the differences between the two populations:

Case 1: The populations have equal dispersions, but differ with respect to their locations.
Case 2: The populations have equal locations, but differ with respect to their dispersions. Class 1 has a larger dispersion than class 2.

For both cases, two types of atypical cases will be considered in the contamination of the training data (see Table 3.1 for more detail):

- *Mislabelled case*: place one observation in class 2 which is generated from the distribution of class 1.
- *Moderate outlier*: place one observation in class 2 which lies “further away” than the mislabelled case.

A schematic illustration of the types of atypical cases for *Case* 1 and *Case* 2 are given in Figures 3.4 and 3.5 respectively.

The atypical cases will be generated from the same underlying distribution as the typical cases of class 2, but with a different mean vector or covariance matrix. For *Case* 1, the atypical cases will be generated using different mean vectors and for *Case* 2, the atypical cases will be generated using different covariance matrices. Table 3.1 contains a summary of the types of atypical cases as well as the mean vectors and covariance matrices that will be used to generate these observations. This table also displays the parameters with which the typical class 1 and class 2 cases will be generated. The two distributions discussed in Section 3.4, normal and lognormal, will be used.

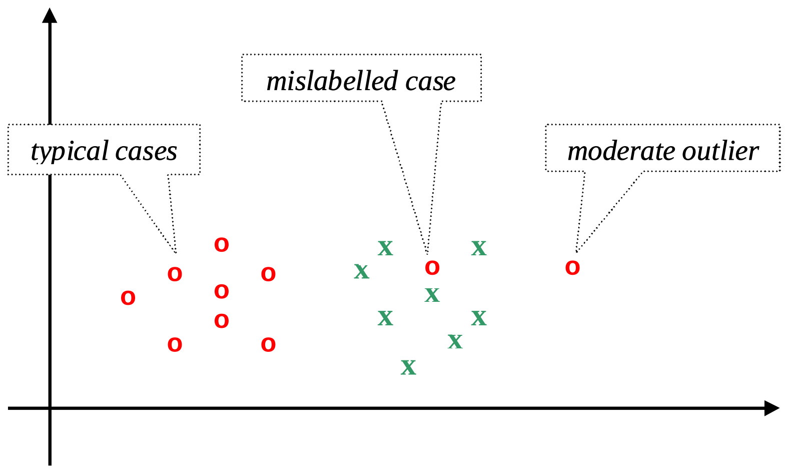


FIGURE 3.4: *Schematic illustration of the contamination of class 2 for different locations but equal dispersions in populations. The circles represent class 2 cases and the crosses the class 1 cases. The atypical cases are generated using the same dispersion as class 2 but with larger locations.*

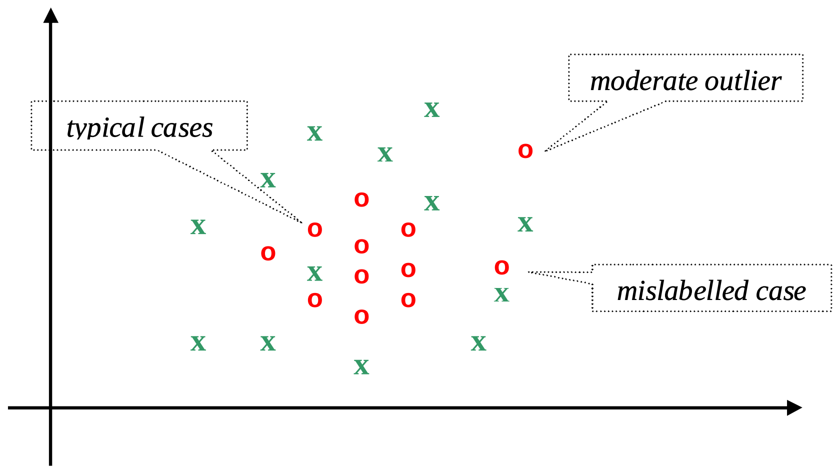


FIGURE 3.5: *Schematic illustration of the contamination of class 2 for different dispersions but equal locations in populations. The circles represent class 2 cases and the crosses the class 1 cases. The atypical cases are generated using the same location as class 2 but with larger dispersions.*

TABLE 3.1: The distribution parameters of the two populations for *Case* 1 and *Case* 2, as well as the parameters with which the two types of atypical cases will be generated.

Case 1: Location differences

	Normal $\mathrm{m}_1 = 2.5\mathbf{1}$; $\mathrm{m}_2 = \mathbf{0}$ $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}^*$	Lognormal $\mathrm{m}_1 = \mathbf{1}$; $\mathrm{m}_2 = \mathbf{0}$ $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}^*$
1. Mislabelled case	$\mathrm{m} = 2.5\mathbf{1}$	$\mathrm{m} = \mathbf{1}$
2. Moderate outlier	$\mathrm{m} = 3.5\mathbf{1}$	$\mathrm{m} = 2\mathbf{1}$

Case 2: Dispersion differences

	Normal $\mathrm{m}_1 = \mathrm{m}_2 = \mathbf{0}$ $\boldsymbol{\Sigma}_1 = 4^2\boldsymbol{\Sigma}^*$; $\boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}^*$	Lognormal $\mathrm{m}_1 = \mathrm{m}_2 = \mathbf{0}$ $\boldsymbol{\Sigma}_1 = 4^2\boldsymbol{\Sigma}^*$; $\boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}^*$
1. Mislabelled case	$\boldsymbol{\Sigma} = 4^2\boldsymbol{\Sigma}^*$	$\boldsymbol{\Sigma} = 4^2\boldsymbol{\Sigma}^*$
2. Moderate outlier	$\boldsymbol{\Sigma} = 5^2\boldsymbol{\Sigma}^*$	$\boldsymbol{\Sigma} = 5^2\boldsymbol{\Sigma}^*$

$\boldsymbol{\Sigma}^*$ is a $p \times p$ matrix with diagonal elements 1, and off-diagonal elements ρ.

The values $\rho = -0.1, 0, 0.7$ will be used in the simulation study.

3.4.5 Standardization of the data

Before applying KFDA, for all the studies in this thesis, each input variable in the training data will be standardized to have zero mean and unit variance across classes. The input variables for the test data are standardized using the means and standard deviations obtained from the training data. This is a common practice in applications of kernel methods and often leads to improved classification performance (*cf.* Schölkopf and Smola, 2002).

3.5 Parameter estimation in kernel Fisher discriminant analysis

As mentioned in Section 2.8, KFDA is not an "off-the-shelf" procedure. This section looks in more detail at the choice of the kernel function and estimation of the hyperparameters needed to perform KFDA. We will use the regularization (2.29), in which the regularization parameter λ has to be estimated. In Chapter 2 it was shown that the KFD classifier can be obtained as (2.33). To use (2.33) a kernel function has to be chosen. The Gaussian kernel (*cf.* Table 2.1) is a popular choice in kernel-based methods and has only one hyperparameter, γ, that has to be estimated. We will use this kernel in all the simulation studies unless we specify another kernel. As mentioned by Louw and Steel (2005), the classification performance of the KFD classifier is quite sensitive to the values of the parameters (γ, λ). The most commonly used method to estimate these parameters is *v*-fold cross-validation. Consider the following two scenarios.

3.5.1 Estimating both λ and γ via cross-validation

In this case a grid of (γ, λ) pairs is specified. Suppose we have *r* γ-values $(\gamma_1, \gamma_2, ..., \gamma_r)$ and *s* λ-values $(\lambda_1, \lambda_2, ..., \lambda_s)$ yielding a total of $t = rs$ pairs. These values are often specified on a log-scale. The training data are then divided into *v* subsets. One subset is used as a test data set while the other $v-1$ subsets are used to train a KFD classifier. The error rate for each of the *t* pairs of parameter values is then obtained by classifying the test data set. This is repeated for all the other subsets and an average error rate is calculated for each hyperparameter pair. The (γ, λ) pair corresponding to the lowest average error rate is used as the final hyperparamater values to obtain the KFD classifier on the full data set.

3.5.2 Estimating γ via other methods and λ via cross-validation

Several other methods to estimate γ have been proposed in the literature. As an example, consider a proposal by Wang *et al.* (2003) where a Gaussian kernel is used in an SVM context. They state that γ in the Gaussian kernel strongly affects the generalization performance of SVMs. If γ is too large or too small, it may lead to poor generalization performance. They argue that an optimal value for γ is needed and propose that such a γ can be obtained by minimizing

$$f(\gamma)=\frac{\mathrm{v}_1^2+\mathrm{v}_2^2}{d\left(\bar{\mathbf{x}}_1^{\Phi},\bar{\mathbf{x}}_2^{\Phi}\right)},$$

where

$$\mathrm{v}_j^2=n_j-\frac{1}{n_j}\sum_{k\in J_j}\sum_{l\in J_j}k(\mathbf{x}_k,\mathbf{x}_l),\ \text{for}\ \ j=1,2$$

and $d\left(\bar{\mathbf{x}}_1^{\Phi},\bar{\mathbf{x}}_2^{\Phi}\right)$ was defined in (3.10). Note that their criterion is proportional to the reciprocal of (3.11) defined in Section 3.3. In a simulation study they applied this optimization using an SVM classifier. A range of values from 0.1 to 500 is specified for γ. The value corresponding to the minimum of $f(\gamma)$ is chosen. Their simulation results show that this approach is effective in selecting an appropriate γ, *i.e.* that the selected γ corresponds to a low generalization error for the resulting SVM.

Other contributions to finding an appropriate γ for the Gaussian kernel include Cristianini *et al*. (1998), Chapelle *et al*. (2002), Keerthi (2002), Keerthi and Lin (2003), and Louw and Steel (2005). Even though methods such as those mentioned in the references here work very well in obtaining an optimal γ, empirical evidence has shown that using $\gamma=1/p$ often works quite well when employing the Gaussian kernel in kernel-based classification.

Once the value for γ has been obtained, by using for example any of the procedures above, one can obtain λ via cross-validation similar to the first scenario but much faster, since we only have to search over the s λ-values. Finding the best estimates for the parameters is still an area of ongoing research. It should be noted that the first scenario that we described in Section 3.5.1 is computationally quite intensive, especially when dealing with large data sets. Hence, we will use a cross-validation search for λ together with $\gamma=1/p$ throughout this thesis.

3.6 The effect of atypical cases on the KFDA decision boundary

This section is aimed at demonstrating the effect of the two types of atypical cases (described in Table 3.1) on the KFDA decision boundary by means of two-dimensional graphs. Four examples are given, which represent a combination of two distributions together with a location or dispersion difference between populations. Figures 3.6 and 3.7 contain scatter plots and the decision boundaries representing the two-class case with data generated from bivariate normal distributions, while Figures 3.8 and 3.9 contain the scatter plots and decision boundaries when the data are generated from bivariate lognormal distributions. Figures 3.6 and 3.8 represent the situation where the populations have different locations, but equal dispersions, while Figures 3.7 and 3.9 represent the case where populations have equal locations but different dispersions. A Gaussian kernel with $\gamma = 0.5$ was used together with a regularization parameter of $\lambda = 0.001$ for all examples. The green squares represent class 1 and the red diamonds represent class 2. The atypical cases in the scatter plots are marked with a solid red point.

The first example refers to the scatter plots in Figure 3.6 which represents the case of two normal populations with equal covariance matrices $\left(\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{I}\right)$ but different locations $\left(\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2\right)$. The scatter plots look quite similar, but note that two of the graphs contain an atypical case. The scatter plot in panel (a) contains the training data with only typical cases for both classes, *i.e.* the observations for class 1 were generated from a population with mean $\mathtt{m}_1 = 2.5\mathbf{1}$ and the observations for class 2 were generated from a population with mean $\mathtt{m}_2 = \mathbf{0}$. The scatter plot in panel (b) represents the same data cases, but one observation which was generated from a population with mean $\mathtt{m} = 2.5\mathbf{1}$, *i.e.* a mislabelled case, is placed in class 2. The scatter plot in panel (c) is the case where a moderate outlier is present in class 2, *i.e.* an atypical case was generated from a population with mean $\mathtt{m} = 3.5\mathbf{1}$. The dotted blue line represents the KFD decision boundary. The change in the decision boundary is quite significant as we progress from a typical case situation to a moderate outlier situation.

In Figure 3.7 the case of equal mean vectors $(\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0})$ but different covariance matrices $(\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2)$ for two normal populations are displayed. Panel (a) contains the training data with typical cases for the classes, *i.e*. class 1 has covariance matrix $\boldsymbol{\Sigma}_1 = 4^2\mathbf{I}$ and class 2 has covariance matrix $\boldsymbol{\Sigma}_2 = \mathbf{I}$. Similar to Figure 3.6, the rest of the scatter plots represent the same data but also contain an atypical case. Class 2 in panel (b) contains a mislabelled case which was generated from a population with covariance matrix $\boldsymbol{\Sigma} = 4^2\mathbf{I}$. Panel (c) contains a moderate outlier in class 2, *i.e*. an atypical case generated from a population with covariance matrix $\boldsymbol{\Sigma} = 5^2\mathbf{I}$. Again we observe dramatic changes in the decision boundaries due to the presence of the atypical case.

The next example pertains to Figure 3.8. The scatter plots in this figure represent the lognormal training data with equal covariance matrices $(\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \mathbf{I})$, but location differences between classes. Panel (a) represents the typical cases where class 1 has unit mean vector $\mathtt{m}_1 = \mathbf{1}$ and class 2 has mean $\mathtt{m}_2 = \mathbf{0}$. Panel (b) is the situation where the mislabelled case, which was generated from a population with $\mathtt{m} = \mathbf{1}$, is inserted. Panel (c) is the case where the moderate outlier, which was generated from a population with $\mathtt{m} = 2\mathbf{1}$, is inserted. The change in the decision boundary is evident.

The last example, corresponding to Figure 3.9, is the case where lognormal training data are used with equal locations $(\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0})$, but different dispersions for the classes. Similar to the normal case in Figure 3.7, panel (a) in Figure 3.9 represents the typical cases where class 1 has covariance matrix $\boldsymbol{\Sigma}_1 = 4^2\mathbf{I}$ and class 2 has covariance matrix $\boldsymbol{\Sigma}_2 = \mathbf{I}$. Panel (b) contains the mislabelled class 2 case which was generated from a population with covariance matrix $\boldsymbol{\Sigma} = 4^2\mathbf{I}$. Panel (c) contains the moderate outlier in class 2 which was generated from a population with covariance matrix $\boldsymbol{\Sigma} = 5^2\mathbf{I}$. Again we see how the decision boundary changes as we introduce the atypical cases.

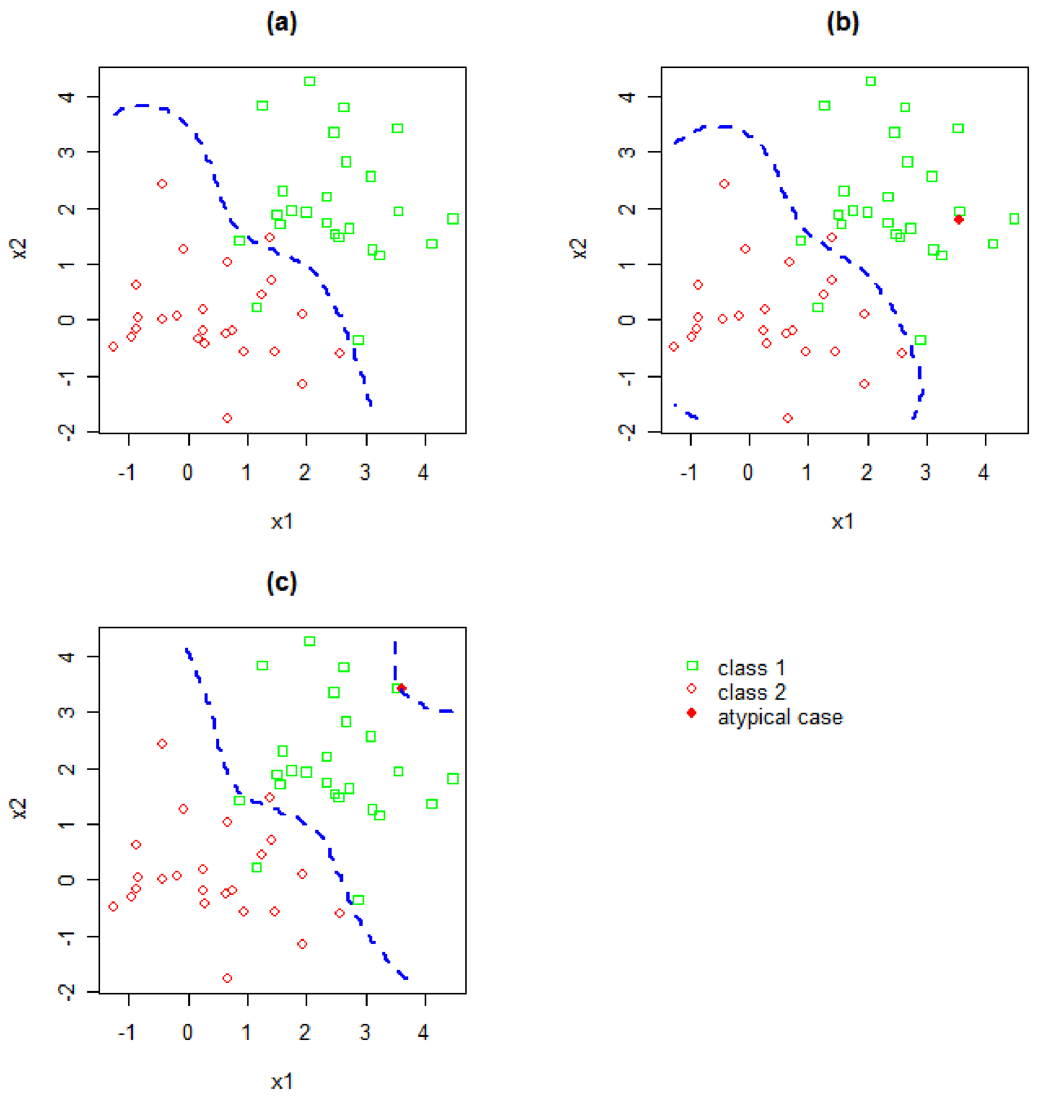


FIGURE 3.6: *Examples of the KFDA decision boundary when the Gaussian kernel is used on normal data with location differences between classes. Panel (a) represents the uncontaminated data set, panel (b) contains the mislabelled case and panel (c) contains the moderate outlier.*

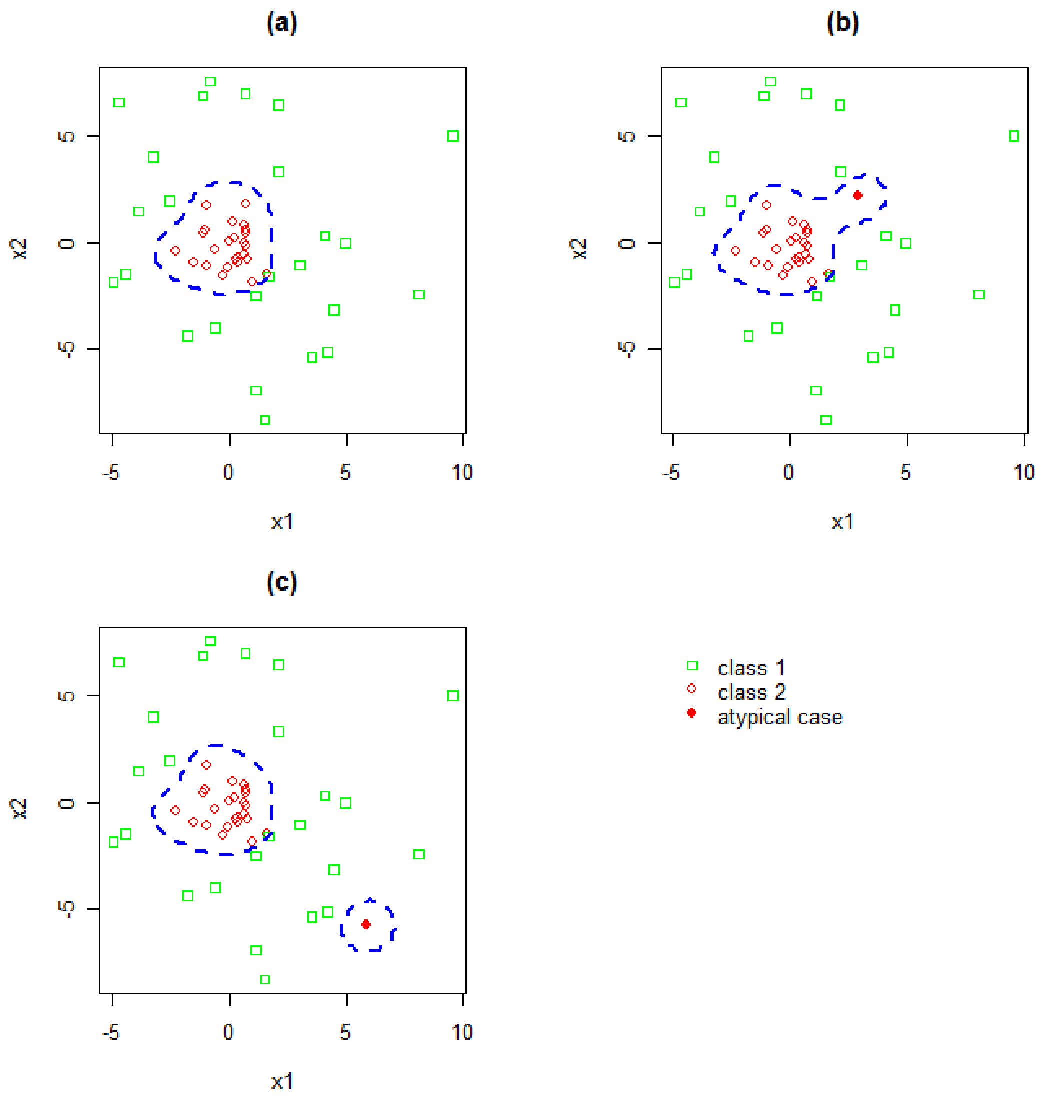


FIGURE 3.7: *Examples of the KFDA decision boundary when the Gaussian kernel is used on normal data with dispersion differences between classes. Panel (a) represents the uncontaminated data set, panel (b) contains the mislabelled case and panel (c) contains the moderate outlier.*

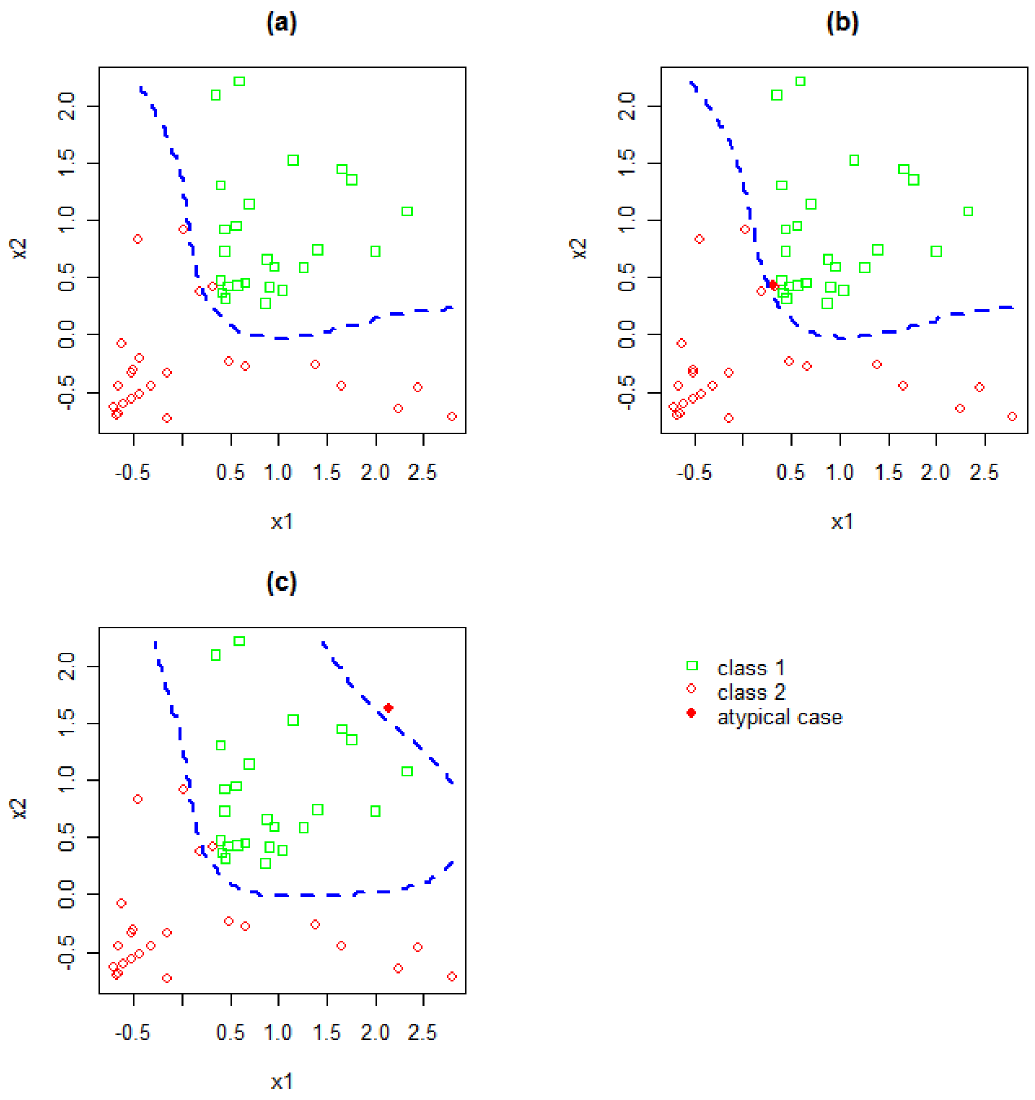


FIGURE 3.8: *Examples of the KFDA decision boundary when the Gaussian kernel is used on lognormal data with location differences between classes. Panel (a) represents the uncontaminated data set, panel (b) contains the mislabelled case and panel (c) contains the moderate outlier.*

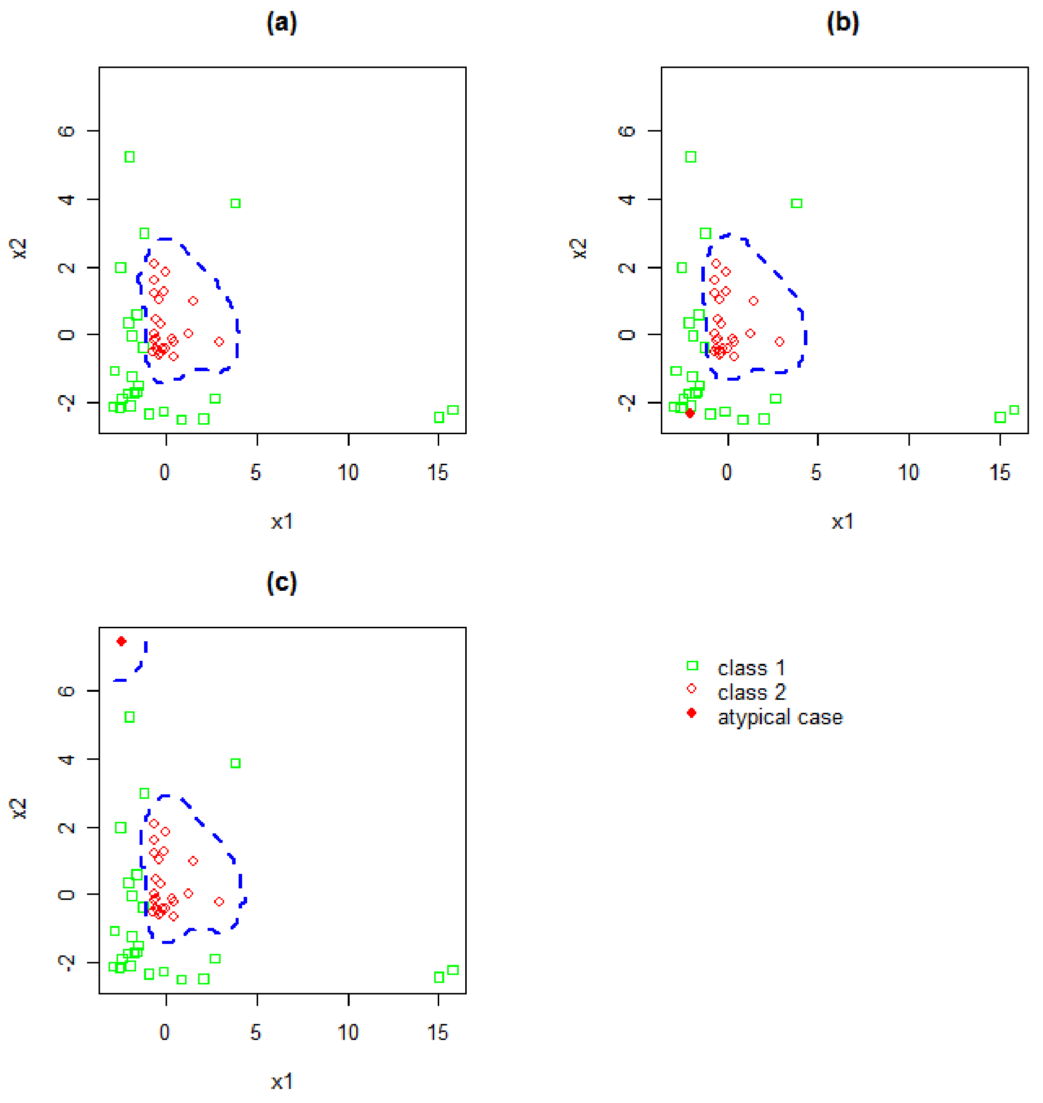


FIGURE 3.9: *Examples of the KFDA decision boundary when the Gaussian kernel is used on lognormal data with dispersion differences between classes Panel (a) represents the uncontaminated data set, panel (b) contains the mislabelled case and panel (c) contains the moderate outlier.*

Judging the change in the decision boundaries in Figures 3.6 to 3.9, one can definitely expect that the generalization performance of the KFD classifier will be influenced by the presence of the atypical case in the training data. Naturally, one would expect the generalization performance to deteriorate. However, these examples are too few and too limited to conclude what will happen to the generalization performance. To investigate the generalization performance of the KFD classifier properly, we will describe a simulation study in the next section. In this study the KFD classifier, based on a contaminated and an uncontaminated training data set respectively, will be used to obtain the test error rates. Our first aim with this study is to compare the test error rates with and without the atypical case in the training data. To test whether the difference in these test error rates are significant, a two-sample hypothesis test will be conducted at a 5% level of significance. The second aim with this study is to investigate the effect of the atypical case on the other measures defined in Section 3.3.

3.7 The effect of atypical cases on the measures: a simulation study

3.7.1 Quantifying the effect of the atypical case on the measures

As mentioned earlier, our main objective in this section is to study the effect of atypical cases on the eight measures introduced in Section 3.3. The question that arises is, how to quantify this effect? For measures (3.2), (3.3) and (3.6) the effect of the atypical case is quantified by calculating the percentage decrease in the measure upon omitting the atypical case. For measures (3.8), (3.9), (3.11), (3.13) and (3.14) the effect of the atypical case is quantified by calculating the percentage increase in the measure upon omitting the atypical case.

These percentages were obtained as follows in the simulation study: Let $\mathcal{T}^{\mathrm{C}} = \{(\mathbf{x}_i, \mathrm{y}_i), i = 1,...,n\}$ denote a contaminated training data set and let the uncontaminated data set (where only the atypical case is omitted from $\mathcal{T}^{\mathrm{C}}$) be denoted by $\mathcal{T} = \{(\mathbf{x}_i, \mathrm{y}_i), i = 1,...,n-1\}$. To quantify the effect of the atypical case, each measure needs to be calculated on both T and T^{C} respectively. Furthermore, let θ denote any one of the measures and let K be the number of simulation repetitions. For the k-th $(k = 1,\ldots,K)$ simulation repetition the data sets T and T^{C} are generated, and the measures θ^{T} and $\theta^{T^{C}}$ are calculated. The averages of these measures over the K repetitions can therefore be obtained as

$$\bar{\theta}^{T} = \frac{1}{K}\sum_{k=1}^{K}\theta_k^{T} \text{ and } \bar{\theta}^{T^{C}} = \frac{1}{K}\sum_{k=1}^{K}\theta_k^{T^{C}}$$

for the uncontaminated and contaminated data respectively. From these values, the percentage decrease when the atypical case is omitted is obtained as

$$P_{\mathrm{D}} = 100\times\left(\bar{\theta}^{T^{C}} - \bar{\theta}^{T}\right)\Big/\bar{\theta}^{T^{C}} ,$$

and the percentage increase when the atypical case is omitted is

$$P_{\mathrm{I}} = 100\times\left(\bar{\theta}^{T} - \bar{\theta}^{T^{C}}\right)\Big/\bar{\theta}^{T^{C}} .$$

Thus, P_{D} was calculated for measures (3.2), (3.3) and (3.6), while P_{I} was calculated for measures (3.8), (3.9), (3.11), (3.13) and (3.14) for each configuration in the simulation study.

To test whether the change in the test error, due to the omission of the atypical case, is significant, we also performed a hypothesis test for two independent normally distributed samples (normality can be assumed because of the large number of simulations and the Central limit theorem). The z-test statistic was calculated for each configuration and a z-value greater than 1.96 or less than -1.96 indicates a significant difference between the test errors at a 5% level of significance.

3.7.2 Design of the simulation study

We will now elaborate more on the design of the Monte Carlo simulation study that was undertaken to obtain P_{I} and P_{D} for the measures. The simulation experiment was conducted for each combination of the following factors and their levels over $K = 1000$ simulation repetitions:

- The underlying distributions from which the training data were generated. We used the normal and lognormal distributions described in Section 3.4.
- With respect to sample sizes, three levels were used, *viz.* small $(n_1 = n_2 = 25)$, large $(n_1 = n_2 = 100)$ and mixed $(n_1 = 75 \text{ and } n_2 = 25)$. These sizes will be denoted by the letters S, L and M respectively.
- The number of input variables used are $p = 5$.
- The correlations between input variables had three levels. Using a common correlation (ρ) for all pairs of variables, we investigated the cases where $\rho = -0.1,\ 0$ and 0.7.
- The parameters with respect to which the two populations differ. We investigated two cases, *viz.*

  Case 1: Difference in location. In this case we have identical covariance structures for both populations,

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \rho & \cdots & \cdots & 1 \end{bmatrix} = \boldsymbol{\Sigma}^*, \tag{3.23}$$

  but with a difference in their respective mean vectors. We used $\boldsymbol{\mu}_2 = \mathbf{0}$ for Π_2 throughout the study, but set $\boldsymbol{\mu}_1 = c\mathbf{1}$ for Π_1, where $c = 2.5$ was used for normal data and $c = 1$ for lognormal data. A smaller value for c is used in the case of the lognormal data, since a larger value results in the classes being (almost always) completely separated (having no overlap).

Case 2: Difference in dispersion. Here we used equal locations for both Π_1 and Π_2, $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0}$. The covariance matrix $\boldsymbol{\Sigma}_2$ remained the same as in (3.23), but we inflated $\boldsymbol{\Sigma}_1$ by putting $\boldsymbol{\Sigma}_1 = \sigma^2 \boldsymbol{\Sigma}^*$, with $\sigma = 4$.

The data set was then contaminated as described in Section 3.4.4, standardized as described in Section 3.4.5 and used to obtain the KFD classifier $\varphi_{T^C}(\mathbf{x})$. The atypical case was then omitted from the training set to obtain the uncontaminated data. This data set was also standardized and used to obtain the second KFD classifier $\varphi_T(\mathbf{x})$. Regarding the test data, independent uncontaminated samples were generated from the same populations as the training data. A test data set of 10000 observations $(n_1 = n_2 = 5000)$ was used for small and large training data sets. The size of the test data for the mixed training data set was 10000 observations $(n_1 = 7500 \text{ and } n_2 = 2500)$. The test data set was also standardized using the means and standard deviations of the contaminated and uncontaminated training data respectively, and then classified by both $\varphi_{T^C}(\mathbf{x})$ and $\varphi_T(\mathbf{x})$ in order to estimate the respective error rates. The hyperparameter value of $\gamma = 1/p$ was used throughout. Regarding the values of the regularization parameter λ, a search was done over the following ranges,

- normal data with location differences: $\lambda = e^x, x = -8, -7, -6, ..., 7$
- normal data with dispersion differences: $\lambda = e^x, x = -13, -12, -11, ..., 19$
- lognormal data with location differences: $\lambda = e^x, x = -17, -16, -15, ..., 2$
- lognormal data with dispersion differences: $\lambda = e^x, x = -14, -13, -12, ..., 8$.

For each configuration, the value of λ corresponding to the minimum test error rate for $\varphi_T(\mathbf{x})$ was selected.

3.7.3 Discussion of the simulation results

Tables 3.2 to 3.5 contain the results of the simulation study. Each table is divided into the following three parts: the configurations, the test error information and the rest of the measures as described in Section 3.3. Tables 3.2 and 3.4 contain the results when the mislabelled case was omitted from the training data, while Tables 3.3 and 3.5 contain the results when the moderate outlier was omitted from the training data. For our discussion we will interpret each of the measures separately but over all the tables.

Consider first the information regarding the test error (R_{test}). Note that the decrease in the test error as well as the *z*-value is reported here. An absolute *z*-value of 1.96 or more indicates that the change in the test error, due to omission of the atypical case, is significant. In each table, the significant cases are highlighted using **bold face** figures. Normally one would expect atypical cases to influence the generalization performance of a classifier in a negative way. As expected, it is clear from the tables that the omission of an atypical case generally results in a decrease in the test error. Most of the associated *z*-values are larger than 1.96, meaning that these decreases are in fact significant. In Table 3.4 there were two cases where an increase was observed of which only one case is significant. As to be expected, the decrease in the test error is more significant (larger *z*-values) for the small sample cases than for the large sample cases.

Secondly, consider the percentage decrease in R_{emp} and $\bar{d}$:

- Similar to the test error, one would expect R_{emp} to decrease in the absence of atypical cases. The results show that the percentage decrease in R_{emp} is quite high for most of the configurations. The high values indicate that this measure is very sensitive to atypical cases.
- It seems reasonable to expect that $\bar{d}$ will decrease when an atypical case is omitted from the training data. The columns representing $\bar{d}$ show that this measure does in fact decrease for most configurations. There are also a few configurations where an

increase is observed, but this increase is relatively small in magnitude compared to the rest of the results.

Finally, consider the percentage increase in $\overline{m}$, r, v, a and $\|\mathbf{v}\|$:

- As mentioned in Section 3.3.4, a good generalization performance is often associated with a large value of $\overline{m}$. Since the generalization performance usually improves in the absence of an atypical case, we can expect $\overline{m}$ to increase as well. The simulation results show that there was an increase in $\overline{m}$ for all the configurations.
- The Rayleigh coefficient (r) is also expected to increase in the absence of atypical cases since it is the optimization criterion for KFDA. The simulation results indicate that there is an increase in r due to the omitted atypical cases.
- The results representing the increase in v show that this measure increases when an atypical case is omitted from the training data.
- The alignment (a) also increases in the absence of an atypical case. However, the percentage increase for a is quite low compared to the other measures. This indicates that this measure is not as sensitive to atypical cases as the other measures.
- In KFDA the optimal vector **v** can explicitly be obtained as a function of the within-class scatter matrix and the difference between the class mean vectors in feature space, *i.e* $\tilde{\mathbf{v}} = \left(\mathbf{S}_{\mathrm{w}}^{\Phi}\right)^{-1}\left(\overline{\mathbf{x}}_1^{\Phi} - \overline{\mathbf{x}}_2^{\Phi}\right)$. From this expression it can easily be shown that $\|\mathbf{v}\|$ will increase in the absence of an atypical case. The simulation results show that this is indeed the case. Only one configuration has a small decrease for this measure.

TABLE 3.2: Percentage increase/ decrease; normal data; mislabelled case

	Configurations		Test error		Other measures						
	sample size	ρ	z-value	R^*_{test}	R^*_{emp}	$\bar{d}^*$	$\bar{m}^{**}$	r^{**}	v^{**}	a^{**}	$\lVert\mathbf{v}\rVert^{**}$
Location difference	S	-0.1	3.923	**16.612**	99.492	99.978	26.689	197.081	12.416	8.299	21.533
	S	0	2.553	**3.057**	89.176	97.444	24.947	152.792	12.176	8.059	19.915
	S	0.7	1.13	0.318	19.381	24.294	19.66	43.91	12.292	8.159	14.723
	M	-0.1	3.5	**15.858**	99.1	99.877	32.115	73.127	11.997	4.658	28.053
	M	0	2.333	**2.851**	85.398	97.653	30.228	63.14	11.988	4.738	26.268
	M	0.7	1.119	0.275	13.927	15.582	20.999	25.953	11.976	4.838	17.055
	L	-0.1	0.5	1.765	98.022	99.779	15.614	44.193	2.956	2.026	14.43
	L	0	0.458	0.407	67.708	92.544	13.819	33.885	2.917	2.008	12.661
	L	0.7	0.258	0.043	5.555	6.758	6.853	8.764	2.927	2.012	5.74
Dispersion difference	S	-0.1	11.2	**9.926**	34.453	38.341	28.813	46.378	12.164	1.479	23.71
	S	0	10.652	**9.888**	34.637	42.666	28.497	46.761	11.972	1.282	23.493
	S	0.7	11.614	**11.029**	10.443	8.179	16.547	14.135	11.655	1.035	13.068
	M	-0.1	5.145	**4.089**	20.476	46.148	13.098	16.355	12.223	4.495	9.991
	M	0	5.063	**4.15**	20.942	47.292	13.724	17.254	12.566	4.377	10.554
	M	0.7	3.677	**1.896**	13.774	45.225	10.625	12.482	12.24	5.272	7.636
	L	-0.1	10.901	**5.095**	9.02	-7.244	7.97	6.099	3.056	0.363	2.208
	L	0	10.912	**5.162**	8.517	-8.629	8.026	6.047	3.086	0.393	2.25
	L	0.7	7.248	**4.186**	6.981	-3.411	6.072	5.586	2.924	0.189	2.729

[*] Percentage decrease in measure
[**] Percentage increase in measure

TABLE 3.3: Percentage increase/ decrease; lognormal data; mislabelled case

	Configurations		Test error		Other measures						
	sample size	ρ	z-value	R^{*}_{test}	R^{*}_{emp}	$\bar{d}^{*}$	$\bar{m}^{**}$	r^{**}	v^{**}	a^{**}	$\lVert\mathbf{v}\rVert^{**}$
Location difference	S	-0.1	11.574	**14.12**	99.833	99.966	149.668	178.232	10.119	8.314	121.726
	S	0	7.501	**7.869**	94.164	97.962	93.452	118.75	10.056	8.324	77.8
	S	0.7	2.316	**1.548**	61.961	78.007	65.19	80.635	11.167	8.404	52.052
	M	-0.1	4.122	**4.57**	97.271	99.266	80.996	127.624	10.326	4.925	71.252
	M	0	3.009	**3.159**	85.912	95.661	65.077	98.545	10.314	4.94	56.646
	M	0.7	0.23	0.161	20.347	29.738	26.254	33.531	11.374	5.137	20.978
	L	-0.1	5.592	**5.258**	94.795	98.521	64.068	78.67	2.474	2.023	56.87
	L	0	3.012	**2.3**	72.218	87.671	36.891	45.862	2.474	2.038	32.973
	L	0.7	2.36	**1.162**	26.544	37.803	21.646	24.371	2.662	2.061	17.936
Dispersion difference	S	-0.1	15.001	**22.925**	55.485	45.044	38.929	34.058	11.836	2.257	26.559
	S	0	11.174	**10.594**	45.735	36.626	33.88	32.877	11.685	2.346	22.901
	S	0.7	7.28	**9.498**	44.329	50.858	55.194	59.491	12.49	4.356	39.865
	M	-0.1	9.23	**11.708**	43.249	52.565	23.065	25.771	11.635	5.773	16.414
	M	0	6.042	**5.244**	21.648	23.732	16.326	18.129	11.185	5.927	11.169
	M	0.7	3.608	**3.196**	15.047	15.968	14.251	16.18	11.697	5.851	10.994
	L	-0.1	7.373	**7.115**	37.715	40.718	15.412	16.242	2.973	0.136	11.816
	L	0	5.531	**3.616**	18.984	10.344	10.141	10.537	3.141	0.464	6.931
	L	0.7	2.423	**1.711**	9.658	7.773	8.664	9.402	3.113	0.794	6.698

[*] Percentage decrease in measure
[**] Percentage increase in measure

TABLE 3.4: Percentage increase/ decrease; normal data; moderate outlier

	Configurations		Test error		Other measures						
	sample size	ρ	z-value	R^*_{test}	R^*_{emp}	$\bar{d}^*$	$\bar{m}^{**}$	r^{**}	v^{**}	a^{**}	$\|\mathbf{v}\|^{**}$
Location difference	S	-0.1	19.752	**81.245**	100	100	125.04	121.551	9.487	7.029	93.514
	S	0	20.656	**41.866**	96.319	98.127	107.195	107.267	9.665	7.321	79.803
	S	0.7	8.062	**4.171**	22.898	11.37	44.357	45.706	11.931	8.503	24.35
	M	-0.1	-5.067	**-22.125**	100	100	136.079	184.474	10.198	3.413	113.04
	M	0	-1.35	-2.354	95.485	98.431	111.877	145.193	10.209	3.501	94.97
	M	0.7	2.664	**1.248**	16.36	5.151	29.642	30.568	12.503	4.251	16.299
	L	-0.1	2.143	**6.383**	98.922	99.717	49.1	66.897	2.211	1.7	43.486
	L	0	1.364	1.351	77.93	89.502	35.018	42.973	2.246	1.784	30.854
	L	0.7	0.671	0.132	6.112	4.922	10.206	10.277	2.765	2.06	7.829
Dispersion difference	S	-0.1	16.189	**13.855**	9.665	-3.811	22.918	12.345	14.14	3.606	7.242
	S	0	16.015	**13.646**	9.316	-1.922	22.942	12.421	14.169	3.641	7.118
	S	0.7	10.28	**7.751**	18.687	25.367	12.595	25.456	11.655	1.035	7.797
	M	-0.1	8.819	**6.688**	10.296	-5.997	11.994	9.466	15.078	3.592	4.233
	M	0	9.352	**6.857**	11.8	-6.533	12.418	10.014	15.108	3.591	4.916
	M	0.7	6.034	**4.426**	11.883	0.146	11.759	10.242	15.048	4.196	6.915
	L	-0.1	5.527	**2.753**	10.528	16.762	5.981	11.402	3.056	0.363	4.818
	L	0	5.34	**2.774**	10.988	16.764	6.119	11.909	3.086	0.393	4.942
	L	0.7	3.331	**1.488**	4.794	8.223	5.405	6.417	2.924	0.189	4.27

* Percentage decrease in measure
** Percentage increase in measure

TABLE 3.5: Percentage increase/ decrease; lognormal data; moderate outlier

	Configurations		Test error		Other measures						
	sample size	ρ	z-value	R^{*}_{test}	R^{*}_{emp}	$\bar{d}^{*}$	$\bar{m}^{**}$	r^{**}	v^{**}	a^{**}	$\lVert\mathbf{v}\rVert^{**}$
Location difference	S	-0.1	12.058	**13.057**	39	12.945	16.449	4.449	8.199	7.571	5.343
	S	0	14.52	**14.053**	53.176	53.201	23.597	13	8.423	7.816	12.578
	S	0.7	18.928	**12.049**	40.952	45.11	37.606	38.664	12.041	8.668	32.785
	M	-0.1	10.409	**11.18**	70.326	76.05	19.507	12.91	9.543	2.867	9.744
	M	0	12.24	**11.204**	50.389	48.286	22.756	20.234	9.84	2.884	13.509
	M	0.7	2.379	**1.645**	22.515	22.336	25.226	32.391	13.021	3.359	21.999
	L	-0.1	8.055	**7.048**	20.465	4.606	4.347	2.139	2.326	1.969	-1.058
	L	0	12.852	**9.17**	26.561	16.793	9.155	6.398	2.381	2.023	3.584
	L	0.7	12.607	**6.159**	18.714	10.027	14.197	13.142	3.214	2.165	8.03
Dispersion difference	S	-0.1	13.06	**23.298**	35.519	26.242	32.179	27.248	14.63	4.881	21.962
	S	0	12.592	**12.251**	33.481	24.741	33.546	30.191	14.532	4.791	21.886
	S	0.7	9.291	**12.466**	27.724	25.005	41.309	43.623	15.217	6.453	33.487
	M	-0.1	12.572	**16.175**	42.288	47.423	24.224	23.73	14.755	4.955	14.431
	M	0	7.253	**6.036**	22.7	18.838	17.99	17.942	14.986	5.061	10.879
	M	0.7	5.786	**5.158**	17.473	15.09	16.097	17.295	16.048	5.441	16.04
	L	-0.1	10.928	**10.384**	31.802	31.907	15.392	14.775	3.823	0.936	10.86
	L	0	6.912	**5.317**	16.689	7.283	10.745	10.283	3.883	1.135	7.624
	L	0.7	4.538	**3.103**	9.767	5.431	8.986	9.059	3.903	1.433	9.165

* Percentage decrease in measure
** Percentage increase in measure

3.8 Concluding remarks

From the results for the test error we conclude that atypical cases do have a detrimental effect on the generalization performance of the KFD classifier. For most configurations in the simulation results the test error rate increases significantly when an atypical case is present in the training data. We saw in the simulation study that the other measures, defined in Section 3.3, are also influenced by the presence of an atypical case in the training data.

However, in practice we do not know which cases in the training data are the atypical cases, but since we now know that the above mentioned measures are affected by atypical cases, the question arises whether these measures can be used as criteria to identify such cases in a data set. In Chapter 5 we will address this question by investigating the effectiveness of these measures in identifying cases that influence the generalization performance of the KFD classifier.

Other questions that arise from this chapter are the following:

- In the simulation study we only worked with the Gaussian kernel. What will the effect of atypical cases be for other kernel functions?
- We studied the effect of single atypical cases. It would also be interesting to see the effect of multiple atypical cases in KFDA.
- In the simulation study, the parameter for the Gaussian kernel was fixed at $\gamma = 1/p$ and potentially different values for λ was used for each configuration. During the analysis it was found that the solutions to KFDA is highly sensitive to the choice of λ. Obtaining the appropriate parameter values in KFDA still needs further investigation.

CHAPTER 4

ESTIMATING POSTERIOR PROBABILITIES IN KERNEL FISHER DISCRIMINANT ANALYSIS

"The most important questions of life are, for the most part, really only problems of probability."

- Pierre-Simon Laplace

4.1 Introduction

When performing a discriminant analysis, the classification of new observations is often only a part of the research investigation. Researchers are often interested in additional aspects of the data, for example: are there any outliers or influential cases in the data, which are the most important variables, how many missing values are in the data. Classifiers that produce posterior probabilities (conditional probability) of class membership for the observations are often quite useful in practice, since it enables researchers to get more insight into the data. For example, Fung (1995a) used posterior probabilities to identify influential cases in FLDA. A posterior probability quantifies how confident we are that an observation belongs to a specific class. One of the advantages of KFDA is that posterior probabilities of class membership can be obtained from its output. In this chapter we explain how to estimate such posterior probabilities in the two-class case by using two different approaches. In Chapter 5 we will use these posterior probabilities to develop a criterion for identifying atypical cases in KFDA.

- The first approach is suggested by Schölkopf and Smola (2002, *p*.464), who state that the projections (*cf*. Section 4.2.1) of the data in each class in feature space onto the direction of discrimination are approximately normally distributed. The normal densities of the projections can therefore be estimated and by applying Bayes' rule the estimated posterior probability of class membership can be obtained. The detail of this approach is discussed in Section 4.3.1.

- The second approach is based on a proposal by Platt (2000) who estimated posterior probabilities in an SVM context. He proposed using logistic regression on the SVM output to obtain the posterior probabilities of class membership. In Section 4.3.2 we apply Platt’s idea to KFDA by performing logistic regression on the discriminant scores.

The remainder of this chapter is organized as follows: In Section 4.2 we explain how the projections are obtained and evaluate the normality assumption for the projections in a simulation study using three different kernels. In Section 4.3 we give the detail of the two posterior probability approaches mentioned above. Instead of using the KFD classifier (2.33), a classifier based on the posterior probabilities of group membership can also be constructed. In Section 4.4 we compare the generalization performance of classifiers based on the two posterior probability approaches and the KFD classifier (2.33) in a simulation study. Section 4.5 contains general conclusions and a summary of the important findings of the simulation studies.

4.2 Evaluating the assumption of normality of the projections

In this section we will evaluate by means of a Monte Carlo simulation study whether the assumption of normality of the projections in feature space, stated by Schölkopf and Smola (2002, *p*.464), is valid. Before we evaluate the normality assumption, we first show how these projections are obtained.

4.2.1 Obtaining the projections for KFDA

To obtain the projections in KFDA, we first have to find the optimal vector $\mathbf{v} \in F$ which corresponds to the direction of discrimination in feature space along which the classes are maximally separated when the observations are projected onto it.

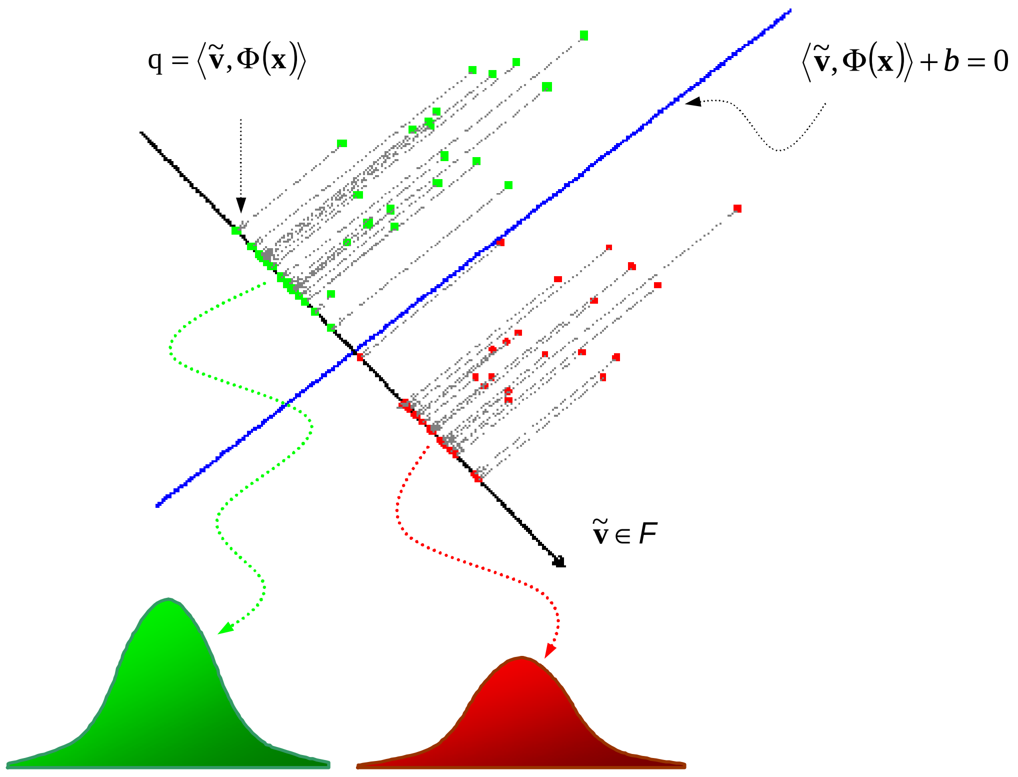

$q = \langle \tilde{\mathbf{v}}, \Phi(\mathbf{x}) \rangle$
$\langle \tilde{\mathbf{v}}, \Phi(\mathbf{x}) \rangle + b = 0$
$\tilde{\mathbf{v}} \in F$

FIGURE 4.1: *Schematic illustration of the projections of the observations onto* $\tilde{\mathbf{v}}$ *in feature space. The blue line is the decision boundary and the arrow represents the direction of discrimination. Schölkopf and Smola (2002) state that these projections for each class follow a normal distribution.*

This optimal vector will be denoted by $\tilde{\mathbf{v}}$ and is illustrated in Figure 4.1. To obtain the projection of the *i*-th training observation in feature space, $\Phi(\mathbf{x}_i)$, onto $\tilde{\mathbf{v}}$, we calculate the inner product $\langle \tilde{\mathbf{v}}, \Phi(\mathbf{x}_i) \rangle$. By using expansion (2.21), the projection of point $\Phi(\mathbf{x}_i)$ onto $\tilde{\mathbf{v}}$ is defined in Schölkopf and Smola (2002, *p*.459) as

$$\mathrm{q}_i = \left\langle \tilde{\mathbf{v}}, \Phi(\mathbf{x}_i) \right\rangle = \sum_{j=1}^{n} \tilde{\alpha}_j \left\langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_i) \right\rangle, \; i = 1, \ldots, n, \tag{4.1}$$

with $\tilde{\alpha}_j$ being the *j*-th element of the vector $\tilde{\mathrm{a}}$ defined in (2.28). The projections (4.1) cannot be calculated explicitly but by replacing the inner product with a kernel function we obtain the kernel representation of (4.1) as,

$$\mathrm{q}_i = \sum_{j=1}^{n} \tilde{\alpha}_j k(\mathbf{x}_j, \mathbf{x}_i), \quad i = 1, \ldots, n. \tag{4.2}$$

According to Schölkopf and Smola (2002, *p*.464), empirical evidence suggests that the histogram of each class in the training data as projected onto the direction of discrimination $\tilde{\mathbf{v}}$ can be closely approximated by a normal distribution. Except for a few graphs comparing these histograms to estimated normal distributions, they do not give any other evidence backing this statement. In the following example we give two graphical illustrations which shows that the projections are not always normally distributed.

Figures 4.2 and 4.3 represent histograms of the projections for the two classes compared to estimated normal density functions. Note that these estimated density functions were obtained using the empirical mean and standard deviation (*cf.* (4.10)) as parameters for the normal distribution. In both figures the projections were obtained according to (4.2) using a Gaussian kernel with $\gamma = 1/p$ and a regularization parameter of $\lambda = 0.1$. The training data set of size 200 $(n_1 = n_2 = 100)$ with 50 variables was generated from two multivariate normal distributions having different locations and equal dispersions for the two classes. The parameters for the normal distributions are given in each of the Figures. In Figure 4.2 the variables are uncorrelated, while the variables used to produce Figure 4.3 have pairwise correlations of 0.5. It is evident from Figure 4.2 that the projections for each class in this particular case approximately follow a normal distribution. Both histograms are symmetrical in shape and the estimated normal densities fit quite well on these projections. However, in Figure 4.3 where the correlation between the variables is changed, the shape of the histograms becomes skewed and the estimated normal density does not fit the distribution of the projections. From Figure 4.3 it is evident that the distribution of the projections may deviate from normality under certain circumstances.

In the next subsections we describe a simulation study to investigate the validity of the normality assumption for different kernel functions and for different distributions of the training data by means of hypothesis tests for normality. The investigation will be done for three kernels, *i.e.* the linear kernel, the Gaussian kernel and the inverse multi-quadric kernel (*cf.* Table 2.1). Using (4.2) we can write an expression for the projections using each of these kernels. For the linear kernel the expression to calculate the projections can be written as

$$\mathrm{q}_i = \sum_{j=1}^{n} \tilde{\alpha}_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle, \tag{4.3}$$

for the Gaussian kernel we express the projections as

$$\mathrm{q}_i = \sum_{j=1}^{n} \tilde{\alpha}_j e^{-\gamma \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2}, \tag{4.4}$$

and for the inverse multi-quadric kernel we have

$$\mathrm{q}_i = \sum_{j=1}^{n} \tilde{\alpha}_j \frac{1}{\sqrt{\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 + d^2}}, \qquad i = 1, \ldots, n. \tag{4.5}$$

The hyperparameters $\gamma = 1/p$ for the Gaussian kernel and $d = 0.1$ for the inverse multi-quadric kernel will be used. A regularization parameter $\lambda = 0.1$ will be used throughout. The design and the results of the simulation study are discussed in Sections 4.2.2 and 4.2.3 respectively.

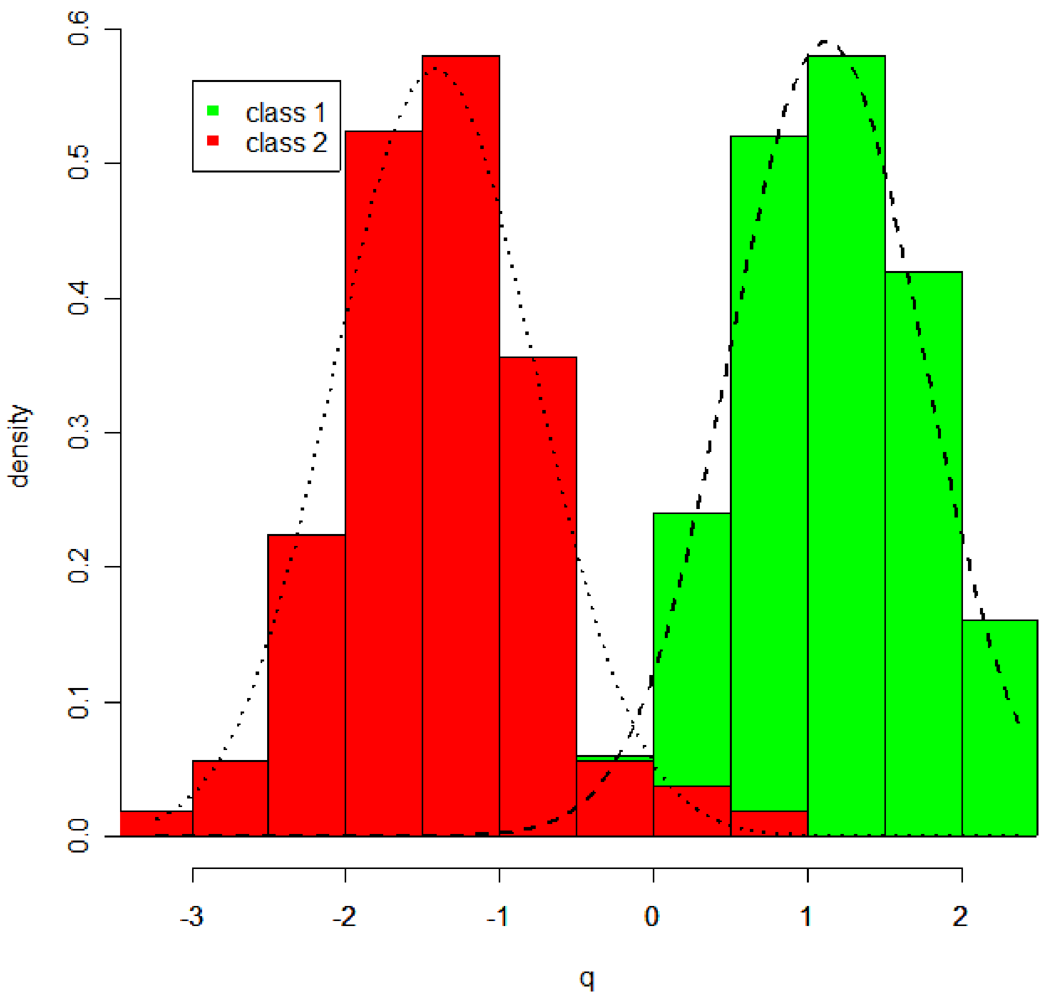


FIGURE 4.2: *An illustrative example of the histograms of the projections with an estimated normal density superimposed. The training data set was generated from two normal distributions with equal covariance matrices,* $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{I}$, *and uncorrelated variables. The mean vectors for class 1 and class 2 are* $\mathrm{m}_1 = \mathbf{0}$ *and* $\mathrm{m}_2 = 0.45\mathbf{1}$ *respectively.*

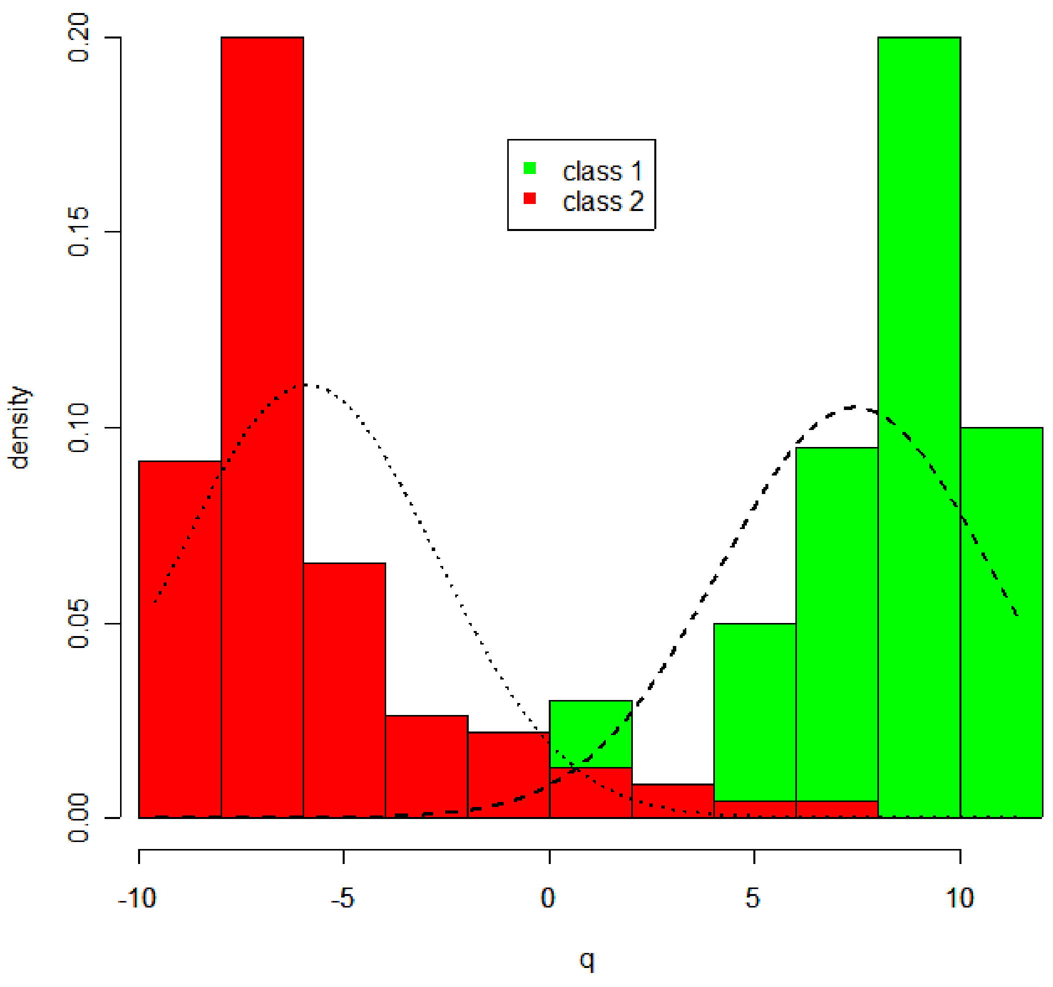


FIGURE 4.3: *An illustrative example of the histograms of the projections with an estimated normal density superimposed. The training data set was generated from two normal distributions with equal covariance matrices,* $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{I}$*, and with 0.5 correlation between variables. The mean vectors for class 1 and class 2 are* $\mathrm{m}_1 = \mathbf{0}$ *and* $\mathrm{m}_2 = 2.251$ *respectively.*

4.2.2 Design of the simulation study

We will now proceed with the investigation of the validity of the normality assumption by means of a Monte Carlo simulation study. Again, we consider the two-class case with populations denoted by Π_1 and Π_2. Training data will be generated from two different distributions and the projections for each class will be obtained using (4.2). Each of the three kernels mentioned in Section 4.2.1 will be used to obtain the respective projections, and a hypothesis test for normality will then be performed on these projections. The results of the simulation study are reported in Section 4.2.3. To calculate the projections and to do the investigation, a simulation study was performed using $K = 100$ simulation repetitions for each combination of the following factors:

- Training data were generated from the normal and lognormal distributions.
- The sizes of the training data sets used are small (S) with $n_1 = n_2 = 25$, and large (L) with $n_1 = n_2 = 100$.
- Three levels for the number of input variables were used, *viz.* p = 5, 50 and 100.
- Different levels for the correlations between each pair of input variables were used, *viz.* ρ = 0, 0.5 and 0.9.

We consider the case where the two populations have equal covariance matrices, $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}^*$ (*cf.* (3.23)), but differ with respect to their locations. The location difference was obtained by specifying a value $\Delta^2 = 5$ for the squared Mahalanobis distance,

$$\Delta^2 = (\mathrm{m}_1 - \mathrm{m}_2)' \mathbf{\Sigma}^{*-1} (\mathrm{m}_1 - \mathrm{m}_2), \tag{4.6}$$

between the two populations. The following parameterization was used to give this distance: the mean of Π_1 was chosen as $\mathrm{m}_1 = \mathbf{0}$, while the mean of Π_2 was chosen as $\mathrm{m}_2 = c\mathbf{1}$, with

$$c = \Delta \Big/ \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p}\sigma^{ij}}\ , \tag{4.7}$$

where σ^{ij}, $i, j = 1,...p$, are the elements of $\mathbf{\Sigma}^{*-1}$.

The Shapiro-Wilk hypothesis test for normality (*cf.* Shapiro and Wilk, 1965) was employed to ascertain whether the projections calculated by means of (4.3), (4.4) and (4.5) approximately follow a normal distribution for the configurations described above. The null and alternative hypotheses of this test are as follows:

H_0 : The projections of class *j* are normally distributed

H_A : The projections of class *j* are not normally distributed.

A 5% level of significance was used throughout. The *p*-value associated with the test statistic was obtained and if this value was greater than or equal to the level of significance, H_0 was accepted.

4.2.3 Discussion of the simulation results

The results of the simulation study (*cf.* Table 4.1) are reported as the relative frequency of acceptance of the null hypothesis. This was obtained as follows: Let N index the set of cases for which H_0 was not rejected in class 1 and class 2 over the K simulation repetitions, then the relative frequencies of normality are calculated as

$$f_N = \frac{1}{2K}\sum_{i \in N} I[\mathrm{p}_i \geq 0.05], \tag{4.8}$$

where p_i represents the *p*-value associated with the test statistic. This was computed for each simulation configuration. By studying the results in Table 4.1, the following conclusions are drawn:

Consider first the case with the normal training data. When the linear kernel is used to obtain the projections, we observe that the relative frequencies of normality are very high, which indicates that the projections are approximately normally distributed. The low relative frequencies when the Gaussian and inverse multi-quadric kernels are employed, indicate that the projections are frequently not normally distributed when using these kernels. For these two kernels and uncorrelated variables, we see that the relative frequencies of normality increases with an increase in the number of variables. This indicates that with no correlation between input variables, the projections tend to be normally distributed in high dimensions. For the other configurations, these kernels do not produce normally distributed projections.

For the case where lognormal training data are used, the picture changes for the linear kernel. The relative frequencies are only high if the number of input variables is high and the variables are uncorrelated. If there is a strong correlation between variables, the relative frequency is quite low, which indicates that the projections do not appear to follow a normal distribution for such configurations. When the Gaussian and inverse multi-quadric kernels were used, we see that the results for the lognormal training data are quite similar to the situation where normal training data were used. The relative frequencies of normality are high only for uncorrelated variables in high dimensions. For other correlation structures we observe low relative frequencies of normality.

The general conclusion based on this empirical study is that the projections for each class are not always normally distributed. As seen in Table 4.1, there are many configurations in which the assumption of normality of the projections was violated. We therefore conclude that the normality assumption is not valid in general.

TABLE 4.1: The relative frequencies of normality obtained by using the three kernels.

High frequencies indicate that the projections for each class are approximately normally distributed.

			Linear			Gaussian			Inverse multi-quadric		
	sample size	ρ	*p*=5	*p*=50	*p*=100	*p*=5	*p*=50	*p*=100	*p*=5	*p*=50	*p*=100
Normal data	S	0	0.950	0.945	0.935	0.715	0.905	0.930	0.605	0.965	0.965
	S	0.5	0.950	0.960	0.925	0.265	0.030	0.010	0.220	0.010	0.000
	S	0.9	0.960	0.925	0.960	0.030	0.025	0.010	0.035	0.005	0.010
	L	0	0.950	0.950	0.945	0.140	0.925	0.935	0.005	0.930	0.930
	L	0.5	0.980	0.965	0.945	0.000	0.000	0.000	0.000	0.000	0.000
	L	0.9	0.940	0.940	0.955	0.000	0.000	0.000	0.000	0.000	0.000
Lognormal data	S	0	0.515	0.800	0.680	0.260	0.465	0.630	0.415	0.705	0.820
	S	0.5	0.255	0.465	0.250	0.020	0.000	0.000	0.215	0.260	0.205
	S	0.9	0.205	0.270	0.410	0.055	0.015	0.000	0.265	0.145	0.180
	L	0	0.000	0.960	0.850	0.010	0.185	0.350	0.005	0.440	0.595
	L	0.5	0.000	0.265	0.130	0.000	0.000	0.000	0.000	0.000	0.000
	L	0.9	0.000	0.190	0.000	0.000	0.000	0.000	0.000	0.000	0.000

4.3 Estimating posterior probabilities in KFDA

We now return to the problem of estimating posterior probabilities in KFDA. As mentioned in Section 4.1, we will describe two approaches. We will show in Section 4.4 how these probabilities can be used for classification.

4.3.1 Assuming a normal distribution for the projections

For this approach we assume that the distribution of the projections defined in (4.2) is known for each of the two classes. In such a case it is possible to obtain posterior probabilities for KFDA by using these projections and applying Bayes' rule. Although the projections are not always normally distributed, we make the assumption here that the projections for class *j* follow a normal distribution with density

$$\mathrm{p}(\mathrm{q} \mid \mathrm{y} = j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} exp\left[-\frac{1}{2}\left(\frac{\mathrm{q} - \mu_j}{\sigma_j}\right)^2\right], \quad j \in \{\pm 1\}, \tag{4.9}$$

where μ_j and σ_j^2 are the population mean and variance of the projections for the *j*-th class. The quantities μ_j and σ_j^2 are usually unknown but can be estimated by calculating

$$\hat{\mu}_j = \frac{1}{n_j}\sum_{k \in J_j} \mathrm{q}_k \text{ and } \hat{\sigma}_j^2 = \frac{1}{n_j - 1}\sum_{k \in J_j}\left(\mathrm{q}_k - \hat{\mu}_j\right)^2 \tag{4.10}$$

respectively from the training data. Then by applying Bayes' rule, an estimated posterior probability that the *k*-th observation belongs to Π_j can be calculated as

$$\hat{\mathrm{p}}(\mathrm{y} = j \mid \mathrm{q}_k) = \frac{\pi_j \hat{\mathrm{p}}(\mathrm{q}_k \mid \mathrm{y} = j)}{\pi_1 \hat{\mathrm{p}}(\mathrm{q}_k \mid \mathrm{y} = +1) + \pi_2 \hat{\mathrm{p}}(\mathrm{q}_k \mid \mathrm{y} = -1)}, \tag{4.11}$$

where π_j, the prior probability of the *j*-th class, is often estimated from the training data as $n_j/(n_1+n_2)$. The estimated density of the projections of each class is denoted by $\hat{\mathrm{p}}(\mathrm{q}_k \mid \mathrm{y}=j)$ and is obtained by using the estimates (4.10) in (4.9).

An illustrative example of the posterior probabilities (4.11) is given in Figure 4.4. In this figure $\hat{\mathrm{p}}(\mathrm{y}=+1 \mid \mathrm{q}_k)$ is plotted against the projections $(\mathrm{q}_k, k=1,\ldots,n)$. The training data set of 200 observations and 5 variables were generated from a normal distribution using the same parameters as those that were used to produce Figure 4.2. A Gaussian kernel $(\gamma = 1/p)$ and $\lambda = 0.1$ was used to perform KFDA. From this figure we observe a clear distinction between the classes with just a few overlapping cases. Ideally we would like $\hat{\mathrm{p}}(\mathrm{y}=+1 \mid \mathrm{q}_k)$ to be high for class 1 but low for class 2, which is indeed the situation for most of the cases depicted in the figure. Similarly we would like $\hat{\mathrm{p}}(\mathrm{y}=-1 \mid \mathrm{q}_k)$ to be high for class 2 and low for class 1. The dotted line represents a posterior probability of 0.5, which is the threshold between the two classes. In Section 4.4.1 we will use the posterior probability estimates discussed in this section to derive a classifier.

4.3.2 Logistic regression on the discriminant scores

The second approach to calculating the posterior probability of class membership of a case entails performing logistic regression on the discriminant scores obtained in KFDA. Platt (2000) proposed estimating posterior probabilities for SVMs by fitting a logistic regression model on the SVM output (2.47). We will apply the same idea in KFDA.

For the two-class case, logistic regression models the log-odds via a linear function in the input variable, *x*. This can be expressed as the linear model

$$log\left(\frac{\mathrm{P}(\mathrm{Y}=+1 \mid \mathrm{X}=x)}{\mathrm{P}(\mathrm{Y}=-1 \mid \mathrm{X}=x)}\right) = \beta_0 + \beta_1 x, \tag{4.12}$$

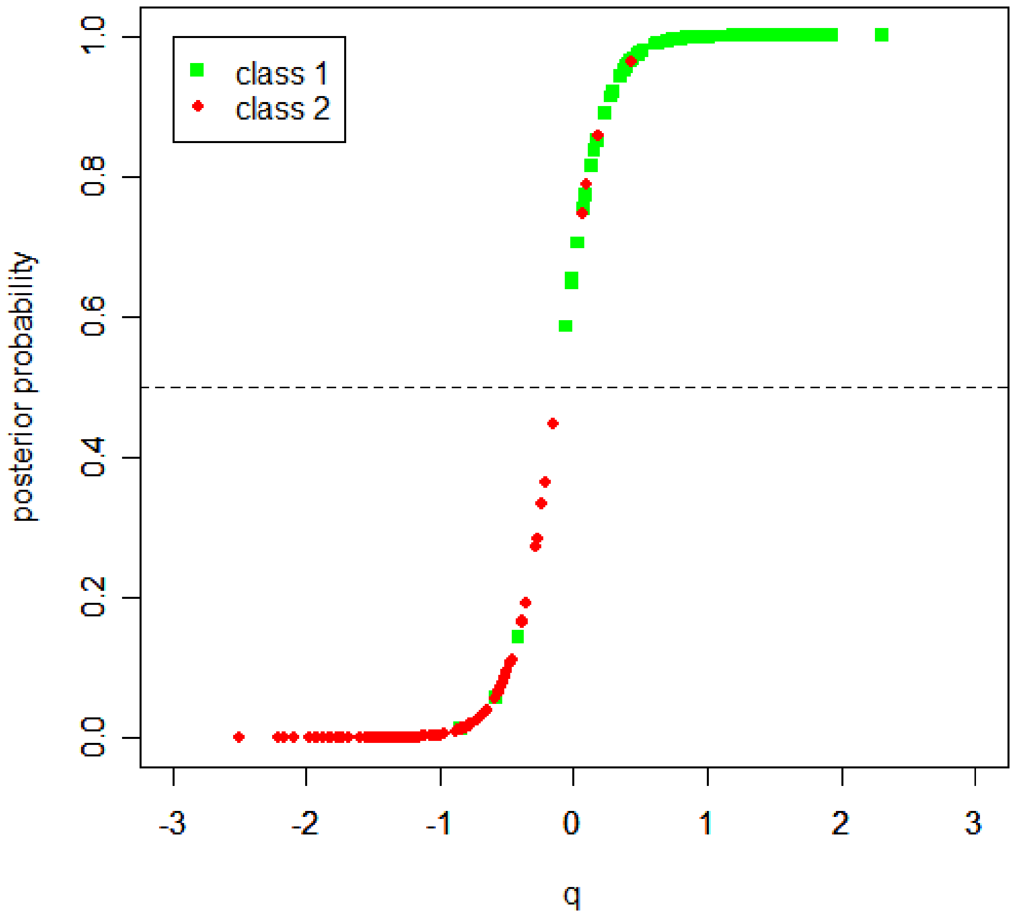


FIGURE 4.4: *An illustrative example of the posterior probabilities* $\hat{p}(y=+1\,|\,q_k)$ *which is obtained by applying Bayes' rule to the projections. The probabilities are high for class 1 and low for class 2. The dotted line represents the threshold of 0.5.*

where $\mathrm{P}(\mathrm{Y}=+1\,|\,\mathrm{X}=x)$ is the probability that observation x belongs to Π_1 and $\mathrm{P}(\mathrm{Y}=-1\,|\,\mathrm{X}=x)$ is the probability that the observation belongs to Π_2. The unknown parameters β_0 and β_1 in (4.12) are usually estimated by means of maximum likelihood based on a conditional distribution $\mathrm{P}(\mathrm{Y}\,|\,\mathrm{X})$. For the setup above, the binomial distribution is appropriate (*cf.* Hastie *et al.* (2001, *pp*.98-99) for a detailed explanation how these estimates are obtained for the binomial distribution). We will denote the estimates of the above parameters by $\hat{\beta}_0$ and $\hat{\beta}_1$ respectively. Using the fact that

$$\mathrm{P}(\mathrm{Y}=-1\,|\,\mathrm{X}=x)=1-\mathrm{P}(\mathrm{Y}=+1\,|\,\mathrm{X}=x),$$

we obtain the posterior probabilities that case x belongs to either of the two classes as

$$\mathrm{P}(\mathrm{Y}=+1\,|\,\mathrm{X}=x)=\frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \tag{4.13}$$

and

$$\mathrm{P}(\mathrm{Y}=-1\,|\,\mathrm{X}=x)=\frac{1}{1+e^{\beta_0+\beta_1 x}}. \tag{4.14}$$

To obtain (4.13) and (4.14) within the KFDA framework, we define a new training data set (t,δ), where $t=(\mathrm{y}+1)/2\in\{0,1\}$ is the binary response variable and δ is the input variable, which is obtained as

$$\delta_i=\sum_{l=1}^{n}\tilde{\alpha}_l k(\mathbf{x}_l,\mathbf{x}_i)+\tilde{b}, \quad i=1,\ \ldots,n. \tag{4.15}$$

We will refer to (4.15) as the discriminant scores of the training data. The $\tilde{\alpha}_l$-values and threshold $\tilde{b}$ are obtained using (2.28) and (2.31) respectively. Performing a logistic regression on (t,δ) gives the following estimated posterior probability that the k-th observation belongs to Π_1,

$$\hat{\mathrm{p}}(\mathrm{y}=+1\,|\,\delta_k)=\frac{e^{\hat{\beta}_0+\hat{\beta}_1\delta_k}}{1+e^{\hat{\beta}_0+\hat{\beta}_1\delta_k}}, \tag{4.16}$$

with

$$\hat{\mathrm{p}}(\mathrm{y}=-1\,|\,\delta_k)=\frac{1}{1+e^{\hat{\beta}_0+\hat{\beta}_1\delta_k}} \tag{4.17}$$

the probability that the observation belongs to Π_2.

An illustrative example of the posterior probabilities (4.16) is given in Figure 4.5. In this graph we plotted $\hat{\mathrm{p}}(\mathrm{y}=+1\,|\,\delta_k)$ versus the discriminant scores, $(\delta_k, k=1,\ldots,n)$, using the same training data that was used in Figure 4.4. Similar to Figure 4.4, we observe a good separation between the classes with few cases overlapping. Again, we observe that the posterior probabilities are high for class 1 and low for class 2, and that 0.5 is a threshold for separating the classes. The advantage of this approach compared to the one in Section 4.3.1, is that we do not make any distributional assumptions in the analysis. The posterior probability estimates discussed in this section will also be used in Section 4.4.1 to derive a classifier.

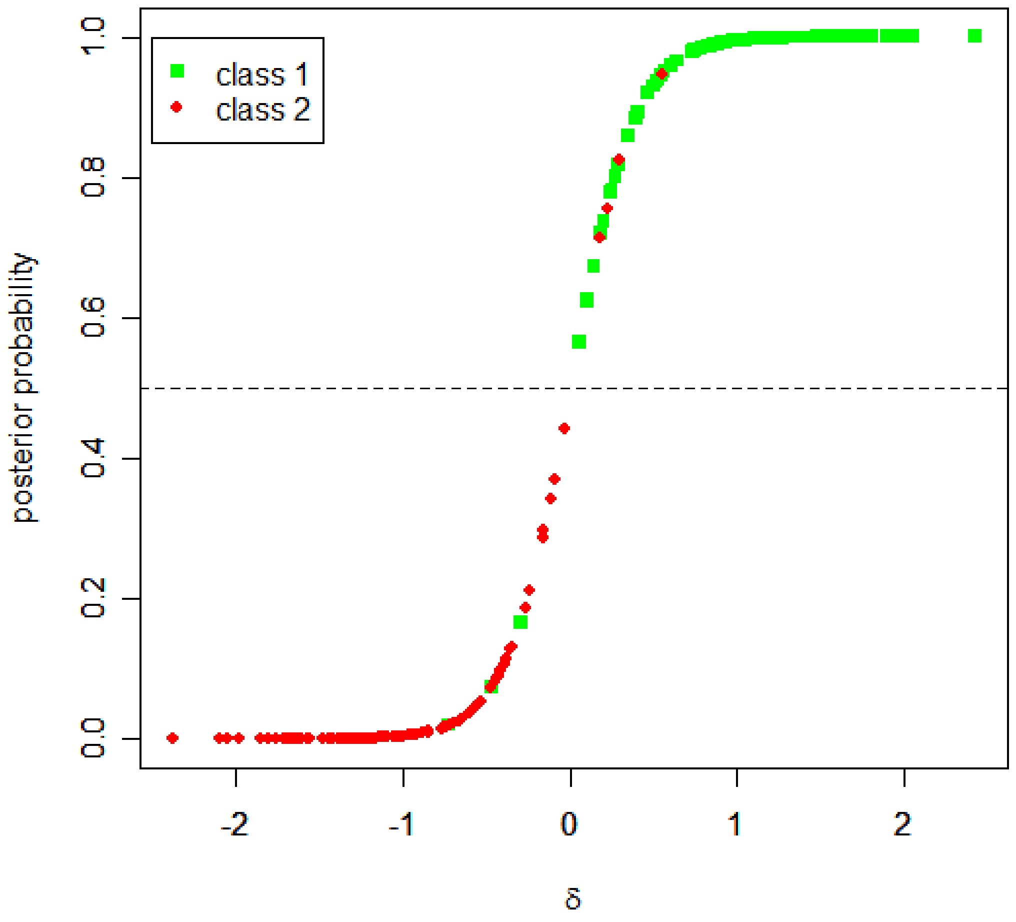


FIGURE 4.5: *An illustrative example of the posterior probabilities* $\hat{p}(y=+1\,|\,\delta_k)$, *obtained from a logistic regression analysis on the discriminant scores. The probabilities are high for class 1 and low for class 2. The dotted line represents the threshold of 0.5.*

4.4 Comparison of three classifiers

The purpose of this section is to investigate the generalization performance of three KFD classifiers in a simulation study.

4.4.1 Summary of the three classifiers

The following is a brief summary of the three classifiers that will be compared. More detail about the classifiers can be found in Chapter 2 as well as in Sections 4.3.1 and 4.3.2 respectively.

Classifier 1:

The first classifier employs the sign rule, which was defined in (2.33). We express the classifier here as

$$\varphi_1(\mathbf{x}) = \begin{cases} +1 & \text{if } sign\{f(\mathbf{x})\} > 0 \\ -1 & \text{otherwise,} \end{cases}$$

where $f(\mathbf{x}) = \sum_{l=1}^{n} \tilde{\alpha}_l k(\mathbf{x}_l, \mathbf{x}) + \tilde{b}$. The vector $\mathbf{x}$ represents the new case that has to be classified.

Classifier 2:

The second classifier is based on the posterior probabilities obtained in (4.11). We define this classifier as

$$\varphi_2(\mathbf{x}) = \begin{cases} +1 & \text{if } \hat{\mathrm{p}}(\mathrm{y} = +1 \mid \mathrm{q}) \geq 0.5 \\ -1 & \text{otherwise.} \end{cases}$$

To use this classifier the new case $\mathbf{x}$ is projected onto the direction of discrimination by using (4.2). The estimated posterior probability, $\hat{p}(y = +1 \mid q)$ for this projection, q, is then obtained by using (4.11). As illustrated in Figure 4.4, the probability 0.5 represents a threshold between the two classes. We will assume that the projections of the training data follow a normal distribution. Thus, the estimates (4.10) were used to obtain the estimated normal densities employed in (4.11).

Classifier 3:

A third classifier can be constructed by using the posterior probabilities obtained from logistic regression (see (4.16) and (4.17)). This classifier is defined as

$$\varphi_3(\mathbf{x}) = \begin{cases} +1 & \text{if } \hat{p}(y = +1 \mid \delta) \geq 0.5 \\ -1 & \text{otherwise.} \end{cases}$$

The discriminant score (δ) for the new case $\mathbf{x}$ is obtained by using (4.15) and the 0.5 threshold is used to do the classification. The estimated posterior probability for this case, $\hat{p}(y = +1 \mid \delta_i)$, is obtained using (4.16) with $\hat{\beta}_0$ and $\hat{\beta}_1$ estimated from the training data using maximum likelihood.

The following are illustrative examples of the decision boundaries of the three classifiers. In Figure 4.6 we generated the training data from a mixture of normal distributions. The three classifiers were obtained for this data set by using a Gaussian kernel with $\gamma = 0.5$ and a regularization parameter of $\lambda = 0.1$. The data in Figure 4.7 were generated similarly. The three classifiers were then obtained by using a 2nd degree polynomial kernel and $\lambda = 0.1$. These graphs reveal that the classifiers produce decision boundaries which are quite similar.

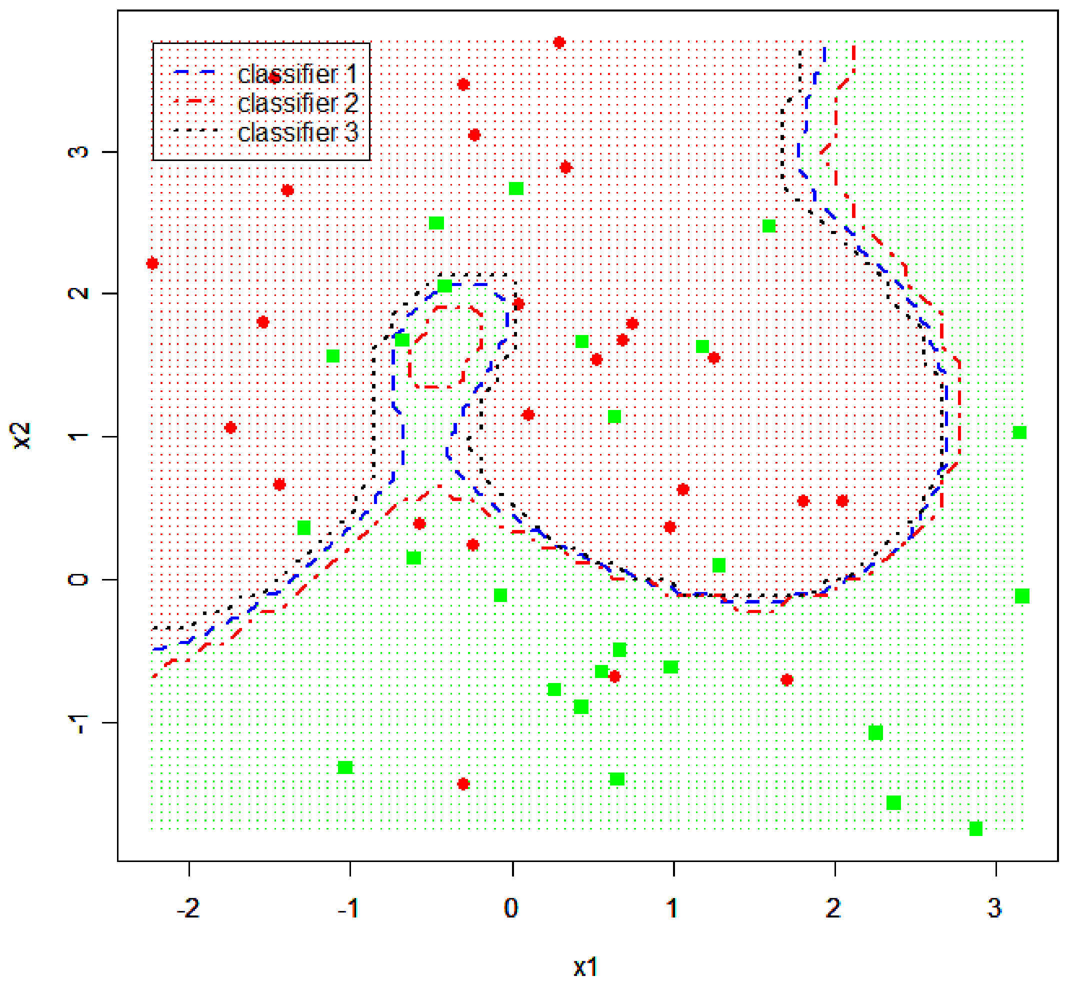


FIGURE 4.6: *An illustrative example of the non-linear decision boundaries produced by the three KFD classifiers. A Gaussian kernel with* $\gamma = 0.5$ *and* $\lambda = 0.1$ *was used. The training data were generated from a mixture of normal distributions.*

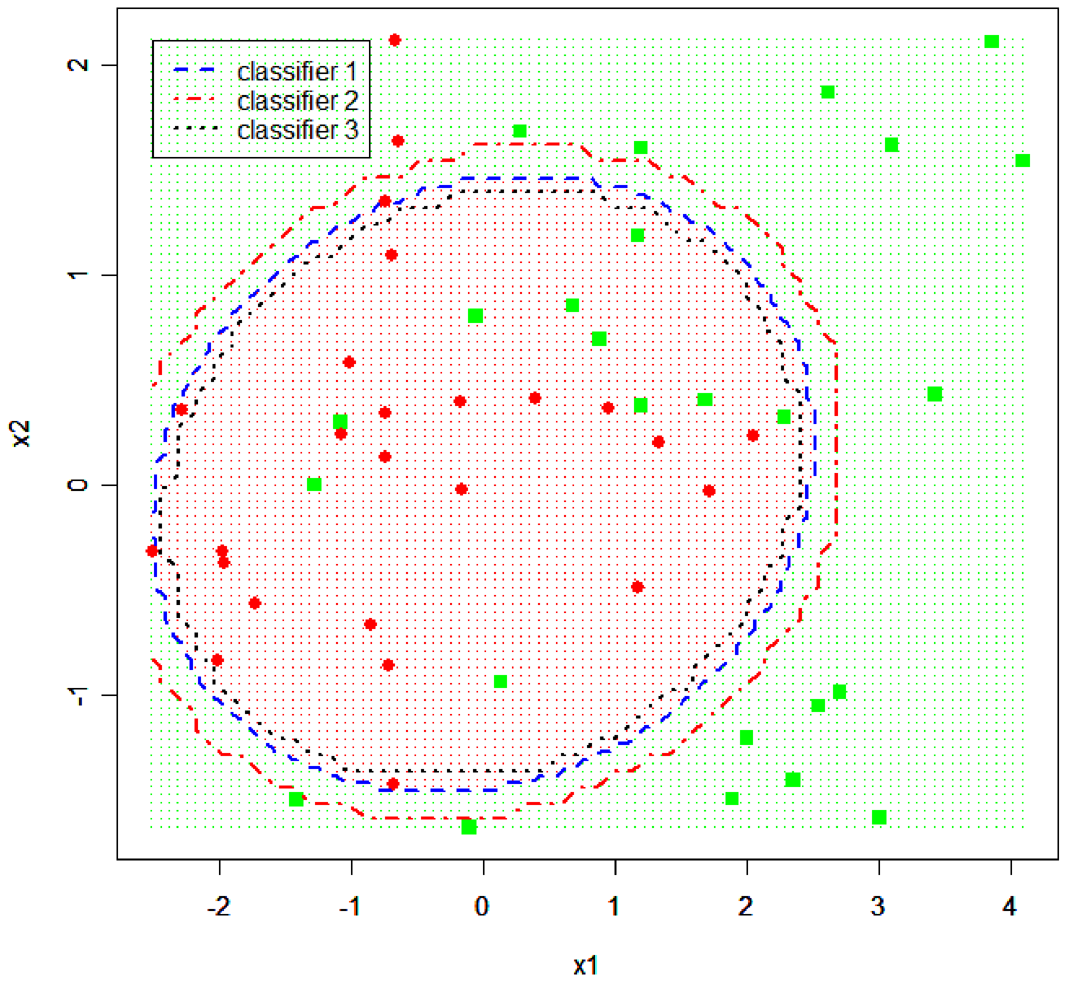


FIGURE 4.7: *Illustrations of the three non-linear decision boundaries when a 2nd degree polynomial kernel and* $\lambda = 0.1$ *are used. The training data are again simulated from a mixture of normal distributions.*

4.4.2 Design of the simulation study

To evaluate the generalization performance of the three classifiers we carried out a simulation study. The investigation was carried out only for the Gaussian kernel $(\gamma = 1/p)$ and a regularization parameter of $\lambda = 0.1$. Consider again the two-class case represented by populations Π_1 and Π_2. The simulation study was conducted with $p = 5$ input variables over $K = 100$ simulation repetitions. The following factors were also employed:

- Training data for the study were generated from the normal and lognormal distributions.
- Three sample sizes were used to obtain the training data, *viz*. small samples $(n_1 = n_2 = 25)$, mixed samples $(n_1 = 75 \text{ and } n_2 = 25)$ and large samples $(n_1 = n_2 = 100)$.
- Three levels were used for the correlations between input variables, *viz*. ρ = -0.1, 0 and 0.7.
- Regarding the differences between the two populations we investigated two cases:

  Case 1: Using different locations but equal dispersions for the populations. The covariance matrices for the two populations are equal, $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}^*$ (*cf*. (3.23)). In this study, the values of the squared Mahalanobis distance between the populations, $\Delta^2 = 2$ and 4 were used. A mean of $\mathtt{m}_1 = \mathbf{0}$ was used for Π_1, and for Π_2 the mean $\mathtt{m}_2 = c\mathbf{1}$, where c is obtained as in (4.7).

  Case 2: Using equal locations but different dispersions for the populations. In this case we used population means $\mathtt{m}_1 = \mathtt{m}_2 = \mathbf{0}$. The covariance matrix for Π_1 was $\mathbf{\Sigma}_1 = \mathbf{\Sigma}^*$ while for Π_2 the covariance matrix $\mathbf{\Sigma}_2 = \sigma^2\mathbf{\Sigma}^*$ was used. Values of $\sigma^2 = 2$ and 4 were used.

In each of the configurations, the training data were standardized before the three classifiers were obtained.

4.4.3 Discussion of the simulation results

The error rates (R_{test}) of the three classifiers were obtained by classifying an independent test data set using each of the three classifiers. The test data were generated from the same distributions as the training data and were standardized by using the means and standard deviations of the unstandardized training data. A test data set consisting of 2000 observations $(n_1 = n_2 = 1000)$ was used for the small and large sample size cases. The test data set for the mixed sample size case consisted of 4000 observations $(n_1 = 3000 \text{ and } n_2 = 1000)$. The classifiers were obtained on the same training data and were used to classify the same test data. The mean and the standard error of the error rates over the *K* simulation repetitions were calculated as

$$\overline{R}_{test} = \frac{1}{K}\sum_{k=1}^{K} R_{test}^{k} \text{ and } s_e = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(R_{test}^{k} - \overline{R}_{test}\right)^2} \tag{4.18}$$

for each configuration, and are reported in Tables 4.2 to 4.4. These tables reveal that the generalization performance of the three classifiers over the different configurations is quite similar. None of the classifiers consistently outperforms any of the others. The trends in the error rates of the three classifiers are very similar over all the tables, in the sense that:

- if we consider *Case* 1, we see that the error rates are lower for the high correlation and higher for the small correlations. As expected, for $\Delta^2 = 4$ we see lower error rates than for $\Delta^2 = 2$. For the lognormal data, we have lower error rates than for the normal data.
- when studying *Case* 2, we observe just the opposite. For the high correlation we see larger error rates than for the small correlations. For $\sigma^2 = 4$, the error rates is much smaller than the case where $\sigma^2 = 2$. When lognormal data were used, we see higher error rates compared to the normal data cases.

TABLE 4.2: Average error rates (with standard errors between parentheses) over 100 simulations for the three classifiers using small samples $(n_1 = n_2 = 25)$.

Case 1: Location differences

		$\Delta^2 = 2$			$\Delta^2 = 4$		
		$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$	$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$
Normal data	classifier 1	0.293 (0.025)	0.283 (0.027)	0.252 (0.017)	0.202 (0.025)	0.186 (0.017)	0.166 (0.012)
	classifier 2	0.293 (0.025)	0.283 (0.027)	0.253 (0.018)	0.202 (0.025)	0.187 (0.018)	0.167 (0.012)
	classifier 3	0.293 (0.025)	0.282 (0.027)	0.252 (0.017)	0.202 (0.024)	0.186 (0.018)	0.166 (0.013)
Lognormal data	classifier 1	0.260 (0.037)	0.147 (0.018)	0.096 (0.013)	0.169 (0.026)	0.094 (0.017)	0.063 (0.010)
	classifier 2	0.259 (0.036)	0.146 (0.018)	0.092 (0.012)	0.168 (0.027)	0.092 (0.016)	0.060 (0.013)
	classifier 3	0.259 (0.037)	0.147 (0.018)	0.091 (0.012)	0.167 (0.027)	0.091 (0.016)	0.059 (0.012)

Case 2: Dispersion differences

		$\sigma^2 = 2$			$\sigma^2 = 4$		
		$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$	$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$
Normal data	classifier 1	0.328 (0.019)	0.330 (0.020)	0.358 (0.030)	0.169 (0.013)	0.166 (0.012)	0.222 (0.022)
	classifier 2	0.328 (0.018)	0.330 (0.020)	0.358 (0.029)	0.174 (0.016)	0.171 (0.015)	0.220 (0.022)
	classifier 3	0.329 (0.019)	0.331 (0.021)	0.359 (0.030)	0.174 (0.015)	0.170 (0.014)	0.219 (0.021)
Lognormal data	classifier 1	0.379 (0.036)	0.359 (0.035)	0.377 (0.040)	0.233 (0.035)	0.212 (0.034)	0.242 (0.032)
	classifier 2	0.380 (0.036)	0.361 (0.035)	0.376 (0.040)	0.227 (0.034)	0.206 (0.033)	0.239 (0.032)
	classifier 3	0.376 (0.036)	0.357 (0.034)	0.377 (0.040)	0.225 (0.033)	0.205 (0.031)	0.242 (0.034)

TABLE 4.3: Average error rates (with standard errors between parentheses) over 100 simulations for the three classifiers using mixed samples $(n_1 = 75 \text{ and } n_2 = 25)$.

Case 1: Location differences

		$\Delta^2 = 2$			$\Delta^2 = 4$		
		$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$	$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$
Normal data	classifier 1	0.212 (0.012)	0.209 (0.014)	0.192 (0.012)	0.154 (0.014)	0.144 (0.013)	0.131 (0.008)
	classifier 2	0.216 (0.016)	0.211 (0.014)	0.196 (0.013)	0.152 (0.014)	0.146 (0.013)	0.135 (0.010)
	classifier 3	0.212 (0.014)	0.208 (0.014)	0.192 (0.011)	0.151 (0.013)	0.144 (0.013)	0.131 (0.007)
Lognormal data	classifier 1	0.195 (0.024)	0.102 (0.013)	0.089 (0.014)	0.116 (0.025)	0.058 (0.010)	0.056 (0.009)
	classifier 2	0.187 (0.035)	0.105 (0.016)	0.091 (0.013)	0.112 (0.027)	0.063 (0.011)	0.058 (0.009)
	classifier 3	0.184 (0.033)	0.103 (0.014)	0.085 (0.016)	0.110 (0.026)	0.059 (0.010)	0.050 (0.007)

Case 2: Dispersion differences

		$\sigma^2 = 2$			$\sigma^2 = 4$		
		$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$	$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$
Normal data	classifier 1	0.218 (0.011)	0.217 (0.011)	0.226 (0.010)	0.116 (0.009)	0.114 (0.008)	0.135 (0.010)
	classifier 2	0.229 (0.016)	0.230 (0.014)	0.233 (0.014)	0.123 (0.012)	0.123 (0.011)	0.137 (0.012)
	classifier 3	0.224 (0.014)	0.224 (0.012)	0.229 (0.013)	0.117 (0.010)	0.116 (0.008)	0.134 (0.011)
Lognormal data	classifier 1	0.266 (0.015)	0.262 (0.016)	0.246 (0.023)	0.200 (0.016)	0.179 (0.013)	0.174 (0.018)
	classifier 2	0.276 (0.020)	0.269 (0.022)	0.240 (0.026)	0.187 (0.017)	0.171 (0.013)	0.167 (0.015)
	classifier 3	0.273 (0.018)	0.267 (0.020)	0.241 (0.027)	0.193 (0.017)	0.176 (0.014)	0.170 (0.017)

TABLE 4.4: Average error rates (with standard errors between parentheses) over 100 simulations for the three classifiers using large samples $(n_1 = n_2 = 100)$.

Case 1: Location differences

		$\Delta^2 = 2$			$\Delta^2 = 4$		
		$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$	$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$
Normal data	classifier 1	0.260 (0.013)	0.250 (0.010)	0.246 (0.011)	0.171 (0.010)	0.168 (0.010)	0.162 (0.008)
	classifier 2	0.260 (0.014)	0.251 (0.010)	0.246 (0.011)	0.171 (0.010)	0.168 (0.010)	0.161 (0.008)
	classifier 3	0.260 (0.014)	0.250 (0.010)	0.245 (0.011)	0.171 (0.010)	0.168 (0.010)	0.161 (0.008)
Lognormal data	classifier 1	0.195 (0.014)	0.120 (0.010)	0.089 (0.008)	0.120 (0.014)	0.067 (0.007)	0.056 (0.007)
	classifier 2	0.195 (0.014)	0.119 (0.010)	0.084 (0.008)	0.119 (0.013)	0.066 (0.007)	0.053 (0.008)
	classifier 3	0.193 (0.014)	0.119 (0.010)	0.083 (0.008)	0.117 (0.013)	0.066 (0.008)	0.052 (0.008)

Case 2: Dispersion differences

		$\sigma^2 = 2$			$\sigma^2 = 4$		
		$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$	$\rho = -0.1$	$\rho = 0$	$\rho = 0.7$
Normal data	classifier 1	0.315 (0.011)	0.313 (0.013)	0.324 (0.014)	0.156 (0.008)	0.153 (0.008)	0.188 (0.013)
	classifier 2	0.313 (0.012)	0.311 (0.012)	0.325 (0.013)	0.157 (0.009)	0.155 (0.009)	0.184 (0.013)
	classifier 3	0.316 (0.011)	0.314 (0.013)	0.324 (0.015)	0.158 (0.009)	0.156 (0.010)	0.182 (0.013)
Lognormal data	classifier 1	0.326 (0.024)	0.333 (0.020)	0.369 (0.020)	0.172 (0.020)	0.182 (0.013)	0.240 (0.021)
	classifier 2	0.337 (0.026)	0.338 (0.021)	0.360 (0.016)	0.171 (0.022)	0.181 (0.014)	0.238 (0.018)
	classifier 3	0.320 (0.025)	0.332 (0.021)	0.370 (0.019)	0.161 (0.020)	0.177 (0.014)	0.247 (0.023)

In Section 4.2.3 we concluded that, when using the Gaussian kernel, the assumption of normality of the projections was not valid at certain configurations. However, if we study the generalization performance of classifier 2 at the same configurations, it seems that the validity of the assumption is not crucial when performing classification. In other words, in situations where the assumption of normality of the projections is violated, the generalization performance of classifier 2 is not inferior to that of the other classifiers where such an assumption is not made.

4.5 Concluding remarks

We have shown in the first simulation study that the projections obtained from KFDA are not always normally distributed. For certain configurations in Table 4.1 we observe very high relative frequencies of normality and for others very low relative frequencies. Therefore we conclude that the statement by Schölkopf and Smola (2002) is not always true.

In Section 4.3 we have discussed two approaches that can be used to obtain posterior probabilities in KFDA. The first approach is based on the assumption that the projections follow a normal distribution while the second approach entails applying logistic regression to the discriminant scores. Both approaches produce posterior probabilities from which a classification rule was also obtained. The posterior probabilities allow us to quantify how certain we are that a case belongs to a given class. Other applications of these probabilities will receive attention in Chapter 5.

Even though we showed that the projections are not always normally distributed, we still used this assumption to obtain the posterior probabilities for the projections. In Section 4.4 we did a second simulation study to compare three classifiers and we saw that their classification results did not differ very much. Thus the violation of the normality assumption did not influence the generalization performance of classifier 2, compared to the other two classifiers. The results and conclusions drawn from this chapter will be utilized in Chapter 5 in the development of a criterion for identifying influential cases.

CHAPTER 5

IDENTIFYING CASES HAVING A DETRIMENTAL EFFECT ON THE KFD GENERALIZATION PERFORMANCE

"*Far better an approximate answer to the right question, which is often vague, than the exact answer to the wrong question, which can always be made precise*."

- John Tukey

5.1 Introduction

In Chapter 3 it was shown that the generalization performance of the KFD classifier often deteriorates when an atypical case (*viz*. a mislabelled case and moderate outlier, *cf*. Section 3.4.4) is inserted into the training data. It was also shown that quantities such as the Rayleigh coefficient, the alignment, the average margin and various other measures described in Section 3.3, are also influenced by the insertion of an atypical case into the data. However, in practice we will rarely have prior knowledge about which cases are atypical. Thus we need a procedure(s) to identify such cases. The main aim of this chapter is to develop criteria which identify cases that have a detrimental effect on the generalization performance of the KFD classifier. We refer to such cases as (classification) influential. Several criteria to identify atypical cases are proposed in this chapter. The approach that we use is called the leave-one-out approach. In this approach an observation is omitted and if its omission optimizes the criterion, it is considered influential according to the given criterion. The error rate is then estimated without that observation to assess the KFD generalization performance.

The rest of the chapter is set out as follows. Section 5.2 contains a brief overview of some of the literature on identification of influential cases in regression and discriminant analysis. The methods proposed in this literature use the leave-one-out approach to identify influential cases. In Section 5.3 we propose several criteria that can be used to identify influential cases when performing KFDA. To evaluate the merit of the proposed criteria, we conducted an extensive simulation study investigating the generalization performance of the KFD classifier upon omission of the case identified by each criterion. Details of the simulation design and a discussion of the simulation results are given in Section 5.4. For each criterion, the observation whose omission optimizes its value is considered to be the most influential case. This case will then be omitted and the resulting error rate will be estimated. As will be seen in the simulation results, the KFD error rate often decreases significantly when the case identified by each of the criteria is omitted from the data. In Section 5.5, the adequacy of the criteria is evaluated on two real-world data sets. Results of the cross-validation error rate and the optimal values of the criteria are presented and discussed in this section. Section 5.6 concludes this chapter by summarizing the main results, the advantages and the disadvantages of the proposed criteria. Some avenues for further research are also discussed.

5.2 A summary of the literature on leave-one-out influence diagnostics

Using a leave-one-out approach to study the influence of observations on model aspects is quite popular in statistical analyses. The leave-one-out approach refers to the omission of a single observation in order to determine the influence, upon its omission, on certain model aspects. The aim with such an analysis is to determine which observation has the largest influence. In this section we briefly review some of the literature on leave-one-out influence diagnostics in linear regression and discriminant analysis.

Several influence diagnostics using the leave-one-out approach have been proposed in linear regression analysis. The Cook statistic, proposed by Cook (1977), is probably the most well-known diagnostic of single influential observations in a regression context. He proposed that the influence of single observations should be judged by using a distance measure calculated without this observation. A generalization of Cook’s statistic was proposed by Pregibon (1981) to detect influential observations in logistic regression. Other references include Léger and Altman (1993), and Steel and Uys (2006), who studied the influence of individual observations on variable selection in regression analysis using the leave-one-out approach.

One of the first studies on the influence of single cases in discriminant analysis was done by Campbell (1978), who proposed using the influence function as an aid in detecting influential cases in linear discriminant analysis. Johnson (1987) investigated the influence of single observations in linear discriminant analysis within a Bayesian framework, while Critchley and Vitiello (1991) studied the influence of single observations on the misclassification probability estimates in linear discriminant analysis. Fung (1995a, 1995b, 1995c, 1996) investigated the influence of single cases on the estimated discriminant scores as well as on the posterior probabilities of group membership. He proposed several influence diagnostics for linear as well as quadratic discriminant analysis. All of his diagnostics make use of the omission of individual observations to detect the influential cases. Moreno-Roldán *et al.* (2007) proposed two influence diagnostics for usage in linear discriminant analysis. Their criteria, which are based on the L_2-norm, also make use of the leave-one-out approach. Steel and Louw (2001, 2002) investigated the influence of individual observations on variable selection in linear discriminant analysis and proposed an influence diagnostic in a variable selection context.

Many single case diagnostics can be extended in order to assess the effect of the omission of groups of observations. However, it is not clear which groups of observations should be evaluated. As a result one has to consider all pairs, triples, *etc*., making this approach computationally prohibitive when dealing with large data sets. The literature on multiple case diagnostics is small, since this is a more difficult problem to address. We also restrict attention to the identification of single influential cases.

5.3 Proposed criteria to identify influential cases in KFDA

In this section we propose nine criteria which can be used in an attempt to identify cases that are influential on the KFD classifier. Each criterion is calculated by using the leave-one-out principle, *i.e.* for $i = 1, 2, ..., n$, the *i*-th observation is omitted from the training data and each criterion is calculated without this case. The case whose omission results in optimization of the criterion is considered to be the influential case. A superscript, *e.g.* $\theta^{(i)}$, will be used to denote the value of each criterion when the *i*-th case is omitted. Seven of the proposed criteria are based on measures defined in Section 3.3. These measures are recalled in Section 5.3.1 and two additional criteria are introduced in Sections 5.3.2 and 5.3.3.

5.3.1 Criteria in Section 3.3

The measures defined in Section 3.7 are affected by the presence of an atypical case in the training data. Since these measures are sensitive to atypical cases, we propose using them as criteria to identify such cases. The training error (R_{emp}) was given in (3.2) and the average distance from the decision boundary for the misclassified training observations was defined in (3.6) and is denoted by $\overline{d}$. Since small values of these measures are desirable, we will consider the observation whose omission leads to the minimization of $\overline{d}^{(i)}$ or $R_{emp}^{(i)}$ as an influential case. The other measures from Section 3.3 are the average margin, the Rayleigh coefficient, the ratio of the between-class to within-class variation, the alignment and the length of **v**. These measures were respectively denoted by $\overline{m}$, r, v, a and $\|\mathbf{v}\|$, and their definitions were given in (3.8), (3.9), (3.11), (3.13) and (3.14) respectively. We propose using these quantities as criteria to identify influential cases. For each of these criteria, a large value is desirable, and therefore the observation whose omission leads to the maximization of $\overline{m}^{(i)}$, $r^{(i)}$, $v^{(i)}$, $a^{(i)}$ or $\|\mathbf{v}\|^{(i)}$ respectively, is considered the influential case according to that specific criterion. A summary of the criteria is given in Table 5.1.

TABLE 5.1: A summary of proposed criteria (defined in Section 3.3)

Criterion	Influential case
$R_{emp} = \frac{1}{n}\sum_{i=1}^{n} I[\varphi(\mathbf{x}_i) \neq \mathrm{y}_i]$	$\min_{1\le i\le n}\left\{R_{emp}^{(i)}\right\}$
$\bar{d} = \frac{1}{n_L}\sum_{i\in L} d_i$	$\min_{1\le i\le n}\left\{\bar{d}^{(i)}\right\}$
$\bar{m} = \frac{1}{n}\sum_{i=1}^{n} m_i$	$\max_{1\le i\le n}\left\{\bar{m}^{(i)}\right\}$
$r = \frac{\tilde{\mathrm{a}}'\mathbf{M}\tilde{\mathrm{a}}}{\tilde{\mathrm{a}}'\mathbf{N}_\lambda\tilde{\mathrm{a}}}$	$\max_{1\le i\le n}\left\{r^{(i)}\right\}$
$v = \frac{d\left(\bar{\mathbf{x}}_1^{\Phi}, \bar{\mathbf{x}}_2^{\Phi}\right)}{\mathrm{s}_1^2 + \mathrm{s}_2^2}$	$\max_{1\le i\le n}\left\{v^{(i)}\right\}$
$a = \frac{\mathbf{y}'\mathbf{G}\mathbf{y}}{n\Vert\mathbf{G}\Vert_{\mathrm{F}}}$	$\max_{1\le i\le n}\left\{a^{(i)}\right\}$
$\Vert\mathbf{v}\Vert = \sqrt{\tilde{\boldsymbol{\alpha}}'\mathbf{G}\tilde{\boldsymbol{\alpha}}}$	$\max_{1\le i\le n}\left\{\Vert\mathbf{v}\Vert^{(i)}\right\}$

5.3.2 Average squared difference between the estimated discriminant scores

Fung (1995b) proposed using the average squared difference between the estimated discriminant scores obtained from the full training data set and the data set without the *i*-th observation to quantify the influence of the omitted observation in linear discriminant analysis. His proposed measure can be expressed as

$$F^{(i)} = \frac{1}{n}\sum_{l=1}^{n}\left[\left\{\langle\tilde{\mathbf{v}},\mathbf{x}_l\rangle + \tilde{a}\right\} - \left\{\langle\tilde{\mathbf{v}}^{(i)},\mathbf{x}_l\rangle + \tilde{a}^{(i)}\right\}\right]^2,$$

with $\tilde{\mathbf{v}}$ and $\tilde{a}$ obtained as in (2.8) and (2.12) from the full data set. By omitting the *i*-th case from the data, $\tilde{\mathbf{v}}^{(i)}$ and $\tilde{a}^{(i)}$ are obtained similarly. Fung (1995b) also shows that $F^{(i)}$ is analogous to the Cook statistic in regression diagnostics.

Since KFDA is an extension of FLDA, Fung's measure can also be extended to the KFDA framework. Thus, we propose the following as an eighth criterion for identification of influential cases in KFDA. For KFDA we can write the average squared difference between the discriminant scores obtained from the full training data set and the data set without the *i*-th observation as

$$\bar{f}^{(i)} = \frac{1}{n}\sum_{l=1}^{n}\left[\left\{\langle \mathbf{v}, \Phi(\mathbf{x}_l)\rangle + \tilde{b}\right\} - \left\{\langle \mathbf{v}^{(i)}, \Phi(\mathbf{x}_l)\rangle + \tilde{b}^{(i)}\right\}\right]^2$$

$$= \frac{1}{n}\sum_{l=1}^{n}\left[\left\{\sum_{k=1}^{n}\tilde{\alpha}_k\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l)\rangle + \tilde{b}\right\} - \left\{\sum_{\substack{k=1\\k\neq i}}^{n}\tilde{\alpha}_k^{(i)}\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l)\rangle + \tilde{b}^{(i)}\right\}\right]^2,$$

which we calculate as

$$\bar{f}^{(i)} = \frac{1}{n}\sum_{l=1}^{n}\left[\left\{\sum_{k=1}^{n}\tilde{\alpha}_k k(\mathbf{x}_k, \mathbf{x}_l) + \tilde{b}\right\} - \left\{\sum_{\substack{k=1\\k\neq i}}^{n}\tilde{\alpha}_k^{(i)} k(\mathbf{x}_k, \mathbf{x}_l) + \tilde{b}^{(i)}\right\}\right]^2. \tag{5.1}$$

Here $\tilde{b}^{(i)}$ denotes the intercept and $\tilde{\alpha}_k^{(i)}$ an element of the α-vector, $\tilde{a}^{(i)}$, of the KFDA solution obtained when the *i*-th case is omitted. The observation whose omission leads to the maximization of $\bar{f}^{(i)}$ is considered the influential case.

5.3.3 Deviation of the posterior probabilities from their ideal values

Our ninth proposal makes use of the estimated posterior probabilities of the projections in KFDA (*cf.* Section 4.3.1). The projections for KFDA were defined in (4.2). Assume that the projections for class $j = 1, 2$ can be approximated by a normal distribution with estimates of the population mean and variance given by (4.10). Then the estimated normal density for an observation from class *j* is given by

$$\hat{f}_j(\mathrm{q}_k) = \frac{1}{\sqrt{2\pi\hat{\sigma}_j^2}} exp\left[-\frac{1}{2}\left(\frac{\mathrm{q}_k - \hat{\mu}_j}{\hat{\sigma}_j}\right)^2\right],\ k \in J_j,\ j = 1, 2.$$

Let the prior probability that an observation belongs to class *j* be π_j. The estimated posterior probability for a class *j* case to be (correctly) classified into class *j* can be obtained from Bayes' rule as $\hat{\mathrm{p}}_j(\mathrm{q}_k) = \pi_j \hat{f}_j(\mathrm{q}_k) \big/ \left(\pi_1 \hat{f}_1(\mathrm{q}_k) + \pi_2 \hat{f}_2(\mathrm{q}_k)\right),\ j = 1,2$. Ideally we would like the value $\hat{\mathrm{p}}_j(\mathrm{q}_k)$ to be close to 1. The quantity

$$\overline{h} = \frac{1}{n}\sum_{j=1}^{2}\sum_{k\in J_j}\left(1-\hat{\mathrm{p}}_j(\mathrm{q}_k)\right)^2 \tag{5.2}$$

is a measure of the extent to which the posterior probabilities deviate from their ideal values. We propose $\overline{h}$ as an influence measure and the observation whose omission leads to a minimization of $\overline{h}^{(i)}$ is declared influential. Note that we can also use other estimates for $\hat{\mathrm{p}}_j(\mathrm{q}_k)$ as well, for example the posterior probabilities obtained from a logistic regression analysis as discussed in Section 4.3.2.

5.4 Identifying influential cases in KFDA: a simulation study

A detailed Monte Carlo simulation study was carried out to investigate the merit of our criteria proposed in Section 5.3. In this study the estimated error rate is used to compare the generalization performance of the KFD classifier based on the full data set and the KFD classifier based on the data set from which the single case identified by each criterion, was omitted.

The results of the study are reported as the percentage decrease in the average test error rates, which is defined as follows: let the test error obtained from a test data set be given by

$$R_{test} = \frac{1}{n_{\mathrm{T}}} \sum_{l=1}^{n_{\mathrm{T}}} I\left[sign\{f(\mathbf{x}_l)\} \neq \mathrm{y}_l\right], \tag{5.3}$$

where $sign\{.\}$ is the classifier obtained from the full training data set and n_{T} the size of the test data set having the same distribution as the training data. The average test error over all K simulation repetitions is therefore

$$\overline{R}_{test}^{[1]} = \frac{1}{K} \sum_{k=1}^{K} R_{test}^{k} \,. \tag{5.4}$$

We used $K = 1000$ throughout. If a criterion is optimized when the *j*-th case is omitted, such a case is the influential case identified by the criterion. After removing this influential case from the training data, KFDA is performed on the reduced data set and the test error, $R_{test}^{(j)}$, is calculated similar to (5.3) above. Let the average test error over all the simulation repetitions, when the influential case is omitted, be denoted by $\overline{R}_{test}^{[2]}$ (calculated similar to (5.4)). Then the value

$$P_e = 100 \times \left(\overline{R}_{test}^{[1]} - \overline{R}_{test}^{[2]}\right) / \overline{R}_{test}^{[1]}$$

represents the percentage change in the average test error due to the omission of the influential case. If positive, P_e represents a decrease in the average error rate, *i.e.* a better generalization performance of the KFD classifier.

5.4.1 Design of the simulation study

We once again consider the two-class case. The different factors and their levels that were used in the design of the simulation study, are as follows:

- We used the normal and lognormal distributions to generate the training data.

- With respect to sample sizes, three levels were used, *viz.* small $(n_1 = n_2 = 25)$, large $(n_1 = n_2 = 100)$ and mixed $(n_1 = 75$ and $n_2 = 25)$.
- The number of input variables used was $p = 5$.
- The correlations between input variables have three levels. Using a common correlation (ρ) for all pairs of variables, we investigated the cases where $\rho = -0.1$, 0 and 0.7.
- For the differences between the two populations we investigated two cases, *viz.*

  Case 1: Difference in location. In this case we have identical covariance structures for both populations, $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}^*$ (as given in (3.23)), but with a difference in their respective mean vectors. We used $\boldsymbol{\mu}_2 = \mathbf{0}$ throughout the study, but set $\boldsymbol{\mu}_1 = c\mathbf{1}$ where $c = 2.5$ was used for normal data and $c = 1$ for lognormal data.

  Case 2: Difference in dispersion. Here we used equal locations, $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0}$. The covariance matrix $\mathbf{\Sigma}_2 = \mathbf{\Sigma}^*$ was used for Π_2, but we used $\mathbf{\Sigma}_1 = \sigma^2\mathbf{\Sigma}^*$ with $\sigma = 4$ for Π_1.

Note that the configurations described above are the same as those used for the simulation study in Section 3.7. In this study we therefore use the same optimal λ-values that were obtained in that study for each of the configurations. Again, the Gaussian kernel with $\gamma = 1/p$ was used throughout. A test data set of 10000 observations $(n_1 = n_2 = 5000)$ was used for small and large training data sets. The size of the test data for the mixed training data set was also 10000 observations $(n_1 = 7500$ and $n_2 = 2500)$.

Part of the simulation design is to deliberately insert an atypical case in the training data (note that the test data contain only typical cases). The reason for this is to make a comparison between the decrease in the test errors without this deliberately inserted atypical case and the decrease in the test errors without the case identified by each of the criteria as influential. Note that the atypical case and the influential case are not in all instances the same case. Both types of atypical cases defined in Table 3.1 were investigated.

To test whether the decrease in the test error is significant we also performed a hypothesis test for two independent normally distributed samples, similar to what was done in Section 3.7. The *z*-test statistic was calculated for each configuration and a *z*- test statistic greater than 1.96 or less than -1.96 indicates a significant difference between the test errors at a 5% level of significance.

5.4.2 Discussion of the simulation results

Tables 5.2 to 5.5 contain the percentage decrease in the average error rates (P_e) for the two distributions and two types of atypical cases. In each table, the first three columns contain the configurations, the next nine columns contain the P_e when the case identified by the criteria (*cf.* Section 5.3) is omitted from the training data and the last column contains the P_e when the atypical case is omitted from the training data.

In each table, we should take note of the following:

The <u>underlined</u> numbers, for each configuration, represent the cases where the P_e corresponding to a criterion is equal to or greater than the P_e in the last column. If the underlined number is equal to the last column, it means that the criterion consistently identified the deliberately inserted atypical case as influential. If it is larger, it means that a criterion identified cases which are even more detrimental to the generalization performance than the atypical case. As represented by the underlined numbers, there are quite a few instances where the criteria identified observations which are equally or more harmful to the KFD generalization performance. This is especially the case when populations differ with respect to location. For the configurations where the two populations have different dispersions this did not occur so frequently.

The **bold face** numbers represent the cases where the *z*-test statistic of the hypothesis test corresponding to the difference $\left(\overline{R}_{test}^{[1]} - \overline{R}_{test}^{[2]}\right)$ is greater than 1.96. In other words, there is a significant difference between the test error based on the training data with and without the identified case. For most of the configurations, the difference is significant and this is a clear indication that the omission of cases identified by the criteria often leads to a significantly improved generalization performance of the KFD classifier.

There are also a few negative numbers in the tables. These numbers indicate that there were configurations in which the omission of the identified case leads to a deterioration in the generalization performance of KFDA. However, these negative numbers are relatively few and mostly represent an insignificant increase in the test error (except for a couple of cases in Table 5.4).

Interpreting the results for the nine criteria:

If we ignore the last column, the entries indicate the extent to which the different criteria succeed in identifying a case which has a detrimental influence on the KFD generalization performance, in the sense that removal of such a case prior to construction of the KFD classifier, leads to a lower generalization error rate. Since the majority of the entries are positive (often significantly so; especially for the dispersion differences) we conclude that the criteria are largely successful in this regard.

From the tables we also observe that the P_e for small and mixed samples are quite high, and for the large samples very low. This can be expected since leaving out one case in small samples has a larger (more significant) effect than in large samples.

Interpreting the results in the last column:

This column represent the P_e when the deliberately inserted atypical case is omitted from the training data. It is clear from the results that this case has a detrimental effect on the KFD generalization performance (see also Chapter 3) and most of the time significantly so.

By comparing the entries for the various criteria to those in the last column, it is clear from these results that the inserted atypical case are in many instances not the case identified by the criteria. If this had been the case, the decrease in the error rates in columns corresponding to the criteria would be exactly the same as in the last column.

We also see that for some configurations, omission of the case identified by the criteria results in a larger decrease in the test error than the omission of the deliberately inserted case (the underlined numbers). This indicates that in these configurations there are often cases present in the training data which have an even more detrimental effect on the KFD generalization performance than the atypical case and that the criteria were successful in detecting those cases.

It is important to note that in practice we do not know which case in the training data is the atypical case. Thus, we will not be able to obtain the results in the last column. Hence, we need a criterion or criteria with which we can identify those cases that are harmful to the generalization performance of the classifier. From these results it is clear that most of the criteria identified the cases that are detrimental to the KFD generalization performance. Omitting those cases often lead to a significant improvement in the generalization performance of KFDA. There were also cases where omitting these cases lead to a deterioration in the KFD generalization performance. However these were insignificant.

Selecting the best criterion:

It is not straightforward to pick an overall “winner” from the criteria that will always perform the best. The simulation study shows that omission of the cases identified as influential, lead to a better generalization performance for most of the configurations considered here. This indicates that the criteria are most of the time successful in identifying cases that are detrimental to the KFD generalization performance. We suggest that a practitioner performing KFDA should calculate all the criteria and investigate the cases identified by each criterion more closely.

TABLE 5.2: Percentage decrease in the test error; normal data; mislabelled case

	Configurations		Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\bar{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	S	-0.1	**13.805**	**13.805**	**13.583**	**13.805**	**13.583**	**13.805**	**13.583**	**13.583**	**13.805**	**13.805**
	S	0	**3.679**	**3.586**	**3.632**	**3.669**	**3.632**	**3.666**	**3.555**	**3.632**	**3.534**	**3.669**
	S	0.7	0.019	-0.261	-0.25	-0.219	-0.424	0.094	-0.257	-0.201	-0.395	0.436
	M	-0.1	**16.055**	**16.055**	**16.055**	**16.055**	**16.055**	**16.055**	**16.055**	**16.055**	**16.055**	**16.055**
	M	0	**2.834**	**2.812**	**2.838**	**2.853**	**2.845**	**2.742**	**2.687**	**2.838**	**2.735**	**2.845**
	M	0.7	-0.045	-0.131	0.048	-0.036	0.178	0.159	-0.07	0.037	-0.026	0.167
	L	-0.1	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	1.043	0.92
	L	0	0.059	0.11	0.099	0.102	0.11	0.091	0.099	0.099	0.04	0.106
	L	0.7	0.002	-0.042	-0.036	-0.026	-0.069	0.006	-0.044	-0.04	-0.043	0.021
Dispersion difference	S	-0.1	**32.951**	**18.651**	**22.464**	**12.682**	**28.304**	**-48.501**	**24.643**	**18.425**	**26.562**	**34.294**
	S	0	**33.721**	**18.246**	**21.339**	**12.84**	**27.543**	**-50.048**	**24.344**	**18.182**	**25.674**	**35.2**
	S	0.7	**2.654**	**2.561**	**-3.29**	**-3.108**	**8.022**	0.015	0.982	**2.435**	**-2.537**	**10.884**
	M	-0.1	**2.841**	**1.984**	**2.076**	-0.077	**3.114**	**4.116**	0.311	**2.53**	**2.614**	**3.306**
	M	0	**2.891**	**2.303**	**2.539**	0.267	**3.392**	**4.425**	0.639	**2.965**	**2.882**	**3.475**
	M	0.7	0.924	0.589	0.733	-0.245	**1.115**	-0.807	-0.197	0.737	1.039	**2.02**
	L	-0.1	-0.194	-1	-0.787	**-1.235**	**4.642**	-0.932	-0.74	0.621	**-1.091**	**4.944**
	L	0	0.173	-0.947	-0.878	**-1.148**	**4.835**	-0.846	-0.399	0.507	-0.913	**5.13**
	L	0.7	-0.511	**-1.847**	**-1.664**	**-2.014**	**2.815**	0.576	**-2.219**	0.801	**-1.399**	**3.992**

TABLE 5.3: Percentage decrease in the test error; lognormal data; mislabelled case

	Configurations		Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\overline{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	S	-0.1	**13.198**	**13.198**	**8.905**	**7.754**	**8.915**	-0.562	**7.371**	**7.884**	**5.492**	**13.198**
	S	0	**7.003**	**6.91**	**3.457**	**2.827**	**3.515**	**-4.332**	**2.917**	2.021	**2.32**	**7.248**
	S	0.7	0.757	0.057	-0.541	-0.834	**2.339**	-1.251	-0.292	-0.909	-1.303	**1.715**
	M	-0.1	**3.033**	**3.051**	1.441	1.112	1.42	-1.548	0.758	1.435	0.185	**3.152**
	M	0	**3.142**	**3.086**	1.606	0.989	**2.181**	-1.65	0.91	1.431	0.477	**3.552**
	M	0.7	-0.266	-1.163	-0.362	-0.864	**1.531**	-0.987	-1.587	-0.344	-1.157	-0.135
	L	-0.1	**5.164**	**5.129**	**3.901**	**3.533**	**3.611**	**-3.851**	**3.223**	**3.827**	1.872	**5.295**
	L	0	1.587	1.346	1.038	0.813	1.026	**-2.711**	0.889	0.876	0.418	**2.155**
	L	0.7	-0.215	-0.518	-0.027	-0.095	**1.632**	-0.144	-0.492	-0.409	-0.348	0.335
Dispersion difference	S	-0.1	**22.26**	**22.192**	**8.565**	**6.32**	**6.132**	**-3.992**	**5.252**	**10.374**	3.525	**23.706**
	S	0	**7.544**	**7.3**	**2.017**	1.35	**4.388**	1.122	**2.677**	**3.781**	0.869	**10.893**
	S	0.7	**8.102**	**7.509**	**4.328**	**3.411**	**5.745**	1.592	**4.504**	**4.938**	1.200	**10.342**
	M	-0.1	**6.505**	**4.413**	**4.461**	**3.669**	**7.863**	0.103	1.248	**5.547**	2.645	**11.488**
	M	0	1.681	-0.583	**2.271**	**2.162**	**4.71**	1.085	1.025	**3.963**	0.827	**5.171**
	M	0.7	0.83	-0.88	**2.192**	1.596	**4.238**	1.427	1.482	**3.652**	1.468	**3.219**
	L	-0.1	**4.407**	**3.268**	**4.088**	**3.711**	**2.681**	**3.683**	**3.626**	**5.442**	**2.393**	**7.133**
	L	0	**1.383**	0.467	**1.724**	**1.808**	**1.906**	**3.124**	1.316	**2.611**	1.026	**3.479**
	L	0.7	**3.09**	0.663	**4.768**	**4.6**	**5.376**	**5.196**	**2.241**	**4.831**	**2.95**	**1.463**

TABLE 5.4: Percentage decrease in the test error; normal data; moderate outlier

	Configurations		Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\bar{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	S	-0.1	**25.151**	**25.151**	**25.209**	**25.151**	**24.315**	**25.151**	**25.209**	**25.209**	**25.123**	**25.151**
	S	0	**3.816**	**3.719**	**3.741**	**3.782**	**3.334**	**3.823**	**3.647**	**3.722**	**3.672**	**3.807**
	S	0.7	0.187	-0.167	0.021	0.018	-0.139	0.217	-0.086	0.029	-0.172	0.5
	M	-0.1	**32.648**	**32.566**	**32.621**	**32.648**	**32.648**	**31.634**	**30.921**	**32.538**	**31.606**	**32.648**
	M	0	**5.258**	**4.754**	**4.74**	**5.066**	**5.312**	**4.551**	**3.577**	**4.747**	**4.504**	**5.414**
	M	0.7	0.055	-0.207	-0.021	-0.097	0.176	0.058	-0.205	-0.017	-0.044	0.405
	L	-0.1	-0.409	-0.409	-0.351	-0.409	-0.409	-0.409	-0.409	-0.409	-0.409	-0.409
	L	0	-0.007	0.004	-0.007	-0.011	0.004	0	-0.011	0.015	-0.004	-0.029
	L	0.7	-0.03	-0.013	-0.014	-0.007	-0.034	0.016	-0.021	-0.008	-0.04	0.007
Dispersion difference	S	-0.1	**35.067**	**24.116**	**30.215**	**25.478**	**32.536**	**-46.6**	**31.374**	**28.989**	**31.189**	**36.045**
	S	0	**36.756**	**24.865**	**32.353**	**28.549**	**34.74**	**-46.625**	**33.528**	**31.278**	**32.085**	**37.071**
	S	0.7	**2.659**	2.03	**-4.328**	**-3.421**	**9.553**	0.409	0.818	**3.65**	**-3.062**	**11.768**
	M	-0.1	**1.728**	**1.286**	**1.363**	0.364	**1.821**	**3.455**	0.539	**1.592**	**1.586**	**1.832**
	M	0	**1.721**	1.399	1.376	0.286	**1.865**	**4.044**	0.549	**1.662**	**1.546**	**1.915**
	M	0.7	**1.103**	**1.497**	**1.504**	0.81	**1.624**	-0.825	0.881	**1.439**	**1.651**	**2.19**
	L	-0.1	-0.408	**-1.052**	**-1.162**	**-1.37**	**3.693**	-0.819	-0.832	0.088	**-1.072**	**3.816**
	L	0	-0.479	-1.013	**-1.088**	**-1.376**	**3.554**	-0.78	-0.865	-0.044	**-1.334**	**3.751**
	L	0.7	-0.662	**-1.769**	**-1.543**	**-1.804**	**3.429**	0.578	**-2.227**	**2.031**	-1.109	**4.307**

TABLE 5.5: Percentage decrease in the test error; lognormal data; moderate outlier

	Configurations		Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\overline{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\|\mathbf{v}\|^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	S	-0.1	**13.199**	**13.199**	**-2.944**	**-2.322**	**4.603**	-1.307	**-3.69**	**-4.209**	-1.143	**13.161**
	S	0	**13.41**	**13.251**	1.294	-1.142	**7.141**	0.244	-0.36	0.068	-0.842	**13.904**
	S	0.7	**7.958**	**6.951**	**4.313**	**3.209**	**7.503**	0.467	**4.881**	**3.56**	**2.105**	**11.852**
	M	-0.1	**10.987**	**10.922**	2.135	-0.5	**8.672**	-1.667	-1.069	**2.417**	0.22	**11.311**
	M	0	**10.005**	**9.934**	**3.305**	0.411	**8.15**	-1.09	0.044	**3.228**	0.294	**11.017**
	M	0.7	-0.076	-1.279	-0.003	-0.46	1.069	-0.304	-1.104	-0.159	-0.6	1.58
	L	-0.1	**6.99**	**6.99**	-0.548	-0.743	**4.679**	-2.443	-1.074	-1.664	-0.397	**7.371**
	L	0	**6.495**	**6.203**	1.28	-0.099	**6.06**	-1.415	-0.32	0.684	0.116	**8.957**
	L	0.7	**1.758**	0.154	**1.436**	**1.082**	**3.143**	0.006	**1.196**	0.458	0.437	**4.992**
Dispersion difference	S	-0.1	**22.195**	**22.167**	**7.161**	**3.92**	**7.901**	-3.673	**5.325**	**9.524**	1.97	**23.476**
	S	0	**8.91**	**8.606**	**3.319**	**2.154**	**6.439**	**2.336**	**2.973**	**5.858**	0.835	**12.806**
	S	0.7	**9.063**	**9.128**	2.810	1.767	**6.312**	1.358	**5.536**	**4.728**	0.806	**11.463**
	M	-0.1	**10.288**	**7.436**	**6.772**	**5.305**	**11.18**	0.116	-0.484	**9.927**	**6.709**	**16.268**
	M	0	**2.115**	-0.137	**2.801**	**1.953**	**5.715**	0.981	1.18	**4.842**	1.633	**6.279**
	M	0.7	1.293	-0.589	1.913	1.54	**4.742**	1.38	1.873	**3.038**	1.533	**5.216**
	L	-0.1	**5.771**	**4.67**	**5.269**	**4.929**	**2.64**	**3.298**	**3.132**	**7.135**	**3.272**	**9.898**
	L	0	**2.27**	0.929	**2.664**	**2.487**	**2.838**	**3.425**	1.342	**4.108**	**1.783**	**5.662**
	L	0.7	**3.071**	0.276	**4.336**	**4.033**	**4.965**	**4.106**	**1.576**	**4.117**	**3**	**3.174**

TABLE 5.6: A comparison of the performance of the nine criteria

Characteristic		$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\overline{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$
# of test error reductions	*signif.*	47	39	38	31	59	19	27	43	26
	insignif.	13	13	14	15	8	25	20	18	23
	Total	**60**	**52**	**52**	**46**	**67**	**44**	**47**	**61**	**49**
	*Rank**	*3*	*4*	*5*	*8*	*1*	*9*	*7*	*2*	*6*
# of test error inflations	*signif.*	0	3	7	9	0	8	3	1	6
	insignif.	12	17	13	17	5	20	22	10	17
	Total	**12**	**20**	**20**	**26**	**5**	**28**	**25**	**11**	**23**

*Ranking of the criteria: 1= the best; 9=the worst

However, we do want to highlight the results given in Table 5.6. These results were obtained from Table 5.2 to 5.5, and therefore represent a summary from the 72 configurations displayed in these tables. In Table 5.6 we give the nine criteria and a summary of how each criterion performed in terms of reducing the error rates. The purpose of the table is to compare the criteria and also to rank them from best to worst. The number of test error reductions (the positive entries in Table 5.2 to 5.5) were counted for each criterion, and split into whether the reduction was significant or not significant. In 67 of the 72 configurations, the criterion $v^{(i)}$ corresponds to a reduction in the test error, of which 59 configurations represent a significant reduction. The lowest frequency of test error reductions were found for $a^{(i)}$. For the alignment, only 44 out of the 72 configurations showed a decrease in the test error, of which 19 was significant. The rest of the criteria can be interpreted similarly. The row representing the rank in Table 5.6, ranks the criteria from best (1) to worst (9), with the best being $v^{(i)}$, then $\bar{f}^{(i)}$, *etc*., and the worst being $a^{(i)}$. For interest sake, we report the number of test error inflations (the negative entries in Table 5.2 to 5.5) and split them into significant or insignificant as well.

5.5 Identifying influential cases in KFDA: practical applications

In addition to the simulation study, we applied the nine criteria to real-world data sets to get an idea of the performance of the nine criteria in a practical situation. Results obtained on two data sets, are given here. A Gaussian kernel with $\gamma = 1/p$ and a hyperparameter value of $\lambda = 0.1$ was used to perform KFDA. Before KFDA was applied, the data was standardized similar to what was done for the simulation studies.

5.5.1 Flea beetle data

The first data set that we consider consists of $n = 39$ $(n_1 = 19$ and $n_2 = 20)$ observations and 4 variables pertaining to the morphological properties of two species of flea beetles (*cf.* Flury, 1997, *p*.306). A pairs plot of this data set is given in Figure 5.1. In each of the two dimensional scatter plots the two classes are displayed by the red circles and green squares.

The criteria were then calculated for this data set using the leave-one-out approach. Thus, upon the omission of the *i*-th case, KFDA was performed and each criterion was calculated. This was done for each of the 39 observations. In Figure 5.2, the values obtained for each criterion are plotted against the index of the *i*-th omitted case. This figure contains nine graphs (known as index plots) which corresponds to the nine criteria. The symbol used for each criterion is displayed on the vertical axis of each index plot. These plots reveal interesting patterns and definite changes in the values of the criteria are visible when some observations are left out from the analysis. These changes are represented by the large spikes in each graph.

- The criteria $\bar{d}^{(i)}$, $\bar{m}^{(i)}$, $r^{(i)}$, $\|\mathbf{v}\|^{(i)}$, $\bar{f}^{(i)}$ and $\bar{h}^{(i)}$ identified case 27 as the most influential case, meaning that the omission of this case optimizes each of these criteria. The second most influential case is observation 21.

- The criteria $v^{(i)}$ and $a^{(i)}$ identified case 38 as the most influential. The graphs representing these two criteria also indicate that case 27 is the second most influential.
- There were several cases that caused the criterion $R_{emp}^{(i)}$ to be a minimum. Thus, for this data set $R_{emp}^{(i)}$ could not identify a single most influential case.

The three cases mentioned above are numbered in Figure 5.1 to demonstrate their positions relative to the rest of the cases in their class. To study the effect of the omission of these cases on the generalization performance of the KFD classifier, we calculated the cross-validation error rate (CVE). The CVE was obtained with 10000 random splits of the data into training data (75%) and a test data (25%). On the full data set, the CVE was obtained as 0.0778.

- When only case 27 was omitted the CVE decreased to 0.0537 (a 30.97% decrease).
- When only case 38 was omitted the CVE was 0.0576 (a 25.96% decrease).
- Studying Figure 5.1, we observe that cases 27 and 38 surprisingly do not deviate much from the rest of the data in its class. However, case 21 seems to deviate somewhat from the data in its class. We also calculated the CVE when only case 21 was omitted. The CVE was obtained as 0.0648, a 16.70% decrease.

In the above cases we see that the error rate is reduced when omitting the cases identified by the criteria. To determine whether omission of cases 27 and 38 respectively leads to the lowest error rates that can be achieved by omitting a single case, the CVE was also calculated after omitting each of the other cases, one by one. The resulting CVEs are displayed in Figure 5.3. From this index plot it is clear that the lowest error rate is indeed obtained when omitting case 27, the second lowest when omitting case 38 and the third lowest when omitting case 21. From this example it is evident that the cases identified by the criteria are also those cases that have the most detrimental effect on the KFD generalization performance, in the sense that omission of each of the cases prior to obtaining the KFD classifier, results in a lower estimated error rate.

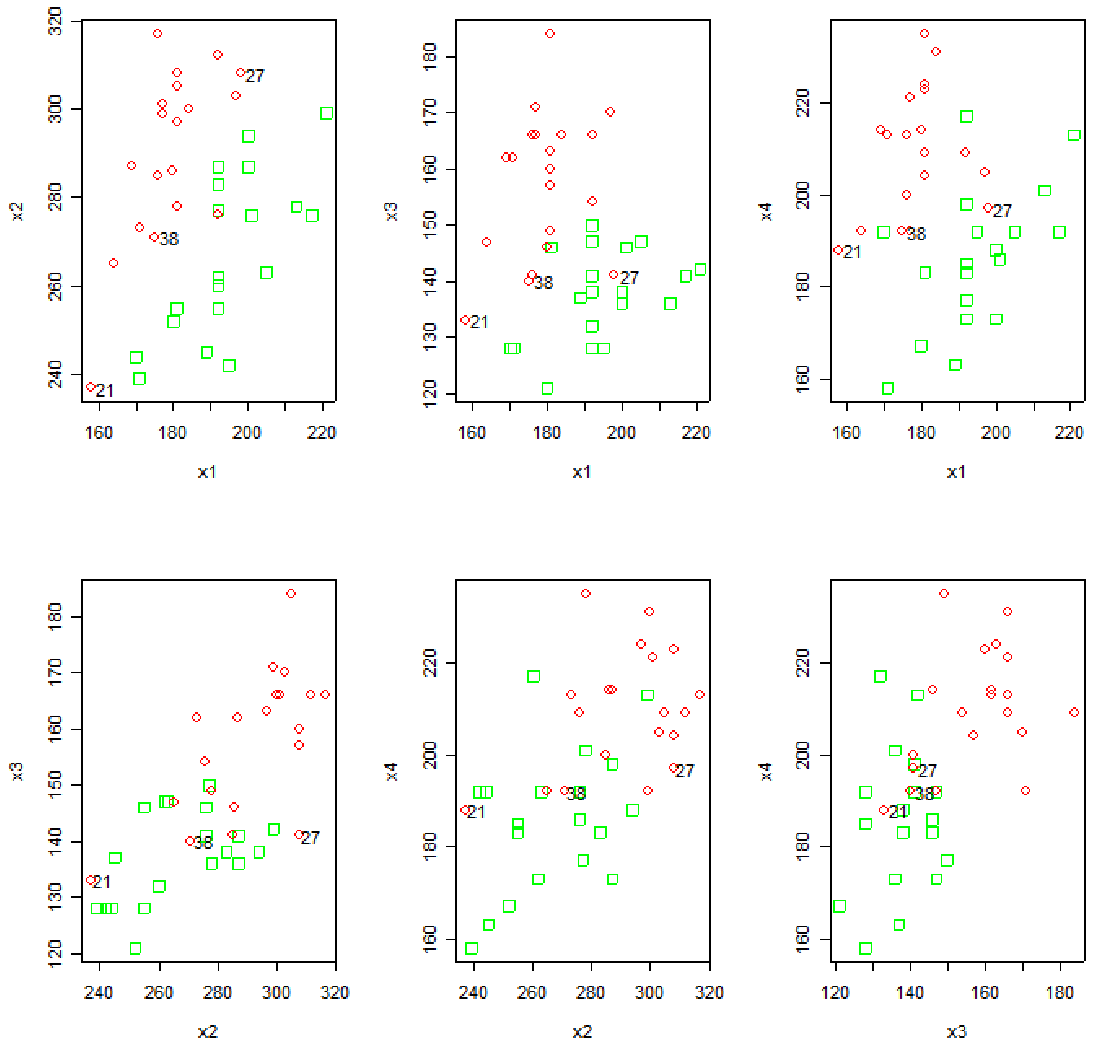


FIGURE 5.1: *Pairs plot of the flea beetle data. Three observations are numbered to illustrate their positions.*

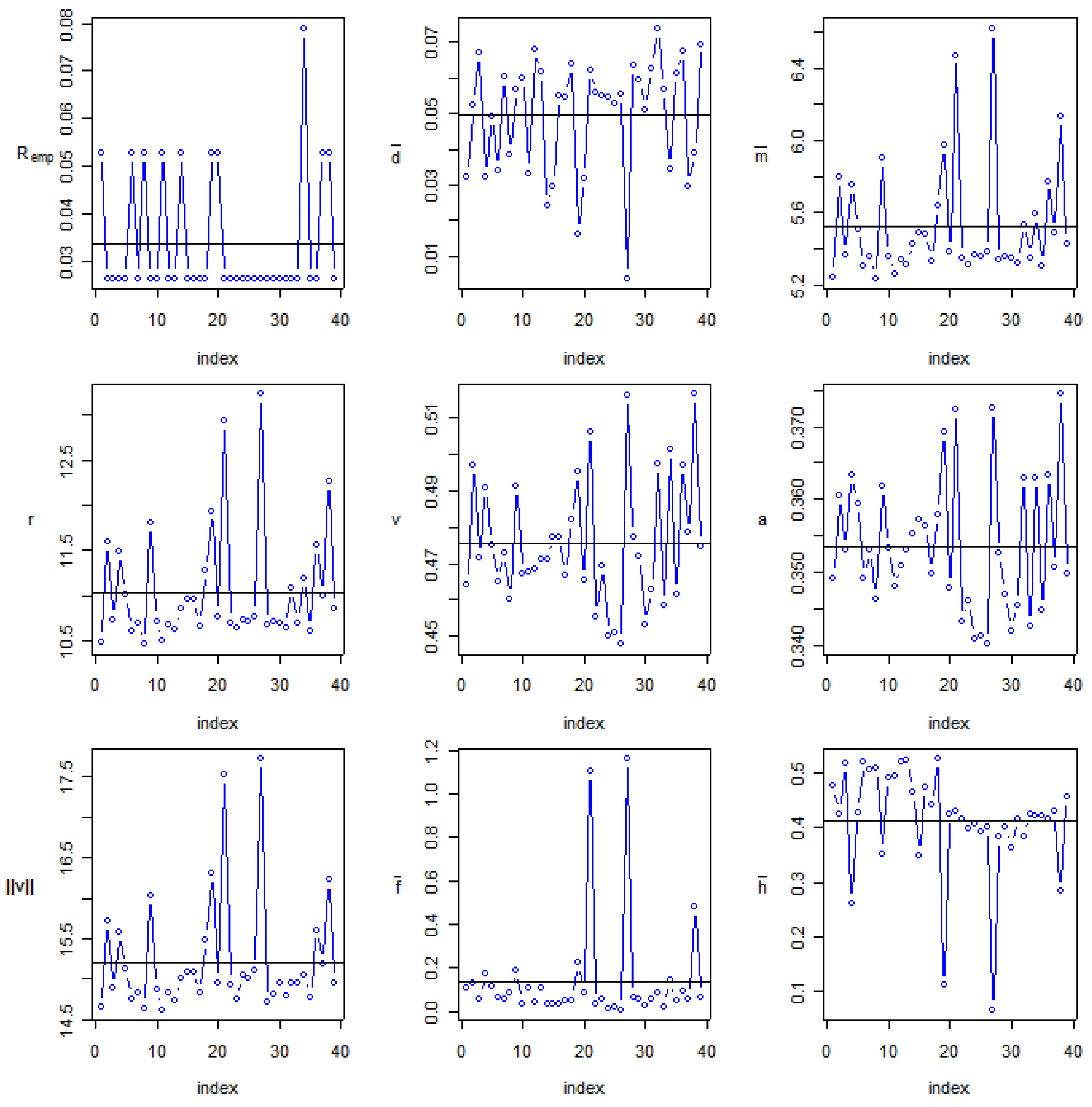


FIGURE 5.2: *Index plots of the nine criteria for the flea beetle data. The horizontal lines are the averages for each criterion.*

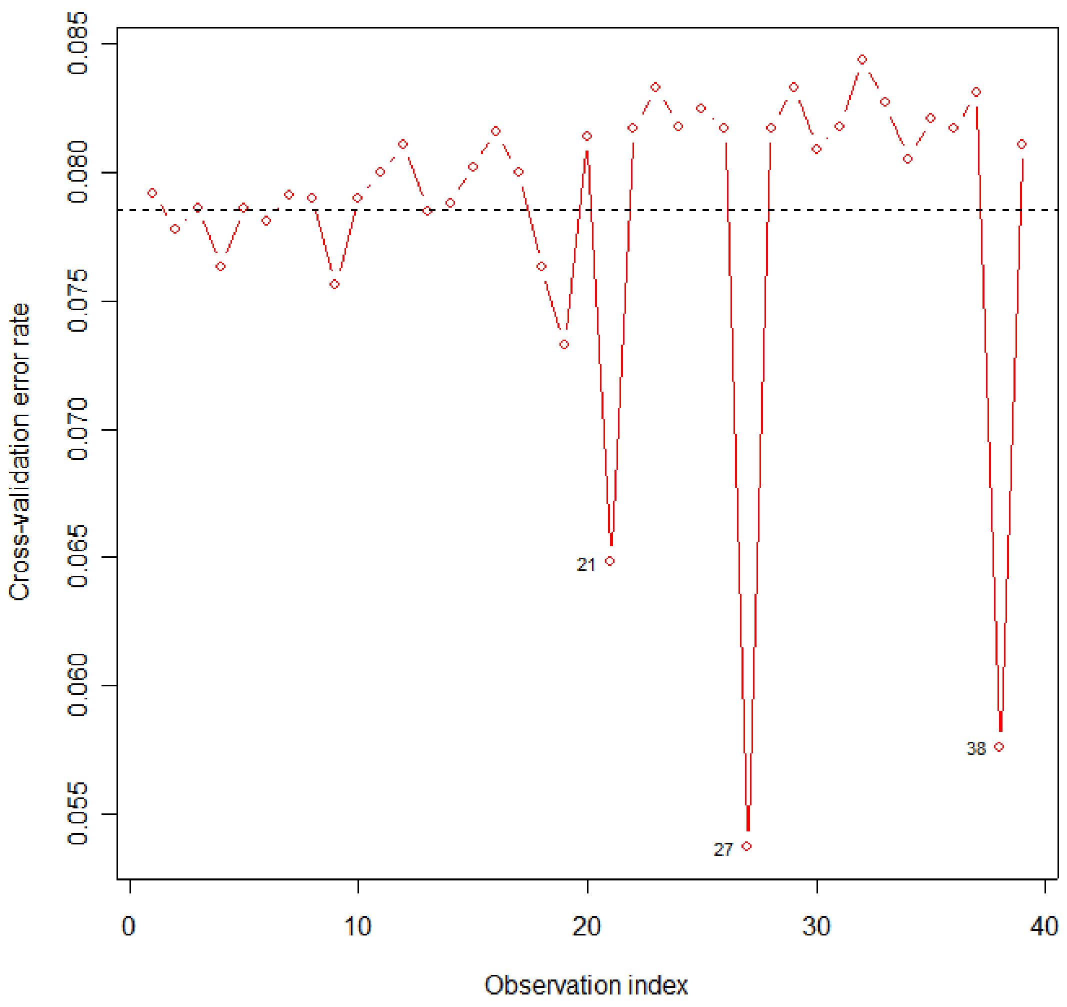


FIGURE 5.3: *Index plot of the CVEs for the flea beetle data after the omission of a single case. The horizontal line represents the average CVE value (0.0785).*

5.5.2 Swiss bank note data

The second data set consisted of $n_1 = 100$ genuine and $n_2 = 100$ forged Swiss frank notes (*cf*. Flury and Riedwyl, 1988). Six variables, amongst which the length, width and diagonal length, were measured on these notes. A pairs plot of the six variables is given in Figure 5.4 for the two classes. The cases coloured in green represent the genuine notes and the red cases represent the forged notes.

Again, the nine criteria were calculated on a leave-one-out basis, which resulted in the index plots displayed in Figure 5.5. It is clear from these graphs that case 70 (which is part of the genuine class) is definitely the most influential in all the criteria.

Similar to the flea beetle data, we also obtain the CVE by omitting the most influential case. The CVE of the KFD classifier obtained from the full data set was 0.0135, which drops to 0.00756 after case 70 was omitted (a 44% decrease). We then calculated the CVEs after the omission of all the 200 observations one by one. In 198 instances the VE was higher than 0.00756, ranging from 0.00902 to 0.01371. However, when case 1 was omitted the CVE was 0.00586 (a 56.59% decrease). Figure 5.6 is an index plot of the CVEs when observations are omitted one at a time. For this example the omission of the case identified by the criteria resulted in the second lowest CVE. This is still an indication that the criteria are quite capable of identifying cases that influence the KFD generalization performance.

In a further analysis, we considered the data set without case 70, and recalculated the criteria for each of the remaining 199 cases. All the criteria except $v^{(i)}$ achieved their optimal value for case 1, indicating that if the procedure was carried out in a sequential way, case 70 would be identified first, and then case 1. The CVE after removal of both these cases was 0.00140 (an 89.62% decrease when compared to the error rate on the full data set).

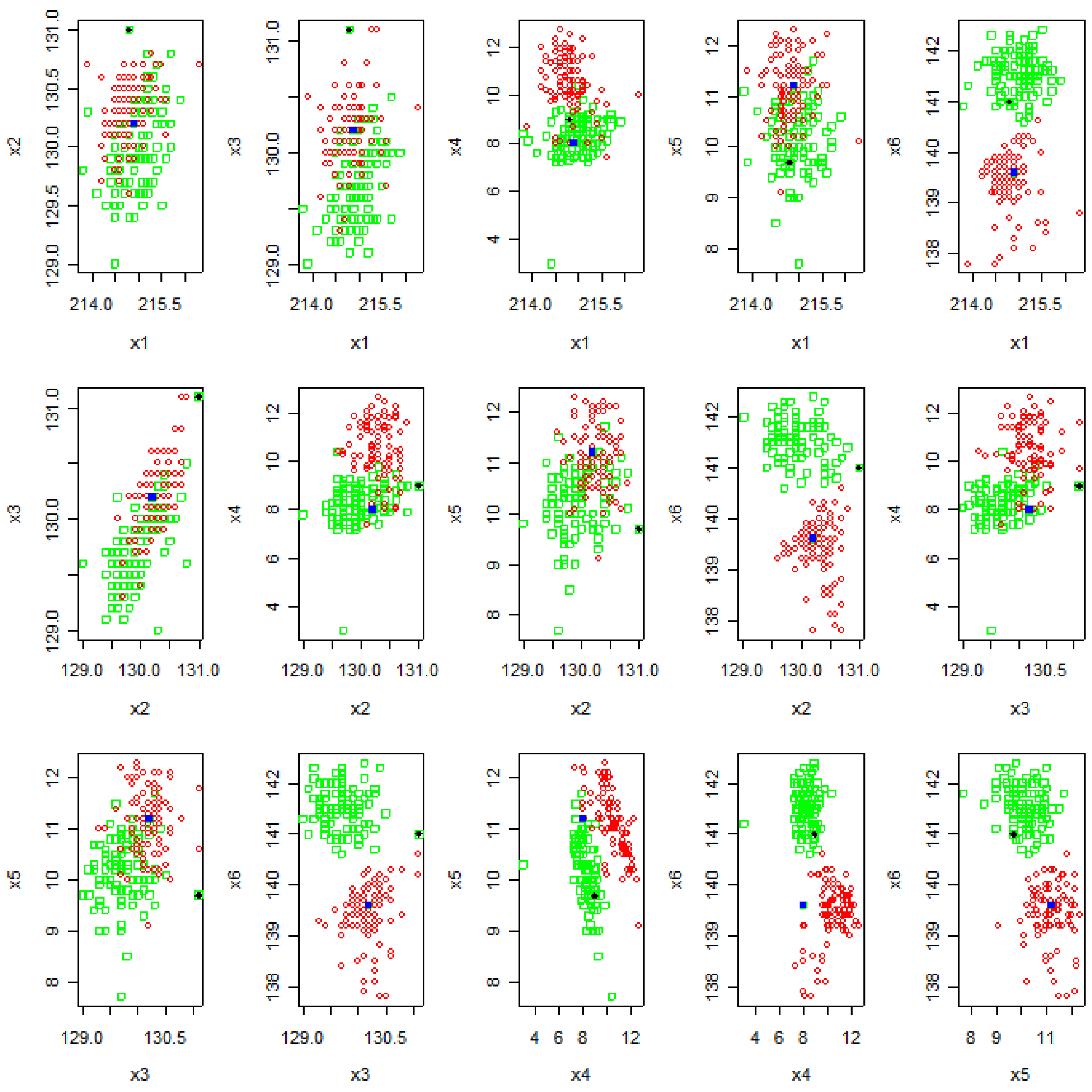


FIGURE 5.4: *Pairs plot of the Swiss bank note data. The blue observation in each scatter plot is case 70 and the black observation is case 1. Both these cases belong to class 1 (genuine notes) and are marked here to illustrate their positions relative to the rest.*

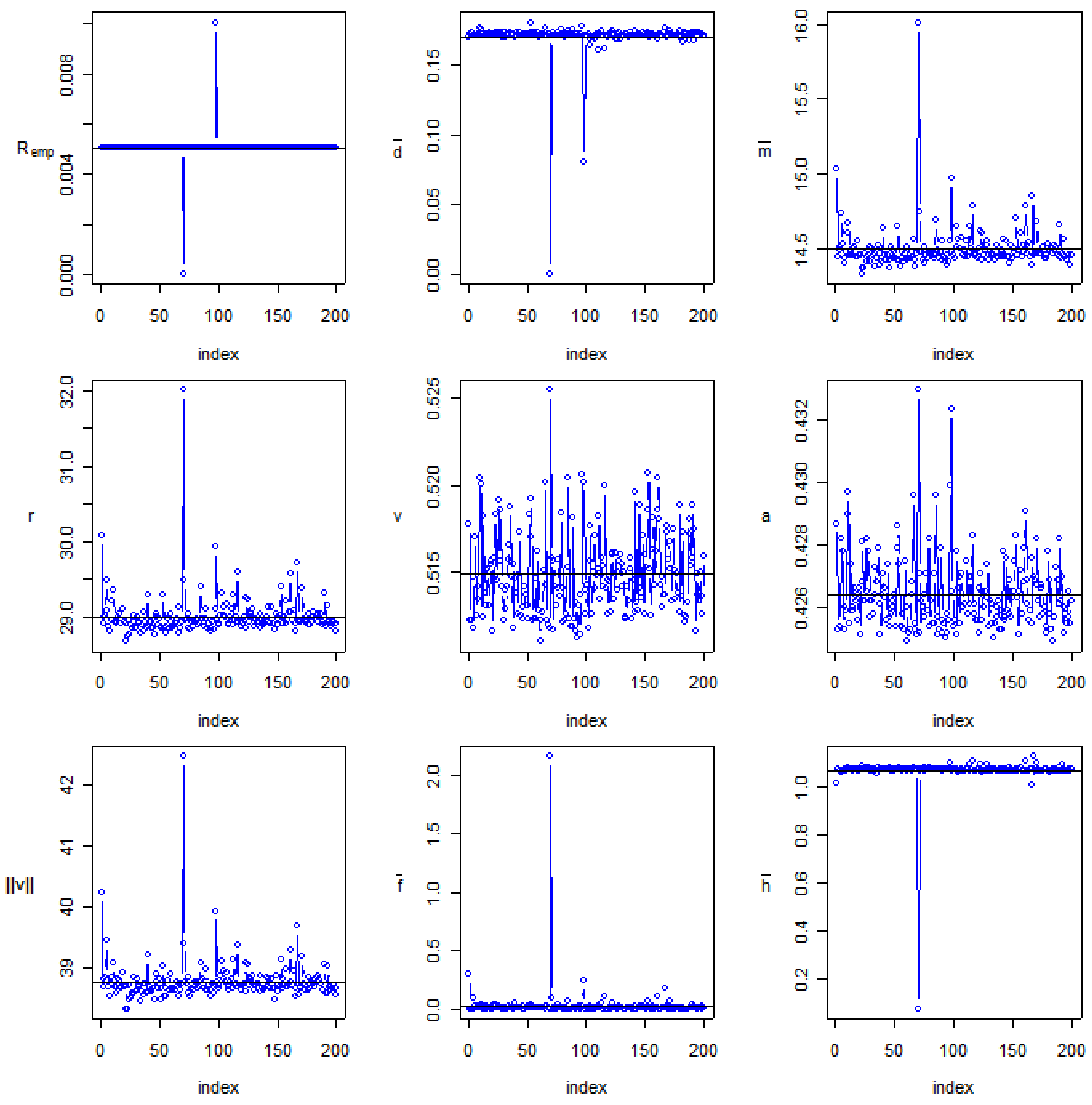


FIGURE 5.5: *Index plots of the nine criteria for the Swiss bank note data. In all the plots we observe that case 70 is different from the rest of the cases. The horizontal lines are the averages for each criterion.*

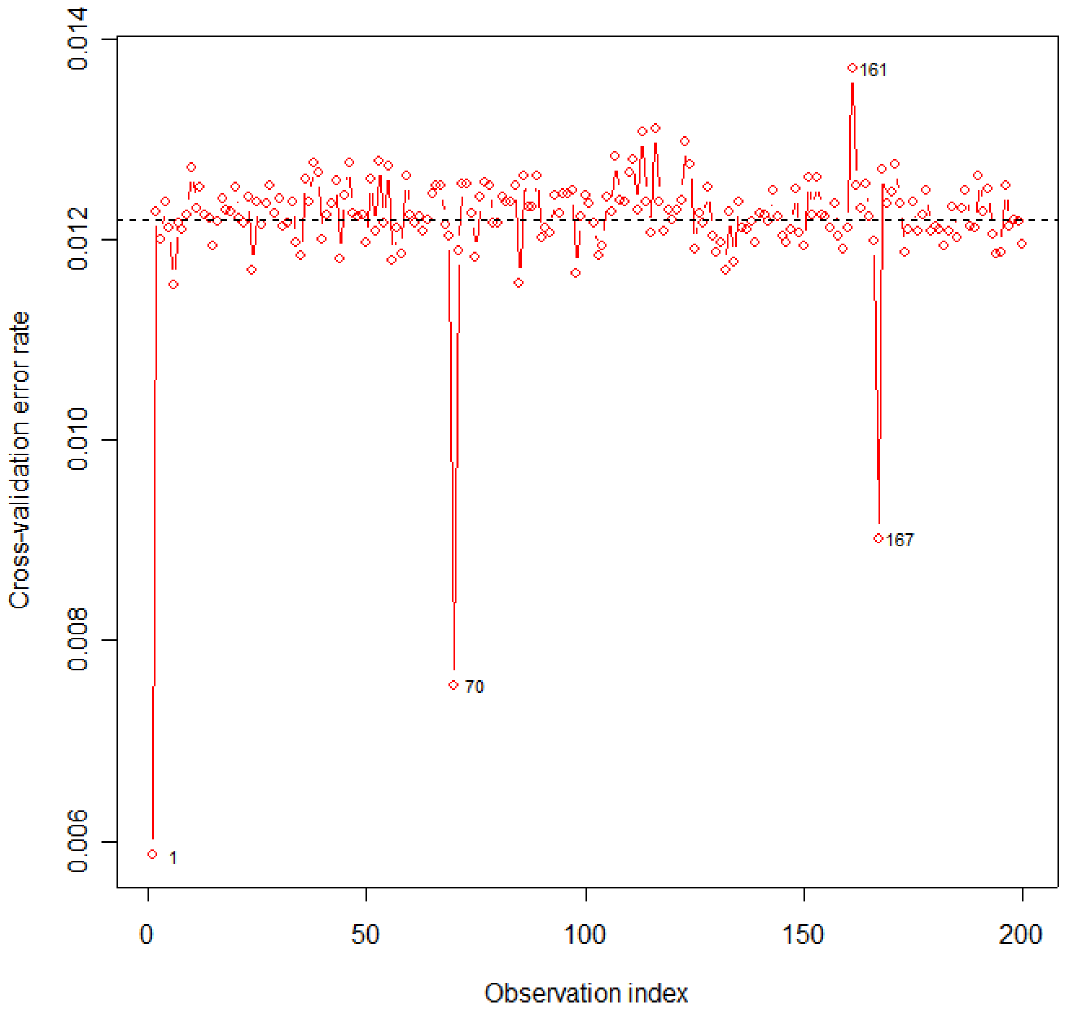


FIGURE 5.6: *Index plot of the CVEs for the Swiss bank note data after the omission of a single case. The horizontal line represents the average CVE value (0.0122).*

5.6 Concluding remarks

In this chapter we considered identification of a single influential case. As was demonstrated in the examples, the procedure could be carried out in a sequential manner to identify further influential points. It is however possible that cases may be jointly influential, and that this may not be detected when applying the procedure in a sequential manner, *i.e.* the most influential pair of cases is not necessarily the cases identified first and second when performing the procedure sequentially.

The criteria proposed in this chapter can also be used to evaluate the influence of more than one observation. Omitting sets of points, *i.e.* all possible pairs, triples, *etc.*, rather than single cases seems like a straightforward option, but even with small data sets, this quickly becomes computationally prohibitive. Finding procedures in which the influence of groups of cases can be assessed without explicitly considering omission of all possible groups, remains an open problem and an avenue for further research.

One drawback of our proposal is that calculation of the criteria on a leave-one-out basis poses a problem with regard to computing time, especially when dealing with large data sets. This problem will be addressed in Chapter 6, where we propose a procedure that can be used as a pre-screening or filter mechanism to reduce the number of cases which will be evaluated on a leave-one-out basis.

The criteria can also be extended for application in other kernel classification methods as well. Note that for methods similar to KFDA, like SVM and regKFDA described in Chapter 2, we can derive criteria similar to $R_{emp}^{(i)}$, $\overline{d}^{(i)}$, $\overline{m}^{(i)}$, $\overline{f}^{(i)}$, $\|\mathbf{v}\|^{(i)}$ and $\overline{h}^{(i)}$. The measure $r^{(i)}$ (the Rayleigh coefficient) is specific to KFDA, and could not readily be extended for application in other kernel methods. On the other hand the criteria $a^{(i)}$ and $v^{(i)}$ can directly be applied in any kernel based classification technique that makes use of the Gram matrix, even if these techniques are not related to KFDA. Since our focus is restricted to KFDA only, we do not report results on the behaviour of these criteria for other classification techniques.

The index plots (Figures 5.3 and 5.6) of the nine criteria show that they definitely are successful in identifying influential cases. Omitting the identified case results in a better generalization performance of the KFD classifier.

When considering Section 5.5, the question may arise whether the index plot of the CVEs could be used to identify the most influential cases. Calculating CVEs on a leave-one-out approach can be time consuming, especially for large data sets. Computing the criteria takes less time. Also, the criteria incorporate other geometrical aspects of KFDA which can give new insight into the data, since we are often interested in more than just improving the generalization performance.

Avenues for further research:

- The leave-one-out approach poses a computational challenge for large data sets. How can we reduce the computational time?
- The leave-one-out approach does not address the problem of masking. How should the influence of groups of observations be evaluated?
- In this study we only work with the Gaussian kernel. How will the criteria behave for other kernel functions?
- The interaction between the KFDA parameters (such as γ and λ) and the presence of an influential case should also be investigated more thoroughly.
- We only considered the two-class scenario. What about the influence of observations in multi-class KFDA?
- When is the value of a criterion large/ small enough to flag an influential case as significantly influential? Using a *p*-value associated with a criterion may be a possible solution. For this we need the distribution of each criterion (which is difficult to find) or alternatively a bootstrap approach could be followed.

CHAPTER 6

IMPLEMENTING THE HYPERSPHERE IN THE IDENTIFICATION OF CASES HAVING A DETRIMENTAL EFFECT ON THE KFD GENERALIZATION PERFORMANCE

"*Nothing is more important than to see the sources of invention which are, in my opinion more interesting than the inventions themselves*."

- Gottfried Wilhelm von Leibniz

6.1 Introduction

In Chapter 5, nine criteria that can be used to identify influential cases in KFDA were introduced and their performance evaluated in a simulation study as well as on real-world data sets. Each of these criteria is calculated on a leave-one-out basis, which can become computationally prohibitive when dealing with large data sets. In practice we are often faced with large data sets and therefore finding a way of identifying influential cases that does not entail evaluating criteria on a leave-one-out basis on the entire data set is important. In this chapter we propose a two-step procedure for identifying influential cases in large data sets. During the first step a subset of data cases that may potentially be influential is found and in the second step the criteria discussed in Chapter 5 are employed to identify influential cases, but only cases in the subset are considered on a leave-one-out basis, leading to a substantial reduction in computation time.

The first step entails constructing the smallest enclosing hypersphere in feature space for each class, and using these hyperspheres to find a subset of cases (in each class) to be evaluated on a leave-one-out basis in the second step. The hypersphere is constructed in a high dimensional feature space and corresponds to a support region in input space. In

Section 6.2 the notion of the support region is discussed, Section 6.3 explains the basic idea of a hypersphere, while details on obtaining the smallest enclosing hypersphere are given in Section 6.4. Once hyperspheres are constructed for the data in each class, the so-called support vectors are obtained. The support vectors of both classes then form the subset of cases that will be considered on a leave-one-out basis using each of the criteria introduced in Chapter 5. In Section 6.5 we present two illustrative examples to support our argument that we can use the hypersphere as a filter to obtain the subset of cases (in the simulation studies we will refer to the application of the hypersphere as a filter). A simulation study in which large data sets are generated, is then performed to test the proposed filtering mechanism. Section 6.6 contains the simulation design and results. In this simulation study KFDA is performed on all the data cases, but only the support vectors obtained from the hyperspheres are evaluated in the calculation of the criteria. The results show that when the case identified by each criterion is omitted, the error rates decrease most of the time.

6.2 The support region

There are many problems in statistics where researchers are interested in estimating the confidence region or support region for a data set. The support region can generally be defined as the region that contains most of the data. For example, when the data follow a multivariate normal distribution, an ellipse can be used to estimate the support region for such a data set. Such a region serves two purposes:

- it gives a good description (support region) of normal data.
- it can be used to detect outliers in normal data.

However, it is often the case that data do not follow a multivariate normal distribution. In such a case we may need another method to estimate the support region. Tax and Duin (1999) proposed a method to obtain support regions for data sets without making any assumptions about the underlying probability distribution of the data. These authors suggest that a minimal (smallest) radius hypersphere in feature space should be constructed using the training data. Once this has been done, a corresponding support region in input space can be obtained. In Section 6.3 we explain the basic idea of the hypersphere. In Section 6.4 the underlying theory of the technique is explained and three illustrative examples of this procedure are given to show the estimated support regions which enclose all the data points.

6.3 The basic idea of the hypersphere

The basic idea is to find the smallest hypersphere in feature space which contains all the data. A schematic illustration of the hypersphere in feature space and its corresponding support region in input space is given in Figure 6.1. The hypersphere in feature space can be obtained as follows: Let the training data set in input space $X \subseteq \Re^p$ be denoted by $T = \{\mathbf{x}_i, i = 1,...,n\}$. Let Φ be a non-linear mapping function and $F \subseteq \Re^N$ be a high-dimensional feature space $(N \leq \infty)$.

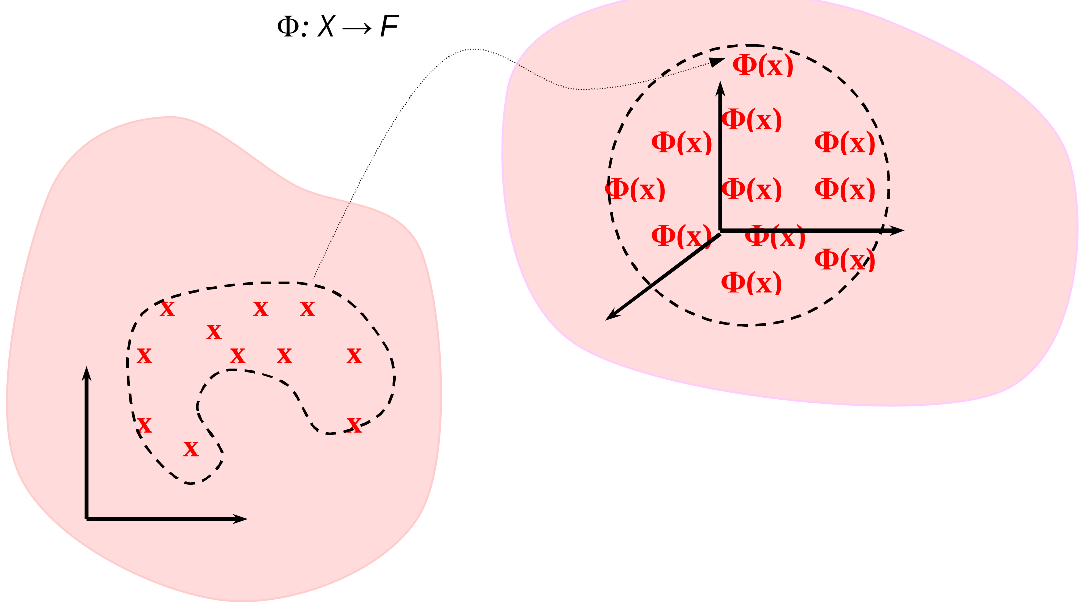


FIGURE 6.1: *An illustrative example of mapping the training data into feature space where a hypersphere is constructed which contains all the data cases. This hypersphere corresponds to a "bean-shaped" support region in input space.*

Using Φ, map T into F, resulting in the mapped data set $\{\Phi(\mathbf{x}_i), i = 1,\ldots,n\}$. The hypersphere which is then constructed for this mapped data is characterised by a center **c** and a radius r (*cf.* Figure 6.2). If all the observations fall inside the hypersphere, it is called an enclosing hypersphere (*cf.* Section 6.4). This hypersphere has the following advantages:

- it can be used to obtain a description (support region) for the data set in input space.
- it can be modified to obtain an outlier detector for the data set in input space.

As was the case for KFDA (*cf.* Section 2.3.1), we also face a computational problem in obtaining the hypersphere. The mapping function is unknown and even if it were known, the feature space might still be too large (possibly infinite) to perform explicit calculations. Similar to the derivation of the KFD classifier, we will make use of the mathematical results in Section 2.3.2 to obtain the hypersphere. This implies that the procedure to obtain the hypersphere needs to be formulated in terms of inner products of the feature space vectors. These inner products will then be replaced with a kernel function which will obviate explicit calculations in feature space. The feature space in which the hypersphere is constructed, is therefore also an RKHS (*cf.* Definition 2.2).

6.4 The smallest enclosing hypersphere

The theory of the smallest enclosing hypersphere is explained in Tax and Duin (1999), Tax (2001) and Shawe-Taylor and Cristianini (2004). It is constructed as follows: Let $\mathbf{c}^*$ be defined by

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \left\lfloor \max_{1\le i\le n} \left\| \Phi(\mathbf{x}_i) - \mathbf{c} \right\| \right\rfloor, \tag{6.1}$$

i.e. $\mathbf{c}^*$ is the center of the enclosing hypersphere having the smallest possible radius. A schematic illustration of the above is given in Figure 6.2.

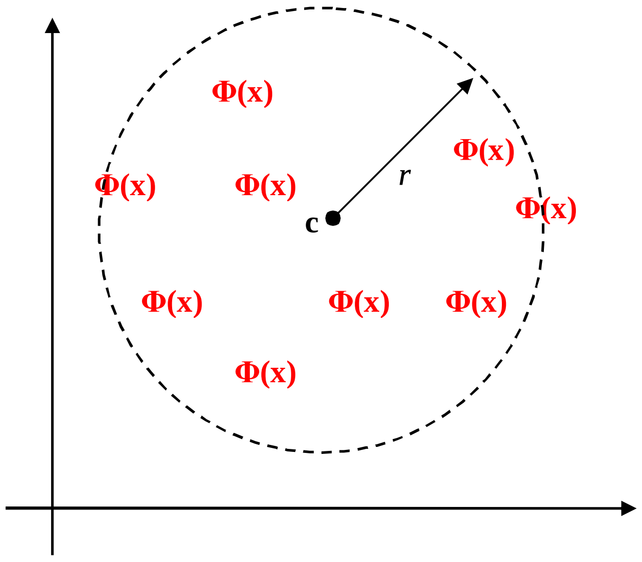


FIGURE 6.2: *Schematic illustration of the smallest enclosing hypersphere in feature space*

Since the mapping function is unknown, finding $\mathbf{c}^*$ in (6.1) is impossible and therefore the hypersphere cannot be obtained from this formulation. However, constructing the hypersphere in feature space is equivalent to solving the following quadratic optimization problem (*cf*. Shawe-Taylor and Cristianini, 2004, *p*.197):

$$\min_{\mathbf{c},r}\left[r^2\right]$$

$$\textit{subject to:}\ \left\|\Phi(\mathbf{x}_i)-\mathbf{c}\right\|^2 \le r^2,\ i=1,....,n, \qquad (6.2)$$

where r denotes the radius of the hypersphere. Note that minimizing r^2 is equivalent to minimizing r. Solving problem (6.2) can be done by defining a Lagrangian, similar to what was done to obtain the SVM classifier in Section 2.6.2. Introducing a Lagrange multiplier $(\alpha_i \ge 0)$ for the constraint in (6.2), we arrive at the following primal Lagrangian:

$$\begin{aligned} L_P(\mathbf{c}, r, \mathrm{a}) &= r^2 + \sum_{i=1}^{n} \alpha_i \left[\left\| \Phi(\mathbf{x}_i) - \mathbf{c} \right\|^2 - r^2 \right] \\ &= r^2 + \sum_{i=1}^{n} \alpha_i \left[\left\langle \Phi(\mathbf{x}_i) - \mathbf{c}, \Phi(\mathbf{x}_i) - \mathbf{c} \right\rangle - r^2 \right] \\ &= r^2 + \sum_{i=1}^{n} \alpha_i \left[\left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \right\rangle - 2\left\langle \mathbf{c}, \Phi(\mathbf{x}_i) \right\rangle + \left\langle \mathbf{c}, \mathbf{c} \right\rangle \right] - \sum_{i=1}^{n} \alpha_i r^2 . \qquad (6.3) \end{aligned}$$

This function has to be minimized with respect to **c** and *r*, and maximized with respect to $\boldsymbol{\alpha}$. Taking partial derivatives with respect to the primal variables $(\mathbf{c}, r)$ and setting them equal to zero, yields

$$\begin{aligned} \frac{\partial L_P}{\partial \mathbf{c}} &= 2\sum_{i=1}^{n} \alpha_i \left(\Phi(\mathbf{x}_i) - \mathbf{c} \right) = \mathbf{0} \text{ and} \\ \frac{\partial L_P}{\partial r} &= 2r\left(1 - \sum_{i=1}^{n} \alpha_i \right) = 0 . \qquad (6.4) \end{aligned}$$

From the second equation above we obtain $\sum_{i=1}^{n} \alpha_i = 1$, which from the first equation implies that $\mathbf{c} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i)$. Substituting these results into (6.3) gives the following dual Lagrangian:

$$\begin{aligned} L_D(\mathbf{c}, r, \mathrm{a}) = r^2 &+ \sum_{i=1}^{n} \alpha_i \left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \right\rangle - 2\sum_{j=1}^{n} \alpha_j \left\langle \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \right\rangle \\ &+ \sum_{i=1}^{n} \alpha_i \left\langle \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k), \sum_{j=1}^{n} \alpha_j \Phi(\mathbf{x}_j) \right\rangle - \sum_{i=1}^{n} \alpha_i r^2 \\ = r^2 &+ \sum_{i=1}^{n} \alpha_i \left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \right\rangle - 2\sum_{j=1}^{n} \alpha_j \left\langle \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \right\rangle \\ &+ \left\langle \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i), \sum_{j=1}^{n} \alpha_j \Phi(\mathbf{x}_j) \right\rangle - r^2 \end{aligned}$$

$$= \sum_{i=1}^{n} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2\sum_{i,j=1}^{n} \alpha_i \alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle + \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

$$= \sum_{i=1}^{n} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle . \qquad (6.5)$$

By replacing the inner product in (6.5) with a kernel function, the optimal α's can be obtained by solving

$$\max_{a} \left[\sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right]$$

$$\textit{subject to}: \sum_{i=1}^{n} \alpha_i = 1 \text{ and } \alpha_i \geq 0 \quad i=1,\ldots,n. \qquad (6.6)$$

Finding the optimal values $(\alpha_1^*, \alpha_2^*, ..., \alpha_n^*)$ is done by using a quadratic programming solver. Once these optimal values are known, the center and the radius of the hypersphere can be obtained.

An important sparseness property, which can be derived from the Karush-Kuhn-Tucker conditions (*cf.* Shawe-Taylor and Cristianini, 2004, *p*.199), implies that only the observations lying on the surface of the hypersphere have non-zero α_i^*'s. For the remaining observations in the training data we have $\alpha_i^* = 0$. The data cases with non-zero α_i^*'s are also referred to as support vectors (similar to the SVM) and only these points play a role in constructing the hypersphere. Thus, using any of the support vectors $\mathbf{x}_i$ (with *i* corresponding to a case where $\alpha_i^* > 0$) the radius of the smallest enclosing hypersphere can now be obtained as

$$\begin{aligned} r &= \left\|\Phi(\mathbf{x}_i) - \mathbf{c}\right\| \\ &= \left\|\Phi(\mathbf{x}_i) - \sum_{j=1}^{n} \alpha_j \Phi(\mathbf{x}_j)\right\| \\ &= \sqrt{\left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \right\rangle - 2\sum_{j=1}^{n} \alpha_j \left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \right\rangle + \sum_{k,j=1}^{n} \alpha_k \alpha_j \left\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_j) \right\rangle}\,. \end{aligned} \tag{6.7}$$

Replacing the inner product with a kernel function and the α 's with their optimal values $(\alpha_1^*, \alpha_2^*, ..., \alpha_n^*)$ we get

$$r^* = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{j=1}^{n} \alpha_j^* k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{k,j=1}^{n} \alpha_k^* \alpha_j^* k(\mathbf{x}_k, \mathbf{x}_j)}\,. \tag{6.8}$$

We can write the center of the hypersphere as the expansion $\mathbf{c}^* = \sum_{i=1}^{n} \alpha_i^* \Phi(\mathbf{x}_i)$. As mentioned in Section 6.3, the smallest enclosing hypersphere in feature space corresponds to a support region in input space. For a given training data set, the radius r^*, center $\mathbf{c}^*$ and support vector $\mathbf{x}_i$ can be used to construct a support region in input space, *viz.*

$$\{\mathbf{x} \in \Re^p : g(\mathbf{x}) \leq 0\}, \tag{6.9}$$

where

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

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

Figure 6.3 contains three illustrative examples of a support region that was constructed using (6.9). The three bivariate data sets of size $n = 100$ were generated as follows: the first data set was generated from a bivariate normal distribution with the following parameters

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N_2\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}\right).$$

The second data set, which we will call the v-shaped data, was generated using

$$X_1 \sim N(0,1) \text{ and } X_2 = X_1^2 + e, \text{ with } e \sim N(0,1).$$

The third data set, which we will call the s-shaped data, was generated as follows

$$X_2 \sim UNIF(0, 2\pi) \text{ and } X_1 = sin(X_2) + u, \text{ where } u \sim UNIF(-1,1).$$

The aim of these two-dimensional data sets is to illustrate how powerful the smallest enclosing hypersphere is in estimating the support region of a data set. A Gaussian kernel was used to obtain (6.9). The parameter value $\gamma = 0.2$ was used for the normal and v-shaped data, while the value $\gamma = 0.5$ was used for the s-shaped data. The three panels in Figure 6.3 clearly demonstrate how well the support region follows the distribution of the data set. All the observations fall either inside the support region or on the boundary. The cases on the boundary of the support region are the support vectors. Figure 6.4 contains index plots of the α^*-values corresponding to each data set. The spikes in each plot indicate which cases are the support vectors, *i.e.* the cases for which $\alpha_i^* > 0$. Only these cases are needed to construct the support regions in Figure 6.3.

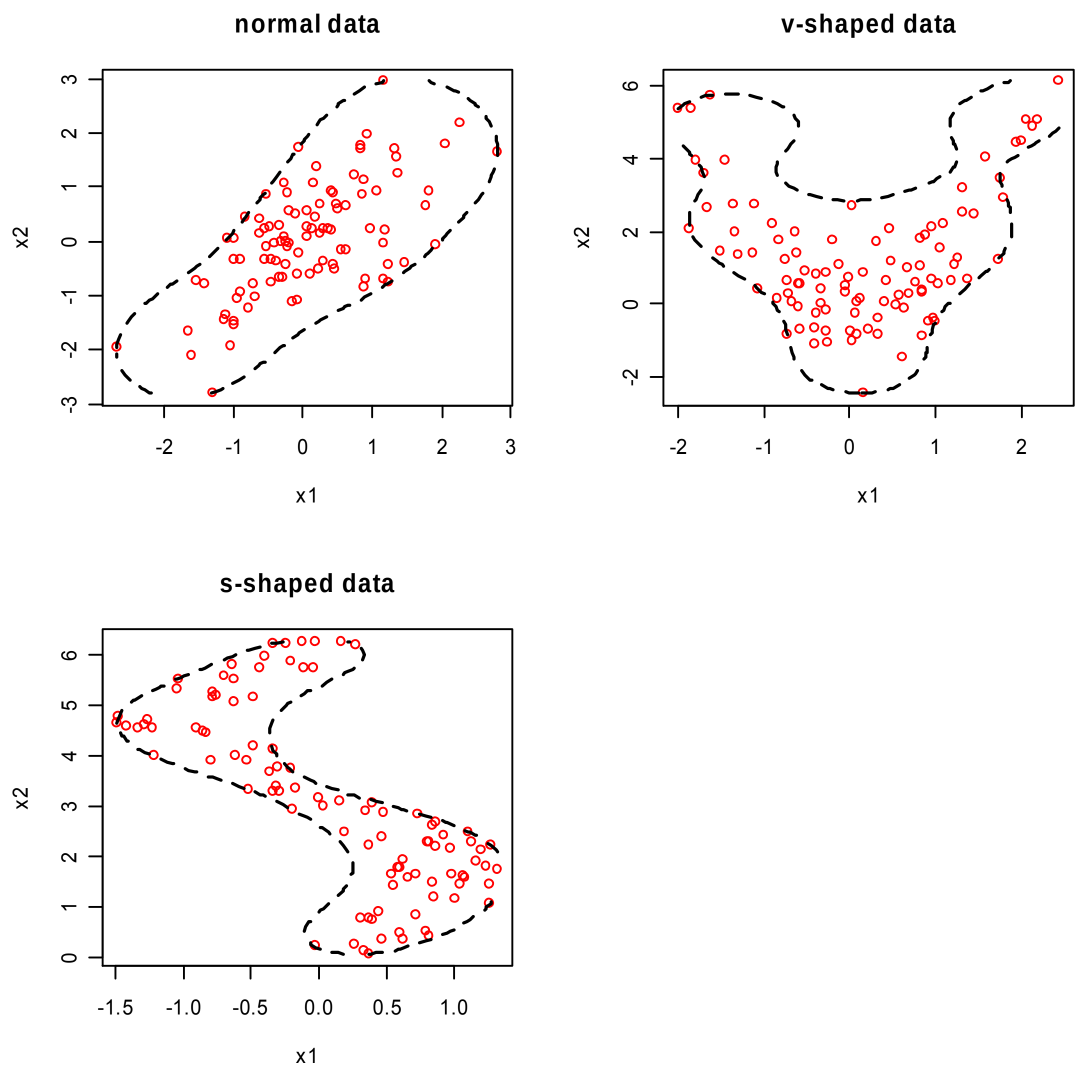


FIGURE 6.3: *Three illustrative examples of the support regions in input space which corresponds to enclosing hyperspheres in feature space. The normal data have 7 support vectors, the v-shaped data have 11 support vectors and the s-shaped data have 13 support vectors.*

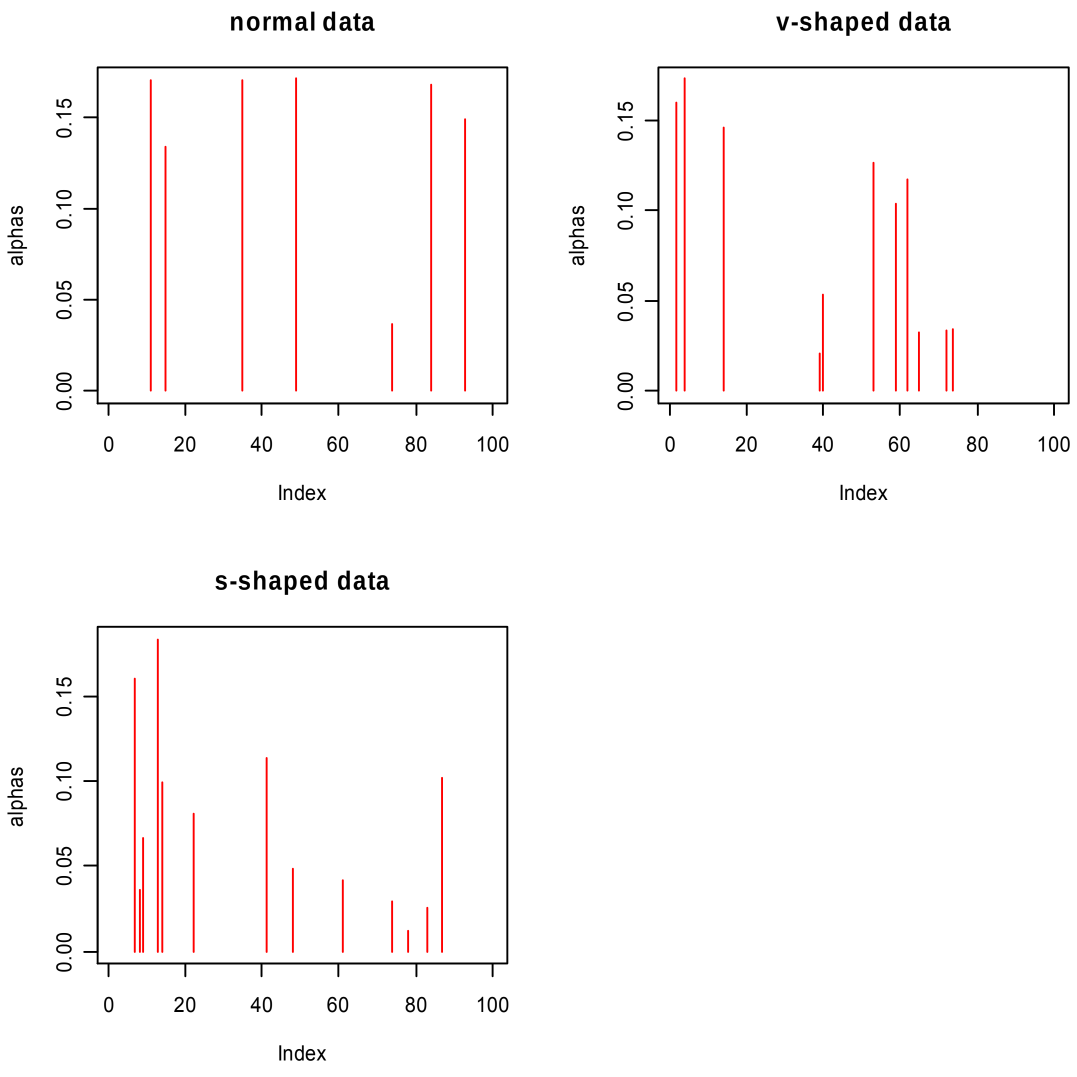


FIGURE 6.4: *Index plots of the support vectors of the three data sets in Figure 6.3.*

Choosing a proper kernel function and its corresponding parameter values is crucial for such an analysis since it affects the number of support vectors as well as the shape of the support region. The Gaussian kernel seems to work well for this type of analysis (*cf*. Tax, 2001, *p*.35) and we will continue to use it for all further examples and simulation studies in this chapter.

6.5 Obtaining a subset of observations by using the hypersphere

In this section we will implement the smallest enclosing hypersphere as a procedure to pre-screen or filter the training data before applying the criteria given in Section 5.3. In practical situations we are often faced with large data sets and calculating the values of the criteria upon omission of each of the cases may become computationally prohibitive. For this reason we propose the following strategy to reduce the number of cases that should be evaluated by the criteria on a leave-one-out basis. We propose using the smallest enclosing hypersphere (Section 6.4) in a preliminary step to identify a subset of cases. In a subsequent step, only the cases in this subset will then be individually evaluated on a leave-one-out basis as cases that potentially may have a detrimental effect on the generalization performance of the KFD classifier.

We argue that the cases that potentially have a detrimental effect on the KFD classifier are those cases in a class that deviate most from the typical cases in that class. If these cases can be identified, only they need to be evaluated by each criterion and not the entire training data set. As illustrated in the following examples, the hypersphere can be used to obtain a subset of cases that deviate from the rest of the cases in a class.

6.5.1 Illustrative examples

The first example, presented in Figure 6.5, contains two classes with equal covariance matrices but different locations, *i.e.* $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{I}$ and $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$. For each class, 200 observations were generated from a bivariate normal distribution. The mean vector for class 1 is $\mathrm{m}_1 = 2.5\mathbf{1}$ and class 2 has mean vector $\mathrm{m}_2 = \mathbf{0}$. An enclosing hypersphere was obtained for each class separately using a Gaussian kernel with parameter $\gamma = 0.1$. The support regions corresponding to these hyperspheres are plotted in Figure 6.5. From the figure we observe that all the cases lie inside the support regions and the boundary of the support regions passes through the support vectors. The number of support vectors obtained for class 1 (green squares) is six and for class 2 (red circles) is five. These support vectors have non-zero α-values which are also plotted as spikes in Figure 6.6 for each class separately. The rest of the data points have zero α-values. From Figure 6.5, we see that the observations that deviate most from the bulk of the data in a class lie on the boundary of the support region, *i.e.* they form part of the support vectors. We propose that only the support vectors should be evaluated by the criteria from Section 5.3 as cases that may possibly have a detrimental effect on the KFD generalization performance. As we see in this example, there are 400 observations in the training data set and the subset (support vectors) contains 11 observations. Thus, in our proposed two-step procedure for identification of influential cases, instead of evaluating all 400 cases on a leave-one-out basis, only the 11 cases will be evaluated. This obviously leads to a considerable reduction in the number of computations that have to be performed.

The second example is illustrated in Figures 6.7 and 6.8. The scatter plot in Figure 6.7 contains lognormal training data with equal covariance matrices $(\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \mathbf{I})$ but location differences between the two classes. The class 1 mean vector is $\mathrm{m}_1 = \mathbf{1}$ and the class 2 mean vector is $\mathrm{m}_2 = \mathbf{0}$. Again, there are 200 observations in each class. The smallest enclosing hyperspheres for each of the classes were obtained using a Gaussian kernel with $\gamma = 0.1$ and the corresponding support regions are shown in Figure 6.7. For this example, the number of support vectors in each class was 6. Again we observe that the observations that deviate from the bulk of the data are support vectors and these lie on the boundaries of the support regions. In Figure 6.8 the α-values of the support vectors for each class are plotted as spikes, while the rest of the data cases have zeros. Implementing our two-step procedure in this example implies that 12 cases, instead of 400, will be evaluated on a leave-one-out basis.

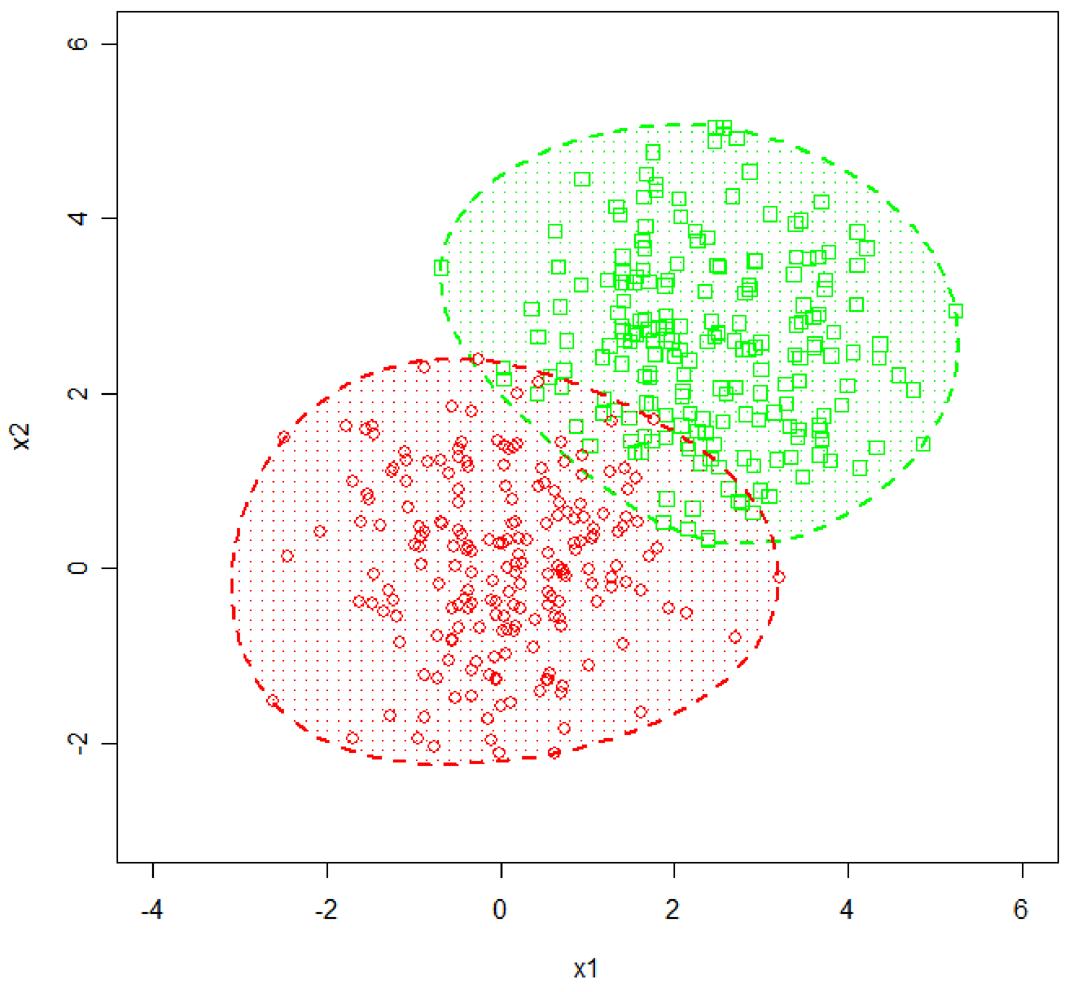
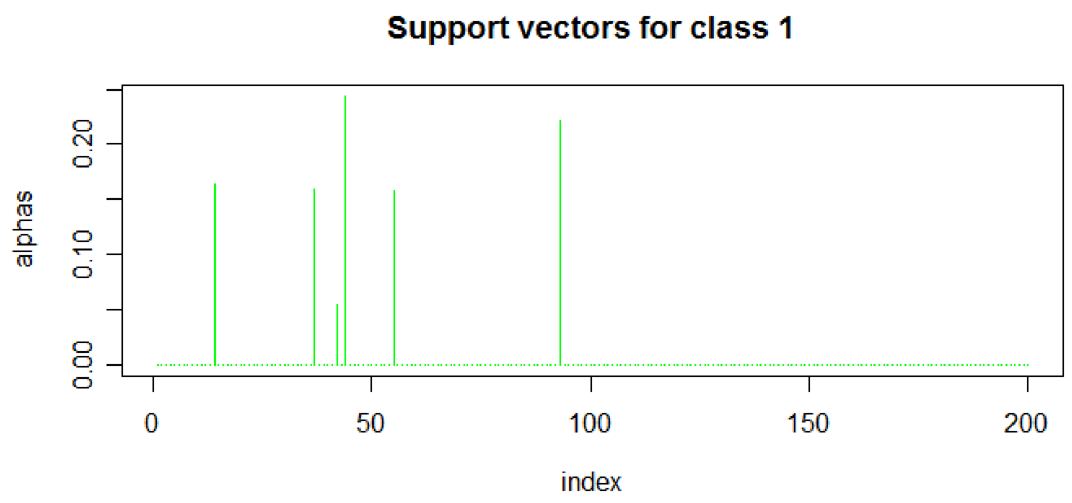


FIGURE 6.5: *Illustrative example of the support regions for normal training data. The Gaussian kernel with* $\gamma = 0.1$ *was used. The green squares represent class 1 and the red circles represent class 2.*



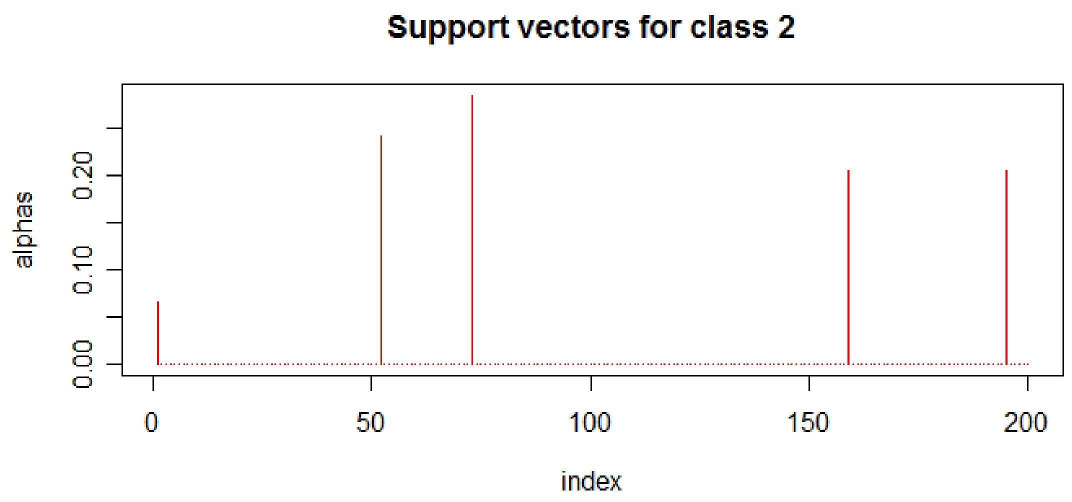


FIGURE 6.6: *Index plot of the α-values of the support vectors for each class. These plots correspond to the same data used in Figure 6.5. There were 6 support vectors in class 1 and 5 support vectors in class 2.*

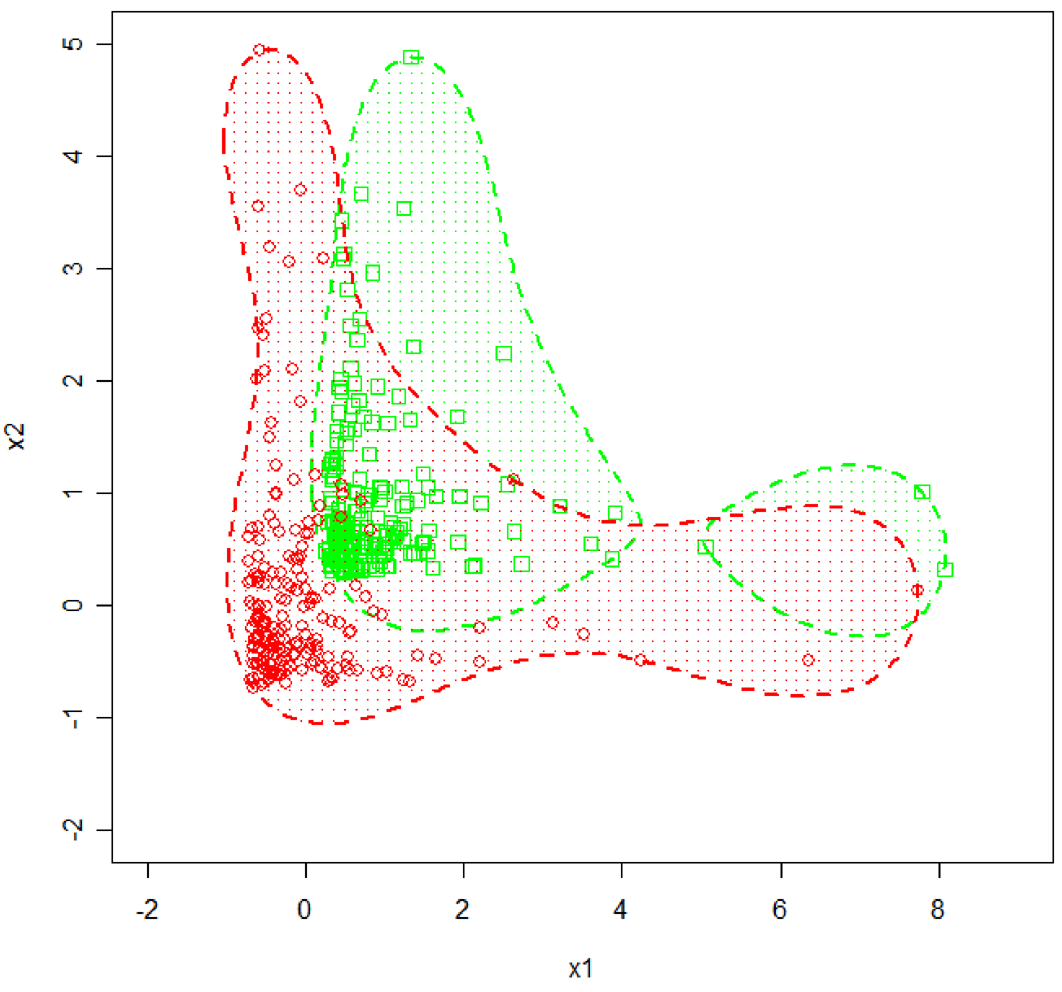


FIGURE 6.7: *Illustrative example of the support regions for the lognormal training data. The Gaussian kernel with* $\gamma = 0.1$ *was used.*

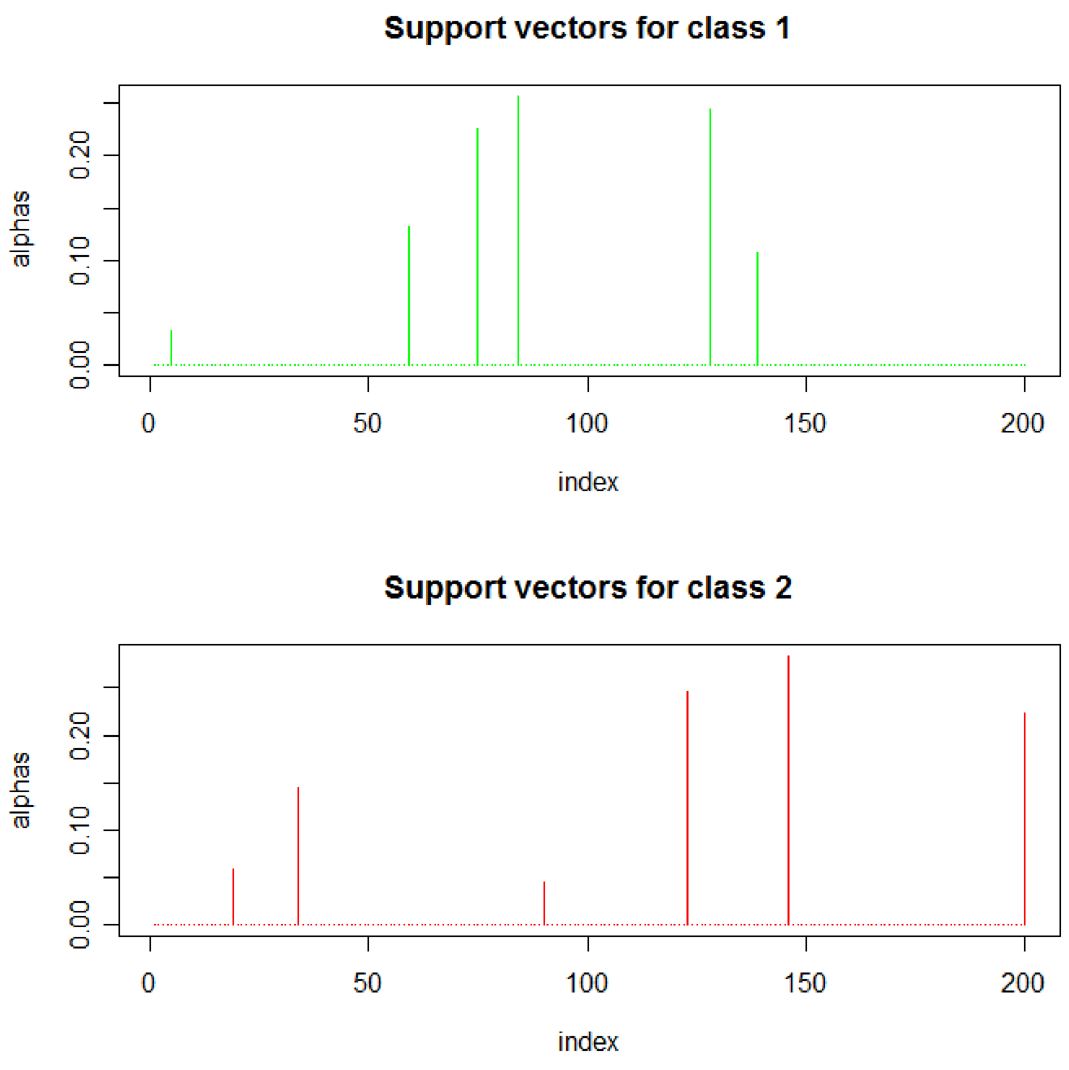


FIGURE 6.8: *Index plot of the* α*-values of the support vectors for each class. These plots correspond to the same data used in Figure 6.7. There were 6 support vectors in class 1 as well as in class 2.*

From these examples it is clear that the observations that deviate most from the bulk of the data in a class form part of the support vectors and therefore has $\alpha_i^* > 0$. Referring back to our argument, we therefore will consider the support vectors as the cases that most probably will have a detrimental effect on the KFD generalization performance. In the next section we will investigate the relationship between the parameter γ of the Gaussian kernel and the number of support vectors.

6.5.2 Relationship between γ and the number of support vectors

In this section we will demonstrate how the number of support vectors in each class can be controlled by varying the γ-value. In Figures 6.5 and 6.7 we obtained the support regions by using $\gamma = 0.1$. For these examples only a few support vectors were needed to construct the support region. Thus, the subset was quite small. A question that one may ask is whether one can control the size of the subset, *i.e.* change the number of support vectors. The following shows how the support region and the number of support vectors change as γ changes. Using the same data that were used in Figures 6.5 and 6.7, we selected $\gamma = 5$ and constructed the new support regions. The resulting support regions are shown in Figures 6.9 and 6.10. A significant change in the shapes of support regions took place for both the normal and the lognormal data. The support regions for $\gamma = 5$ is not as smooth as the ones when $\gamma = 0.1$ was used. The increase in the number of support vectors is also very dramatic. For the normal data the number of support vectors in the training data increased to 264, and for the lognormal data the number of support vectors increased to 96. Note that in Figures 6.9 and 6.10 there are some points falling outside of the support regions. These points are also support vectors having $\alpha^* > 0$.

It is clear from Figures 6.5 and 6.7 as well as 6.9 and 6.10 that the choice of the γ-value has a significant effect on the number of support vectors. A simulation study was performed as a further investigation of the relationship between the γ-value and the number of support vectors.

We used the following configurations:

- Normal and lognormal distributions were used.
- Five uncorrelated variables were used for both distributions.
- The training data consisted of 400 observations $(n_1 = n_2 = 200)$.
- The two populations differed with respect to location. We used $\boldsymbol{\mu}_2 = \mathbf{0}$ and $\boldsymbol{\mu}_1 = c\mathbf{1}$ where $c = 2.5$ was used for normal data and $c = 1$ for lognormal data. The identity matrix was used as covariance matrix for both populations $(\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \mathbf{I})$.
- Twenty equally spaced γ-values were selected between 0.01 and 10.

For each of these γ-values the number of support vectors in each class as well as for the entire training data set were obtained. These were then used to calculate the fraction of the data corresponding to support vectors in each class as well as in the entire training data set. For each of the configurations described above, 100 simulation repetitions were performed and an average fraction of support vectors was calculated over these repetitions. The results are presented in Figure 6.11 for the normal and lognormal data separately.

Figure 6.11 shows an interesting relationship between the γ-values and the fraction of support vectors. For small γ-values the fraction of support vectors is also small, but as γ increases, the fraction of support vectors also increases. For large γ-values, all the training data cases will become support vectors. When comparing the normal with the lognormal case another interesting pattern is observed. For the normal data, the curve shows a rapid increase in the fraction of support vectors for $0 < \gamma < 2$. For $\gamma > 2$, all the training data are support vectors (fraction of support vectors is equal to one). In the case of the lognormal data, the curve increases at a lower rate compared to the normal data case. It seems that only after $\gamma > 10$, all the training data cases become support vectors. Since we observe a clear relationship between the γ-values and the number of support vectors, we can use γ to control the number of support vectors.

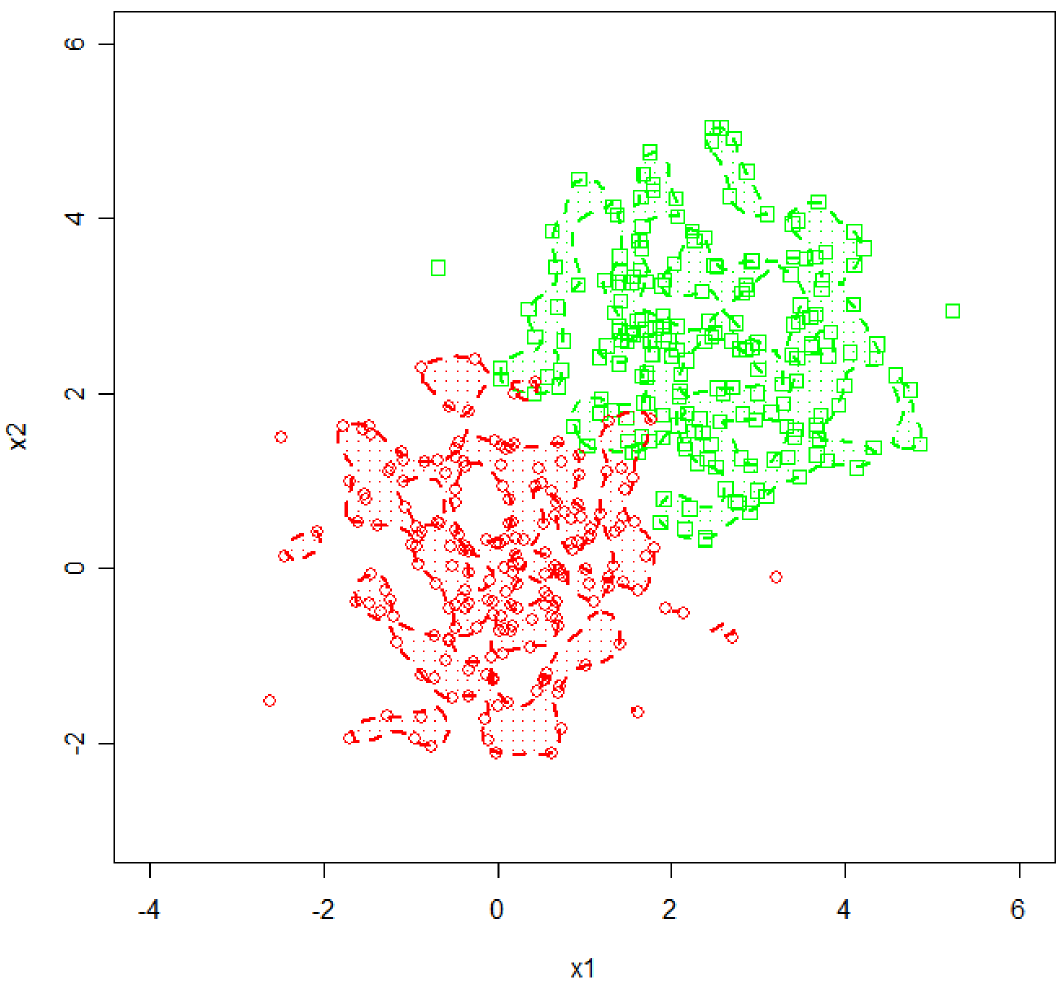


FIGURE 6.9: *Support regions for the same normal training data as in Figure 6.5. A Gaussian kernel with* $\gamma = 5$ *was used. Compared to Figure 6.5 the number of support vectors increases (from 11 to 264 for the entire data set) which also changes the shape of the support regions dramatically.*

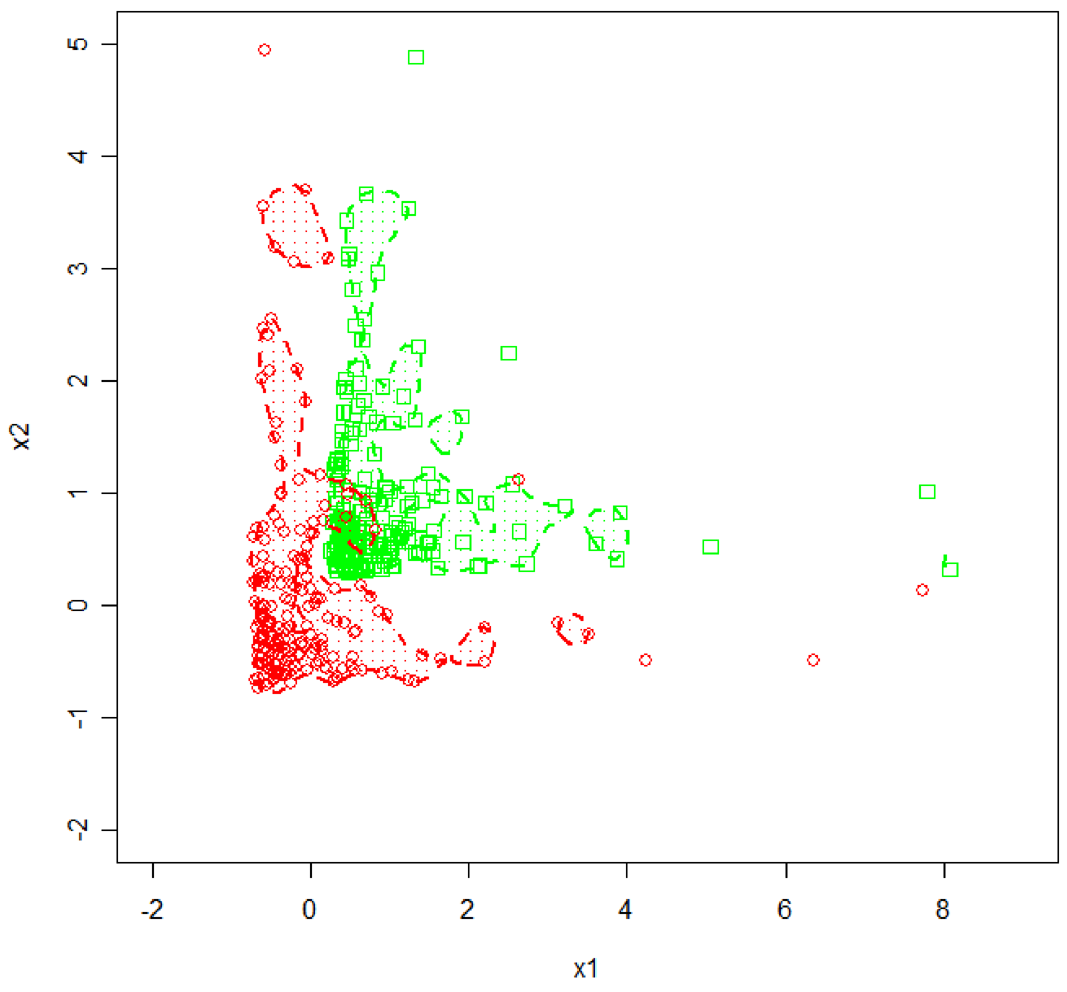


FIGURE 6.10: *Support regions for the same lognormal training data as in Figure 6.7. A Gaussian kernel with* $\gamma = 5$ *was used. Compared to Figure 6.7 the number of support vectors increases dramatically (from 12 to 96 for the entire data set). The shapes of the support regions change accordingly.*

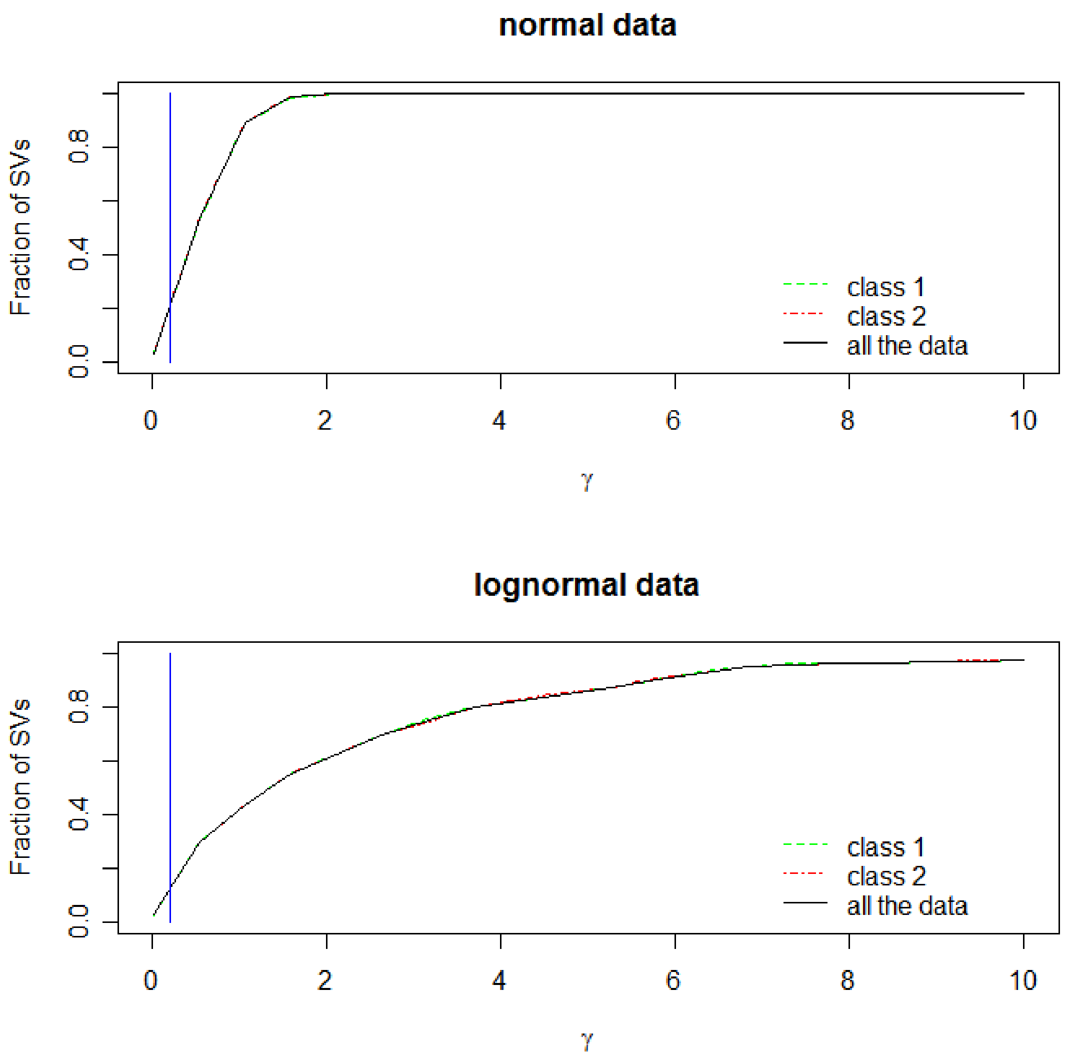


FIGURE 6.11: *Relationship between the parameter* γ *and the fraction of the data corresponding to support vectors for the normal and lognormal training data. The vertical blue line represents* $\gamma = 1/p$*, which corresponds to a number of support vectors of about 15% to 20% of the training data.*

As mentioned in the previous section, we want to use the support vectors as a subset of cases that should be evaluated by the leave-one-out criteria. By selecting an appropriate γ-value, we are able to control the size of the subset. The question that now arises is, which γ-value is an appropriate value. In the next section a simulation study will be conducted where the smallest enclosing hypersphere is applied as a filter to obtain a subset of cases, which is then evaluated by each leave-one-out criterion to identify the cases that are most detrimental to the KFD generalization performance. We will use $\gamma = 1/p$ to construct the hypersphere in the simulation study. For the congifurations considered above, this γ-value corresponds to a subset size of about 15% to 20% of the training data, as represented by the vertical line in Figure 6.11.

6.6 Identifying influential cases in KFDA after the application of a filter: a simulation study

In this section we describe a simulation study in which we used the hypersphere as a filter. We first constructed the smallest enclosing hypersphere for each class in the training data as a preliminary step to obtain the support vectors. Only these support vectors (a subset) were then evaluated by the criteria in Section 5.3. The case (from the subset) that a criterion identified as the most influential will then be omitted. Upon the omission of this case, the average test error (P_e) will be calculated (*cf*. Section 5.4).

Similar to what was done in the simulation study in Section 5.4, we inserted an atypical case in the training data to compare the error when the identified case is omitted *versus* the error when the atypical case is omitted. The next two subsections contain more detail on the simulation design and the results.

6.6.1 Design of the simulation study

For this study we fixed the number of variables at $p = 5$. A total of 1000 simulation repetitions were performed for each combination of the following factors:

- The underlying distributions which were used to generate the training data are the normal and lognormal distributions.
- Two levels for the sample sizes were chosen: $n_1 = n_2 = 100$ and $n_1 = n_2 = 200$.
- Three levels were used for the correlations between input variables. We investigated the cases where $\rho = -0.1$, 0 and 0.7.
- With respect to the differences between the two populations, we investigated the following two cases, *viz*.

  Case 1: For the difference in location we have identical covariance structures for both populations, $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}^*$ (*cf*. (3.23)), but with a difference in their respective means. We used $\boldsymbol{\mu}_2 = \mathbf{0}$ throughout, but set $\boldsymbol{\mu}_1 = c\mathbf{1}$ where $c = 2.5$ was used for normal data and $c = 1$ for lognormal data.

  Case 2: For the difference in dispersion we used equal locations, $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0}$. The covariance matrix $\mathbf{\Sigma}_2$ remained the same as before, but we inflated $\mathbf{\Sigma}_1$ by putting $\mathbf{\Sigma}_1 = \sigma^2\mathbf{\Sigma}^*$, with $\sigma = 4$.

The Gaussian kernel was used with $\gamma = 1/p$ for KFDA as well as for the hypersphere. The optimal λ-values for the above configurations are the same as those used for the large sample sizes in Section 3.7. A test data set of 10000 observations was used $(n_1 = n_2 = 5000)$ to obtain an estimate of the error rate. Similar to the other simulation studies in this thesis, the training data and test data were also standardized here. A hypothesis test for two independent normally distributed samples was also conducted to test whether the decrease in the average error rates was significant.

6.6.2 Discussion of the simulation results

Tables 6.1 to 6.4 contain the percentage decrease in the average error rates (P_e) for the two distributions and two types of atypical cases (*cf.* Section 5.4). These tables should be interpreted similar to Tables 5.2 to 5.5. However, one should keep in mind that the filter was applied in this study.

Again one should take note of the following in each table:

The underlined numbers represent the cases where the P_e for a criterion is equal to or greater than P_e in the last column (where the deliberately inserted atypical case is omitted). Thus for these configurations the criteria identified either the atypical case or a case more harmful to the KFD generalization performance.

The **bold face** numbers represent the cases where the difference between the average error rates, $\overline{R}_{test}^{[1]}$ and $\overline{R}_{test}^{[2]}$ (*cf.* (5.4)), is significant, according to the hypothesis test. For almost all the configurations where the populations differ with respect to dispersion there was a significant decrease in the average error rates. Where the populations differ with respect to location we observe substantially fewer significant cases and note that some of these cases represent an increase in the average error rate. It is therefore clear that the criteria performed appreciably better for the configurations of dispersion differences than for location differences.

The negative entries refer to an increase in the P_e. These entries appear in most of the configurations corresponding to location differences. However, many of them are insignificant. There were only six such entries in the configurations corresponding to dispersion differences.

Selecting the best criterion:

Similar to Table 5.6, we counted the number of test error reductions (the positive entries in Tables 6.1 to 6.4) for each criterion and split them into the two categories, *i.e.* significant and non significant reductions. These results are presented in Table 6.5 and should be interpreted similar to Table 5.6. However, note that Tables 6.1 to 6.4 represent a total of 48 configurations. In 44 of the 48 configurations, the criterion $R_{emp}^{(i)}$ corresponded to a reduction in the test error, of which 31 were significant. The criterion $v^{(i)}$ corresponds to 39 reductions in the test error of which 31 were significant. As was the case in Table 5.6, the alignment performed the worst with 31 test error reductions, of which only 14 were significant reductions. Again, the criteria were ranked from best to worst according to the simulation results. In this case the best criterion is $R_{emp}^{(i)}$ and the worst is $a^{(i)}$. However, we want to emphasize that the rankings in Table 6.5 do not guarantee that one single criterion will always be the superior one in the identification of the case that is most detrimental to the KFD generalization performance.

Comparison of the results with the filter and the results without the filter:

The sample size of 100 in Tables 6.1 to 6.4 (with filter) is the same size as the large samples (L) in Tables 5.2 to 5.5 (without filter). Thus, the configurations for this sample size enable us to make a comparison between the simulation results with the filter and without the filter. Overall, it seems as if the P_e is quite similar with or without the filter. Thus the criteria are successful in identifying cases that have a detrimental effect on the KFD generalization performance before or after the application of the filter. However, the advantages of using the filter is that only a subset of cases in the training data are evaluated, which in turn leads to a large reduction in the number of computations. We therefore conclude that our proposal of using the smallest enclosing hypersphere as a filter, before application of the criteria, is quite successful in reducing the number of computations.

TABLE 6.1: Percentage decrease in the test error; normal data; mislabelled case

Configurations			Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\overline{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	100	-0.1	-1.295	-1.295	-1.295	-1.295	-1.295	-1.295	-1.295	-1.295	-1.295	-1.295
	100	0	0.281	0.296	0.296	0.296	0.292	0.284	0.284	0.296	0.262	0.262
	100	0.7	0.002	0	-0.021	-0.021	-0.036	-0.003	-0.027	-0.02	0.003	0.043
	200	-0.1	0.188	0.188	0.188	0.188	0.188	0.188	0.188	0.188	0.188	0.188
	200	0	0.106	0.091	0.098	0.091	0.098	0.087	0.091	0.091	-0.038	0.117
	200	0.7	0.001	-0.002	-0.004	0	-0.008	0.013	-0.005	-0.003	-0.025	0.004
Dispersion difference	100	-0.1	**3.205**	0.649	**3.497**	**2.055**	**4.565**	0.633	1.405	**4.412**	**1.332**	**4.55**
	100	0	**3.267**	0.735	**3.842**	**2.552**	**4.967**	0.549	1.716	**4.852**	**1.438**	**4.946**
	100	0.7	**2.163**	0.47	**2.557**	**1.907**	**3.029**	**1.328**	0.768	**3.258**	**1.491**	**3.931**
	200	-0.1	**1.13**	0.484	**1.418**	**1.263**	**1.568**	0.409	**0.971**	**1.534**	**1.009**	**1.558**
	200	0	**1.333**	0.52	**1.506**	**1.366**	**1.662**	0.417	**1.08**	**1.613**	**1.148**	**1.644**
	200	0.7	**1.013**	-0.365	**1.172**	**0.909**	**1.301**	-0.774	-0.319	0.024	**0.938**	**1.82**

TABLE 6.2: Percentage decrease in the test error; lognormal data; mislabelled case

	Configurations		Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\bar{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\Vert\mathbf{v}\Vert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	100	-0.1	**4.463**	**4.178**	**2.657**	**2.308**	**2.606**	**-3.779**	1.784	1.786	1.239	**4.853**
	100	0	**2.078**	1.357	0.687	0.627	1.473	**-3.112**	0.215	-0.671	0.152	**2.645**
	100	0.7	0.151	-0.828	-0.14	-0.101	**0.926**	-0.256	-0.73	-0.636	-0.389	0.599
	200	-0.1	1.568	1.195	0.674	0.711	0.733	**-2.834**	0.53	-0.216	0.478	**1.882**
	200	0	0.048	-0.353	-0.31	-0.307	-0.619	**-1.581**	-0.516	**-1.289**	-0.658	0.62
	200	0.7	0.289	-0.101	0.558	0.562	0.245	0.549	0.162	0.244	0.053	0.128
Dispersion difference	100	-0.1	**6.827**	**4.324**	**4.419**	**4.461**	**2.035**	**3.394**	**4.721**	**5.953**	**3.799**	**7.43**
	100	0	**2.841**	**1.527**	**2.985**	**3.077**	**2.453**	**3.49**	**2.092**	**3.574**	**1.932**	**3.835**
	100	0.7	**3.822**	1.254	**5.045**	**4.955**	**5.332**	**5.084**	**2.715**	**4.681**	**3.523**	**1.569**
	200	-0.1	**3.41**	1.616	**4.506**	**4.558**	**2.614**	**4.857**	**3.658**	**5.123**	**2.368**	**3.185**
	200	0	**1.864**	1.026	**3.11**	**3.283**	**2.596**	**3.722**	**2.99**	**3.536**	**1.981**	**1.703**
	200	0.7	**2.791**	0.088	**3.479**	**3.467**	**3.679**	**3.562**	0.926	**3.219**	**2.744**	0.904

TABLE 6.3: Percentage decrease in the test error; normal data; moderate outlier

	Configurations		Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\bar{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\Vert\mathbf{v}\Vert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	100	-0.1	1.54	1.54	1.718	1.54	1.422	1.54	1.659	1.659	1.54	1.54
	100	0	0.059	0.052	0.022	0.044	-0.041	0.019	0	0.022	0.026	0.044
	100	0.7	-0.004	-0.027	-0.05	-0.04	-0.065	0.003	-0.057	-0.049	-0.043	0.015
	200	-0.1	-0.377	-0.377	-0.377	-0.377	-0.063	-0.377	-0.377	-0.377	-0.377	-0.377
	200	0	-0.053	-0.038	-0.061	-0.049	-0.084	-0.038	-0.057	-0.065	-0.148	-0.053
	200	0.7	0.002	-0.001	-0.014	-0.009	-0.031	-0.002	-0.017	-0.015	-0.02	0.012
Dispersion difference	100	-0.1	**2.321**	0.521	**2.547**	**1.565**	**3.62**	0.553	**1.047**	**3.544**	0.818	**3.605**
	100	0	**2.346**	0.565	**2.713**	**1.757**	**3.501**	0.612	**1.194**	**3.439**	**0.994**	**3.471**
	100	0.7	**2.12**	0.444	**2.6**	**1.788**	**3.728**	**1.206**	0.461	**3.916**	**1.152**	**4.428**
	200	-0.1	**0.843**	0.402	**1.062**	**0.957**	**1.109**	0.461	**0.745**	**1.104**	**0.794**	**1.119**
	200	0	**0.906**	0.453	**1.182**	**1.042**	**1.176**	0.39	**0.786**	**1.166**	**0.852**	**1.17**
	200	0.7	**1.005**	-0.484	**1.219**	0.724	**1.795**	-0.795	-0.641	0.127	**0.792**	**2.164**

TABLE 6.4: Percentage decrease in the test error; lognormal data; moderate outlier

Configurations			Omitting case identified by criterion									Omitting atypical case
	sample size	ρ	$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\overline{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$	
Location difference	100	-0.1	**7.244**	**7.095**	-0.935	-1.37	**5.069**	**-2.544**	-1.503	**-2.162**	-0.964	**7.432**
	100	0	**7.525**	**6.181**	**1.934**	0.414	**6.162**	**-1.577**	-0.276	0.917	-0.046	**8.979**
	100	0.7	**1.944**	-0.058	**1.199**	0.849	**2.368**	-0.292	0.382	0.083	0.234	**4.915**
	200	-0.1	**5.252**	**4.574**	-0.613	-0.923	**4.049**	**-1.951**	-0.903	**-1.983**	-0.667	**5.672**
	200	0	**4.229**	**1.995**	**1.579**	0.771	**3.778**	-1.108	0.708	0.264	0.368	**5.853**
	200	0.7	0.506	-0.252	0.509	0.433	0.405	0.409	0.166	0.219	0.05	2.002
Dispersion difference	100	-0.1	**8.504**	**6.308**	**7.638**	**6.983**	**3.251**	**3.915**	**5.188**	**9.307**	**5.893**	**10.618**
	100	0	**3.019**	**1.594**	**2.594**	**2.798**	**2.598**	**2.963**	1.104	**3.982**	1.21	**5.267**
	100	0.7	**3.762**	0.952	**3.966**	**3.73**	**4.607**	**3.843**	1.312	**3.575**	**3.215**	**3.142**
	200	-0.1	**4.885**	**2.46**	**5.152**	**5.127**	**2.696**	**4.684**	**3.918**	**6.194**	**3.113**	**5.296**
	200	0	**2.139**	**1.191**	**3.23**	**3.364**	**2.738**	**3.611**	**2.889**	**3.709**	**1.957**	**2.555**
	200	0.7	**2.898**	-0.212	**3.089**	**3.082**	**3.184**	**2.889**	0.261	**3.087**	**2.67**	**1.99**

TABLE 6.5: A comparison of the performance of the nine criteria

Characteristic		$R_{emp}^{(i)}$	$\bar{d}^{(i)}$	$\overline{m}^{(i)}$	$r^{(i)}$	$v^{(i)}$	$a^{(i)}$	$\lVert\mathbf{v}\rVert^{(i)}$	$\bar{f}^{(i)}$	$\bar{h}^{(i)}$
# of test error reductions	*signif.*	31	11	28	24	31	14	14	22	22
	insignif.	13	23	9	14	8	17	20	13	14
	Total	**44**	**34**	**37**	**38**	**39**	**31**	**34**	**35**	**36**
	*Rank**	*1*	*8*	*4*	*3*	*2*	*9*	*7*	*6*	*5*
# of test error inflations	*signif.*	0	0	0	0	0	7	0	3	0
	insignif.	4	14	11	10	9	10	14	10	12
	Total	**4**	**14**	**11**	**10**	**9**	**17**	**14**	**13**	**12**

*Ranking of the criteria: 1 = the best; 9 = the worst

6.7 Identifying influential cases in KFDA after the application of a filter: application to real-world data sets

We applied the filter procedure to the Flea beetle data and the Swiss bank note data (*cf.* Section 5.5). The following two-step procedure was performed on each of these data sets respectively:

- The smallest enclosing hypersphere was obtained for each of the two classes in the data set using the Gaussian kernel with $\gamma = 1/p$. The support vectors for each class was obtained (the subset of cases from the training data).
- We then performed KFDA on all the data using the Gaussian kernel with $\gamma = 1/p$ and a hyperparameter of $\lambda = 0.1$. The subset was then evaluated using the leave-one-out criteria. The results for the two data sets are given in the following subsections.

6.7.1 Flea beetle data

This data set consisted of 39 cases having 19 cases in class 1 and 20 cases in class 2. The number of support vectors was 12 for each class respectively. Thus the subset for the Flea beetle data that had to be evaluated comprised 24 cases (62% of the training data).

In Figure 6.12 the index plot of the α^*-values show us which observations formed the subset. Among this subset are cases 27 and 38. Similar to what was found in Section 5.5.1, the leave-one-out criteria identified these cases as the most influential. The criteria $\bar{d}^{(i)}$, $\overline{m}^{(i)}$, $r^{(i)}$, $\|\mathbf{v}\|^{(i)}$, $\bar{f}^{(i)}$ and $\bar{h}^{(i)}$ identified case 27, while $v^{(i)}$ and $a^{(i)}$ identified case 38. The $R_{emp}^{(i)}$ criterion did not perform well on this small data set since it did not identify a single most influential case. See Section 5.5.1 for the results on the KFD generalization performance without these cases.

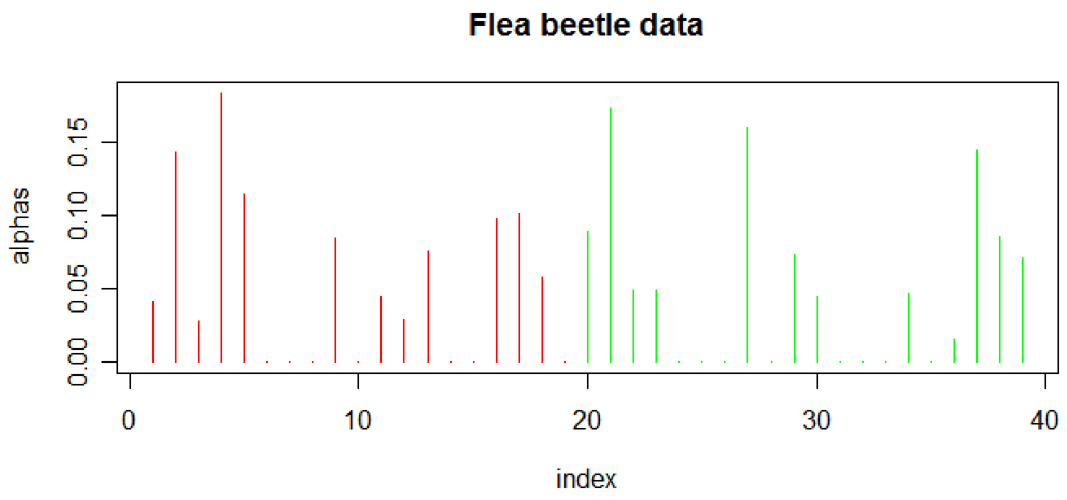
6.7.2 Swiss bank note data

This two-class data set consisted of 200 observations, *i.e.* 100 cases in each class. For class 1 we obtained 29 support vectors and for class 2 we obtained 31 support vectors. Thus, a subset of size 60 (30% of the training data) was obtained. Figure 6.12 contains an index plot of the α^*-values indicating which cases form part of the subset. Among this subset is case 70 and similar to the results is Section 5.5.2, this case was identified as the most influential case by all the criteria. Refer to Section 5.5.2 for further discussions on the KFD generalization performance without these cases.

Based on the examples above, we conclude the results of the leave-one-out criteria are the same with or without the filter. However, by using the hypersphere as a filter, a lot of computing time is saved since the evaluation is done on a subset of the training data. The results also show that the smallest enclosing hypersphere is not very effective for small data sets. For the Flea beetle data (a small data set), 62% of the training data formed the subset. For the Swiss bank note data (a much larger data set), 30% of the training data was selected for the subset.

6.8 Concluding remarks

This chapter focused mostly on a discussion of the smallest enclosing hypersphere and the corresponding support region. We illustrated with examples how the hypersphere in



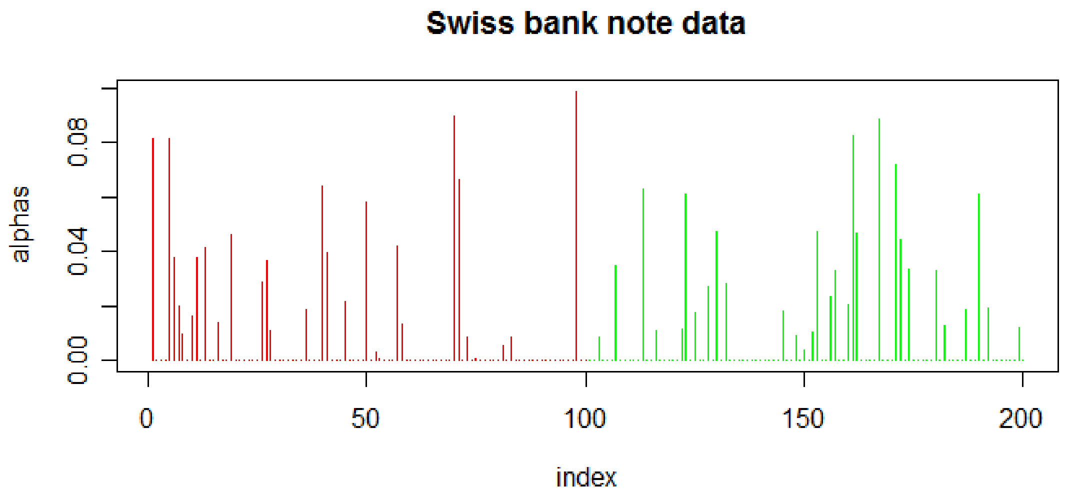


FIGURE 6.12: *Index plots of the alphas for the Flea beetle and Swiss bank note data. The spikes represent the cases with non-zero alphas, i.e. support vectors. The Flea beetle data have 12 support vectors per class (a subset of size 24). The Swiss bank note data have 29 and 31 support vectors for class 1 and 2 respectively (a subset of size 60).*

feature space corresponds to a support region in input space. It was demonstrated that the hypersphere can be constructed by using only the support vectors, *i.e.* the observations that fall on the surface of the hypersphere. The aim of this chapter was to obtain a procedure to reduce the number of computations when calculating the nine criteria on a leave-one-out basis. In Section 6.5 we argued that the smallest enclosing hypersphere should be used as a filter to obtain a subset of observations, *i.e.* the support vectors from the training data. We then proposed that only this subset should be evaluated using the nine criteria proposed in Chapter 5. Thus, a two-step procedure was given in this chapter where:

- the first step entails constructing the smallest enclosing hypersphere in feature space for each class to find a subset of cases in each class, and
- the second step entails evaluating this subset, using the criteria, on a leave-one-out basis.

The simulation results showed that this two-step procedure worked effectively. Using the hypersphere as a filter reduced the number of computations, since only a few cases needed to be evaluated. Comparing the simulation results with and without the filter, we found that the decrease in the average error rates for these two situations is quite similar. The practical applications also showed promising results, proving that the application of the hypersphere as a filter works effectively on real-world data.

Avenues for further research:

- The analysis in this chapter was carried out only for the Gaussian kernel. Experiments should also be conducted with other kernel functions when applying the hypersphere.
- We have also seen that by specifying different γ-values for the Gaussian kernel the number of support vectors also changes. Selecting the appropriate γ-value to construct the hypersphere also requires further investigation. Similarly, one can also do this investigation for hyperparameters of other kernel functions.

CHAPTER 7

GENERAL CONCLUSION

"*What we know is not much. What we do not know is immense.*"

- Pierre-Simon Laplace

This thesis started with a detailed explanation of KFDA in Chapter 2. It was demonstrated that the KFD classifier is a non-linear generalization of the FLD classifier. Several classification methods related to KFDA were discussed as well as some advantages and disadvantages of KFDA. The rest of the thesis focused mainly on the effect of atypical cases on KFDA and the development of criteria to identify such cases. The contributions made in this thesis can be summarized as follows.

We studied the effect of atypical cases on various aspects of KFDA in Chapter 3. The purpose of this study was to determine whether atypical cases have an effect on KFDA, specifically on the KFD classifier's generalization performance. The simulation results in Section 3.7 showed that atypical cases do have a detrimental effect on the KFD classifier's generalization performance. The study also revealed that certain other aspects (*cf*. Section 3.3) of KFDA are also affected by atypical cases. This leads to the question whether these aspects can be used to develop criteria to identify cases having a detrimental effect on the KFD generalization performance.

In Chapter 4 we discussed the estimation of posterior probabilities in KFDA. We showed how certain output from KFDA (the projections and the discriminant scores) can be used to estimate posterior probabilities. In a simulation study we also showed that the projections are not always normally distributed, as stated in Schölkopf and Smola (2002, *p*.464). Two classifiers, making use of posterior probabilities to classify observations, were also derived in this chapter. These classifiers showed similar classification results to the classifier given in Chapter 2 (*cf*. (2.33)). However, the classifiers making use of the posterior probabilities allow us to obtain additional information. Posterior probabilities can be used to assign a confidence to the final classification.

Chapter 5 dealt with the detection of cases that are detrimental to the KFD classifier's generalization performance. In this chapter we proposed nine criteria that can be used to detect such cases. To study the effectiveness of the proposed criteria we first performed a simulation study in which the average error rate was used to evaluate the effect of cases on the KFD generalization performance. The simulation study revealed that omission of the cases that were identified by the criteria, resulted in a decrease in the error rate in a majority of the configurations studied. Thus, it showed that the criteria have the ability to detect cases that have a detrimental effect on the KFD classifier. Secondly, the criteria were applied to two real-world data sets. The cross-validation error rates of the data sets were also calculated and we saw that the error rate decreased when the cases identified by the criteria were omitted. Both the simulation study and the real-world data application showed that the criteria do have merit for the detection of cases that are detrimental to the KFD generalization performance.

In Chapter 6 a filter or pre-screening method was proposed. Because of the computational challenge when calculating the criteria on a leave-one-out basis, we propose that the smallest enclosing hypersphere should be used to filter the training data. A subset of cases (support vectors) was obtained and only these cases need to be evaluated by each criterion. A simulation study was also performed to test the effectiveness of the filter. The results showed that the filter works well in identifying the cases that are detrimental to the KFD generalization performance. Evaluating only a small subset of the training data using the criteria, saves a lot of computing time. This can be quite useful in applications involving large data sets.

Several remarks and open problems for further research are given at the end of each chapter. Even though the studies in this thesis were quite extensive, these problems indicate that there are still many investigations that can be done. However, the studies in this thesis revealed that KFDA is sensitive to atypical cases and as a result, these cases should be identified and investigated more thoroughly. Such cases may reveal important characteristics about the training data and it may affect model aspects (such as the KFD generalization performance). As shown in the results of the simulation studies and practical applications, the criteria and filtering method proposed in this thesis are to a great extent successful in this regard. It is also worth mentioning that the proposals in this thesis may also be extended to other kernel-based classification methods.

--------------- ÒÑÒÑ Ω Ð Ï Ð Ï ---------------

BIBLIOGRAPHY

1. Aitchison, J. and Brown J. (1957), *The lognormal distribution*, Cambridge: Cambridge University Press.
2. Aizerman, M., Braverman, E. and Rozonoer, L. (1964), Theoretical foundations of the potential function method in pattern recognition learning, *Automation and Remote Control*, **25**: 821-837.
3. Anderson, T.W. (2003), *An introduction to multivariate statistical analysis*, 3rd edition, New York: John Wiley.
4. Aronszajn, N. (1950), Theory of reproducing kernels, *Transactions of the American Mathematical Society*, **68**: 337-404.
5. Baudat, G. and Anouar, F. (2000), Generalized discriminant analysis using a kernel approach, *Neural Computation*, **12**(10): 2385-2404.
6. Bertsekas, D.P. (1995), *Non-linear Programming*, Belmont, MA: Athena Scientific.
7. Boser, B.E., Guyon, I.M. and Vapnik, N.V. (1992), A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, *pp*.144-152.
8. Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984), *Classification and Regression Trees*, Wadsworth.
9. Campbell, N.A. (1978), The influence function as an aid in outlier detection in discriminant analysis, *Applied Statistics*, **27**: 251-258.
10. Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S. (2002), Choosing multiple parameters for support vector machines, *Machine Learning*, **46**(1): 131-159.
11. Cook, R.D. (1977), Detection of influential observations in linear regression, *Technometrics*, **19**: 15-18.
12. Cristianini, N., Campbell, C., and Shawe-Taylor, J. (1998), Dynamically adapting kernels in support vector machines. *Advances in Neural Information Processing Systems*, **11**: 204 – 210.
13. Cristianini, N., Shawe-Taylor, J., Elisseeff, A. and Kandola, J. (2002), On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, **14**:367–373.

14. Critchley, F. and Vitiello, C. (1991), The influence of observations on misclassification probability estimates in linear discriminant analysis, *Biometrika*, **78**: 677-690.
15. Debnath, L and Mikusiński, P (1999), An introduction to Hilbert Spaces with Applications, San Diego: Academic Press.
16. Fisher, R.A. (1936), The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, **7**:179-188.
17. Flury , B.W. (1997), *A first course in multivariate statistics*, New York: Springer-Verlag.
18. Flury, B.W. and Riedwyl, H. (1988), *Multivariate Statistics: A Practical Approach*. London: Chapman and Hall.
19. Fung, W.K. (1995a), Detecting influential observations for estimated probabilities in multiple discriminant analysis, *Computational Statistics and Data Analysis*, **20**: 557-568.
20. Fung, W.K. (1995b), Diagnostics in linear discriminant analysis, *Journal of the American Statistical Association*, **90**: 952-956.
21. Fung, W.K. (1995c), Influence on classification and probability of misclassification, *Indian Journal of Statistics*, **57**: 377-384.
22. Fung, W.K. (1996), Diagnosing influential observations in quadratic discriminant analysis, *Biometrics*, **52**: 1235-1241.
23. Girolami, M. (2002), Mercer Kernel-Based Clustering in Feature Space, *I.E.E.E. Transactions on Neural Networks*, **13**(3): 780-784.
24. Hamers, B. (2004), *Kernel models for large scale applications*, PhD Thesis, Katolieke Universiteit Leuven.
25. Härdle, W. and Simar, L. (2003), *Applied multivariate statistical analysis*, New York: Springer.
26. Hastie, T., Tibshirani, R. and Friedman, J. (2001), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York: Springer.
27. Herbrich, R. (2001), *Learning Kernel Classifiers: Theory and Algorithms*, London: MIT Press.

28. Johnson, M.E. (1987), *Multivariate Statistical Simulation*, New York: Wiley & Sons Inc.
29. Johnson, R.A. and Wichern, D.W. (2007), *Applied Multivariate Statistical Analysis*, 6rd edition, London: Prentice-Hall International Inc.
30. Johnson, W. (1987), The detection of influential observations for allocation, separation and the determination of probabilities in a Bayesian framework, *Journal of Business Economics and Statistics*, **5**: 369-381.
31. Joossens, K. (2006), *Robust discriminant analysis,* PhD thesis, Katolieke Universiteit Leuven.
32. Jordan, M.I. (2004), Lectures notes on “Reproducing kernel Hilbert Spaces” from http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/ .
33. Karush, W. (1939), *Minima of functions of several variables with inequalities as side constraints*, Master’s thesis, Dept. of Mathematics, Univ. of Chicago.
34. Keerthi, S.S. (2002), Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms, *IEEE Transactions on Neural Networks*, **13**(5): 1225-1229.
35. Keerthi, S.S. and Lin, C.J. (2003), Asymptotic behaviors of support vector machines with Gaussian kernel, *Neural Computation*, **15**: 1667-1689.
36. Kolmogorov, A.N. (1941), Stationary sequences in Hilbert spaces, *Moscow University Mathematics*, **2**: 1-40.
37. Kuhn, H.W. and Tucker, A.W. (1951), Non-linear programming, *In Proc. 2’nd Berkeley Symposium on Mathematical Statistics and Probabilistics*, *pp*. 481-492, Berkeley: Univ. of California Press.
38. Kuss, M. and Graepel, T. (2003), The geometry of kernel canonical correlation analysis, *Technical report no. 108*, Max Planck institute for biological cybernetics.
39. Law, A. and Kelton, W. (2000), *Simulation modeling and analysis*, 3rd edition, Boston: McGraw Hill.
40. Lee, Y., Lin, Y. and Wahba, G. (2001), Multicategory support vector machines, *Technical report no. 1043*, Department of Statistics, University of Wisconsin.
41. Léger, C. and Altman, N. (1993), Assessing influence in variable selection problems, *Journal of the American Statistical Association*, **88**: 547-556.

42. Louw, N. and Steel, S.J. (2005), A review of kernel Fisher discriminant analysis for statistical classification, *South African Statistical Journal*, **39**(1): 1-21.
43. Mangasarian, O.L. (1997), Mathematical Programming in data mining, *Data Mining and Knowledge Discovery*, **42**: 183-201.
44. Mercer, J. (1909), Functions of positive and negative type and their connection with the theory of integral equations, *Philos. Trans. Roy. Soc. London*, **209**: 415-446.
45. Mika, S., Rätsch, G., Weston, J., Schölkopf, B. and Müller, K.-R. (1999), Fisher discriminant analysis with kernels, In Y.-H. Hu, J. Larsen, E. Wilson and S. Douglas, editors, *Neural Networks for Signal Processing IX*, IEEE, *pp*.41-48.
46. Mika, S. (2002), *Kernel Fisher Discriminants*, PhD thesis, Technischen Universität Berlin.
47. Moreno-Roldán, D., Muñoz-Pichardo, J.M. and Enguix-González, A. (2007), Influence diagnostics in multiple discriminant analysis, *Test*, **16**: 172-187.
48. Navarrete, P. and del Solar, J.R. (2002), On the generalization of kernel machines, *Pattern Recognition with SVM*, In S. Whan-Lee and A. Verri, editors, *pp*. 24-39.
49. Platt, J. (2000), Probabilities for SV machines, In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, *pp*.61-74.
50. Pregibon, D. (1981), Logistic regression diagnostics, *Annals of Statistics*, **9**: 705-724.
51. Schölkopf, B. (1997), *Support vector learning*, PhD thesis, Technischen Universität Berlin.
52. Schölkopf, B. and Smola, A.J. (2002), *Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, Cambridge, MA: MIT press.
53. Shapiro, S.S. and Wilk, M.B. (1965), An analysis of variance test for normality (complete samples), *Biometrika*, **52**: 591-611.
54. Shashua, A. (1999), On the relationship between the support vector machine for classification and sparsified Fisher's linear discriminant, *Neural Processing Letters*, **9**(2): 129-139.
55. Shawe-Taylor, J. and Cristianini, N. (2004), *Kernel Methods for Pattern Analysis*, Cambridge: Cambridge University Press.

56. Steel, S.J. and Louw, N. (2001), Variable selection in discriminant analysis: measuring the influence of individual cases, *Computational Statistics and Data Analysis*, **37**: 249-260.
57. Steel, S.J. and Louw, N. (2002), An integrated approach to outlier identification and variable selection in discriminant analysis, *South African Statistical Journal*, **36:** 129-151.
58. Steel, S.J. and Uys, D.W. (2006), Influential data cases when the C_p criterion is used for variable selection in multiple linear regression, *Computational Statistics and Data Analysis*, **50**: 1840-1854.
59. Suykens, J.A.K. and Vandewalle, J. (1999), Least squares support vector machine classifier, *Neural Processing Letters*, **9**(3): 293-300.
60. Tax, D.M.J. and Duin, R.P.W. (1999), Support vector domain description, *Pattern Recognition Letters*, **20**: 1191-1199.
61. Tax, D.M.J. (2001), *One-class classification*, PhD thesis, Technischen Universität, Berlin.
62. Tipping, T.E. (2000), The relevance vector machine, In S.A. Solla, T.K. Leen and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, **12**: 652-658.
63. Vapnik, V.N. (1995), *The Nature of Statistical Learning Theory*, New York: Springer Verlag.
64. Wabha, G. (1973), Convergence rates of certain approximate solutions to first kind integral equations, *Journal of Approximation Theory*, **7**: 167-185.
65. Wang, W., Xu, Z., Lu, W., and Zhang, X. (2003), Determination of the spread parameter in the Gaussian kernel for classification and regression, *Neurocomputing*, **55**: 643-663.
66. Weston, J. (1999), *Extensions to the support vector method*, PhD thesis, University of London.

APPENDIX

THE APPENDIX CONTAINS THE FOLLOWING PROGRAMS

A. KERNEL FISHER DISCRIMINANT ANALYSIS

```
kernel.Fisher<-function(data){

#
# Creating a 2 dimensional scatterplot with KFDA decision boundary and
# classification regions
#
# ---- Specifying the KFDA hyperparameters
#
      opt.gam<-0.5
      opt.lam<-0.1

      class1<-matrix(data[data[,1]==1],ncol=ncol(data))[,2:ncol(data)]
      class2<-matrix(data[data[,1]==-1],ncol=ncol(data))[,2:ncol(data)]
      n1<-nrow(matrix(data[data[,1]==1],ncol=ncol(data)))
      n2<-nrow(matrix(data[data[,1]==-1],ncol=ncol(data)))
      n<-n1+n2
      p<-ncol(class1)

      Inp1<-rbind(as.matrix(rep(1,n1)),as.matrix(rep(0,n2)))
      Inm1<-rbind(as.matrix(rep(0,n1)),as.matrix(rep(1,n2)))
      In<-matrix(rep(0,(n*n)),ncol=n,nrow=n)
      diag(In)<-1
#
# ---- Creating the Gram matrix using a Gaussian kernel function
#
      G<-exp(((-1)*opt.gam)*(as.matrix(dist(data[,- 1],
           method="euclidean")^2)))
      kp11<-(1/n1)*(G%*%Inp1)
      km11<-(1/n2)*(G%*%Inm1)
      N<-((1/n)*((G%*%G)-(n1*(kp11%*%t(kp11)))-
            (n2*(km11%*%t(km11)))))+(opt.lam*In)
#
# ---- The optimal alpha vector and thresshold b
#
      alpha<-solve(N)%*%(kp11-km11)
      b<-(-1/2)*t(kp11-km11)%*%solve(N)%*%(kp11+km11)-log(n2/n1)
#
# ---- The following instructions is used to obtain the two class
#      classification regions and the decision boundary if p=2
#
      xmin<-min(data[,2])
      xmax<-max(data[,2])
      ymin<-min(data[,3])
      ymax<-max(data[,3])

      grid.data<-
          expand.grid(seq(xmin,xmax,length=100),seq(ymin,ymax,length=100))
      aa<-as.matrix(grid.data[,1])
      bb<-as.matrix(grid.data[,2])
      cc<-matrix(rep(1,nrow(aa)),ncol=1)
      grid.data<-cbind(cc,aa,bb)
      ng<-nrow(grid.data)
      signgrid<-matrix(rep(0,ng),nrow=ng,ncol=1)

      for (i in 1:ng){
            combine.grid<-rbind(data,grid.data[i,])
            Gng<-exp(((-1)*opt.gam)*(as.matrix(dist(combine.grid[,-1],
                method="euclidean")^2)))
            vectorzg<-as.matrix(Gng[-nrow(Gng),ncol(Gng)])
```

```
              signgrid[i,]<-sign((t(alpha)%*%vectorzg)+b)
       }

       xp<-seq(min(data[,2]),max(data[,2]),length=100)
       np<-length(xp)
       yp<-seq(min(data[,3]),max(data[,3]),length=100)
       ymat<-factor(data[,1])
       zp<-matrix(signgrid,ncol=1)
#
# ---- Plotting the KFDA decision boundary
#
       plot(data[,2],data[,3],type="n",xlab="x1",ylab="x2")
       par(new=T)
       contour(xp,yp,matrix(zp,np),add=T,drawlabels=F,levels=0,lty=2,lwd=2,
       col="blue")
#
# ---- Obtaining the classification regions
#
       region.data<-cbind(signgrid,grid.data[,2:3])
       region.red<-matrix(region.data[region.data[,1]==1],ncol=3)
       region.green<-matrix(region.data[region.data[,1]==-1],ncol=3)

       par(new=T)
       plot(region.red[,2:3],xlim=c(min(data[,2]),max(data[,2])),
         ylim=c(min(data[,3]),max(data[,3])),col="green",
         xlab="x1",ylab="x2",pch=".")
       points(region.green[,2:3],col="red",pch=".")
#
# ---  Plotting the training data
#
       par(new=T)
       plot(class1,xlim=c(min(data[,2]),max(data[,2])),ylim=c(min(data[,3]),
         max(data[,3])),col="green",xlab="x1",ylab="x2",pch=15,cex=1.25)
       points(class2,col="red",pch=16,cex=1.25)

       list(alpha=alpha,b=b)

       }
```

B. SUPPORT VECTOR MACHINE

```
support.vector<-function(data){

#
# Creating a 2 dimensional scatterplot with SVM decision boundary and
# classification regions

      library(e1071)
#
# ---- Specifying the SVM hyperparameters
#
      costpar<-0.5
      gam<-0.5

      class1<-matrix(data[data[,1]==1],ncol=3)
      class2<-matrix(data[data[,1]==-1],ncol=3)
      n1<-nrow(as.matrix(data[,1][data[,1]==1]))
      n2<-nrow(as.matrix(data[,1][data[,1]==-1]))
      n<-n1+n2
      p<-ncol(data[,-1])
#
# ---- Performing the SVM
#
      svm1<-svm(as.factor(data[,1])~.,data=data[,-1],scale=TRUE,shrink=TRUE,
      kernel="radial",cost=costpar,gamma=gam,cachesize=500,tolerance=0.0001)
#
# ---- The following instructions is used to obtain the two class
#      classification regions and the decision boundary if p=2
#
      xmin<-min(data[,2])
      xmax<-max(data[,2])
      ymin<-min(data[,3])
      ymax<-max(data[,3])

      grid.data<-
        expand.grid(seq(xmin,xmax,length=100),seq(ymin,ymax,length=100))
      aa<-as.matrix(grid.data[,1])
      bb<-as.matrix(grid.data[,2])
      cc<-matrix(rep(1,nrow(aa)),ncol=1)
      grid.data<-cbind(cc,aa,bb)

      pred2<-as.matrix(predict(svm1,grid.data[,-1],type="class"))
      newdat<-cbind(as.numeric(pred2),grid.data)

      zp<-matrix(as.numeric(pred2),ncol=1)
      xp<-seq(min(data[,2]),max(data[,2]),length=100)
      np<-length(xp)
      yp<-seq(min(data[,3]),max(data[,3]),length=100)
      ymat<-factor(data[,1])
#
# ---- Plotting the SVM decision boundary
#
      plot(data[,2],data[,3],type="n",xlab="x1",ylab="x2")
      par(new=T)
      contour(xp,yp,matrix(zp,np),add=T,drawlabels=F,levels=0,lty=2,
        col="blue",lwd=2)
#
# ---- Obtaining the classification regions
#
      region.red<-matrix(newdat[newdat[,1]==1],ncol=4)
      region.green<-matrix(newdat[newdat[,1]==-1],ncol=4)
      par(new=T)
      plot(region.red[,3:4],xlim=c(min(newdat[,3]),max(newdat[,3])),
        ylim=c(min(newdat[,4]),max(newdat[,4])),col="green",xlab="x1",
        ylab="x2",pch=".")
      points(region.green[,3:4],col="red",pch=".")
#
# ---  Plotting the training data
#
      par(new=T)
      plot(class1[,2:3],xlim=c(min(newdat[,3]),max(newdat[,3])),
        ylim=c(min(newdat[,4]),max(newdat[,4])),col="green",xlab="x1",
        ylab="x2",pch=15)
      points(class2[,2:3],col="red",pch=16)

      }
```

C. REGULARIZED KERNEL FISHER DISCRIMINANT ANALYSIS

```
regKFDA<-function (data) {

#
# Creating a 2 dimensional scatterplot with regKFDA decision boundary and
# classification regions

#
# ---- Specifying the regKFDA hyperparameters
#
      opt.gam<-0.5
      opt.lam<-0.1

      class1<-matrix(data[data[,1]==1],ncol=ncol(data))[,2:ncol(data)]
      class2<-matrix(data[data[,1]==-1],ncol=ncol(data))[,2:ncol(data)]
      n1<-nrow(matrix(data[data[,1]==1],ncol=ncol(data)))
      n2<-nrow(matrix(data[data[,1]==-1],ncol=ncol(data)))
      n<-n1+n2
      p<-ncol(class1)

      d1<-matrix(rep(2*n2/n,n1),ncol=1);d2<-matrix(rep(2*n1/n,n2),ncol=1)
      d<-rbind(d1,d2)
      D<-matrix(rep(0,n*n),ncol=n)
      diag(D)<-d

      Cp1<-matrix(rep(0,n*n),ncol=n)
      Cp1[1:n1,1:n1]<-matrix(rep(((2*n2)/(n*n1)),n1*n1),ncol=n1)

      Cm1<-matrix(rep(0,n*n),ncol=n)
      Cm1[(n1+1):n,(n1+1):n]<-matrix(rep(((2*n1)/(n*n2)),n2*n2),ncol=n2)

      B<-D-Cp1-Cm1
#
# ---- Obtaining the Gram matrix using the Gaussian kernel function
#
      G<-exp(((-1)*opt.gam)*(as.matrix(dist(data[,1],
         method="euclidean")^2)))

      y<-data[,1]

      I<-matrix(rep(0,n*n),ncol=n);diag(I)<-1

      t1<-matrix(rep(1/n1,n1),ncol=1);t2<-matrix(rep(1/n2,n2),ncol=1)
      t<-rbind(t1,t2)
#
# ---- The optimal alpha vector and the thresshold b
#
      alpha<-solve((B%*%G)+(opt.lam*I))%*%y
      b<--0.5*t(alpha)%*%G%*%t
#
# ---- The following instructions is used to obtain the two class
#      classification regions and the decision boundary if p=2
#
      xmin<-min(data[,2])
      xmax<-max(data[,2])
      ymin<-min(data[,3])
      ymax<-max(data[,3])

      grid.data<-
        expand.grid(seq(xmin,xmax,length=100),seq(ymin,ymax,length=100))
```

```
       aa<-as.matrix(grid.data[,1])
       bb<-as.matrix(grid.data[,2])
       cc<-matrix(rep(1,nrow(aa)),ncol=1)
       grid.data<-cbind(cc,aa,bb)
       ng<-nrow(grid.data)
       signgrid<-matrix(rep(0,ng),nrow=ng,ncol=1)

       for (i in 1:ng){
              combine.grid<-rbind(data,grid.data[i,])
              Gng<-exp(((-1)*opt.gam)*(as.matrix(dist(combine.grid[,-
              1],method="euclidean")^2)))
              vectorzg<-as.matrix(Gng[-nrow(Gng),ncol(Gng)])
              signgrid[i,]<-sign((t(alpha)%*%vectorzg)+b)
       }

       xp<-seq(min(data[,2]),max(data[,2]),length=100)
       np<-length(xp)
       yp<-seq(min(data[,3]),max(data[,3]),length=100)
       ymat<-factor(data[,1])
       zp<-matrix(signgrid,ncol=1)
#
# ---- Plotting the regKFDA decision boundary
#
       plot(data[,2],data[,3],type="n",xlab="x1",ylab="x2")
       par(new=T)
       contour(xp,yp,matrix(zp,np),add=T,drawlabels=F,levels=0,lty=2,lwd=2,
       col="blue")

       region.data<-cbind(signgrid,grid.data[,2:3])

       region.green<-matrix(region.data[region.data[,1]==1],ncol=3)
       region.red<-matrix(region.data[region.data[,1]==-1],ncol=3)
#
# ---- Plotting the classification regions
#
       par(new=T)
       plot(region.green[,2:3],xlim=c(min(data[,2]),max(data[,2])),ylim=c(min
          (data[,3]),max(data[,3])),col="green",xlab="x1",ylab="x2",pch=".")
       points(region.red[,2:3],col="red",pch=".")
#
# ---- Plotting the training data
#
       par(new=T)
       plot(class1,xlim=c(min(data[,2]),max(data[,2])),ylim=c(min(data[,3]),
          max(data[,3])),col="green",xlab="x1",ylab="x2",pch=17)
       points(class2,col="red",pch=16)

       }
```

D. KFDA USING POSTERIOR PROBABILITIES OBTAINED FROM BAYES' RULE

```
kernel.Fisher.prob<-function(data){

#
# Creating a 2 dimensional scatterplot with KFDA decision boundary and
# classification regions using posterior probabilities obtained from
# estimated normal densities on the projections

# ---- Hyperparameters

      opt.gam<-0.5
      opt.lam<-0.1

      class1<-matrix(data[data[,1]==1],ncol=ncol(data))[,2:ncol(data)]
      class2<-matrix(data[data[,1]==-1],ncol=ncol(data))[,2:ncol(data)]
      n1<-nrow(matrix(data[data[,1]==1],ncol=ncol(data)))
      n2<-nrow(matrix(data[data[,1]==-1],ncol=ncol(data)))
      n<-n1+n2
      p<-ncol(class1)

      Inp1<-rbind(as.matrix(rep(1,n1)),as.matrix(rep(0,n2)))
      Inm1<-rbind(as.matrix(rep(0,n1)),as.matrix(rep(1,n2)))
      In<-matrix(rep(0,(n*n)),ncol=n,nrow=n)
      diag(In)<-1
#
# ---- Creating the Gram matrix using a Gaussian kernel function
#
      G<-exp(((-1)*opt.gam)*(as.matrix(dist(data[,-1],
          method="euclidean")^2)))
      kp11<-(1/n1)*(G%*%Inp1)
      km11<-(1/n2)*(G%*%Inm1)
      N<-((1/n)*((G%*%G)-(n1*(kp11%*%t(kp11)))-
      (n2*(km11%*%t(km11)))))+(opt.lam*In)
#
# ---- The optimal alpha vector and the thresshold b
#
      alpha<-solve(N)%*%(kp11-km11)
      b<-(-1/2)*t(kp11-km11)%*%solve(N)%*%(kp11+km11)-log(n2/n1)
#
# ---- Obtaining the projections
#
      qdata<-matrix(t(alpha)%*%G,ncol=1)
      qdata<-cbind(data[,1],qdata)
      qmax<-max(qdata[(n1+1):n,2])
      qmin<-min(qdata[1:n1,2])

       if(qmin>qmax)
      qdata[,1][qdata[,2]==qmax]<-1
      qdata[,1][qdata[,2]==qmin]<--1

      mu1q<-mean(qdata[,2][qdata[,1]==1])
      mu2q<-mean(qdata[,2][qdata[,1]==-1])
      var1q<-var(qdata[,2][qdata[,1]==1])
      var2q<-var(qdata[,2][qdata[,1]==-1])
      varpq<-sqrt(((n1-1)*var1q+(n2-1)*var2q)/(n-2))
```

```
#
# ---- The following instructions is used to obtain the two class
#      classification regions and the decision boundary if p=2
#
      xmin<-min(data[,2])
      xmax<-max(data[,2])
      ymin<-min(data[,3])
      ymax<-max(data[,3])

      grid.data<-
        expand.grid(seq(xmin,xmax,length=50),seq(ymin,ymax,length=50))
      aa<-as.matrix(grid.data[,1])
      bb<-as.matrix(grid.data[,2])
      cc<-matrix(rep(1,nrow(aa)),ncol=1)
      grid.data<-cbind(cc,aa,bb)
      ng<-nrow(grid.data)

      gram1data<-rbind(data,grid.data)
      Gram1<-exp(((-1)*opt.gam)*(as.matrix(dist(gram1data[,-
      1],method="euclidean")^2)))
      Gram1<-Gram1[1:n,(n+1):ncol(Gram1)]

      q<-matrix((t(alpha)%*%Gram1),ncol=1)
#
# ---- Estimating normal densities on the projections
#
      dens<-matrix(rep(0,ng*2),ncol=2)
      signgrid<-matrix(rep(0,ng),nrow=ng,ncol=1)

      for (i in 1:ng){
      dens[i,1]<-dnorm(q[i,],mu1q,sqrt(var1q))/(dnorm(q[i,],mu1q,
                 sqrt(var1q))+dnorm(q[i,],mu2q,sqrt(var2q)))
      dens[i,2]<-dnorm(q[i,],mu2q,sqrt(var2q))/(dnorm(q[i,],mu1q,
                 sqrt(var1q))+dnorm(q[i,],mu2q,sqrt(var2q)))

      if (dens[i,1]>=0.5)
      signgrid[i,]<-1
      else signgrid[i,]<--1
      }

      xp<-seq(min(data[,2]),max(data[,2]),length=50)
      np<-length(xp)
      yp<-seq(min(data[,3]),max(data[,3]),length=50)
      ymat<-factor(data[,1])
      zp<-matrix(signgrid,ncol=1)
#
# ---- Plotting the decision boundary of KFDA
#
      plot(data[,2],data[,3],type="n",xlab="x1",ylab="x2")
      par(new=T)
      contour(xp,yp,matrix(zp,np),add=T,drawlabels=F,levels=0,lty=2,lwd=2,
      col="blue")

      region.data<-cbind(signgrid,grid.data[,2:3])
      region.green<-matrix(region.data[region.data[,1]==1],ncol=3)
      region.red<-matrix(region.data[region.data[,1]==-1],ncol=3)
#
# ---- Creating the classification regions
#
      par(new=T)
      plot(region.green[,2:3],xlim=c(min(data[,2]),max(data[,2])),ylim=c(min
      (data[,3]),max(data[,3])),col="green",xlab="x1",ylab="x2",pch=".")
      points(region.red[,2:3],col="red",pch=".")
```

```
#
# ---- Plotting the training data
#
       par(new=T)
       plot(class1,xlim=c(min(data[,2]),max(data[,2])),ylim=c(min(data[,3]),
        max(data[,3])),col="green",xlab="x1",ylab="x2",pch=17,cex=1.25)
       points(class2,col="red",pch=16,cex=1.25)

       }
```

E. KFDA USING POSTERIOR PROBABILITIES OBTAINED FROM LOGISTIC REGRESSION

```
kernel.Fisher.log<-function(data){


# Creating a 2 dimensional scatterplot with KFDA decision boundary and
# classification regions based on posterior probabilities from logistic
# regression

       library(nnet)

# ---- Hyperparameters

       opt.gam<-0.5
       opt.lam<-0.1

       class1<-matrix(data[data[,1]==1],ncol=ncol(data))[,2:ncol(data)]
       class2<-matrix(data[data[,1]==-1],ncol=ncol(data))[,2:ncol(data)]
       n1<-nrow(matrix(data[data[,1]==1],ncol=ncol(data)))
       n2<-nrow(matrix(data[data[,1]==-1],ncol=ncol(data)))
       n<-n1+n2
       p<-ncol(class1)

       Inp1<-rbind(as.matrix(rep(1,n1)),as.matrix(rep(0,n2)))
       Inm1<-rbind(as.matrix(rep(0,n1)),as.matrix(rep(1,n2)))
       In<-matrix(rep(0,(n*n)),ncol=n,nrow=n)
       diag(In)<-1
#
# ---- Creating the Gram matrix using a Gaussian kernel function
#
       G<-exp(((-1)*opt.gam)*(as.matrix(dist(data[,1],
           method="euclidean")^2)))
       kp11<-(1/n1)*(G%*%Inp1)
       km11<-(1/n2)*(G%*%Inm1)
       N<-((1/n)*((G%*%G)-(n1*(kp11%*%t(kp11)))-(n2*(km11%*%t(km11)))))
             +(opt.lam*In)
#
# ---- The optimal alpha vector and the thresshold b
#
       alpha<-solve(N)%*%(kp11-km11)
       b<-(-1/2)*t(kp11-km11)%*%solve(N)%*%(kp11+km11)-log(n2/n1)
#
# ---- Calculating the discriminant scores for training data
#
       mat.b<-matrix(rep(b,nrow(G)),nrow=1)
       scores<-matrix((t(alpha)%*%G)+mat.b,ncol=1)
#
# ---- The following instructions is used to obtain the two class
#      classification regions and the decision boundary if p=2
#
       xmin<-min(data[,2])
       xmax<-max(data[,2])
       ymin<-min(data[,3])
       ymax<-max(data[,3])

       grid.data<-
         expand.grid(seq(xmin,xmax,length=50),seq(ymin,ymax,length=50))
       aa<-as.matrix(grid.data[,1])
```

```
      bb<-as.matrix(grid.data[,2])
      cc<-matrix(rep(1,nrow(aa)),ncol=1)
      grid.data<-cbind(cc,aa,bb)
      ng<-nrow(grid.data)
      gram1data<-rbind(data,grid.data)
      Gram1<-exp(((-1)*opt.gam)*(as.matrix(dist(gram1data[,-
      1],method="euclidean")^2)))
      Gram1<-Gram1[1:n,(n+1):ncol(Gram1)]

      mat2.b<-matrix((rep(b,ncol(Gram1))),nrow=1)
      scores.grid.data<-matrix(((t(alpha)%*%Gram1)+mat2.b),ncol=1)
      scores.grid.data<-cbind(grid.data[,1],scores.grid.data)
      ti<-(data[,1]+1)/2
      fi<-scores
      input<-cbind(ti,fi)
#
# ---- Performing logistic regression
#
      output<-multinom(input[,1]~input[,2],maxit=250,drop=FALSE)
      coef<-coef(output)
      A<-coef[[1]]
      B<-coef[[2]]
#
# ---- Calculating the posterior probabilities - to obtain decision boundary
#
      probs<-
      matrix(exp(A+(B*scores.grid.data[,2]))/(1+exp(A+(B*scores.grid.data[,2
             ]))),ncol=1)
      signgrid<-matrix(rep(0,ng),nrow=ng,ncol=1)

      for (i in 1:ng){
            if (probs[i,]>=0.5)
             signgrid[i,]<-1
            else signgrid[i,]<--1
      }

      xp<-seq(min(data[,2]),max(data[,2]),length=50)
      np<-length(xp)
      yp<-seq(min(data[,3]),max(data[,3]),length=50)
      ymat<-factor(data[,1])
      zp<-matrix(signgrid,ncol=1)
#
# ---- Plotting the decision boundary
#
      plot(data[,2],data[,3],type="n",xlab="x1",ylab="x2")
      par(new=T)
      contour(xp,yp,matrix(zp,np),add=T,drawlabels=F,levels=0,lty=2,lwd=2,
       col="blue")
#
# ---- Obtaining and plotting the classification regions
#
      region.data<-cbind(signgrid,grid.data[,2:3])
      region.green<-matrix(region.data[region.data[,1]==1],ncol=3)
      region.red<-matrix(region.data[region.data[,1]==-1],ncol=3)
      par(new=T)
      plot(region.green[,2:3],xlim=c(min(data[,2]),max(data[,2])),ylim=c(min
       (data[,3]),max(data[,3])),col="green",xlab="x1",ylab="x2",pch=".")
      points(region.red[,2:3],col="red",pch=".")
```

```
#
# ---- Plotting the training data
#
       par(new=T)
       plot(class1,xlim=c(min(data[,2]),max(data[,2])),ylim=c(min(data[,3]),
        max(data[,3])),col="green",xlab="x1",ylab="x2",pch=17)
       points(class2,col="red",pch=16)

       }
```

F. THE SMALLEST ENCLOSING HYPERSPHERE

```
enclosing.hypersphere<-function (data) {


#
# Creating a two dimensional scatterplot and using the smallest enclosing
# hypersphere to obtain a support region for a class
#

       library(kernlab)
       library(base)
       library(MASS)

       class1<-matrix(data[data[,1]==1],ncol=ncol(data))[,1:ncol(data)]
       class2<-matrix(data[data[,1]==-1],ncol=ncol(data))[,1:ncol(data)]
       class2<-cbind(class2[,1]*-1,class2[,2:3])
       n1<-nrow(class1)
       n2<-nrow(class2)
       xaxis<-c(min(data[,2]),max(data[,2]))
       yaxis<-c(min(data[,3]),max(data[,3]))

       plot(data[,-1],type="n",ylim=yaxis,xlim=xaxis,xlab="x1",ylab="x2")
#
# ---- Performing the hypersphere analysis
#
       hypersphere.plot<-
       function (data,xaxis,yaxis,pch,col){

       library(kernlab)
       library(base)

       n<-nrow(data)
#
# ---- The hyperparameter for the Gaussian kernel
#
       opt.lam<-0.5
#
# ---- Obtaining the Gram matrix using the Gaussian kernel function
#
       G<-exp(((-1)*opt.lam)*(as.matrix(dist(data[,-1],
          method="euclidean")^2)))
#
# ---- Solving the QP problem using interior point optimization - Obtaining
#      the optimal alpha vector
#
       cvec<- -0.5*matrix(diag(G),ncol=1)
       lvec<-matrix(rep(0,n),ncol=1)
       uvec<-matrix(rep(1,n),ncol=1)
       b<-0
       r<-1
       Amat<-t(data[,1])
       alpha<-primal(ipop(cvec,G,Amat,b,lvec,uvec,r))

       index<-seq(1:n)[round(alpha,5)!=0][1]
#
# ---- Finding the squared radius
#
       r2<-G[index,index]-(2*t(alpha)%*%G[,index])+(t(alpha)%*%G%*%alpha)

       D<-t(alpha)%*%G%*%alpha-r2
#
# ---- The following instructions is used to obtain the support region if
#      p=2
#
       xmin<-xaxis[1]
       xmax<-xaxis[2]
       ymin<-yaxis[1]
       ymax<-yaxis[2]

       grid.data<-expand.grid(seq(xmin,xmax,length=80),
                  seq(ymin,ymax,length=80))
       aa<-as.matrix(grid.data[,1])
       bb<-as.matrix(grid.data[,2])
       cc<-matrix(rep(1,nrow(aa)),ncol=1)
       grid.data<-cbind(cc,aa,bb)
       ng<-nrow(grid.data)

       grid<-matrix(rep(0,ng),nrow=ng,ncol=1)

       for (i in 1:ng){
             combine.grid<-rbind(data,grid.data[i,])
             Gng<-exp(((-1)*opt.lam)*(as.matrix(dist(combine.grid[,-1],
             method="euclidean")^2)))
             vectorzg<-as.matrix(Gng[-nrow(Gng),ncol(Gng)])
             grid[i,]<-Gng[nrow(Gng),ncol(Gng)]-2*t(alpha)%*%vectorzg+D
       }

       xp<-seq(xmin,xmax,length=80)
       np<-length(xp)
       yp<-seq(ymin,ymax,length=80)
       ymat<-factor(data[,1])
       zp<-matrix(grid,ncol=1)
#
# ---- Obtaining and plotting the support region for a class
#
       region.data<-cbind(grid,grid.data[,2:3])
       region<-matrix(region.data[region.data[,1]<0],ncol=3)
       par(new=T)
       contour(xp,yp,matrix(zp,np),add=T,drawlabels=F,levels=0,lty=2,lwd=2,
         col=col)
       points(region[,2:3],col=col,pch=".")
       points(data[,2:3],col=col,pch=pch,cex=1.25)
       alpha
       }

       alpha1<-hypersphere.plot(class1,xaxis,yaxis,16,"red")
       alpha2<-hypersphere.plot(class2,xaxis,yaxis,15,"green")

       list(data,alpha1,alpha2)
       }
```

G. AN EXAMPLE OF A R SIMULATION PROGRAM PERFORMING THE SHAPIRO- WILK HYPOTHESIS TEST ON THE PROJECTIONS OF KFDA AS DISCUSSED IN CHAPTER 4

```
normal.test.simulation<-function (n1,n2) {


#
# ---- the output of this program is a table containing the relative
#      frequencies of normality for the projections at three different
#      number of variables, two Mahalanobis distances and three correlations
#

      normal.test.q<-function (n1,n2,mal,cor,nmc) {

# ---- mal=Mahalanobis distance between classes
#      n1=sample size of class 1
#      n2=sample size of class 2
#      cor=correlation between variables
#      nmc=number of Monte Carlo simulations

      library(MASS)

      p.mat1<-matrix(rep(0,nmc),ncol=1)
      p.mat2<-matrix(rep(0,nmc),ncol=1)
      q.mat<-matrix(rep(0,(n1+n2)*nmc),ncol=nmc)
#
# ---- Specifying the number of variables used in the simulation
#
      var<-matrix(c(5,50,100),ncol=1)

      dim<-nrow(var)

      p.mat<-matrix(rep(0,dim*(2*nmc)),nrow=(2*nmc))

      for (j in 1:nrow(var)){
            p<-var[j,]
#
# ---- Starting the simulation based on p-variables
#
      for(k in 1:nmc){
#
# ---- Generating the training data from multivariate normal distributions
#
      sigma<-matrix(rep(cor,(p*p)),ncol=p)
      diag(sigma)<-1
      mu1<-rep(0,p)
      gem2<-sqrt(mal/sum(solve(sigma)))
      mu2<-rep(gem2,p)

      class1<-mvrnorm(n1,mu1,sigma)
      class2<-mvrnorm(n2,mu2,sigma)
      n<-n1+n2

      x<-rbind(class1,class2)
#
# ---- Standardizing the training data
#
      scale(x)
      y1<-as.matrix(rep(1,n1)); y2<-as.matrix(rep(-1,n2))
      y<-rbind(y1,y2)
      data<-as.matrix(cbind(y,x))
#
# ---- Specifying the hyperparameters for KFDA
#
      opt.gam<-1/p
      opt.lam<-0.1

      Inp1<-rbind(as.matrix(rep(1,n1)),as.matrix(rep(0,n2)))
      Inm1<-rbind(as.matrix(rep(0,n1)),as.matrix(rep(1,n2)))
      In<-matrix(rep(0,(n*n)),ncol=n,nrow=n)
      diag(In)<-1
#
# ---- Calculating the Gram matrix using the Gaussian kernel function
#
      G<-exp(((-1)*opt.gam)*(as.matrix(dist(data[,-1],
            method="euclidean")^2)))
      kp11<-(1/n1)*(G%*%Inp1)
      km11<-(1/n2)*(G%*%Inm1)
      N<-((1/n)*((G%*%G)-(n1*(kp11%*%t(kp11)))-
                  (n2*(km11%*%t(km11)))))+(opt.lam*In)
#
# ---- Optimal alpha vector and threshold b
#
      alpha<-solve(N)%*%(kp11-km11)
      b<-(-1/2)*t(kp11-km11)%*%solve(N)%*%(kp11+km11)-log(n2/n1)
#
# ---- Calculating the projections
#
      qdata<-t(alpha)%*%G
#
# ---- Performing the Shapiro-Wilk test for normality on the projections of
#      each class
#
      p.mat1[k,1]<-shapiro.test(qdata[1:n1])$p.value
      p.mat2[k,1]<-shapiro.test(qdata[(n1+1):(n1+n2)])$p.value
      q.mat[,k]<-qdata

      }

      p.mat[,j]<-round(rbind(p.mat1,p.mat2),4)

      }

      w1<-as.matrix(rep(1,nmc)); w2<-as.matrix(rep(-1,nmc))
      w<-rbind(w1,w2)
      p.mat<-cbind(w,p.mat)

      indicator<-p.mat[,-1]

      for(i in 1:dim){
            for (j in 1:(2*nmc)){
                  if(indicator[j,i]>=0.05)
                        indicator[j,i]<-1
                  if(indicator[j,i]<0.05)
                        indicator[j,i]<-0
                        }
                  }
```

```
fraction<-apply(indicator,2,sum)
fraction<-fraction/(2*nmc)
fraction
}

correl<-matrix(c(0,0.5,0.9),ncol=1)
table1<-table2<-matrix(rep(0,9),ncol=3)
for (j in 1:nrow(correl)){
       cor<-correl[j,]
      table1[j,]<-normal.test.q(n1,n2,5,cor,100)
      table2[j,]<-normal.test.q(n1,n2,10,cor,100)
}

names1<-c("p=5","p=50","p=100")
names2<-c("cor=0","cor=0.5","cor=0.9")
dimnames(table1)<-list(names2,names1)
dimnames(table2)<-list(names2,names1)
list("Mahalanobis distance=5"=table1,"Mahalanobis distance=10"=table2)

}
```

H. AN EXAMPLE OF A R SIMULATION PROGRAM COMPARING THE CLASSIFICATION PERFORMANCE OF THE THREE KFD CLASSIFIERS AS DISCUSSED IN CHAPTER 4

```
three.classifiers.simulation<-function (n1,n2) {


#
# ---- the output of this program is a table containing the means and
#      standard deviations of the error rates obtained from the three
#      classifiers, as well as the Mahalanobis distance,sample size and
#      correlation in the last column
#
      three.classifiers<-function (n1,n2,mal,p,cor,nmc) {

# ---- mal=Mahalanobis afstand
#      n1=sample size of class 1
#      n2=sample size of class 2
#      cor=correlation
#      nmc=number of Monte Carlo simulations
#      p=number of variables
#      data generated from a lognormal distribution with equal covariances

      library(MASS)
      library(nnet)

      tot.sign.fout<-matrix(rep(0,nmc),ncol=1)
      tot.dens.fout<-matrix(rep(0,nmc),ncol=1)
      tot.log.fout<-matrix(rep(0,nmc),ncol=1)
#
# ---- Constants needed for Johnson’s translation system
#
      E<-exp(1)
      LAM<-sqrt(1/(E*(E-1)))
      EP<--1*sqrt(1/(E-1))
      aa<-log(cor*(E-1)+1)
      cc<-((E^aa)-1)/(E-1)
#
# ---- Starting the simulation
#
      for(k in 1:nmc){

      sigma<-matrix(rep(aa,(p*p)),ncol=p)
      diag(sigma)<-1
      mu1<-rep(0,p)
      gem2<-sqrt(mal/sum(solve(sigma)))
      mu2<-rep(gem2,p)

      mgreen<-LAM*exp(mvrnorm(n1,mu1,sigma))+EP
      mred<-LAM*exp(mvrnorm(n2,mu1,sigma))+EP
      for(i in 1:n2){mred[i,]<-mred[i,]+mu2}

      x<-rbind(mgreen,mred)
#
# ---- Calculating the means and variances of the training data, needed to
#      standardize the test data
      mean<-matrix(apply(x[,1:ncol(x)],2,mean),ncol=p)
      var<-matrix(apply(x[,1:ncol(x)],2,var),ncol=p)
```

```
#
# ---- Standardizing the training data
#
      x<-scale(x)
      y1<-as.matrix(rep(1,n1)); y2<-as.matrix(rep(-1,n2))
      y<-rbind(y1,y2)

      data<-as.matrix(cbind(y,x))

      class1<-matrix(data[data[,1]==1],ncol=ncol(data))[,2:ncol(data)]
      class2<-matrix(data[data[,1]==-1],ncol=ncol(data))[,2:ncol(data)]
      n<-n1+n2
      p<-ncol(class1)
#
# ---- Specifying the hyperparameters for KFDA
#
      opt.gam<-1/p
      opt.lam<-0.1

      Inp1<-rbind(as.matrix(rep(1,n1)),as.matrix(rep(0,n2)))
      Inm1<-rbind(as.matrix(rep(0,n1)),as.matrix(rep(1,n2)))
      In<-matrix(rep(0,(n*n)),ncol=n,nrow=n)
      diag(In)<-1
#
# ---- Obtaining the Gram matrix using the Gaussian kernel
#
      G<-exp(((-1)*opt.gam)*(as.matrix(dist(data[,-1],
           method="euclidean")^2)))
      kp11<-(1/n1)*(G%*%Inp1)
      km11<-(1/n2)*(G%*%Inm1)
      N<-((1/n)*((G%*%G)-(n1*(kp11%*%t(kp11)))-
      (n2*(km11%*%t(km11)))))+(opt.lam*In)
#
# ---- Obtaining the optimal alpha vector and thresshold b
#
      alpha<-solve(N)%*%(kp11-km11)
      b<-(-1/2)*t(kp11-km11)%*%solve(N)%*%(kp11+km11)-log(n2/n1)
#
# ---- Calculating the projections of the training data
#
      qdata<-matrix(t(alpha)%*%G,ncol=1)
      qdata<-cbind(data[,1],qdata)
      qmax<-max(qdata[(n1+1):n,2])
      qmin<-min(qdata[1:n1,2])

      if(qmin>qmax)
      qdata[,1][qdata[,2]==qmax]<-1
      qdata[,1][qdata[,2]==qmin]<--1
#
# ---- Calculating the discriminant scores of the training data
#
      mat.b<-matrix(rep(b,nrow(G)),nrow=1)
      scores<-matrix((t(alpha)%*%G)+mat.b,ncol=1)
      signs<-sign(scores)
#
# ---- Generate the test data of sieze 2000 from the same distribution as
#      the training data
#
      test1<-LAM*exp(mvrnorm(1000,mu1,sigma))+EP
      test2<-LAM*exp(mvrnorm(1000,mu1,sigma))+EP
      for(i in 1:1000) {test2[i,]<-test2[i,]+mu2}

      test<-rbind(test1,test2)
      ty1<-as.matrix(rep(1,1000)); ty2<-as.matrix(rep(-1,1000));
      ty<-rbind(ty1,ty2)
      testdata<-as.matrix(cbind(ty,test))


#
# ---- Standardizing the test data using the means and variances form the
#      unstandardized training data
#
      std.testdata<-matrix(rep(0,(2000*p)),ncol=p)

      for (i in 1:2000){
            for (j in 1:p){
            std.testdata[i,j]<-(testdata[,2:ncol(testdata)][i,j]-
            mean[,j])/sqrt(var[,j])
            }
      }

      std.testdata<-cbind(testdata[,1],std.testdata)

      gram1data<-rbind(data,std.testdata)
      Gram1<-exp(((-1)*opt.gam)*(as.matrix(dist(gram1data[,-
      1],method="euclidean")^2)))
      Gram1<-Gram1[1:n,(n+1):ncol(Gram1)]
#
# ---- Projections of the test data
#
      q.testdata<-matrix((t(alpha)%*%Gram1),ncol=1)
      q.testdata<-cbind(testdata[,1],q.testdata)

      mat2.b<-matrix((rep(b,ncol(Gram1))),nrow=1)
      scores.testdata<-matrix(((t(alpha)%*%Gram1)+mat2.b),ncol=1)
      scores.testdata<-cbind(testdata[,1],scores.testdata)
#
# ---- Calculating the error rate for KFDA using classifier 1
#
      s.test<-matrix(sign(scores.testdata[,2]),ncol=1)
      s.test<-cbind(scores.testdata,s.test)
      s.fout<-matrix(rep(0,nrow(s.test)),ncol=1)

      for (i in 1:nrow(s.test)){
             if(s.test[i,1]==s.test[i,3])
                   s.fout[i,]<-0
             if(s.test[i,1]!=s.test[i,3])
                    s.fout[i,]<-1
            }

      s.test<-cbind(s.test,s.fout)
      tot.sign.fout[k,]<-sum(s.fout)/nrow(s.fout)  # error rate
#
# ---- Calculating the error rate for KFDA using classifier 2
#
      mu1q<-mean(qdata[,2][qdata[,1]==1])
      mu2q<-mean(qdata[,2][qdata[,1]==-1])
      var1q<-var(qdata[,2][qdata[,1]==1])
      var2q<-var(qdata[,2][qdata[,1]==-1])
      varpq<-sqrt(((n1-1)*var1q+(n2-1)*var2q)/(n-2))

      dens<-matrix(rep(0,(2000*2)),ncol=2)
      indicator<-matrix(rep(0,(2000*1)),ncol=1)

      for (i in 1:2000){
      q<-matrix(q.testdata[,-1],ncol=1)
      dens[i,1]<-
      dnorm(q[i,],mu1q,sqrt(var1q))/(dnorm(q[i,],mu1q,sqrt(var1q))+
          dnorm(q[i,],mu2q,sqrt(var2q)))
      dens[i,2]<-
      dnorm(q[i,],mu2q,sqrt(var2q))/(dnorm(q[i,],mu1q,sqrt(var1q))+
          dnorm(q[i,],mu2q,sqrt(var2q)))

      if (dens[i,1]>=0.5)
            indicator[i,]<-1
      else indicator[i,]<--1
      }

      densdata<-cbind(testdata[,1],dens,indicator)

      d.fout<-matrix(rep(0,nrow(densdata)),ncol=1)

      for (i in 1:nrow(densdata)){
            if(densdata[i,1]==densdata[i,4])
            d.fout[i,]<-0
            if(densdata[i,1]!=densdata[i,4])
            d.fout[i,]<-1
      }

      d.test<-cbind(densdata,d.fout)
      tot.dens.fout[k,]<-sum(d.fout)/nrow(d.fout)  # error rate
#
# ---- Calculating the error rate for KFDA using classifier 3
#
      ti<-(data[,1]+1)/2
      fi<-scores
      input<-cbind(ti,fi)
#
# ---- Performing logistic regression
#
      output<-multinom(input[,1]~input[,2],maxit=250,drop=FALSE)
      coef<-coef(output)
      A<-coef[[1]]
      B<-coef[[2]]
      probs<-
      matrix(exp(A+(B*scores.testdata[,2]))/(1+exp(A+(B*scores.testdata[,2])
          )),ncol=1)
      indicator2<-matrix(rep(0,(2000)),ncol=1)

      for (i in 1:2000){
            if (probs[i,]>=0.5)
            indicator2[i,]<-1
            else indicator2[i,]<--1
      }

      logdata<-cbind(testdata[,1],probs,indicator2)

      l.fout<-matrix(rep(0,nrow(logdata)),ncol=1)

      for (i in 1:nrow(logdata)){
            if(logdata[i,1]==logdata[i,3])
            l.fout[i,]<-0
            if(logdata[i,1]!=logdata[i,3])
            l.fout[i,]<-1
      }

      l.test<-cbind(logdata,l.fout)
      tot.log.fout[k,]<-sum(l.fout)/nrow(l.fout)   # error rate
      }

      cat("Mahalanobis distance\n")
      print(mal)

      cat("Observations in Class 1\n")
      print(n1)

      cat("Observations in Class -1\n")
      print(n2)

      cat("Correlation\n")
      print(cor)


      table<-matrix(rep(0,9),ncol=3)
      table[1,1]<-mean(tot.sign.fout);table[1,2]<-mean(tot.dens.fout);
       table[1,3]<-mean(tot.log.fout)
      table[2,1]<-sqrt(var(tot.sign.fout));table[2,2]<-
        sqrt(var(tot.dens.fout)); table[2,3]<-sqrt(var(tot.log.fout))
      table[3,1]<-mal;table[3,2]<-n1;table[3,3]<-cor

      name1<-c("classifier 1","classifier 2","classifier 3")
      name2<-c("Mean","Std deviation","Parameter")

      dimnames(table)<-list(name2,name1)
      cat("Error rates\n")
      print(table)
      }

      correl<-matrix(c(-0.1,0,0.7),ncol=1)
      mahal<-matrix(c(2,4),ncol=1)
      output<-NULL

      for (j in 1:nrow(mahal)){
             for (i in 1:nrow(correl)){
                    mal<-mahal[j,]
                    cor<-correl[i,]
                    mat<-three.classifiers(n1,n2,mal,5,cor,2)
                    output<-append(output,mat)
                    }
             }

      output<-matrix(output,ncol=3,byrow=T)
      name1<-rep(c("classifier 1","classifier 2","classifier 3")
      ,nrow(correl)*nrow(mahal))
      name2<-c("Mean","Std deviation","Parameter")
#
# ---- Note: the column parameter contains the Mahalanobis distance,sample
#       size and correlation. If the Mahalanobis distance is large, the
#       logistic regression cannot find a solution for the parameters
#
      dimnames(output)<-list(name1,name2)
      output
      }
```

I. AN EXAMPLE OF THE FORTRAN PROGRAM THAT WAS USED TO PERFORM THE SIMULATION STUDY IN CHAPTER 3

```
C     XDATA CONTAINS THE DATA WITH THE ATYPICAL CASE
C     XMAT CONTAINS THE DATA WITHOUT THE ATYPICAL CASE
C     THE FOLLOWING CRITERIA ARE CALCULATED
C
C     1  TEST ERROR
C     2  BETWEEN GROUP/WITHIN GROUP VARIANCE RATIO
C     3  NORM OF W
C     4  AVERAGE MARGIN
C     5  AVERAGE DISTANCE OF MISCLASSIFIED CASE FROM DECISION BOUNDARY
C     6  TRAINING ERROR
C     7  RAYLEIGH COEFICIENT
C     8  ALIGNMENT
C
C     DATA ARE GENERATED FROM TWO NORMAL DISTRIBUTIONS WITH THE SAME
C     COVARIANCE MATRIX, BUT DIFFERENT MEAN VECTORS
C
C     NOTATION:
C     IP=NUMBER OF X VARIABLE
C     NTOT=NUMBER OF CASES IN GROUP 1
C     MTOT= NUMBER OF CASES IN GROUP 2
C     NMTOT=NTOT+MTOT
C     NNPMM=NMTOT-1
C     YV=DIE Y-VECTOR WITH +1 AND -1 LABELS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      PARAMETER (NMC=1000)
      PARAMETER (GAM=1.0D0/IP)
      PARAMETER (CORR=-0.1D0)
      DIMENSION AMU1(IP),AMU2(IP)
      DIMENSION SIGMAM(IP,IP),SIGINV(IP,IP),RSIG(IP,IP)
      DIMENSION XM1(NTOT,IP),XM2(MTOT,IP)
      DIMENSION XMAT(NNPMM,IP),YV(NNPMM)
      DIMENSION XT1(NT,IP),XT2(MT,IP),XT(NMT,IP),YVT(NMT)
      DIMENSION XTO(NMT,IP)
      DIMENSION GEM(IP),SA(IP),GEMUIT(IP),SAUIT(IP)
      DIMENSION GRMAT(NNPMM,NNPMM),GRNUUT(NMT,NNPMM)
      DIMENSION EENP(NNPMM),EENM(NNPMM)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM)
      DIMENSION ALPHA(NNPMM),AKM(NNPMM),AKP(NNPMM)
      DIMENSION ALPHAKWAD(NNPMM)
      DIMENSION YVTOT(NMTOT)
      DIMENSION XANDER(1,IP)
      DIMENSION XDATA(NMTOT,IP)
      DIMENSION GRMATTOT(NMTOT,NMTOT),GRNUUTTOT(NMT,NMTOT)
      DIMENSION EENPTOT(NMTOT),EENMTOT(NMTOT)
      DIMENSION BTOT(NMTOT),HTOT(NMTOT,NMTOT),ANTOT(NMTOT,NMTOT)
      DIMENSION ALPHATOT(NMTOT),AKMTOT(NMTOT),AKPTOT(NMTOT)
      DIMENSION ALPHATOTKWAD(NMTOT)
      DIMENSION QVEK(NMTOT)
      DIMENSION QV(NNPMM),SCOREVEK(NNPMM)
      DIMENSION DISTVEK(NNPMM),AMARGVEK(NNPMM)
      DIMENSION AN(NNPMM,NNPMM)
      DIMENSION QVTOT(NMTOT),SCOREVEKTOT(NMTOT)
      DIMENSION DISTVEKTOT(NMTOT),AMARGVEKTOT(NMTOT)
      DIMENSION FOUTVEK(2),CRITM(2),CRITR(2),GEMDIST(2)
      DIMENSION RCRIT(2),WNORM(2),AMARG(2),CRITAL(2),TERR(2)
      DIMENSION CRITMAT(15,2,9),STDVEK(15)

      CHARACTER*70 FILEOUT1,FILEOUT2,FILEOUT3,FILEOUT4

      FILEOUT1='n5c1ks25crits.d'
      FILEOUT2='n5c1ks25foute.d'
      FILEOUT3='n5c1ks25crits1.d'
      FILEOUT4='n5c1ks25foute1.d'

C     DEFINE PARAMETERS

      DO 3 I=1,IP
      AMU1(I)=2.5D0
      AMU2(I)=0.0D0
      DO 2 J=1,IP
      SIGMAM(I,J)=CORR
2     CONTINUE
      SIGMAM(I,I)=1.0D0
3     CONTINUE
      CALL DLINDS(IP,SIGMAM,IP,SIGINV,IP)
      TOL=1.0D2*DMACH(4)
      CALL DCHFAC(IP,SIGMAM,IP,TOL,IRANK,RSIG,IP)

C     CONSTRUCT RESPONSE VECTOR

      DO 6 I=1,NTOT
      YVTOT(I)=-1.0D0
6     CONTINUE
      DO 7 I=NTOT+1,NMTOT
      YVTOT(I)=1.0D0
7     CONTINUE

C     CONSTRUCT INDICATOR VECTOR
C
      DO 8 I=1,NMTOT
      EENPTOT(I)=0.0D0
      EENMTOT(I)=0.0D0
      IF (YVTOT(I).LT.-0.1D0) EENMTOT(I)=1.0D0
      IF (YVTOT(I).GT.0.1D0) EENPTOT(I)=1.0D0
8     CONTINUE

      DO 599 III=1,2
      DO 893 J1=1,15
      DO 892 I=1,2
      DO 891 J=1,9
      CRITMAT(J1,I,J)=0.0D0
891   CONTINUE
892   CONTINUE
893   CONTINUE

      DO 889 J=1,2
      FOUTVEK(J)=0.0D0
889   CONTINUE

      SSS1=0.0D0
      SSS2=0.0D0
C
C     BEGINNING OF SIMULATION DO LOOP
```

```
      DO 495 MC=1,NMC
      WRITE(6,*) MC
C
C     GENERATE TRAINING DATA
C
      CALL DRNMVN(NTOT,IP,RSIG,IP,XM1,NTOT)

      DO 11 I=1,NTOT
      DO 10 J=1,IP
      XMAT(I,J)=XM1(I,J)+AMU1(J)
10    CONTINUE
11    CONTINUE
C
      CALL DRNMVN(MTOT,IP,RSIG,IP,XM2,MTOT)

      DO 13 I=1,MTOT-NUIT
      DO 12 J=1,IP
      XMAT(NTOT+I,J)=XM2(I,J)+AMU2(J)
12    CONTINUE
13    CONTINUE

C     ADD ATYPICAL CASE

      CALL DRNMVN(1,IP,RSIG,IP,XANDER,1)

      DO 14 J=1,IP
      XANDER(1,J)=XANDER(1,J)+AMU1(J)+(III-1.0D0)
14    CONTINUE

C     GENERATE TEST DATA AND RESPONSE VECTOR
C
      CALL DRNMVN(NT,IP,RSIG,IP,XT1,NT)
      CALL DRNMVN(MT,IP,RSIG,IP,XT2,MT)

      DO 18 I=1,NT
      DO 17 J=1,IP
      XTO(I,J)=XT1(I,J)+AMU1(J)
17    CONTINUE
18    CONTINUE
      DO 20 I=1,MT
      DO 19 J=1,IP
      XTO(NT+I,J)=XT2(I,J)+AMU2(J)
19    CONTINUE
20    CONTINUE
      DO 21 I=1,NT
      YVT(I)=-1.0D0
21    CONTINUE
      DO 22 I=NT+1,NMT
      YVT(I)=1.0D0
22    CONTINUE

      DO 492 LLL=1,1

      DO 777 I=1,NMTOT-1
      DO 776 J=1,IP
      XDATA(I,J)=XMAT(I,J)
776   CONTINUE
777   CONTINUE

      DO 778 J=1,IP
      XDATA(NMTOT,J)=XANDER(LLL,J)
778   CONTINUE
```

```
C     COMPUTE TRAINING ERROR ON ALL THE DATA-
C     STANDARDISE THE TRAINING DATA
C
      DO 24 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 23 I=1,NMTOT
      S1=S1+XDATA(I,J)
      S2=S2+XDATA(I,J)*XDATA(I,J)
23    CONTINUE
      GEM(J)=S1/NMTOT
      SA(J)=DSQRT((S2-NMTOT*GEM(J)*GEM(J))/(NMTOT-1))
24    CONTINUE
      DO 26 J=1,IP
      DO 25 I=1,NMTOT
      XDATA(I,J)=(XDATA(I,J)-GEM(J))/SA(J)
25    CONTINUE
26    CONTINUE
C
C     STANDARDISE THE TEST DATA
C
      DO 28 J=1,IP
      DO 27 I=1,NMT
      XT(I,J)=(XTO(I,J)-GEM(J))/SA(J)
27    CONTINUE
28    CONTINUE

      CALL GRAMMATTOT(GAM,XDATA,GRMATTOT)
      DO 491 KK1=1,15
      CPAR=DEXP(-9.0D0+KK1)

      CALL DKFDATOT(EENMTOT,EENPTOT,GRMATTOT,
      &CPAR,BTOT,HTOT,AKPTOT,AKMTOT,ANTOT,ALPHATOT,BOPTTOT)
      CALL GRAMNUUTTOT(GAM,XDATA,XT,GRNUUTTOT)
      CALL BERFOUTTOT(GRNUUTTOT,YVT,ALPHATOT,BOPTTOT,FOUTVEK(LLL))
      CRITMAT(KK1,LLL,1)=CRITMAT(KK1,LLL,1)+FOUTVEK(LLL)
      CRITMAT(KK1,LLL,9)=CRITMAT(KK1,LLL,9)+FOUTVEK(LLL)**2
      CALL BERSOMMATTOT1(GRMATTOT,GRSOM1,GRSOM2,GRSOM3)
      CALL BERSOMDIAGTOT1(GRMATTOT,DIAGSOM1,DIAGSOM2)

      DIST1=DIAGSOM1/NTOT-GRSOM1/(NTOT*NTOT)
      DIST2=DIAGSOM2/MTOT-GRSOM2/(MTOT*MTOT)
      DISTCENT=GRSOM1/(NTOT*NTOT)+GRSOM2/(MTOT*MTOT)
             &-(2.0D0*GRSOM3)/(NTOT*MTOT)
      RCRIT(LLL)=DISTCENT/(DIST1+DIST2)
      CRITMAT(KK1,LLL,2)=CRITMAT(KK1,LLL,2)+RCRIT(LLL)

      CALL BERQTOT(GRMATTOT,ALPHATOT,BOPTTOT,QVTOT,SCOREVEKTOT)
      CALL BERNORMTOT(GRMATTOT,ALPHATOT,BOPTTOT,WNORM(LLL))
      CRITMAT(KK1,LLL,3)=CRITMAT(KK1,LLL,3)+WNORM(LLL)

      CALL BERMARGTOT(SCOREVEKTOT,AMARGVEKTOT,AMARG(LLL))
      CRITMAT(KK1,LLL,4)=CRITMAT(KK1,LLL,4)+AMARG(LLL)

      DO 63 KKK=1,NMTOT
      DISTVEKTOT(KKK)=AMARGVEKTOT(KKK)/WNORM(LLL)
63    CONTINUE

      CALL BERMISCTOT(DISTVEKTOT,GEMDIST(LLL),TERR(LLL))
      CRITMAT(KK1,LLL,5)=CRITMAT(KK1,LLL,5)+GEMDIST(LLL)
      CRITMAT(KK1,LLL,6)=CRITMAT(KK1,LLL,6)+TERR(LLL)

      CALL BERCRITRTOT(BTOT,ANTOT,ALPHATOT,CRITR(LLL))
      CRITMAT(KK1,LLL,7)=CRITMAT(KK1,LLL,7)+CRITR(LLL)

      CALL BERALIGNTOT(GRMATTOT,CRITAL(LLL))
      CRITMAT(KK1,LLL,8)=CRITMAT(KK1,LLL,8)+CRITAL(LLL)

491   CONTINUE
492   CONTINUE

C     CALCULATE ERROR RATE USING DATA WITHOUT THE ATYPICAL CASE

      NN=NTOT
      MM=MTOT-1

C     CONSTRUCT RESPONSE VECTOR
C
      DO 110 I=1,NN
      YV(I)=-1.0D0
110   CONTINUE
      DO 111 I=NN+1,NNPMM
      YV(I)=1.0D0
111   CONTINUE
C
C     CONSTRUCT INDICATOR VECTOR
C
      DO 112 I=1,NNPMM
      EENP(I)=0.0D0
      EENM(I)=0.0D0
      IF (YV(I).LT.-0.1D0) EENM(I)=1.0D0
      IF (YV(I).GT.0.1D0) EENP(I)=1.0D0
112   CONTINUE

      DO 124 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 123 I=1,NNPMM
      S1=S1+XMAT(I,J)
      S2=S2+XMAT(I,J)*XMAT(I,J)
123   CONTINUE
      GEM(J)=S1/NNPMM
      SA(J)=DSQRT((S2-NNPMM*GEM(J)*GEM(J))/(NNPMM-1))
124   CONTINUE
      DO 126 J=1,IP
      DO 125 I=1,NNPMM
      XMAT(I,J)=(XMAT(I,J)-GEM(J))/SA(J)
125   CONTINUE
126   CONTINUE
C
      DO 128 J=1,IP
      DO 127 I=1,NMT
      XT(I,J)=(XTO(I,J)-GEM(J))/SA(J)
127   CONTINUE
128   CONTINUE

      CALL GRAMMAT(NN,MM,GAM,XMAT,GRMAT)
      DO 199 KK1=1,15
      CPAR=DEXP(-9.0D0+KK1)

      CALL DKFDA(NN,MM,EENM,EENP,GRMAT,
      &CPAR,B,H,AKP,AKM,AN,ALPHA,BOPT)
      CALL GRAMNUUT(GAM,XMAT,XT,GRNUUT)
      CALL BERFOUT(GRNUUT,YVT,ALPHA,BOPT,FOUTVEK(2))
      CRITMAT(KK1,2,1)=CRITMAT(KK1,2,1)+FOUTVEK(2)
      CRITMAT(KK1,2,9)=CRITMAT(KK1,2,9)+FOUTVEK(2)**2
      CALL BERSOMMAT1(NN,MM,GRMAT,GRSOM1,GRSOM2,GRSOM3)
      CALL BERSOMDIAG1(NN,MM,GRMAT,DIAGSOM1,DIAGSOM2)

      DIST1=DIAGSOM1/NN-GRSOM1/(NN*NN)
      DIST2=DIAGSOM2/MM-GRSOM2/(MM*MM)
      DISTCENT=GRSOM1/(NN*NN)+GRSOM2/(MM*MM)
     &-(2.0D0*GRSOM3)/(NN*MM)
      RCRIT(2)=DISTCENT/(DIST1+DIST2)
      CRITMAT(KK1,2,2)=CRITMAT(KK1,2,2)+RCRIT(2)

      CALL BERQ(GRMAT,ALPHA,BOPT,QV,SCOREVEK)
      CALL BERNORM(GRMAT,ALPHA,BOPT,WNORM(2))
      CRITMAT(KK1,2,3)=CRITMAT(KK1,2,3)+WNORM(2)

      CALL BERMARG(NN,MM,SCOREVEK,AMARGVEK,AMARG(2))
      CRITMAT(KK1,2,4)=CRITMAT(KK1,2,4)+AMARG(2)

      DO 163 KKK=1,NNPMM
      DISTVEK(KKK)=AMARGVEK(KKK)/WNORM(2)
163   CONTINUE

      CALL BERMISC(DISTVEK,GEMDIST(2),TERR(2))
      CRITMAT(KK1,2,5)=CRITMAT(KK1,2,5)+GEMDIST(2)
      CRITMAT(KK1,2,6)=CRITMAT(KK1,2,6)+TERR(2)

      CALL BERCRITR(B,AN,ALPHA,CRITR(2))
      CRITMAT(KK1,2,7)=CRITMAT(KK1,2,7)+CRITR(2)

      CALL BERALIGN(NN,MM,GRMAT,CRITAL(2))
      CRITMAT(KK1,2,8)=CRITMAT(KK1,2,8)+CRITAL(2)

199   CONTINUE

495   CONTINUE

      DO 521 K1=1,15
      DO 520 I=1,2
      DO 519 J=1,8
      CRITMAT(K1,I,J)=CRITMAT(K1,I,J)/NMC
519   CONTINUE

      CRITMAT(K1,I,9)=DSQRT(((CRITMAT(K1,I,9)-
      NMC*(CRITMAT(K1,I,1)**2))/(NMC**2)))

520   CONTINUE

      STDVEK(K1)=DSQRT((((NMC*CRITMAT(K1,1,9))**2)+
      ((NMC*CRITMAT(K1,2,9))**2))/(2*NMC))*(DSQRT(2.0D0/NMC))

521   CONTINUE

      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
      WRITE(1,*) III
      DO 532 K1=1,15

      WRITE(1,*) K1
      DO 531 J=1,9
      WRITE(1,603) (J,(CRITMAT(K1,I,J),I=1,2))
531   CONTINUE
      WRITE(1,600) STDVEK(K1)
      WRITE(1,*)
532   CONTINUE
```

```
      CLOSE(1)

      OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
      WRITE(1,*) III
      DO 542 K1=1,15
      WRITE(1,*)K1
      DO 541 J=1,1
      WRITE(1,603) (J,(CRITMAT(K1,I,J),I=1,2))
541   CONTINUE
      WRITE(1,*)
542   CONTINUE
      CLOSE(1)

      OPEN(1,FILE=FILEOUT3,ACCESS='APPEND')
      DO 552 K1=1,15
      DO 551 J=1,9
      WRITE(1,608) (III,K1,J,(CRITMAT(K1,I,J),I=1,2))
551   CONTINUE
552   CONTINUE
      CLOSE(1)

      OPEN(1,FILE=FILEOUT4,ACCESS='APPEND')
      DO 562 K1=1,15
      DO 561 J=1,1
      WRITE(1,609) (III,K1,(CRITMAT(K1,I,J),I=1,2),STDVEK(K1))
561   CONTINUE
562   CONTINUE
      CLOSE(1)

599   CONTINUE

601   FORMAT(20(F6.3,1X))
600   FORMAT(20(F16.6))
603   FORMAT(I3,10(F16.6))

608   FORMAT(3(I3),10(F16.6))

609   FORMAT(2(I3),10(F16.6))

602   FORMAT(20I4)

      STOP
      END

      SUBROUTINE GRAMMAT(NN,MM,GAM,XM,GRMAT)
C
C     CALCULATE THE GRAM-MATRIKS USING THE GAUSSIESE KERNEL
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
C
      DO 10 I=1,NNPMM-1
      GRMAT(I,I)=1.0D0
      DO 5 J=I+1,NNPMM
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XM(I,K)-XM(J,K))*(XM(I,K)-XM(J,K))
3     CONTINUE
      GRMAT(I,J)=DEXP(-GAM*S)
5     CONTINUE
10    CONTINUE
```

```
      GRMAT(NNPMM,NNPMM)=1.0D0
      DO 20 I=2,NNPMM
      DO 15 J=1,I-1
      GRMAT(I,J)=GRMAT(J,I)
15    CONTINUE
20    CONTINUE
      RETURN
      END

      SUBROUTINE GRAMNUUT(GAM,XM,XT,GRNUUT)
C
C     CALCULATE GRAM MATRIX FOR TEST DATA
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION XM(NNPMM,IP),GRNUUT(NMT,NNPMM)
      DIMENSION XT(NMT,IP)
      DO 10 I=1,NMT
      DO 5 J=1,NNPMM
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XT(I,K)-XM(J,K))*(XT(I,K)-XM(J,K))
3     CONTINUE
      GRNUUT(I,J)=DEXP(-GAM*S)
5     CONTINUE
10    CONTINUE
      RETURN
      END

      SUBROUTINE DKFDA(NN,MM,EENM,EENP,GRMAT,CPAR,
      & B,H,AKP,AKM,AN,ALPHA,BOPT)
C
C     PERFORM KFDA
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      PARAMETER (LDH=NNPMM)
      DIMENSION GRMAT(NNPMM,NNPMM)
      DIMENSION EENP(NNPMM),EENM(NNPMM)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM),AN(NNPMM,NNPMM)
      DIMENSION ALPHA(NNPMM),AKM(NNPMM),AKP(NNPMM),SOM(NNPMM)
      DO 55 J=1,NNPMM
      S1=0.0D0
      S2=0.0D0
      DO 54 I=1,NNPMM
      S1=S1+GRMAT(I,J)*EENM(I)
      S2=S2+GRMAT(I,J)*EENP(I)
54    CONTINUE
      AKM(J)=S1/NN
      AKP(J)=S2/MM
      SOM(J)=AKP(J)+AKM(J)
      B(J)=AKP(J)-AKM(J)
55    CONTINUE
      DO 58 J1=1,NNPMM
      DO 57 J2=1,NNPMM
      S=0.0D0
      DO 56 I=1,NNPMM
      S=S+GRMAT(I,J1)*GRMAT(I,J2)
```

```
56    CONTINUE
      H(J1,J2)=(S-MM*AKP(J1)*AKP(J2)-NN*AKM(J1)*AKM(J2))/NNPMM
      AN(J1,J2)=H(J1,J2)
57    CONTINUE
      H(J1,J1)=H(J1,J1)+CPAR
58    CONTINUE
      CALL DLSASF(NNPMM,H,LDH,B,ALPHA)
      AS=0.0D0
      DO 59 J=1,NNPMM
      AS=AS+ALPHA(J)*SOM(J)
59    CONTINUE
      BOPT=-0.50*AS-DLOG((1.0D0*NN)/(1.0D0*MM))

      RETURN
      END

      SUBROUTINE BERFOUT(GRNUUT,YVT,ALPHA,BOPT,FOUT)
C
C     CALCULATE TEST ERROR
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION GRNUUT(NMT,NNPMM),ALPHA(NNPMM),YVT(NMT)
      FOUT=0.0D0
      DO 10 I=1,NMT
      TOETS=1.0D0
      S=BOPT
      DO 5 J=1,NNPMM
      S=S+ALPHA(J)*GRNUUT(I,J)
5     CONTINUE

      IF (S.LT.0.0D0) TOETS=-1.0D0
      IF (DABS((YVT(I)-TOETS)).GT.0.1D0) FOUT=FOUT+1.0D0
10    CONTINUE

      FOUT=FOUT/NMT
      RETURN
      END

      SUBROUTINE BERCRITR(B,H,ALPHA,CRITR)
C
C     CALCULATE RAYLEIGH COEFFICIENT
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM),ALPHA(NNPMM)
      DIMENSION AM(NNPMM,NNPMM),ALPHAN(NNPMM),ALPHAM(NNPMM)

      DO 73 I=1,NNPMM
      DO 72 J=1,NNPMM
      AM(I,J)=B(I)*B(J)
72    CONTINUE
73    CONTINUE
      DO 75 I=1,NNPMM
      S=0.0D0
      DO 74 J=1,NNPMM
      S=S+H(I,J)*ALPHA(J)
74    CONTINUE
      ALPHAN(I)=S
```

```
75    CONTINUE
      DO 77 I=1,NNPMM
      S=0.0D0
      DO 76 J=1,NNPMM
      S=S+AM(I,J)*ALPHA(J)
76    CONTINUE
      ALPHAM(I)=S
77    CONTINUE
      S1=0.0D0
      S2=0.0D0
      DO 78 I=1,NNPMM
      S1=S1+ALPHAM(I)*ALPHA(I)
      S2=S2+ALPHAN(I)*ALPHA(I)
78    CONTINUE
      CRITR=S1/S2
      RETURN
      END

      SUBROUTINE GRAMMATTOT(GAM,XMTOTO,GRMATTOT)
C
C     THIS ROUTINE CALCULATES THE GRAM-MATRIX FOR A GIVEN DATA SET
C     USING THE GAUSSIAN KERNEL FUNCTION.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      DIMENSION XMTOTO(NMTOT,IP),GRMATTOT(NMTOT,NMTOT)
      DO 10 I=1,NMTOT-1
      GRMATTOT(I,I)=1.0D0
      DO 5 J=I+1,NMTOT
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XMTOTO(I,K)-XMTOTO(J,K))*(XMTOTO(I,K)-XMTOTO(J,K))
3     CONTINUE
      GRMATTOT(I,J)=DEXP(-GAM*S)
5     CONTINUE
10    CONTINUE
      GRMATTOT(NMTOT,NMTOT)=1.0D0
      DO 20 I=2,NMTOT
      DO 15 J=1,I-1
      GRMATTOT(I,J)=GRMATTOT(J,I)
15    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE GRAMNUUTTOT(GAM,XMTOTO,XT,GRNUUTTOT)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION XMTOTO(NMTOT,IP),GRNUUTTOT(NMT,NMTOT)
      DIMENSION XT(NMT,IP)
      DO 10 I=1,NMT
      DO 5 J=1,NMTOT
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XT(I,K)-XMTOTO(J,K))*(XT(I,K)-XMTOTO(J,K))
3     CONTINUE
      GRNUUTTOT(I,J)=DEXP(-GAM*S)
5     CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE DKFDATOT(EENMTOT,EENPTOT,GRMATTOT,CPAR,
      & BTOT,HTOT,AKP,AKM,ANTOT,ALPHATOT,BOPTTOT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      PARAMETER (LDH=NMTOT)
      DIMENSION GRMATTOT(NMTOT,NMTOT)
      DIMENSION EENPTOT(NMTOT),EENMTOT(NMTOT)
      DIMENSION BTOT(NMTOT),HTOT(NMTOT,NMTOT),ANTOT(NMTOT,NMTOT)
      DIMENSION ALPHATOT(NMTOT),AKM(NMTOT),AKP(NMTOT),SOM(NMTOT)
      DO 55 J=1,NMTOT
      S1=0.0D0
      S2=0.0D0
      DO 54 I=1,NMTOT
      S1=S1+GRMATTOT(I,J)*EENMTOT(I)
      S2=S2+GRMATTOT(I,J)*EENPTOT(I)
54    CONTINUE
      AKM(J)=S1/NTOT
      AKP(J)=S2/MTOT
      SOM(J)=AKP(J)+AKM(J)
      BTOT(J)=AKP(J)-AKM(J)
55    CONTINUE
      DO 58 J1=1,NMTOT
      DO 57 J2=1,NMTOT
      S=0.0D0
      DO 56 I=1,NMTOT
      S=S+GRMATTOT(I,J1)*GRMATTOT(I,J2)
56    CONTINUE
      HTOT(J1,J2)=(S-MTOT*AKP(J1)*AKP(J2)-NTOT*AKM(J1)*AKM(J2))/NMTOT
      ANTOT(J1,J2)=HTOT(J1,J2)
57    CONTINUE
      HTOT(J1,J1)=HTOT(J1,J1)+CPAR
58    CONTINUE
      CALL DLSASF(NMTOT,HTOT,LDH,BTOT,ALPHATOT)
      AS=0.0D0
      DO 59 J=1,NMTOT
      AS=AS+ALPHATOT(J)*SOM(J)
59    CONTINUE
      BOPTTOT=-0.50*AS-DLOG((1.0D0*NTOT)/(1.0D0*MTOT))
      RETURN
      END
      SUBROUTINE BERFOUTTOT(GRNUUTTOT,YVT,ALPHATOT,BOPTTOT,FOUT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION GRNUUTTOT(NMT,NMTOT),ALPHATOT(NMTOT),YVT(NMT)
      FOUT=0.0D0
      DO 10 I=1,NMT
      TOETS=1.0D0
      S=BOPTTOT
      DO 5 J=1,NMTOT
      S=S+ALPHATOT(J)*GRNUUTTOT(I,J)
5     CONTINUE
      IF (S.LT.0.0D0) TOETS=-1.0D0
      IF (DABS((YVT(I)-TOETS)).GT.0.1D0) FOUT=FOUT+1.0D0
10    CONTINUE

      FOUT=FOUT/NMT

      RETURN
      END
      SUBROUTINE BERALIGN(NN,MM,GRMAT,CRITAL)
C
C     CALCULATE ALIGNMENT
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NNPMM,NNPMM)
C
      S1=0.0D0
      S2=0.0D0
      S3=0.0D0
      DO 60 I=1,NN
      DO 59 J=1,NN
      S1=S1+GRMAT(I,J)
      S3=S3+GRMAT(I,J)**2.0D0
59    CONTINUE
60    CONTINUE
      DO 65 I=NN+1,NNPMM
      DO 64 J=NN+1,NNPMM
      S1=S1+GRMAT(I,J)
      S3=S3+GRMAT(I,J)**2.0D0
64    CONTINUE
65    CONTINUE
      DO 70 I=1,NN
      DO 69 J=NN+1,NNPMM
      S2=S2+2.0D0*GRMAT(I,J)
      S3=S3+2.0D0*(GRMAT(I,J)**2.0D0)
69    CONTINUE
70    CONTINUE
      CRITAL=(S1-S2)/(NNPMM*DSQRT(S3))
      RETURN
      END

      SUBROUTINE BERSOMMAT1(NN,MM,AMAT,SOM1,SOM2,SOM3)
C
C     CALCULATE SUM OF ELEMENTS OF MATRIX
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION AMAT(NNPMM,NNPMM)
      S1=0.0D0
      DO 45 I=1,NN
      DO 44 J=1,NN
      S1=S1+AMAT(I,J)
44    CONTINUE
45    CONTINUE
      SOM1=S1

      S2=0.0D0
      DO 47 I=NN+1,NNPMM
      DO 46 J=NN+1,NNPMM
      S2=S2+AMAT(I,J)
46    CONTINUE
47    CONTINUE
      SOM2=S2

      S3=0.0D0
      DO 49 I=1,NN
      DO 48 J=NN+1,NNPMM
      S3=S3+AMAT(I,J)
48    CONTINUE
```

```
49    CONTINUE
      SOM3=S3
      RETURN
      END

      SUBROUTINE BERSOMMATTOT1(AMAT,SOM1,SOM2,SOM3)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      DIMENSION AMAT(NMTOT,NMTOT)
      S1=0.0D0
      DO 45 I=1,NTOT
      DO 44 J=1,NTOT
      S1=S1+AMAT(I,J)
44    CONTINUE
45    CONTINUE
      SOM1=S1

      S2=0.0D0
      DO 47 I=NTOT+1,NMTOT
      DO 46 J=NTOT+1,NMTOT
      S2=S2+AMAT(I,J)
46    CONTINUE
47    CONTINUE
      SOM2=S2

      S3=0.0D0
      DO 49 I=1,NTOT
      DO 48 J=NTOT+1,NMTOT
      S3=S3+AMAT(I,J)
48    CONTINUE
49    CONTINUE
      SOM3=S3
      RETURN
      END

      SUBROUTINE BERSOMDIAG(AMAT,SOM)
C
C     CALCULATE SUM OF DIAGONAL ELEMENTS OF MATRIX
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION AMAT(NNPMM,NNPMM)
      S=0.0D0
      DO 45 I=1,NNPMM
      S=S+AMAT(I,I)
45    CONTINUE
      SOM=S
      RETURN
      END
      SUBROUTINE BERSOMDIAGTOT(AMAT,SOM)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      DIMENSION AMAT(NMTOT,NMTOT)
      S=0.0D0
      DO 45 I=1,NMTOT
      S=S+AMAT(I,I)
45    CONTINUE
      SOM=S
      RETURN
      END

      SUBROUTINE BERSOMDIAG1(NN,MM,AMAT,SOM1,SOM2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION AMAT(NNPMM,NNPMM)
      S1=0.0D0
      DO 45 I=1,NN
      S1=S1+AMAT(I,I)
45    CONTINUE
      SOM1=S1
      S2=0.0D0
      DO 46 I=NN+1,NNPMM
      S2=S2+AMAT(I,I)
46    CONTINUE
      SOM2=S2
      RETURN
      END

      SUBROUTINE BERSOMDIAGTOT1(AMAT,SOM1,SOM2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      DIMENSION AMAT(NMTOT,NMTOT)
      S1=0.0D0
      DO 45 I=1,NTOT
      S1=S1+AMAT(I,I)
45    CONTINUE
      SOM1=S1
      S2=0.0D0
      DO 46 I=NTOT+1,NMTOT
      S2=S2+AMAT(I,I)
46    CONTINUE
      SOM2=S2
      RETURN
      END

      SUBROUTINE BERQ(GRMAT,ALPHA,BOPT,QV,SCOREVEK)
C
C     CALCULATE DISCRIMINANT SCORES

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=4,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NNPMM,NNPMM),ALPHA(NNPMM)
      DIMENSION QV(NNPMM),SCOREVEK(NNPMM)
      DO 62 I=1,NNPMM
      S=0.0D0
      DO 61 J=1,NNPMM
      S=S+ALPHA(J)*GRMAT(I,J)

61    CONTINUE
      SCOREVEK(I)=S+BOPT
      QV(I)=S
62    CONTINUE

      RETURN
      END

      SUBROUTINE BERQTOT(GRMAT,ALPHA,BOPT,QV,SCOREVEK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NMTOT,NMTOT),ALPHA(NMTOT)
      DIMENSION QV(NMTOT),SCOREVEK(NMTOT)
      DO 62 I=1,NMTOT
     S=0.0D0
     DO 61 J=1,NMTOT
     S=S+ALPHA(J)*GRMAT(I,J)
61    CONTINUE
     SCOREVEK(I)=S+BOPT
     QV(I)=S
62    CONTINUE
     RETURN
     END

     SUBROUTINE BERMARG(NN,MM,VEK1,VEK2,AMARG)
C
C     CALCULATE MARGIN
C
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
     PARAMETER (NNPMM=NMTOT-1)
     DIMENSION VEK1(NNPMM),VEK2(NNPMM)
     DO 40 I=1,NN
     VEK2(I)=-1.0D0*VEK1(I)
40    CONTINUE
     S=0.0D0
     DO 41 I=NN+1,NNPMM
     VEK2(I)=VEK1(I)
41    CONTINUE
     S=0.0D0

     DO 45 I=1,NNPMM
     S=S+VEK2(I)
45    CONTINUE
     AMARG=S/NNPMM
     RETURN
     END

     SUBROUTINE BERMARGTOT(VEK1,VEK2,AMARG)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
     DIMENSION VEK1(NMTOT),VEK2(NMTOT)
     DO 40 I=1,NTOT
     VEK2(I)=-1.0D0*VEK1(I)
40    CONTINUE
     S=0.0D0
     DO 41 I=NTOT+1,NMTOT
     VEK2(I)=VEK1(I)
41    CONTINUE
     S=0.0D0

     DO 45 I=1,NMTOT
     S=S+VEK2(I)
45    CONTINUE
     AMARG=S/NMTOT
     RETURN
     END

     SUBROUTINE BERNORM(GRMAT,ALPHA,BOPT,WNORM)
C
C     CALCULATE NORM OF W
C
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
     PARAMETER (NNPMM=NMTOT-1)
     PARAMETER (LDH=NNPMM)
     DIMENSION GRMAT(NNPMM,NNPMM)
     DIMENSION ALPHA(NNPMM)
     S1=0.0D0
     DO 55 J=1,NNPMM
     DO 54 I=1,NNPMM
     S1=S1+ALPHA(I)*ALPHA(J)*GRMAT(I,J)
54    CONTINUE
55    CONTINUE
     WNORM=DSQRT(S1)
     RETURN
     END

     SUBROUTINE BERNORMTOT(GRMAT,ALPHA,BOPT,WNORM)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
     DIMENSION GRMAT(NMTOT,NMTOT)
     DIMENSION ALPHA(NMTOT)
     S1=0.0D0
     DO 55 J=1,NMTOT
     DO 54 I=1,NMTOT
     S1=S1+ALPHA(I)*ALPHA(J)*GRMAT(I,J)
54    CONTINUE
55    CONTINUE
     WNORM=DSQRT(S1)
     RETURN
     END

     SUBROUTINE BERMISC(DISTVEK,GEMDIST,TERR)
C
C     CALCULATE TEST ERROR
C
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
     PARAMETER (NNPMM=NMTOT-1)
     DIMENSION DISTVEK(NNPMM)
     S=0.0D0
     TERR=0.0D0
     DO 55 J=1,NNPMM
     IF (DISTVEK(J).LT.0.0D0) THEN
     TERR=TERR+1.0D0
     S=S+DISTVEK(J)
     ENDIF
55    CONTINUE

     GEMDIST=0.0D0
     IF (TERR.GT.0.0D0) GEMDIST=-1.0D0*S/TERR
     TERR=TERR/NNPMM
     RETURN
     END

     SUBROUTINE BERMISCTOT(DISTVEK,GEMDIST,TERR)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NMTOT=NTOT+MTOT)
     DIMENSION DISTVEK(NMTOT)
     S=0.0D0
     TERR=0.0D0
     DO 55 J=1,NMTOT
     IF (DISTVEK(J).LT.0.0D0) THEN
     TERR=TERR+1.0D0
     S=S+DISTVEK(J)
     ENDIF
55    CONTINUE

     GEMDIST=0.0D0
     IF (TERR.GT.0.0D0) GEMDIST=-1.0D0*S/TERR
     TERR=TERR/NMTOT
     RETURN
     END

     SUBROUTINE BERALIGNTOT(GRMAT,CRITAL)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
     DIMENSION GRMAT(NMTOT,NMTOT)
C
     S1=0.0D0
     S2=0.0D0
     S3=0.0D0
     DO 60 I=1,NTOT
     DO 59 J=1,NTOT
     S1=S1+GRMAT(I,J)
     S3=S3+GRMAT(I,J)**2.0D0
59    CONTINUE
60    CONTINUE
     DO 65 I=NTOT+1,NMTOT
     DO 64 J=NTOT+1,NMTOT
     S1=S1+GRMAT(I,J)
     S3=S3+GRMAT(I,J)**2.0D0
64    CONTINUE
65    CONTINUE
     DO 70 I=1,NTOT
     DO 69 J=NTOT+1,NMTOT
     S2=S2+2.0D0*GRMAT(I,J)
     S3=S3+2.0D0*(GRMAT(I,J)**2.0D0)
69    CONTINUE
70    CONTINUE
     CRITAL=(S1-S2)/(NMTOT*DSQRT(S3))
     RETURN
     END

     SUBROUTINE BERCRITRTOT(B,H,ALPHA,CRITR)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=5,NTOT=25,MTOT=25,NUIT=1,NMTOT=NTOT+MTOT)
     PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
     DIMENSION B(NMTOT),H(NMTOT,NMTOT),ALPHA(NMTOT)
     DIMENSION AM(NMTOT,NMTOT),ALPHAN(NMTOT),ALPHAM(NMTOT)

     DO 73 I=1,NMTOT
     DO 72 J=1,NMTOT
     AM(I,J)=B(I)*B(J)
72    CONTINUE
73    CONTINUE
     DO 75 I=1,NMTOT
     S=0.0D0
     DO 74 J=1,NMTOT
     S=S+H(I,J)*ALPHA(J)
74    CONTINUE
     ALPHAN(I)=S
75    CONTINUE
     DO 77 I=1,NMTOT
     S=0.0D0
     DO 76 J=1,NMTOT
     S=S+AM(I,J)*ALPHA(J)
76    CONTINUE
     ALPHAM(I)=S
77    CONTINUE
     S1=0.0D0
     S2=0.0D0
     DO 78 I=1,NMTOT
     S1=S1+ALPHAM(I)*ALPHA(I)
     S2=S2+ALPHAN(I)*ALPHA(I)
78    CONTINUE
     CRITR=S1/S2
     RETURN
     END
```

J. AN EXAMPLE OF THE FORTRAN PROGRAM THAT WAS USED TO PERFORM THE SIMULATION STUDY IN CHAPTER 5

```
C     DATA ARE GENERATED FROM LOGNORMAL DISTRIBUTIONS WITH DIFFERENT
C     COVARIANCE MATRICES BUT THE SAME MEAN VECTOR

C     NOTATION:
C     IP=NUMBER OF X VARIABLE
C     NN=NUMBER OF CASES IN GROUP 1
C     MM= NUMBER OF CASES IN GROUP 2
C     NNPMM=NNPMM-1
C     YV=DIE Y-VECTOR WITH +1 AND -1 LABELS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      PARAMETER (NMC=1000)
      PARAMETER (CORR=0.7D0,SIGFAKTOR=4.0D0)
      DIMENSION AMU1(IP),AMU2(IP),XMUIT(IP)
      DIMENSION SIGMAM1(IP,IP),RSIG1(IP,IP)
      DIMENSION SIGMAM2(IP,IP),RSIG2(IP,IP)
      DIMENSION SIGMAM3(IP,IP),RSIG3(IP,IP)
      DIMENSION XM1(NTOT,IP),XM2(MTOT,IP)
      DIMENSION XM(NNPMM,IP),YV(NNPMM)
      DIMENSION XT1(NT,IP),XT2(MT,IP),XT(NMT,IP),YVT(NMT)
      DIMENSION XTO(NMT,IP)
      DIMENSION GEM(IP),SA(IP),GEMUIT(IP),SAUIT(IP)
      DIMENSION GRMAT(NNPMM,NNPMM),GRNUUT(NMT,NNPMM)
      DIMENSION EENP(NNPMM),EENM(NNPMM)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM)
      DIMENSION ALPHA1(NNPMM),AKM(NNPMM),AKP(NNPMM)
      DIMENSION ALPHAKWAD(NNPMM)
      DIMENSION XMTOTO(NMTOT,IP),XMTOT(NMTOT,IP),YVTOT(NMTOT)
      DIMENSION GRMATTOT(NMTOT,NMTOT),GRNUUTTOT(NMT,NMTOT)
      DIMENSION EENPTOT(NMTOT),EENMTOT(NMTOT)
      DIMENSION BTOT(NMTOT),HTOT(NMTOT,NMTOT)
      DIMENSION ALPHA2(NMTOT),ALPHA3(NNPMM)
      DIMENSION FUNGVER2(NMTOT),AVER2(NNPMM),XMTOTQ(NMTOT,IP)
      DIMENSION GRMATTOT2(NNPMM,NMTOT)
      DIMENSION QVEK(NMTOT),XWEG(IP),GRWEG(NNPMM)
      DIMENSION POSTVEKTOT1(NMTOT),POSTVEKTOT2(NMTOT)
      DIMENSION POSTVEK1(NNPMM),POSTVEK2(NNPMM)
      DIMENSION POSTVEKWEG1(NNPMM),POSTVER2(NNPMM)
      DIMENSION QV(NNPMM),SCOREVEK(NNPMM)
      DIMENSION DISTVEK(NNPMM),AMARGVEK(NNPMM)
      DIMENSION AN(NNPMM,NNPMM)
      DIMENSION QVTOT(NMTOT),SCOREVEKTOT(NMTOT)
      DIMENSION SCOREVEKWEG(NNPMM),SCOREVER2(NNPMM)

      DIMENSION FREKMAT(NMTOT,9),FREKMAT1(NMTOT,9)
      DIMENSION IUITVEK(9),IUITUNIEK(9)
      DIMENSION FOUTVEK(11),STDVEK(11),SKWADVEK(11)

      CHARACTER*70 FILEOUT1,FILEOUT2,FILEOUT3,FILEOUT4

      FILEOUT1='l5c7gs4foute.dnuut'
      FILEOUT2='l5c7gs4uitskieter.dnuut'
      FILEOUT3='l5c7gs4allefrek.dnuut'
      FILEOUT4='l5c7gs4uitfrek.dnuut'


C     CONSTRUCT PARAMETERS.

      CPAR=DEXP(-11.0D0)

      E=DEXP(1.0D0)
      BLAM=DSQRT(1.0D0/(E*(E-1.0D0)))
      EP=-1.0D0*DSQRT(1.0D0/(E-1.0D0))
      VARFAKTOR=SIGFAKTOR*SIGFAKTOR
      COV1=VARFAKTOR*DLOG(CORR*(E-1)+1.0D0)
      COV2=DLOG(CORR*(E-1.0D0)+1.0D0)
C
      DO 3 I=1,IP
      AMU1(I)=0.0D0
      AMU2(I)=0.0D0
      DO 2 J=1,IP
      SIGMAM1(I,J)=COV1
      SIGMAM2(I,J)=COV2
2     CONTINUE
      SIGMAM1(I,I)=VARFAKTOR
      SIGMAM2(I,I)=1.0D0
3     CONTINUE
C
      TOL=1.0D2*DMACH(4)
      CALL DCHFAC(IP,SIGMAM1,IP,TOL,IRANK,RSIG1,IP)
      CALL DCHFAC(IP,SIGMAM2,IP,TOL,IRANK,RSIG2,IP)

C     CONSTRUCT RESPONSE VECTOR

      DO 6 I=1,NTOT
      YVTOT(I)=-1.0D0
6     CONTINUE
      DO 7 I=NTOT+1,NMTOT
      YVTOT(I)=1.0D0
7     CONTINUE

C     CONSTRUCT INDICATOR VECTOR
C
      DO 8 I=1,NMTOT
      EENPTOT(I)=0.0D0
      EENMTOT(I)=0.0D0
      IF (YVTOT(I).LT.-0.1D0) EENMTOT(I)=1.0D0
      IF (YVTOT(I).GT.0.1D0) EENPTOT(I)=1.0D0
8     CONTINUE

      DO 599 III=1,2

      SIGUIT=SIGFAKTOR+(III-1.0D0)
      VARUIT=SIGUIT*SIGUIT

      DO 9 I=1,IP
      DO 890 J=1,IP
      SIGMAM3(I,J)=VARUIT*DLOG(CORR*(E-1)+1.0D0)
890   CONTINUE
      SIGMAM3(I,I)=VARUIT
9     CONTINUE

      CALL DCHFAC(IP,SIGMAM3,IP,TOL,IRANK,RSIG3,IP)

      DO 888 I=1,NMTOT
      DO 887 J=1,9
      FREKMAT(I,J)=0.0D0
      FREKMAT1(I,J)=0.0D0

887   CONTINUE
888   CONTINUE

      DO 889 J=1,11
      FOUTVEK(J)=0.0D0
      SKWADVEK(J)=0.0D0
      STDVEK(J)=0.0D0
889   CONTINUE

C     BEGINNING OF SIMULATION LOOP

      DO 590 MC=1,NMC
      WRITE(6,*) MC

C     GENERATE THE TRAINING DATA
C
      CALL DRNMVN(NTOT,IP,RSIG1,IP,XM1,NTOT)
      DO 11 I=1,NTOT
      DO 10 J=1,IP
      XMTOTO(I,J)=SIGFAKTOR*((BLAM*DEXP(XM1(I,J)/SIGFAKTOR))+EP)

10    CONTINUE
11    CONTINUE
C
C     GROUP 1 ARE FINISHED.

C     GENERATE DATA FROM GROUP 2

      CALL DRNMVN(MTOT,IP,RSIG2,IP,XM2,MTOT)
      DO 13 I=1,MTOT-NUIT
      DO 12 J=1,IP
      XMTOTO(NTOT+I,J)=(BLAM*DEXP(XM2(I,J)))+EP

12    CONTINUE
13    CONTINUE

C     INSERT ATYPICAL CASE

      CALL DRNMVN(1,IP,RSIG3,IP,XMUIT,1)

      DO 15 I=NMTOT-NUIT+1,NMTOT
      DO 14 J=1,IP
      XMTOTO(I,J)=SIGUIT*((BLAM*DEXP(XMUIT(J)/SIGUIT))+EP)

14    CONTINUE
15    CONTINUE

C     GENERATE TEST DATA AND RESPONSE VECTOR

      CALL DRNMVN(NT,IP,RSIG1,IP,XT1,NT)
      CALL DRNMVN(MT,IP,RSIG2,IP,XT2,MT)
      DO 17 I=1,NT
      DO 16 J=1,IP
      XTO(I,J)=SIGFAKTOR*((BLAM*DEXP(XT1(I,J)/SIGFAKTOR))+EP)

16    CONTINUE
17    CONTINUE
      DO 19 I=1,MT
      DO 18 J=1,IP
      XTO(NT+I,J)=(BLAM*DEXP(XT2(I,J)))+EP
18    CONTINUE
19    CONTINUE
      DO 20 I=1,NT
      YVT(I)=-1.0D0
20    CONTINUE
      DO 21 I=NT+1,NMT
      YVT(I)=1.0D0
21    CONTINUE

      GAM=1.0D0/IP

C     CALCULATE TEST ERROR USING ALL THE DATA

      DO 24 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 23 I=1,NMTOT
      S1=S1+XMTOTO(I,J)
      S2=S2+XMTOTO(I,J)*XMTOTO(I,J)
23    CONTINUE
      GEM(J)=S1/NMTOT
      SA(J)=DSQRT((S2-NMTOT*GEM(J)*GEM(J))/(NMTOT-1))
24    CONTINUE
      DO 26 J=1,IP
      DO 25 I=1,NMTOT
      XMTOT(I,J)=(XMTOTO(I,J)-GEM(J))/SA(J)
25    CONTINUE
26    CONTINUE
C
C     STANDARDISE THE TEST DATA
C
      DO 28 J=1,IP
      DO 27 I=1,NMT
      XT(I,J)=(XTO(I,J)-GEM(J))/SA(J)
27    CONTINUE
28    CONTINUE

      CALL GRAMMATTOT(GAM,XMTOT,GRMATTOT)
      CALL DKFDATOT(EENMTOT,EENPTOT,GRMATTOT,
      &CPAR,BTOT,HTOT,ALPHA2,BOPT2)
      CALL GRAMNUUTTOT(GAM,XMTOT,XT,GRNUUTTOT)
      CALL BERQTOT(GRMATTOT,ALPHA2,BOPT2,QVTOT,SCOREVEKTOT)
      CALL BERFOUTTOT(GRNUUTTOT,YVT,ALPHA2,BOPT2,FOUT)

      FOUTVEK(11)=FOUTVEK(11)+FOUT
      SKWADVEK(11)=SKWADVEK(11)+(FOUT**2)

      CALL POSTPROBTOT(QVTOT,POSTVEKTOT1)

C     LEAVE OUT ONE CASE AT A TIME
C
      AMAX1=-1.0D0
      AMAX2=-1.0D0
      AMAX3=-1.0D0
      AMAX4=-1.0D0
      AMAX5=-1.0D0
      AMAX6=-1.0D0
      AMIN1=1.00D10
      AMIN2=1.00D10
      AMIN3=1.00D10

      DO 80 II=1,NMTOT
      CALL WEGLAAT(II,IP,XMTOTO,XM)

      CALL WEGLAAT1(II,SCOREVEKTOT,SCOREVEKWEG)
      CALL WEGLAAT1(II,POSTVEKTOT1,POSTVEKWEG1)

      IF (II.LE.NTOT) THEN
      NN=NTOT-1
      MM=MTOT
      ENDIF
      IF (II.GT.NTOT) THEN
      NN=NTOT
      MM=MTOT-1
      ENDIF
C
      DO 44 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 43 I=1,NNPMM
      S1=S1+XM(I,J)
      S2=S2+XM(I,J)*XM(I,J)
43    CONTINUE
      GEM(J)=S1/NNPMM
      SA(J)=DSQRT((S2-NNPMM*GEM(J)*GEM(J))/(NNPMM-1))
44    CONTINUE
      DO 46 J=1,IP
      DO 45 I=1,NNPMM
      XM(I,J)=(XM(I,J)-GEM(J))/SA(J)
45    CONTINUE
46    CONTINUE
C
      DO 47 J=1,IP
      XWEG(J)=(XMTOTO(II,J)-GEM(J))/SA(J)
47    CONTINUE
C
      DO 54 J=1,IP
      DO 53 I=1,NMT
      XT(I,J)=(XTO(I,J)-GEM(J))/SA(J)
53    CONTINUE
54    CONTINUE
C
      DO 58 I=1,NN
      YV(I)=-1.0D0
58    CONTINUE
      DO 59 I=NN+1,NNPMM
      YV(I)=1.0D0
59    CONTINUE
C
      DO 60 I=1,NNPMM
      EENP(I)=0.0D0
      EENM(I)=0.0D0
      IF (YV(I).LT.-0.1D0) EENM(I)=1.0D0
      IF (YV(I).GT.0.1D0) EENP(I)=1.0D0
60    CONTINUE
C
C     CALCULATE THE KFD CLASSIFIER WITHOUT THE OMITTED CASE
C     AND CALCULATE CRITERIA

      CALL GRAMMAT(NN,MM,GAM,XM,GRMAT)
      CALL DKFDA(NN,MM,EENM,EENP,GRMAT,CPAR,B,H,AKP,AKM,AN,ALPHA1,BOPT1)
      CALL GRAMWEG(GAM,XM,XWEG,GRWEG)
      CALL BERSOMMAT1(NN,MM,GRMAT,GRSOM1,GRSOM2,GRSOM3)
      CALL BERSOMDIAG1(NN,MM,GRMAT,DIAGSOM1,DIAGSOM2)
      DIST1=DIAGSOM1/NN-GRSOM1/(NN*NN)
      DIST2=DIAGSOM2/MM-GRSOM2/(MM*MM)
      DISTCENT=GRSOM1/(NN*NN)+GRSOM2/(MM*MM)-(2.0D0*GRSOM3)/(NN*MM)
      RCRIT=DISTCENT/(DIST1+DIST2)

      CALL BERQ(GRMAT,GRWEG,ALPHA1,BOPT1,QV,SCOREVEK,QWEG,SCOREWEG)
      CALL BERNORM(GRMAT,ALPHA1,BOPT1,WNORM)
      CALL BERMARG(NN,MM,SCOREVEK,AMARGVEK,AMARG)

      DO 65 KKK=1,NNPMM
      DISTVEK(KKK)=AMARGVEK(KKK)/WNORM
65    CONTINUE

      CALL BERMISC(DISTVEK,GEMDIST,TERR)
      CALL BERCRITR(B,H,ALPHA1,CPAR,CRITR)
      CALL BERALIGN(NN,MM,GRMAT,CRITAL)
      CALL POSTPROB(NN,MM,QV,QWEG,POSTVEK1,POSTWEG1)

      S1=0.0D0
      DO 66 I=1,NN
      S1=S1+(1.0D0-POSTVEK1(I))**2
66    CONTINUE
      S2=0.0D0
      DO 67 I=NN+1,NNPMM
      S2=S2+(0.0D0-POSTVEK1(I))**2
67    CONTINUE
      CRITP=S1+S2

      DO 72 I=1,NNPMM
      SCOREVER2(I)=(SCOREVEKWEG(I)-SCOREVEK(I))**2
72    CONTINUE

      S3=0.0D0
      DO 75 I=1,NNPMM
      S3=S3+SCOREVER2(I)
75    CONTINUE

      S4=S3+(SCOREVEKTOT(II)-SCOREWEG)**2
      CRITF=S4/NMTOT

      IF (CRITR.GE.AMAX1) THEN
      AMAX1=CRITR
      IUITR=II
      IUITVEK(6)=II
      ENDIF

      IF (CRITAL.GE.AMAX2) THEN
      AMAX2=CRITAL
      IUITAL=II
      IUITVEK(7)=II
      ENDIF

      IF (CRITF.GE.AMAX3) THEN
      AMAX3=CRITF
      IUITF=II
      IUITVEK(8)=II
      ENDIF

      IF (RCRIT.GE.AMAX4) THEN
      AMAX4=RCRIT
      IUITRCRIT=II
      IUITVEK(1)=II
      ENDIF

      IF (WNORM.GE.AMAX5) THEN
      AMAX5=WNORM
      IUITWNORM=II
      IUITVEK(2)=II
      ENDIF

      IF (AMARG.GE.AMAX6) THEN
      AMAX6=AMARG
      IUITAMARG=II
      IUITVEK(3)=II
      ENDIF

      IF (CRITP.LE.AMIN1) THEN
      AMIN1=CRITP
      IUITP=II
      IUITVEK(9)=II
      ENDIF

      IF (GEMDIST.LE.AMIN2) THEN
      AMIN2=GEMDIST
      IUITGEMDIST=II
      IUITVEK(4)=II
      ENDIF

      IF (TERR.LE.AMIN3) THEN
      AMIN3=TERR
      IUITTERR=II
      IUITVEK(5)=II
      ENDIF

80    CONTINUE

      IUITUNIEK(1)=IUITVEK(1)
      ITEL=1
      DO 82 I=2,9
      IW=0

      DO 81 J=1,ITEL
      IF(IUITVEK(I).EQ.IUITUNIEK(J)) IW=1
81    CONTINUE

      IF(IW.EQ.0) THEN
      ITEL=ITEL+1
      IUITUNIEK(ITEL)=IUITVEK(I)
      ENDIF

82    CONTINUE

      DO 90 K=1,9
      FREKMAT1(IUITVEK(K),K)=FREKMAT1(IUITVEK(K),K)+1.0D0
90    CONTINUE

      OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')

      WRITE(1,602) (IUITVEK(J),J=1,9)
      WRITE(1,602) ITEL
      WRITE(1,602) (IUITUNIEK(J),J=1,ITEL)

      WRITE(1,*)
      CLOSE(1)
      DO 444 JJJ=1,ITEL
C
      CALL WEGLAAT(IUITUNIEK(JJJ),IP,XMTOTO,XM)

      IF (IUITUNIEK(JJJ).LE.NTOT) THEN
      NN=NTOT-1
      MM=MTOT
      ENDIF

      IF (IUITUNIEK(JJJ).GT.NTOT) THEN
      NN=NTOT
      MM=MTOT-1
      ENDIF

      DO 94 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 93 I=1,NNPMM
      S1=S1+XM(I,J)
      S2=S2+XM(I,J)*XM(I,J)
93    GEM(J)=S1/NNPMM
      SA(J)=DSQRT((S2-NNPMM*GEM(J)*GEM(J))/(NNPMM-1))
94    CONTINUE
      DO 96 J=1,IP
      DO 95 I=1,NNPMM
      XM(I,J)=(XM(I,J)-GEM(J))/SA(J)
95    CONTINUE
96    CONTINUE
C
      DO 104 J=1,IP
      DO 103 I=1,NMT
      XT(I,J)=(XTO(I,J)-GEM(J))/SA(J)
103   CONTINUE
104   CONTINUE

      DO 108 I=1,NN
      YV(I)=-1.0D0
108   CONTINUE
      DO 109 I=NN+1,NNPMM
      YV(I)=1.0D0
109   CONTINUE
C
      DO 110 I=1,NNPMM
      EENP(I)=0.0D0
      EENM(I)=0.0D0
      IF (YV(I).LT.-0.1D0) EENM(I)=1.0D0
      IF (YV(I).GT.0.1D0) EENP(I)=1.0D0
110   CONTINUE

      CALL GRAMMAT(NN,MM,GAM,XM,GRMAT)
      CALL DKFDA(NN,MM,EENM,EENP,GRMAT,CPAR,B,H,AKP,AKM,AN,ALPHA1,BOPT1)
      CALL GRAMNUUT(GAM,XM,XT,GRNUUT)
      CALL BERFOUT(GRNUUT,YVT,ALPHA1,BOPT1,FOUT)

      DO 120 K=1,9
      IF(IUITVEK(K).EQ.IUITUNIEK(JJJ)) THEN
      FOUTVEK(K)=FOUTVEK(K)+FOUT
      SKWADVEK(K)=SKWADVEK(K)+(FOUT**2)
      ENDIF
120   CONTINUE

444   CONTINUE
```

```
C     CALCULATE THE TEST ERROR WITHOUT THE ATYPICAL CASE

      CALL WEGLAAT(NMTOT,IP,XMTOTO,XM)

      NN=NTOT
      MM=MTOT-1

      DO 571 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 570 I=1,NNPMM
      S1=S1+XM(I,J)
      S2=S2+XM(I,J)*XM(I,J)
570   GEM(J)=S1/NNPMM
      SA(J)=DSQRT((S2-NNPMM*GEM(J)*GEM(J))/(NNPMM-1))
571   CONTINUE
      DO 573 J=1,IP
      DO 572 I=1,NNPMM
      XM(I,J)=(XM(I,J)-GEM(J))/SA(J)
572   CONTINUE
573   CONTINUE
C
      DO 575 J=1,IP
      DO 574 I=1,NMT
      XT(I,J)=(XTO(I,J)-GEM(J))/SA(J)
574   CONTINUE
575   CONTINUE

      DO 578 I=1,NN
      YV(I)=-1.0D0
578   CONTINUE
      DO 579 I=NN+1,NNPMM
      YV(I)=1.0D0
579   CONTINUE
C
      DO 580 I=1,NNPMM
      EENP(I)=0.0D0
      EENM(I)=0.0D0
      IF (YV(I).LT.-0.1D0) EENM(I)=1.0D0
      IF (YV(I).GT.0.1D0) EENP(I)=1.0D0
580   CONTINUE

      CALL GRAMMAT(NN,MM,GAM,XM,GRMAT)
      CALL DKFDA(NN,MM,EENM,EENP,GRMAT,CPAR,B,H,AKP,AKM,AN,ALPHA1,BOPT1)
      CALL GRAMNUUT(GAM,XM,XT,GRNUUT)
      CALL BERFOUT(GRNUUT,YVT,ALPHA1,BOPT1,FOUT)
      FOUTVEK(10)=FOUTVEK(10)+FOUT
      SKWADVEK(10)=SKWADVEK(10)+(FOUT**2)

590   CONTINUE

      DO 591 J=1,11
      FOUTVEK(J)=FOUTVEK(J)/NMC
591   CONTINUE

      DO 592 J=1,11
      STDVEK(J)=DSQRT(((SKWADVEK(J)-NMC*(FOUTVEK(J)**2))/(NMC**2)))
592   CONTINUE

      DO 595 I=1,NMTOT
      DO 594 J=1,9
C     FREKMAT(I,J)=FREKMAT(I,J)/NMC
      FREKMAT1(I,J)=FREKMAT1(I,J)/NMC
```

```
594   CONTINUE
595   CONTINUE

      OPEN(1,FILE=FILEOUT3,ACCESS='APPEND')
      DO 596 I=1,NMTOT
      WRITE(1,603) (FREKMAT1(I,J),J=1,9)

596   CONTINUE
      WRITE(1,*)
      WRITE(1,*)
      CLOSE(1)

      OPEN(1,FILE=FILEOUT4,ACCESS='APPEND')
      WRITE(1,603) (FREKMAT1(NMTOT,J),J=1,9)
      WRITE(1,*)
      WRITE(1,*)
      CLOSE(1)

      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
      WRITE(1,600) (FOUTVEK(J),J=1,11)
      WRITE(1,*)
      WRITE(1,600) (STDVEK(J),J=1,11)
      WRITE(1,*)
      CLOSE(1)
599   CONTINUE

601   FORMAT(11(F6.3,1X))
600   FORMAT(11(F9.7,1X))
603   FORMAT(11(F6.3))
602   FORMAT(11I4)

      STOP
      END

      SUBROUTINE GRAMMAT(NN,MM,GAMPAR,XM,GRMAT)
C
C     THIS ROUTINE CALCULATES THE GRAM MATRIX USING THE GUASSIAN KERNEL
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
C
      DO 10 I=1,NNPMM-1
      GRMAT(I,I)=1.0D0
      DO 5 J=I+1,NNPMM
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XM(I,K)-XM(J,K))*(XM(I,K)-XM(J,K))
3     CONTINUE
      GRMAT(I,J)=DEXP(-GAMPAR*S)
5     CONTINUE
10    CONTINUE
      GRMAT(NNPMM,NNPMM)=1.0D0
C
      DO 20 I=2,NNPMM
      DO 15 J=1,I-1
      GRMAT(I,J)=GRMAT(J,I)
15    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE GRAMNUUT(GAMPAR,XM,XT,GRNUUT)
C
C     THIS ROUTINE CALCULATES THE GRAM MATRIX FOR THE TEST DATA USING A
C     GAUSSIAN KERNEL
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION XM(NNPMM,IP),GRNUUT(NMT,NNPMM)
      DIMENSION XT(NMT,IP)
      DO 10 I=1,NMT
      DO 5 J=1,NNPMM
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XT(I,K)-XM(J,K))*(XT(I,K)-XM(J,K))
3     CONTINUE
      GRNUUT(I,J)=DEXP(-GAMPAR*S)
5     CONTINUE
10    CONTINUE
      RETURN
      END

      SUBROUTINE GRAMWEG(GAMPAR,XM,XWEG,GRWEG)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION XM(NNPMM,IP),GRWEG(NNPMM)
      DIMENSION XWEG(IP)
      DO 5 J=1,NNPMM
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XWEG(K)-XM(J,K))*(XWEG(K)-XM(J,K))
3     CONTINUE
      GRWEG(J)=DEXP(-GAMPAR*S)
5     CONTINUE
      RETURN
      END

      SUBROUTINE DKFDA(NN,MM,EENM,EENP,GRMAT,CPAR,
      & B,H,AKP,AKM,AN,ALPHA,BOPT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      PARAMETER (LDH=NNPMM)
      DIMENSION GRMAT(NNPMM,NNPMM)
      DIMENSION EENP(NNPMM),EENM(NNPMM)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM),AN(NNPMM,NNPMM)
      DIMENSION ALPHA(NNPMM),AKM(NNPMM),AKP(NNPMM),SOM(NNPMM)
      DO 55 J=1,NNPMM
      S1=0.0D0
      S2=0.0D0
      DO 54 I=1,NNPMM
      S1=S1+GRMAT(I,J)*EENM(I)
      S2=S2+GRMAT(I,J)*EENP(I)
54    CONTINUE
      AKM(J)=S1/NN
      AKP(J)=S2/MM
      SOM(J)=AKP(J)+AKM(J)
      B(J)=AKP(J)-AKM(J)
55    CONTINUE
      DO 58 J1=1,NNPMM
      DO 57 J2=1,NNPMM
      S=0.0D0
      DO 56 I=1,NNPMM
      S=S+GRMAT(I,J1)*GRMAT(I,J2)
56    CONTINUE
      H(J1,J2)=(S-MM*AKP(J1)*AKP(J2)-NN*AKM(J1)*AKM(J2))/NNPMM
      AN(J1,J2)=H(J1,J2)
57    CONTINUE
      H(J1,J1)=H(J1,J1)+CPAR
58    CONTINUE
      CALL DLSASF(NNPMM,H,LDH,B,ALPHA)
      AS=0.0D0
      DO 59 J=1,NNPMM
      AS=AS+ALPHA(J)*SOM(J)
59    CONTINUE
      BOPT=-0.50*AS-DLOG((1.0D0*NN)/(1.0D0*MM))
      RETURN
      END

      SUBROUTINE BERFOUT(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION GRNUUT(NMT,NNPMM),ALPHA(NNPMM),YVT(NMT)
      FOUT=0.0D0
      DO 10 I=1,NMT
      TOETS=1.0D0
      S=BOPT
      DO 5 J=1,NNPMM
      S=S+ALPHA(J)*GRNUUT(I,J)
5     CONTINUE

      IF (S.LT.0.0D0) TOETS=-1.0D0
      IF (DABS((YVT(I)-TOETS)).GT.0.1D0) FOUT=FOUT+1.0D0
10    CONTINUE
      FOUT=FOUT/NMT
      RETURN
      END

      SUBROUTINE BERCRITR(B,H,ALPHA,CPAR,CRITR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM),ALPHA(NNPMM)
      DIMENSION AM(NNPMM,NNPMM),ALPHAN(NNPMM),ALPHAM(NNPMM)
      DO 70 J1=1,NNPMM
      H(J1,J1)=H(J1,J1)-CPAR
70    CONTINUE
      DO 73 I=1,NNPMM
      DO 72 J=1,NNPMM
      AM(I,J)=B(I)*B(J)
72    CONTINUE
73    CONTINUE
      DO 75 I=1,NNPMM
      S=0.0D0
      DO 74 J=1,NNPMM
      S=S+H(I,J)*ALPHA(J)
74    CONTINUE
```

```
      ALPHAN(I)=S
75    CONTINUE
      DO 77 I=1,NNPMM
      S=0.0D0
      DO 76 J=1,NNPMM
      S=S+AM(I,J)*ALPHA(J)
76    CONTINUE
      ALPHAM(I)=S
77    CONTINUE
      S1=0.0D0
      S2=0.0D0
      DO 78 I=1,NNPMM
      S1=S1+ALPHAM(I)*ALPHA(I)
      S2=S2+ALPHAN(I)*ALPHA(I)
78    CONTINUE
      CRITR=S1/S2
      RETURN
      END

      SUBROUTINE WEGLAAT(II,NKOL,X,X1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION X(NMTOT,NKOL),X1(NNPMM,NKOL)
      N=NMTOT
      IF (II.EQ.1) THEN
      DO 5 I=1,N-1
      DO 1 J=1,NKOL
      X1(I,J)=X(I+1,J)
1     CONTINUE
5     CONTINUE
      ENDIF
      IF ((II.GT.1).AND.(II.LT.N)) THEN
      DO 15 I=1,II-1
      DO 10 J=1,NKOL
      X1(I,J)=X(I,J)
10    CONTINUE
15    CONTINUE
      DO 25 I=II,N-1
      DO 20 J=1,NKOL
      X1(I,J)=X(I+1,J)
20    CONTINUE
25    CONTINUE
      ENDIF
      IF (II.EQ.N) THEN
      DO 35 I=1,N-1
      DO 30 J=1,NKOL
      X1(I,J)=X(I,J)
30    CONTINUE
35    CONTINUE
      ENDIF
      RETURN
      END

      SUBROUTINE GRAMMATTOT(GAMPAR,XMTOTO,GRMATTOT)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      DIMENSION XMTOTO(NMTOT,IP),GRMATTOT(NMTOT,NMTOT)
C
      DO 10 I=1,NMTOT-1
      GRMATTOT(I,I)=1.0D0
      DO 5 J=I+1,NMTOT
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XMTOTO(I,K)-XMTOTO(J,K))*(XMTOTO(I,K)-XMTOTO(J,K))
3     CONTINUE
      GRMATTOT(I,J)=DEXP(-GAMPAR*S)
5     CONTINUE
10    CONTINUE
      GRMATTOT(NMTOT,NMTOT)=1.0D0
C
      DO 20 I=2,NMTOT
      DO 15 J=1,I-1
      GRMATTOT(I,J)=GRMATTOT(J,I)
15    CONTINUE
20    CONTINUE
      RETURN
      END

      SUBROUTINE GRAMNUUTTOT(GAMPAR,XMTOTO,XT,GRNUUTTOT)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION XMTOTO(NMTOT,IP),GRNUUTTOT(NMT,NMTOT)
      DIMENSION XT(NMT,IP)
      DO 10 I=1,NMT
      DO 5 J=1,NMTOT
      S=0.0D0
      DO 3 K=1,IP
      S=S+(XT(I,K)-XMTOTO(J,K))*(XT(I,K)-XMTOTO(J,K))
3     CONTINUE
      GRNUUTTOT(I,J)=DEXP(-GAMPAR*S)
5     CONTINUE
10    CONTINUE
      RETURN
      END

      SUBROUTINE DKFDATOT(EENMTOT,EENPTOT,GRMATTOT,CPAR,
      & BTOT,HTOT,ALPHA2,BOPT2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      PARAMETER (LDH=NMTOT)
      DIMENSION GRMATTOT(NMTOT,NMTOT)
      DIMENSION EENPTOT(NMTOT),EENMTOT(NMTOT)
      DIMENSION BTOT(NMTOT),HTOT(NMTOT,NMTOT)
      DIMENSION ALPHA2(NMTOT),AKM(NMTOT),AKP(NMTOT),SOM(NMTOT)
      DO 55 J=1,NMTOT
      S1=0.0D0
      S2=0.0D0
      DO 54 I=1,NMTOT
      S1=S1+GRMATTOT(I,J)*EENMTOT(I)
      S2=S2+GRMATTOT(I,J)*EENPTOT(I)
54    CONTINUE
      AKM(J)=S1/NTOT
      AKP(J)=S2/MTOT
      SOM(J)=AKP(J)+AKM(J)
      BTOT(J)=AKP(J)-AKM(J)
55    CONTINUE
      DO 58 J1=1,NMTOT
      DO 57 J2=1,NMTOT
      S=0.0D0
      DO 56 I=1,NMTOT
      S=S+GRMATTOT(I,J1)*GRMATTOT(I,J2)
56    CONTINUE
      HTOT(J1,J2)=(S-MTOT*AKP(J1)*AKP(J2)-NTOT*AKM(J1)*AKM(J2))/NMTOT
57    CONTINUE
      HTOT(J1,J1)=HTOT(J1,J1)+CPAR
58    CONTINUE
      CALL DLSASF(NMTOT,HTOT,LDH,BTOT,ALPHA2)
      AS=0.0D0
      DO 59 J=1,NMTOT
      AS=AS+ALPHA2(J)*SOM(J)
59    CONTINUE
      BOPT2=-0.50*AS-DLOG((1.0D0*NTOT)/(1.0D0*MTOT))
      RETURN
      END

      SUBROUTINE BERFOUTTOT(GRNUUTTOT,YVT,ALPHA2,BOPT2,FOUT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NT=5000,MT=5000,NMT=NT+MT)
      DIMENSION GRNUUTTOT(NMT,NMTOT),ALPHA2(NMTOT),YVT(NMT)
      FOUT=0.0D0
      DO 10 I=1,NMT
      TOETS=1.0D0
      S=BOPT2
      DO 5 J=1,NMTOT
      S=S+ALPHA2(J)*GRNUUTTOT(I,J)
5     CONTINUE
      IF (S.LT.0.0D0) TOETS=-1.0D0
      IF (DABS((YVT(I)-TOETS)).GT.0.1D0) FOUT=FOUT+1.0D0
10    CONTINUE

      FOUT=FOUT/NMT
      RETURN
      END

      SUBROUTINE BERALIGN(NN,MM,GRMAT,CRITAL)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NNPMM,NNPMM)
C
      S1=0.0D0
      S2=0.0D0
      S3=0.0D0
      DO 60 I=1,NN
      DO 59 J=1,NN
      S1=S1+GRMAT(I,J)
      S3=S3+GRMAT(I,J)**2.0D0
59    CONTINUE
60    CONTINUE
      DO 65 I=NN+1,NNPMM
      DO 64 J=NN+1,NNPMM
      S1=S1+GRMAT(I,J)
      S3=S3+GRMAT(I,J)**2.0D0
64    CONTINUE
65    CONTINUE
      DO 70 I=1,NN
      DO 69 J=NN+1,NNPMM
      S2=S2+2.0D0*GRMAT(I,J)
      S3=S3+2.0D0*(GRMAT(I,J)**2.0D0)
69    CONTINUE
70    CONTINUE
      CRITAL=(S1-S2)/(NNPMM*DSQRT(S3))
      RETURN
      END

      SUBROUTINE WEGLAAT1(II,A,A1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)

      DIMENSION A(NMTOT),A1(NNPMM)
      N=NMTOT
      IF (II.EQ.1) THEN
      DO 5 I=1,N-1
      A1(I)=A(I+1)
5     CONTINUE
      ENDIF
      IF ((II.GT.1).AND.(II.LT.N)) THEN
      DO 15 I=1,II-1
      A1(I)=A(I)
15    CONTINUE
      DO 25 I=II,N-1
      A1(I)=A(I+1)
25    CONTINUE
      ENDIF
      IF (II.EQ.N) THEN
      DO 35 I=1,N-1
      A1(I)=A(I)
35    CONTINUE
      ENDIF
      RETURN
      END

      SUBROUTINE POSTPROBTOT(QVEK,POSTVEK1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      DIMENSION QVEK(NMTOT)
      DIMENSION PVEK1(NMTOT),PVEK2(NMTOT)
      DIMENSION PSOMVEK(NMTOT),POSTVEK1(NMTOT),POSTVEK2(NMTOT)
      S1=0.0D0
      S2=0.0D0
      DO 3 I=1,NTOT
      S1=S1+QVEK(I)
      S2=S2+QVEK(I)*QVEK(I)
3     CONTINUE

      GEM1=S1/NTOT
      VAR1=(S2-NTOT*GEM1*GEM1)/(NTOT-1)
      S1=0.0D0
      S2=0.0D0
      DO 4 I=NTOT+1,NMTOT
      S1=S1+QVEK(I)
      S2=S2+QVEK(I)*QVEK(I)
4     CONTINUE
      GEM2=S1/MTOT
      VAR2=(S2-MTOT*GEM2*GEM2)/(MTOT-1)
      DO 5 I=1,NMTOT
      PVEK1(I)=(DEXP(-1.0D0*((QVEK(I)-GEM1)**2)/(2.0D0*VAR1)))
     &/DSQRT((44.0D0*VAR1)/7.0D0)
      PVEK2(I)=(DEXP(-1.0D0*((QVEK(I)-GEM2)**2)/(2.0D0*VAR2)))
     &/DSQRT((44.0D0*VAR2)/7.0D0)
      PSOMVEK(I)=PVEK1(I)+PVEK2(I)
      POSTVEK1(I)=PVEK1(I)/PSOMVEK(I)
      POSTVEK2(I)=PVEK2(I)/PSOMVEK(I)
5     CONTINUE
```

```
      RETURN
      END

      SUBROUTINE POSTPROB(NN,MM,QVEK,QWEG,POSTVEK1,POSTWEG1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION QVEK(NNPMM)
      DIMENSION PVEK1(NNPMM),PVEK2(NNPMM)
      DIMENSION PSOMVEK(NNPMM),POSTVEK1(NNPMM),POSTVEK2(NNPMM)
      S1=0.0D0
      S2=0.0D0
      DO 3 I=1,NN
      S1=S1+QVEK(I)
      S2=S2+QVEK(I)*QVEK(I)
3     CONTINUE

      GEM1=S1/NN
      VAR1=(S2-NN*GEM1*GEM1)/(NN-1)
      S1=0.0D0
      S2=0.0D0
      DO 4 I=NN+1,NNPMM
      S1=S1+QVEK(I)
      S2=S2+QVEK(I)*QVEK(I)
4     CONTINUE
      GEM2=S1/MM
      VAR2=(S2-MM*GEM2*GEM2)/(MM-1)

      DO 5 I=1,NNPMM
      PVEK1(I)=(DEXP(-1.0D0*((QVEK(I)-GEM1)**2)/(2.0D0*VAR1)))
      &/DSQRT((44.0D0*VAR1)/7.0D0)
      PVEK2(I)=(DEXP(-1.0D0*((QVEK(I)-GEM2)**2)/(2.0D0*VAR2)))
      &/DSQRT((44.0D0*VAR2)/7.0D0)
      PSOMVEK(I)=PVEK1(I)+PVEK2(I)
      POSTVEK1(I)=PVEK1(I)/PSOMVEK(I)
      POSTVEK2(I)=PVEK2(I)/PSOMVEK(I)
5     CONTINUE

      PWEG1=(DEXP(-1.0D0*((QWEG-GEM1)**2)/(2.0D0*VAR1)))
      &/DSQRT((44.0D0*VAR1)/7.0D0)
      PWEG2=(DEXP(-1.0D0*((QWEG-GEM2)**2)/(2.0D0*VAR2)))
      &/DSQRT((44.0D0*VAR2)/7.0D0)
      PSOMWEG=PWEG1+PWEG2
      POSTWEG1=PWEG1/PSOMWEG
      POSTWEG2=PWEG2/PSOMWEG
      RETURN
      END

      SUBROUTINE BERCRITW(NN,MM,GRMAT,CRIT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NUIT=1,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NNPMM,NNPMM)
      S1=0.0D0
      DO 5 I=1,NN
      DO 4 J=1,NN
      S1=S1+GRMAT(I,J)
4     CONTINUE
5     CONTINUE
      S2=0.0D0
      DO 10 I=NN+1,NNPMM
      DO 9 J=NN+1,NNPMM
      S2=S2+GRMAT(I,J)
```

```
9      CONTINUE
10     CONTINUE
       S3=0.0D0
       DO 15 I=1,NN
       DO 14 J=NN+1,NNPMM
       S3=S3+GRMAT(I,J)
14     CONTINUE
15     CONTINUE
       TELLER=S1/(NN*NN)+S2/(MM*MM)-2.0D0*S3/(NN*MM)
       ANOEMER=1.0D0*NNPMM-S1/NN-S2/MM
       CRIT=TELLER/ANOEMER
       RETURN
       END

       SUBROUTINE BERSOMMAT(AMAT,SOM)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
       PARAMETER (NNPMM=NMTOT-1)
       DIMENSION AMAT(NNPMM,NNPMM)
       S=0.0D0
       DO 45 I=1,NNPMM
       DO 44 J=1,NNPMM
       S=S+DABS(AMAT(I,J))
44     CONTINUE
45     CONTINUE
       SOM=S
       RETURN
       END

       SUBROUTINE BERSOMMAT1(NN,MM,AMAT,SOM1,SOM2,SOM3)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
       PARAMETER (NNPMM=NMTOT-1)
       DIMENSION AMAT(NNPMM,NNPMM)
       S1=0.0D0
       DO 45 I=1,NN
       DO 44 J=1,NN
       S1=S1+AMAT(I,J)
44     CONTINUE
45     CONTINUE
       SOM1=S1

       S2=0.0D0
       DO 47 I=NN+1,NNPMM
       DO 46 J=NN+1,NNPMM
       S2=S2+AMAT(I,J)
46     CONTINUE
47     CONTINUE
       SOM2=S2

       S3=0.0D0
       DO 49 I=1,NN
       DO 48 J=NN+1,NNPMM
       S3=S3+AMAT(I,J)
48     CONTINUE
49     CONTINUE
       SOM3=S3
       RETURN
       END

       SUBROUTINE BERSOMDIAG(AMAT,SOM)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
```

```
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION AMAT(NNPMM,NNPMM)
      S=0.0D0
      DO 45 I=1,NNPMM
      S=S+AMAT(I,I)
45    CONTINUE
      SOM=S
      RETURN
      END

      SUBROUTINE BERSOMDIAG1(NN,MM,AMAT,SOM1,SOM2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION AMAT(NNPMM,NNPMM)
      S1=0.0D0
      DO 45 I=1,NN
      S1=S1+AMAT(I,I)
45    CONTINUE
      SOM1=S1
      S2=0.0D0
      DO 46 I=NN+1,NNPMM
      S2=S2+AMAT(I,I)
46    CONTINUE
      SOM2=S2
      RETURN
      END

      SUBROUTINE BERSOMVEK(VEK,SOM)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION VEK(NNPMM)
      S=0.0D0
      DO 45 I=1,NNPMM
      S=S+VEK(I)
45    CONTINUE
      SOM=S
      RETURN
      END

      SUBROUTINE BERQ(GRMAT,GRWEG,ALPHA,BOPT,QV,SCOREVEK,QWEG,SCOREWEG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NNPMM,NNPMM),ALPHA(NNPMM)
      DIMENSION QV(NNPMM),SCOREVEK(NNPMM),GRWEG(NNPMM)
      DO 62 I=1,NNPMM
      S=0.0D0
      DO 61 J=1,NNPMM
      S=S+ALPHA(J)*GRMAT(I,J)

61    CONTINUE
      SCOREVEK(I)=S+BOPT
      QV(I)=S
62    CONTINUE

      S=0.0D0
      DO 70 J=1,NNPMM
      S=S+ALPHA(J)*GRWEG(J)
70    CONTINUE
      QWEG=S
      SCOREWEG=S+BOPT
      RETURN
      END

      SUBROUTINE BERQTOT(GRMAT,ALPHA,BOPT,QV,SCOREVEK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION GRMAT(NMTOT,NMTOT),ALPHA(NMTOT)
      DIMENSION QV(NMTOT),SCOREVEK(NMTOT)
      DO 62 I=1,NMTOT
      S=0.0D0
      DO 61 J=1,NMTOT
      S=S+ALPHA(J)*GRMAT(I,J)
61    CONTINUE
      SCOREVEK(I)=S+BOPT
      QV(I)=S
62    CONTINUE
      RETURN
      END

      SUBROUTINE BERMARG(NN,MM,VEK1,VEK2,AMARG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION VEK1(NNPMM),VEK2(NNPMM)
      DO 40 I=1,NN
      VEK2(I)=-1.0D0*VEK1(I)
40    CONTINUE
      S=0.0D0
      DO 41 I=NN+1,NNPMM
      VEK2(I)=VEK1(I)
41    CONTINUE
      S=0.0D0

      DO 45 I=1,NNPMM
      S=S+VEK2(I)
45    CONTINUE
      AMARG=S/NNPMM
      RETURN
      END

      SUBROUTINE BERNORM(GRMAT,ALPHA,BOPT,WNORM)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      PARAMETER (LDH=NNPMM)
      DIMENSION GRMAT(NNPMM,NNPMM)
      DIMENSION ALPHA(NNPMM)
      S1=0.0D0
      DO 55 J=1,NNPMM
      DO 54 I=1,NNPMM
      S1=S1+ALPHA(I)*ALPHA(J)*GRMAT(I,J)
54    CONTINUE
55    CONTINUE
      WNORM=DSQRT(S1)
      RETURN
      END

      SUBROUTINE BERMISC(DISTVEK,GEMDIST,TERR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      PARAMETER (NNPMM=NMTOT-1)
      DIMENSION DISTVEK(NNPMM)
      S=0.0D0
      TERR=0.0D0
      DO 55 J=1,NNPMM
      IF (DISTVEK(J).LT.0.0D0) THEN
      TERR=TERR+1.0D0
      S=S+DISTVEK(J)
      ENDIF
55    CONTINUE

      GEMDIST=0.0D0
      IF (TERR.GT.0.0D0) GEMDIST=-1.0D0*S/TERR
      TERR=TERR/NNPMM
      RETURN
      END
```

K. AN EXAMPLE OF THE FORTRAN PROGRAM THAT WAS USED TO PERFORM THE HYPERSPHERE ANALYSIS IN CHAPTER 6

```
C     THIS SUBROUTINE IS USED TO DETERMINE WHICH POINTS LIE ON
C     THE SURFACE OF THE SMALLEST ENCLOSING HYPERSPHERE

C     ALH CONTAINS THE ALPHA COEFFICIENTS ASSOCIATED WITH EACH
C     DATA CASE

C     ALPOS IS AN INDICATOR VECTOR IDENTIFYING THE SUPPORT VECTORS

      SUBROUTINE BERALPOS(XM1,XM2,GAMPAR,ALH,ALPOS)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      DIMENSION XM1(NTOT,IP),XM2(MTOT,IP),GEM(IP),SA(IP)
      DIMENSION GRMAT1(NTOT,NTOT),GRMAT2(MTOT,MTOT)
      DIMENSION ALH1(NTOT),ALH2(MTOT),ALH(NMTOT)
      DIMENSION GEWIG1(NTOT),GEWIG2(MTOT),ALPOS(NMTOT)

      DO 18 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 17 I=1,NTOT
      S1=S1+XM1(I,J)
      S2=S2+XM1(I,J)*XM1(I,J)
17    CONTINUE
      GEM(J)=S1/NTOT
      SA(J)=DSQRT((S2-NTOT*GEM(J)*GEM(J))/(NTOT-1))
18    CONTINUE
      DO 20 J=1,IP
      DO 19 I=1,NTOT
      XM1(I,J)=(XM1(I,J)-GEM(J))/SA(J)
19    CONTINUE
20    CONTINUE

      DO 28 J=1,IP
      S1=0.0D0
      S2=0.0D0
      DO 27 I=1,MTOT
      S1=S1+XM2(I,J)
      S2=S2+XM2(I,J)*XM2(I,J)
27    CONTINUE
      GEM(J)=S1/MTOT
      SA(J)=DSQRT((S2-MTOT*GEM(J)*GEM(J))/(MTOT-1))
28    CONTINUE
      DO 30 J=1,IP
      DO 29 I=1,MTOT
      XM2(I,J)=(XM2(I,J)-GEM(J))/SA(J)
29    CONTINUE
30    CONTINUE

      CALL GRAMMATDEEL(NTOT,GAMPAR,XM1,GRMAT1)
      CALL GRAMMATDEEL(MTOT,GAMPAR,XM2,GRMAT2)

C     THE HYPERSHPERE FOR EACH GROUP IS OBTAINED

      CALL BERALPHAH(NTOT,GRMAT1,ALH1)
      CALL BERALPHAH(MTOT,GRMAT2,ALH2)
      DO 35 I=1,NTOT
      ALH(I)=ALH1(I)
35    CONTINUE
      DO 36 I=1,MTOT
      ALH(NTOT+I)=ALH2(I)
36    CONTINUE
      IALTEL=0
      DO 37 I=1,NMTOT
      ALPOS(I)=0.0D0
37    CONTINUE

      DO 38 I=1,NMTOT
      IF (ALH(I).GT.0.1D-10) THEN
      ALPOS(I)=1.0D0
      IALTEL=IALTEL+1
      ENDIF
38    CONTINUE

      RETURN
      END

C     THIS IS THE SUBROUTINE USED TO CALCULATE THE SMALLEST ENCLOSING
C     HYPERSHERE

      SUBROUTINE BERALPHAH(NPUNTE,GRMAT,ALPHAHYPER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=5,NTOT=100,MTOT=100,NMTOT=NTOT+MTOT)
      DIMENSION GRMAT(NPUNTE,NPUNTE)
      DIMENSION A(NPUNTE+1,NPUNTE),B(NPUNTE+1),G(NPUNTE),H(NPUNTE,NPUNTE)
      DIMENSION ALAM(NPUNTE),SOL(NPUNTE),GEWIG(NPUNTE),ALPHAR(NPUNTE)
      DIMENSION IACT(NPUNTE),IPERM(NPUNTE),IPOS(NPUNTE),ALPHAHYPER(NPUNTE)

      NVAR=NPUNTE
      NCON=NPUNTE+1
      NEQ=1
      LDA=NPUNTE+1
      LDH=NPUNTE
C
C     CONSTRAINTS FOR QUADRATIC OPTIMISATION PROBLEM:
C
C     EQUALITY CONSTRAINT: SUM OF ALPHA*Y'S = 1
C
      DO 5 I=1,NPUNTE
      A(1,I)=1.0D0
5     CONTINUE
C
C     N+M ALPHA >= 0 CONSTRAINTS
C
      DO 30 I=1,NPUNTE
      DO 29 J=1,NPUNTE
      A(I+1,J)=0.0D0
29    CONTINUE
      A(I+1,I)=1.0D0
30    CONTINUE

      B(1)=1.0D0
      DO 35 I=1,NPUNTE
      B(I+1)=0.0D0
35    CONTINUE
C
C     LINEAR PART OF OBJECTIVE FUNCTION:
C
```

```
      DO 36 I=1,NPUNTE
      G(I)=GRMAT(I,I)
36    CONTINUE
C
C     QUADRATIC PART OF OBJECTIVE FUNCTION:
C
      DO 40 I=1,NPUNTE
      DO 39 J=1,NPUNTE
      H(I,J)=2.0D0*GRMAT(I,J)
39    CONTINUE
40    CONTINUE

C     DQPROG IS AN IMSL SUBROUTINE FOR SOLVING QUADRATIC OPTIMISATION
C     PROBLEMS

      CALL DQPROG(NPUNTE,NCON,NEQ,A,LDA,B,G,H,LDH,DIAG,SOL,NACT,
      & IACT,ALAM)

      NPOS=0
      DO 42 I=1,NPUNTE
      ALPHAHYPER(I)=SOL(I)
      IF (SOL(I).GT.0.1D-10) THEN
      NPOS=NPOS+1
      IPOS(NPOS)=I
      ALPHAR(NPOS)=SOL(I)
      ENDIF
      IPERM(I)=I
42    CONTINUE
      CALL DSVRGP(NPOS,ALPHAR,ALPHAR,IPERM)

      ALPHAK=SOL(IPOS(IPERM(NPOS)))
      ALPHA1=SOL(IPOS(IPERM(1)))

45    CONTINUE
      RETURN
      END
```