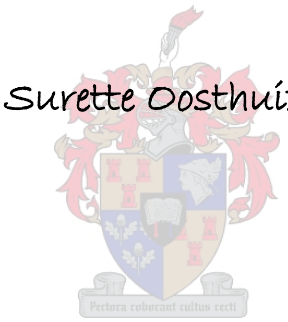
VARIABLE SELECTION FOR KERNEL METHODS WITH APPLICATION TO BINARY CLASSIFICATION

Surette Oosthuizen



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

PROMOTER: Prof. S.J. Steel ***DATE***: March 2008

DECLARATION

I, the undersigned, hereby declare that the work contained in this dissertation is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

***SIGNATURE*: ……**

***DATE*:** 19 November 2007

ABSTRACT

The problem of variable selection in binary kernel classification is addressed in this thesis. Kernel methods are fairly recent additions to the statistical toolbox, having originated approximately two decades ago in machine learning and artificial intelligence. These methods are growing in popularity and are already frequently applied in regression and classification problems.

Variable selection is an important step in many statistical applications. Thereby a better understanding of the problem being investigated is achieved, and subsequent analyses of the data frequently yield more accurate results if irrelevant variables have been eliminated. It is therefore obviously important to investigate aspects of variable selection for kernel methods.

Chapter 2 of the thesis is an introduction to the main part presented in Chapters 3 to 6. In Chapter 2 some general background material on kernel methods is firstly provided, along with an introduction to variable selection. Empirical evidence is presented substantiating the claim that variable selection is a worthwhile enterprise in kernel classification problems. Several aspects which complicate variable selection in kernel methods are discussed.

An important property of kernel methods is that the original data are effectively transformed before a classification algorithm is applied to it. The space in which the original data reside is called input space, while the transformed data occupy part of a feature space. In Chapter 3 we investigate whether variable selection should be performed in input space or rather in feature space. A new approach to selection, so-called feature-to-input space selection, is also proposed. This approach has the attractive property of combining information generated in feature space with easy interpretation in input space.

An empirical study reveals that effective variable selection requires utilisation of at least some information from feature space.

Having confirmed in Chapter 3 that variable selection should preferably be done in feature space, the focus in Chapter 4 is on two classes of selecion criteria operating in feature space: criteria which are independent of the specific kernel classification algorithm and criteria which depend on this algorithm. In this regard we concentrate on two kernel classifiers, *viz*. support vector machines and kernel Fisher discriminant analysis, both of which are described in some detail in Chapter 4. The chapter closes with a simulation study showing that two of the algorithm-independent criteria are very competitive with the more sophisticated algorithm-dependent ones.

In Chapter 5 we incorporate a specific strategy for searching through the space of variable subsets into our investigation. Evidence in the literature strongly suggests that backward elimination is preferable to forward selection in this regard, and we therefore focus on recursive feature elimination. Zero- and first-order forms of the new selection criteria proposed earlier in the thesis are presented for use in recursive feature elimination and their properties are investigated in a numerical study. It is found that some of the simpler zero-order criteria perform better than the more complicated first-order ones.

Up to the end of Chapter 5 it is assumed that the number of variables to select is known. We do away with this restriction in Chapter 6 and propose a simple criterion which uses the data to identify this number when a support vector machine is used. The proposed criterion is investigated in a simulation study and compared to cross-validation, which can also be used for this purpose. We find that the proposed criterion performs well.

The thesis concludes in Chapter 7 with a summary and several discussions for further research.

OPSOMMING

Die veranderlike-seleksie-probleem word in hierdie proefskrif beskou. Kernmetodes is ongeveer twee dekades gelede in die masjienleer- en kunsmatige intelligensie-omgewings voorgestel, en verteenwoordig dus redelike onlangse toevoegings tot statistiese metodes. Hierdie metodes is gaandeweg besig om meer gewild te raak en word dikwels toegepas in regressie- en klassifikasie-probleme.

Veranderlike-seleksie is ’n belangrike stap in baie toepassings in statistiek. Daardeur word ’n beter begrip van die probleem wat ondersoek word, verkry, en wanneer irrelevante veranderlikes elimineer word, lei dit tot akkurater na-seleksie analises van die data. Hieruit spreek die belangrikheid van die ondersoek van die aspekte van veranderlike-seleksie vir kernmetodes vanself.

Hoofstuk 2 van die proefskrif is ’n inleiding tot die hoofgedeelte wat in Hoofstukke 3 tot 6 gegee word. In Hoofstuk 2 word eerstens algemene agtergrond-inligting oor kernmetodes gegee, saam met ’n inleiding tot veranderlike-seleksie. Daarna bied ons empiriese getuienis aan wat bevestig dat veranderlike-seleksie die moeite werd is in klassifikasie-probleme vir kernmetodes. Verskeie aspekte wat veranderlike-seleksie in kernmetodes kompliseer, word ook bespreek.

’n Belangrike eienskap van kernmetodes is dat die oorspronklike data effektiewelik transformeer word voordat ’n klassifikasie-algoritme daarop toegepas word. Die ruimte waarbinne die oorspronklike data lê, word ’n insetruimte genoem, terwyl die getransformeerde data in ’n gedeelte van ’n kernmerkruimte bevat is. In Hoofstuk 3 ondersoek ons of veranderlike-seleksie in die insetruimte gedoen moet word, of eerder in die kernmerkruimte. ’n Nuwe benadering tot seleksie, sogenaamde kernmerk-na-insetruimte seleksie, word ook voorgestel. Die aantreklike eienskap van hierdie benadering is dat inligting wat in die kernmerkruimte genereer word, gekombineer word met maklike

interpretasie in die insetruimte. ’n Empiriese studie toon aan dat effektiewe veranderlike-seleksie die gebruik van minstens sommige van die inligting vanuit die kernmerkruimte vereis.

Nadat in Hoofstuk 3 bevestig is dat veranderlike-seleksie verkieslik in die kernmerkruimte gedoen moet word, lê die klem in Hoofstuk 4 op twee kategorieë vir seleksie-kriteria in laasgenoemde ruimte: kriteria wat onafhanklik is van die spesifieke kern-klassifikasie-algoritme en kriteria wat afhang van hierdie algoritme. In hierdie opsig konsentreer ons op twee klassifikasie-algoritmes, nl. ondersteuningspunt-algoritmes en Fisher se kern-diskriminant-analise. Beide word in redelike besonderhede in Hoofstuk 4 beskryf. Die hoofstuk sluit af met ’n simulasie-studie wat aandui dat twee van die algoritme-onafhanklike kriteria baie mededingend is met die meer gesofistikeerde algoritme-afhanklikes.

In Hoofstuk 5 inkorporeer ons ’n spesifieke strategie om deur die ruimte van veranderlike deelversamelings te soek. Getuienis in die literatuur dui sterk daarop dat terugwaartse seleksie verkies word bo voorwaartse seleksie, daarom is dit ons fokus. Nul- en eerste-orde weergawes van die nuut-voorgestelde seleksie-kriteria vroeër in die tesis word aangebied vir gebruik in terugwaartse seleksie, waarna hulle eienskappe in ’n numeriese studie ondersoek word. Ons bevinding is dat sommige van die eenvoudiger nul-orde kriteria beter vaar as die meer ingewikkelde eerste-orde kriteria.

Tot en met die einde van Hoofstuk 5 word aangeneem dat die aantal veranderlikes om te selekteer bekend is. Ons verwyder hierdie beperking in Hoofstuk 6 en stel ’n eenvoudige data-afhanklike kriterium voor om die aantal veranderlikes te identifiseer binne die konteks van ’n ondersteuningspunt-algoritme. Die voorgestelde kriterium word in ’n simulasie-studie ondersoek en vergelyk met kruis-validasie, wat ook vir hierdie doel aangewend kan word. Ons bevinding is dat die voorgestelde kriterium goed vaar.

Ons sluit die proefskrif in Hoofstuk 7 af, met ’n opsomming en verskeie voorstelle vir verdere navorsing.

ACKNOWLEDGEMENTS

I want to sincerely thank Prof. Sarel Steel for his invaluable guidance and commitment throughout the study, our head of Department, Prof. Tertius de Wet, for his mentorship, and all my colleagues, for their continuous encouragement and support.

A special thank you also to my dad, Klopper Oosthuizen, for many investments in me, and for his love and support, and to my family and friends.

CONTENTS

CHAPTER 1

INTRODUCTION

The field of statistics has often been enriched by contributions from other disciplines. Examples that spring to mind are factor analysis (largely developed in psychology), and kriging (in the earth sciences). Since the early 1990s there have been numerous important developments in *pattern analysis* which are having a significant impact on several areas in statistics. The focus in this thesis is on aspects of one of these developments, *viz*. *kernel methods*.

Pattern analysis may informally be described as a collection of procedures designed to detect patterns in data, which has long been an important objective in various scientific fields. Advances in data collection, data storage and computing capabilities are providing a strong impetus for continuing developments in the pattern analysis field. This has also seen an extension of pattern analysis applications beyond the boundaries of traditional science. Importantly, pattern analysis is not restricted to the detection of relations only in numerical data, and can therefore be utilised to solve a wide variety of problems. Examples are fraud detection and stock market analysis in the financial world, automatic optical character recognition and image and texture analysis in machine learning, gene expression array analysis in genomics, and chromatography diagnosis in chemical engineering.

From a historical perspective, three important stages in the development of pattern analysis may be distinguished. During the 1950s and 1960s, the theory and application of linear methods in pattern analysis were successfully established. A prime example of a procedure developed during this era is the perceptron (Rosenblatt, 1959). During this stage very little progress was however made in terms of developing non-linear methods. The second stage in the development of pattern analysis started with the introduction of neural networks and

decision trees in the early 1980s. These techniques provided non-linear answers to pattern analysis problems, but were largely based on heuristic arguments. Non-linear procedures based on a firm theoretical foundation were still lacking. This gap was filled in the 1990s with the introduction of kernel methods. It is generally acknowledged that the first contribution in this regard was made by Boser *et al*. (1992), with their introduction of the support vector machine (SVM). Extension of the idea in this seminal paper lead to the rapid development of several kernel techniques, for example support vector time series analysis, support vector density estimation and support vector analysis of variance (ANOVA) decomposition; kernel ridge regression and kernel partial least squares in regression; kernel Fisher discriminant analysis (KFDA), kernel logistic regression, import and relevance vector machines in classification; kernel principal component analysis in dimension reduction; and support vector clustering. For an introduction to these techniques the reader is referred to Weston *et al*. (1997), Stitson *et al*. (1997), Mukherjee *et al*. (1997), Schölkopf *et al*. (1999), Mika *et al*. (1999), Herbrich (2002), and Schölkopf and Smola (2002).

In time series analysis, density estimation, ANOVA, regression, classification, dimension reduction and clustering, a number of standard statistical techniques have traditionally been available. Compared to these techniques, kernel procedures frequently offer several advantages. Firstly, in terms of prediction accuracy, kernel methods in many scenarios yield state-of-the-art performance. The results of an empirical study in Louw and Steel (2005) for example show that KFDA (the kernel version of ordinary linear discriminant analysis which will be described in Chapter 5) markedly outperforms ordinary linear discriminant analysis in most situations. A second advantage is that kernel methods can even be applied in cases where the number of variables is much larger than the number of data points. In fact, this property of kernel methods has largely contributed to their extensive application in the analysis of gene profile data, where hundreds of thousands of variables are frequently studied using sample sizes smaller than a hundred (see Guyon and Elisseeff, 2003). Thirdly, as pointed out in the previous paragraph, kernel techniques can handle data of almost any type, ranging from biosequences to image pixels and graph nodes – hence their wide applicability. Finally, Shawe-Taylor and Cristianini (2004) point

out that kernel methods satisfy the requirements of computational efficiency (computation time of the algorithm scales polynomially with the number of data cases), robustness (the algorithm is not particularly sensitive to outliers), and statistical stability (patterns originating from the true source, and not spurious patterns caused by noise, are typically detected).

Since their introduction, kernel methods have become increasingly popular, both as an area of research, and as powerful tools in practice. The current vitality of research in kernel methods is evident from the large number of publications appearing in the literature. Since kernel procedures are typically computer intensive, much of the theory on kernel methods was originally developed by and for researchers and practitioners who are experts in machine learning and computer science (*cf*. for example the special issue on kernel methods in the Journal of Machine Learning Research – Cristianini *et al*., 2001). Gradually other disciplines (including statistics) caught up, leading to for example the formulation of key principles in *statistical learning theory*. Yet at this stage to our view kernel methods still present many opportunities for contributions from the statistics community.

The general focus in this thesis will be on binary classification using kernels. Our specific focus will be on variable selection in kernel classification. Typical examples of cases where variable selection in kernel classification will be beneficial, are gene expression array analysis and fraud detection. In gene analysis, gene expression measurements are used to quantify the abundance of messenger ribonucleic acid in the tissue biopsies of a (usually relatively small) number of persons. In such an analysis, the primary purpose is to identify the genes (variables) which can be used to distinguish between healthy and ill individuals. In fraud detection, a transaction is evaluated in terms of a large number of predetermined attributes. Here the pattern analysis task is typically to obtain the attributes (variables) which distinguish best between honest and fraudulent transactions. In gene expression array analyses the identification of relevant genes facilitates a better understanding of potential susceptibility to certain diseases, possibly leading to early detection and prevention. Additionally, as will be seen in later chapters of this thesis, the

elimination of irrelevant genes can potentially lead to a substantial improvement in the accuracy of the kernel classifier predicting illness. Improved prediction accuracy is also the most important benefit of a reduction in irrelevant attributes in fraud detection. Because of the large number of genes or transaction attributes, these examples of variable selection are however difficult to solve. They are examples of *modern variable selection application domains* where data sets containing hundreds of thousands of variables are common, and even an initial variable filtering process typically only reduces the number of variables to the order of tens of thousands. We would like to emphasise that these examples are representative of a large number of important applications which could potentially benefit from variable selection.

1.1 *NOTATION*

In this section we define notation that will be used throughout the remainder of the thesis. In line with our intended focus, consider the following generic two-group classification problem. We observe a binary response variable $Y \in \{-1, +1\}$, together with classification variables $X_1, X_2, \ldots, X_p$ for $n = n_1 + n_2$ sample cases, where the first n_1 cases belong to group 1 (with $Y = +1$) and the remaining n_2 cases belong to group 2. The sets I_j contain the indices of the cases in group j, $j = 1,2$. The resulting *training data set* is denoted by $\mathcal{T} = \{(\boldsymbol{x}_i, y_i),\ i = 1,2,\ldots,n\}$. Here $\boldsymbol{x}_i$ is a $p-$component column vector containing the values of $X_1, X_2, \ldots, X_p$ for case i in the sample. In the literature the $\boldsymbol{x}_i$ vectors are often referred to as *training patterns*; we will also use the terms *training inputs*, or *sample cases*. Naturally $\mathcal{T}$ can be divided into two training sets corresponding to the two groups, *viz.* $\mathcal{T}_j = \{(\boldsymbol{x}_i, y_i),\ i \in I_j\}$, $j = 1,2$. We write $\boldsymbol{X}$ for the $n \times p$ data matrix with i^{th} row equal to $\boldsymbol{x}_i'$, or equivalently, with j^{th} column containing the n observations on input variable X_j in the training sample.

The objective in classification is to use $\mathcal{T}$ to find a real-valued function $f(\boldsymbol{x})$, called a *discriminant function*, so that $sign\{f(\boldsymbol{x})\}$ can be used to assign a new case with observed values of the classification variables in the vector $\boldsymbol{x}$ to one of the two groups. We will refer to $sign\{f(\boldsymbol{x})\}$ as a *classifier*.

An important quantity when studying a classifier is its *generalisation error*. This is defined by

$$Err(f) = P(Y \neq sign(f(\boldsymbol{X}))), \tag{1.1}$$

where $(\boldsymbol{X}, Y)$ represents a new observation from the same distribution as the one generating the training data. The symbol $err(f)$ will be used to denote the *training error* of $f(\boldsymbol{x})$, *i.e.*

$$err(f) = \tfrac{1}{n}\sum_{i=1}^{n} Ind(y_i \neq f(\boldsymbol{x}_i)), \tag{1.2}$$

where $Ind(A)$ denotes the indicator function of the event A.

In order to address the topic of variable selection in classification, we require additional notation for the classifier obtained using all the available variables, as well as for the post-selection classifier based only on a subset of selected variables. The former will be denoted by $sign\{f(\boldsymbol{x})\}$, and the latter by $sign\{f(\widetilde{\boldsymbol{x}})\}$, where $\widetilde{\boldsymbol{x}}$ represents the measurements on the selected subset of variables. The size of the selected variable subset will be denoted by m.

An important concept in kernel methods is the so-called *feature transformation function*, or feature map, which is usually denoted by Φ. We think of the function Φ as a non-linear transformation of the patterns to a so-called *feature space*. The space to which the patterns originally belong, *i.e.* the *input space*, will be denoted by $\aleph$ (in our applications this will

be $\Re^p$), while the feature space is represented by $\Im$. Conceptually therefore $\Phi(\boldsymbol{x})$ (or simply $\boldsymbol{\varphi}$) is the *feature vector* or *feature pattern* (the data case in $\Im$) corresponding to the input pattern $\boldsymbol{x}$ in $\aleph$. The symbols $\Phi(\boldsymbol{x})$ and $\boldsymbol{\varphi}$ will be used interchangeably. The coordinates of $\boldsymbol{\varphi}$ represent transformations of the original (*input*) *variables* $X_1, X_2, \ldots, X_p$ induced by Φ. They are called *features* and will be denoted by $\{\phi_j, j = 1,2,\ldots,N\}$. Therefore in this thesis Φ is the non-linear transformation function which maps the p-dimensional input space $\aleph$ to a higher dimensional feature space $\Im$, *viz.* $\Phi : \aleph \subseteq \Re^p \rightarrow \Im \subseteq \Re^N$, where the dimension of $\Im$, denoted by N, may be infinite.

In the statistics literature the terms *variable* and *feature* are often used interchangeably. Note that in this thesis there is an important distinction between (*input*) *variable selection* and *feature selection* for kernel methods. In the case of variable selection the objective is to select a subset from the variables $X_1, X_2, \ldots, X_p$, whereas in feature selection interest lies in selection of a subset from the features $\{\phi_j, j = 1,2,\ldots,N\}$. We will use $\Phi(\mathcal{T}) = \{(\boldsymbol{\varphi}_i, y_i), i = 1,2,\ldots,n\}$, $\Phi(\mathcal{T}) \subseteq \Im$, to denote the training data set embedded in feature space and $\Phi(\mathcal{T}_\aleph) = \{\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n\}$ in cases where we wish to omit the sample response values. Here $\mathcal{T}_\aleph = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, $\mathcal{T}_\aleph \subseteq \aleph$, indicates the training set of input patterns.

The concept of an *inner product* plays a central role in kernel methods. We will use the notation $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_i a_i b_i$ to represent the standard inner product between two vectors. The corresponding *norm* will be denoted by $\|\boldsymbol{a}\| = \sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle}$. We will write $\boldsymbol{I}_r$ for the $r \times r$ identity matrix, and $\boldsymbol{E}_{rs}$ for the $r \times s$ matrix with all entries equal to 1. The trace of an $n \times n$ matrix $\boldsymbol{A}$, *i.e.* $\sum_{i=1}^{n} a_{ii}$, will be denoted by $tr(\boldsymbol{A})$.

1.2 *OVERVIEW OF THE THESIS*

In Chapter 2 we firstly provide a fairly extensive introduction to kernel methods. It is shown how the *kernel trick* may be used to implicitly perform calculations in a possibly infinite dimensional feature space, enabling us to fit linear discriminant functions in feature space which correspond to non-linear discriminant functions in input space. We provide examples of different kernel functions and briefly discuss properties of the kernel to be used in the remainder of the thesis. The elegant and useful structure of kernel methods, which also plays an important role in later chapters, is pointed out. We conclude the introductory section on kernel methods by showing how they can be derived using regularisation theory. The remainder of Chapter 2 contains an overview of variable selection, emphasising a binary classification context. We make use of results from a numerical study to confirm the importance of variable selection. We then identify general strategies which may be used for variable selection, and provide an overview of several more recent statistical procedures that inherently facilitate variable selection. Chapter 2 concludes with remarks regarding aspects that complicate variable selection for kernel methods, and possible approaches towards performing kernel variable selection.

Chapters 3 and 4 present part of the contribution of this thesis, *viz*. a new and useful conceptual framework for variable selection in kernel methods. This framework considers firstly the data space in which selection of input variables is performed, and secondly the modular nature of kernel methods. Using this structure, selection criteria may be categorised according to whether they are defined in input or in feature space, and also whether they are specific to a particular kernel algorithm, or not. The framework enables one to identify some of the fairly ad hoc selection criteria proposed in the literature as special cases of a general strategy. The proposed framework therefore unifies a wide array of seemingly unrelated selection criteria in the literature, and also allows a logical development of new ideas and proposals. It provides a structured variable selection methodology for kernel methods, and represents a contribution to the field.

Chapter 3 starts by reporting excerpts from a simulation study which was performed in order to establish whether selection criteria defined in input space can be used for efficient kernel variable selection. We then define and motivate criteria that are defined in feature space, but that ultimately allow the selection of variables to be performed in input space – and indicate the points in favour of and against such an approach. An empirical evaluation of the performance of these so-called feature-to-input space selection criteria is described and reported in the final section of Chapter 3.

Chapters 4 and 5 focus on kernel variable selection carried out entirely in feature space. In Chapter 4 we consider two classes of selection criteria defined in $\mathfrak{I}$, *viz*. algorithm-independent and algorithm-dependent criteria, and empirically evaluate their relative performance. In the case of algorithm-dependent selection criteria we restrict attention to criteria which can be applied in the context of fitting SVMs, or performing KFDA. The results of a simulation study conducted to evaluate the proposed criteria are reported and discussed at the end of Chapter 4. In Chapter 5 we extend the discussion in Chapters 3 and 4 to include a specific selection strategy, *viz*. a backward elimination approach. Instead of using selection criteria to simply rank the input variables and then selecting the m variables with highest ranks, we describe and evaluate recursive feature elimination for kernel methods. This approach entails stepwise elimination of variables deemed most irrelevant, and it has the advantage compared to naïve variable ranking of taking into account inter-dependence amongst the input variables in terms of their effect on the response.

Whereas most of the literature on kernel variable selection assumes the number of input variables to include in the post-selection classifier to be known, we pay specific attention to this aspect in Chapter 6. We propose a criterion which can be used in SVMs for deciding on the number of variables to retain during selection, and evaluate the criterion when it is used in combination with the best performing selection criteria in Chapter 5. We then investigate our criterion when it is used in combination with both a forward and backward selection strategy. Our proposal is shown to perform well in most of the scenarios investigated. With this we provide a comprehensive variable selection procedure that does

not make use of the restrictive assumption of a known value for the number of input variables in the post-selection classifier.

We close the thesis in Chapter 7 with recommendations and a discussion of directions for further research.

CHAPTER 2

VARIABLE SELECTION FOR KERNEL METHODS

2.1 *INTRODUCTION*

Variable selection is frequently the first step in the analysis of a data set. Often the primary objective of a study is to determine which input variables influence the outcome of some response of interest. Even in cases where variable selection is not the goal in itself, suitable reduction of input variables frequently leads to several additional benefits. Especially in small sample cases, a reduction in the number of variables may prevent overfitting the data, and subsequent unstable predictions. Also, a more parsimonious model is simpler to interpret and cheaper than the model containing all variables. The relevance of variable selection in traditional statistical procedures is well established. Standard references are Linhart and Zucchini (1986), Burnham and Anderson (2002), and Miller (2002).

During the past decade it has become clear that variable selection is also important when kernel methods are used, as is substantiated by many contributions in this regard. In the introductory paper to a special issue on kernel variable selection, Guyon and Elisseeff (2003) motivate the importance of variable selection in kernel methods: once again, more parsimonious description of the data yields simpler interpretation, potential cost savings, improved insight regarding the relative importance of the explanatory variables, and frequently also more accurate predictions.

Variable selection is a notoriously difficult problem. It entails a decision on two different levels: the number of variables to include (a value for the model dimension, *i.e.* a value for m), and which subset of m variables this should be. The former is possibly the more difficult decision to make. Two cases may therefore be distinguished. Firstly, it may be

possible to assume that the model dimension is known before the selection process starts. For example, the very large numbers of variables in some application areas makes it essential that an upper bound on the model dimension be specified. In these cases a selection criterion only needs to take into consideration the accuracy of $f(\tilde{\boldsymbol{x}})$, without having to guard against overfitting. Overfitting occurs when noise variables are wrongly selected in an attempt to decrease $err(f)$, usually resulting in an increase in $Err(f)$. The second scenario arises when the value of m is not known. For such situations selection criteria typically provide for a trade-off between accuracy on the training data and complexity, *i.e.* the number of variables included. Establishing the right balance between these conflicting requirements is an important but difficult problem. Hence in the literature, as pointed out by Ishwaran and Rao (2005), determining the model dimension is sometimes not considered part of the classical variable selection problem. Examples of papers dealing with estimation of m include Breiman (1992) and Rao (1999). In this thesis we start by assuming the value of m to be known, and address the problem of estimating the true model dimension in the final chapter.

If the value of m is known, the variable selection problem becomes much simpler. Ideally, one then needs to evaluate all possible models with m input variables, and select the best one. For this purpose an appropriate criterion to evaluate different subsets of variables (a variable selection criterion) is of course required. Specifying variable selection criteria in different application contexts is a difficult problem. We will pay considerable attention to this aspect of kernel variable selection.

The number of models which need to be evaluated during variable selection is often very large. Even if the value of m is fixed, evaluating all $\binom{p}{m}$ possible models of size m may not be feasible. This is even more of a problem in cases where a best model of each possible dimension has to be identified. Such an approach requires evaluation of $\sum_{m=1}^{p}\binom{p}{m}$ models. Variable selection procedures therefore generally require a strategy for reducing all possible combinations of input variables to be evaluated to a more manageable number, and specification of such a strategy is an important aspect of the selection process. Recent

references regarding variable selection strategies in multiple linear regression are Efron *et al*. (2004), Ishwaran (2004), and Shao and Rao (2000). See also Ishwaran and Rao (2005), and references therein.

It is clear from the above that there are basically three important components which need to be specified in a variable selection procedure. One requires:

i. A variable selection criterion in order to rate the various combinations of input variables;
ii. A variable selection strategy which specifies (in combination with the selection criterion) which combinations of input variables should be evaluated;
iii. A method for determining the number of input variables to select.

In this thesis we will discuss these three components in the context of selection for kernel classifiers. The first component receives attention in Chapters 3 and 4, the second component is discussed in Chapter 5, and we propose a new method for selecting the model dimension in kernel methods in Chapter 6.

The aim of the remainder of this chapter is to introduce important concepts in variable selection and in kernel methods. The chapter is structured as follows. In Section 2.2 we provide an overview of kernel methods with special consideration of the properties of kernel functions and the intrinsic modular structure of these techniques. We describe the construction of a kernel classifier and conclude the section with a description of kernel methods from a regularisation perspective. Section 2.3 is devoted to (classical) variable selection in binary classification. We start by providing a historical perspective and introduce separatory and allocatory variable relevance. This is followed by a presentation of empirical evidence supporting the need for variable selection (in classical and kernel application domains), and a discussion of several classical selection strategies and criteria. We conclude the chapter in Section 2.4 with a discussion of the particular aspects which complicate variable selection for kernel methods.

2.2 *AN OVERVIEW OF KERNEL METHODS*

2.2.1 *BASIC CONCEPTS*

In this section we provide an overview of kernel methods for binary classification. Kernel techniques yield classification functions of the form $sign\{f(\boldsymbol{x})\}$, where $f(\boldsymbol{x})$ is non-linear in the training patterns. There are of course a number of advantages if $f(\boldsymbol{x})$ is linear in $\boldsymbol{x}$. A linear decision boundary is simple to interpret and requires only $p+1$ parameters to be estimated from the training data, which reduces the likelihood of overfitting and often yields classifiers with good generalisation properties. Linear methods however do not yield satisfactory generalisation errors in all situations: for example, linear classifiers are inadequate for data that are clearly not linearly separable. A standard approach in such cases is to construct $f(\boldsymbol{x})$ to be linear in a transformed version of $\boldsymbol{x}$. The transformed training patterns reside in a feature space $\Im$, which is usually higher dimensional than the original input space. A linear model is subsequently fitted in $\Im$. This strategy is also motivated by Cover's Theorem (see Haykin, 1999), which essentially states that non-linearly transforming a data set to a higher dimensional space generally renders it more likely to be linearly separable. See also Kroon (2003, *pp.* 122-123).

The following simple example from Schölkopf and Smola (2002) illustrates how data which are not linearly separable may easily be transformed to make it linearly separable.

EXAMPLE 2.1

The one-dimensional data in Figure 2.1 are clearly not linearly separable. However, by transforming the data using $\Phi(x)=\left(x, x^2\right)$, we obtain the two-dimensional data represented in Figure 2.2, where the distinct groups of data points are now linearly separable.

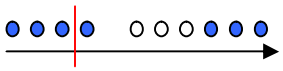


Figure 2.1: *One-dimensional data*

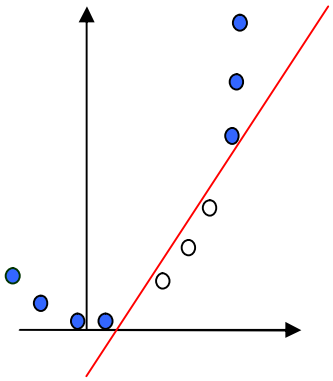


Figure 2.2: *Two-dimensional data after the transformation using* $\Phi(x) = (x, x^2)$

□

There are several issues which need to be addressed in this context. Firstly, it is necessary to specify the transformation Φ. There are many popular choices, one of which is to transform each x to a set of monomials. This leads to polynomial or *spline models*, which are popular in statistics. Of course a decision has to be made regarding the order of the polynomial functions to use. In Example 2.1 we could easily picture the one-dimensional inputs and see that a quadratic transformation would be sufficient. In practice however things are seldom so clearcut. The data are usually multi-dimensional and it quickly becomes impossible to visualise the effect of a non-linear transformation. Even if an appropriate transformation Φ is known, further problems may arise. Calculations in a high-dimensional space may be expensive and in some cases even become intractable. Fitting a model in a high-dimensional space may also entail difficult optimisations, further complicating the problem. Finally there is the danger of overfitting: using too many features in a too high-dimensional space may result in a low error rate on the training data but this may well be accompanied by poor generalisation to new cases.

Conceptually a kernel method also constructs a non-linear discriminant function $f(\boldsymbol{x})$ by fitting a linear function in a transformed higher dimensional feature space. One of the attractive properties of kernel methods is however that the specific non-linear

transformation Φ need not be specified explicitly. In fact, one is not required (and mostly not able) to specify the transformed inputs (or feature vectors), and calculations on feature vectors are performed in an implicit manner by making use of the original input patterns. Thus calculations in feature space are effectively performed in input space, and are therefore typically very fast and efficient – independent of the dimension of the feature space. Although explicit specification of a non-linear transformation Φ is therefore not required when one uses a kernel method, we will see that this requirement is replaced by the need to decide on a *kernel function*. In some respects this is easier than explicitly specifying a non-linear transformation, but inevitably there are subtleties involved which have to be handled carefully.

Regularisation is a standard tool for guarding against overfitting (see for example Hastie *et al.*, 2001, Chapter 5). Since kernel methods can be derived by applying established principles in regularisation theory, this provides inherent protection against potential overfitting. We will however see that successful implementation of regularisation usually requires specification of a regularisation (or smoothing) parameter. If this is done successfully, a kernel method usually delivers state-of-the-art generalisation performance. Details regarding a regularisation perspective on kernel methods will be given in Section 2.4. In the next section we present a more intuitive discussion of the basic ideas in kernel methods.

In summary therefore, a kernel method may be viewed in terms of two steps. The first step is to replace the original input patterns in the input space $\aleph$ with their non-linearly transformed counterparts in a feature space $\Im$. The second step entails construction of a discriminant function which is linear in $\Im$. Requiring linearity of the discriminant function implies that we still benefit from the advantages of linear methods. It must however be kept in mind that the kernel discriminant function thus obtained will typically be highly non-linear in $\aleph$.

2.2.2 *KERNEL FUNCTIONS AND THE KERNEL TRICK*

Transformations used in kernel applications are in most cases much more complicated than the transformation illustrated in Example 2.1. In fact, we will see that frequently the relevant transformation to feature space cannot be specified explicitly, and that it may even lead to an infinite-dimensional feature space. Fortunately, we will still be able to perform the calculations required for constructing a classifier linear in $\Im$. This is accomplished by using an appropriate *kernel function*, which can be defined as follows.

DEFINITION 2.1: ***A KERNEL FUNCTION***

A kernel function $k : \aleph \times \aleph \to \Re$ is a function satisfying $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$ for all $\boldsymbol{x}, \boldsymbol{z} \in \aleph$, where Φ maps $\aleph$ to an inner product feature space $\Im$, *i.e.* $\Phi : \boldsymbol{x} \to \Phi(\boldsymbol{x}) \in \Im$.

The theory underlying kernel methods relies on specific requirements regarding the feature space $\Im$. Specifically, $\Im$ should be a *complete separable inner product space* (*a Hilbert space*). The restriction of $\Im$ to be a Hilbert space is important, and the reader is referred to Section 2.2.4 for more details in this regard.

We see from Definition 2.1 that with a kernel function we associate a transformation function Φ such that $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$ for all $\boldsymbol{x}, \boldsymbol{z} \in \aleph$. A kernel function therefore provides an efficient way of calculating inner products in $\Im$: to find $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$, we merely have to evaluate $k(\boldsymbol{x}, \boldsymbol{z})$. In the literature this shortcut of substituting $k(\boldsymbol{x}, \boldsymbol{z})$ for $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$ is generally known as the *kernel trick*. See also Schölkopf and Smola (2002) and Kroon (2003). Since $k(\boldsymbol{x}, \boldsymbol{z})$ does not require the feature vectors $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{z})$, but only the input vectors $\boldsymbol{x}$ and $\boldsymbol{z}$ in their original form, the kernel trick obviates the need for knowing the transformation function Φ and the explicit construction of feature vectors. A kernel function therefore enables us to work in a higher dimensional feature space without having to define or construct it explicitly, and moreover, via a kernel function we can

calculate inner products efficiently even in cases where Φ yields infinite-dimensional features. In theory we may think of Φ as the transformation function embedding training patterns in $\mathfrak{I}$, but in practice it is sufficient to specify an appropriate kernel function and to work in $\mathfrak{I}$ without further attention to Φ.

To illustrate these points we discuss a simple example taken from Shawe-Taylor and Cristianini (2004, *p*. 34), a standard textbook on kernel methods.

EXAMPLE 2.2

Consider the *quadratic kernel function* $k(\boldsymbol{x},\boldsymbol{z}) = \langle \boldsymbol{x},\boldsymbol{z} \rangle^2$ defined on $\aleph \times \aleph$, $\aleph \subseteq \Re^2$. It is easy to obtain

$$k(\boldsymbol{x},\boldsymbol{z}) = \langle \boldsymbol{x},\boldsymbol{z} \rangle^2 = (x_1 z_1 + x_2 z_2)^2 = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 .$$

Also, let $\Phi(\boldsymbol{x}) = \left[x_1^2, x_2^2, \sqrt{2}x_1 x_2\right]$, mapping the original input space $\aleph$ into a higher dimensional feature space $\mathfrak{I} \subseteq \Re^3$. Since

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = \left\langle \left[x_1^2, x_2^2, \sqrt{2}x_1 x_2\right], \left[z_1^2, z_2^2, \sqrt{2}z_1 z_2\right] \right\rangle = \langle \boldsymbol{x},\boldsymbol{z} \rangle^2 ,$$

we see how the kernel trick facilitates indirect and efficient calculation of the inner product $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$. In this simple special case we are able to identify the features in $\mathfrak{I}$, *viz.* x_1^2, x_2^2 and $\sqrt{2}x_1 x_2$, something which will generally be impossible. Finally in this example, note that there are also other feature maps which correspond to the quadratic kernel, for example $\Phi(\boldsymbol{x}) = \left[x_1^2, x_2^2, x_1 x_2, x_2 x_1\right]$, which maps $\aleph \subseteq \Re^2$ into $\mathfrak{I} \subseteq \Re^4$. Hence we see that it is not necessarily possible to associate a unique feature map with a given kernel function. □

We emphasise that the kernel trick may be used without any restrictions regarding the input domain $\aleph$, other than it being a set (see also Aronszajn, 1950). This is an important reason for the wide applicability of kernel procedures. In order to further appreciate the importance of the kernel trick, one must realise that any algorithm in which the inputs are only involved in terms of inner products in input space, can readily be *kernelised*, *i.e.* formulated in terms of inner products in a feature space. It is in fact this principle which led to the extension of several well known statistical procedures. Examples include kernel logistic regression, kernel principal component analysis, kernel discriminant analysis and kernel regression analysis.

How does one obtain a function satisfying the requirement in Definition 2.1? A basic point in this regard is to ensure that the chosen function allows inner product representations in some higher dimensional feature space. Since the kernel function may be viewed as a generalised inner product, properties of an ordinary inner product in a vector space dictate the kernel to be symmetric, *i.e.* $k(\boldsymbol{x},\boldsymbol{z}) = k(\boldsymbol{z},\boldsymbol{x})$.

The following definitions and results provide properties of kernel functions and indicate ways in which these functions may be constructed.

A key property of kernel functions may be formulated in terms of the so-called *kernel* (or *Gram*) *matrix*.

DEFINITION 2.2: ***THE KERNEL MATRIX***

The kernel matrix for a set of input vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is defined to be the $n \times n$ symmetric matrix $\boldsymbol{K}$ with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j = 1, 2, \ldots, n$.

The following theorem for kernel matrices may be extended to provide a characterisation of kernel functions.

THEOREM 2.1: Kernel matrices are positive semi-definite

PROOF

For any vector $\boldsymbol{v}$ we have $\boldsymbol{v}'\boldsymbol{K}\boldsymbol{v} = \sum_{i,j=1}^{n} v_i v_j k_{ij} = \sum_{i,j=1}^{n} v_i v_j \left\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \right\rangle$

$$= \left\langle \sum_{i=1}^{n} v_i \Phi(\boldsymbol{x}_i), \sum_{j=1}^{n} v_j \Phi(\boldsymbol{x}_j) \right\rangle$$

$$= \left\| \sum_{i=1}^{n} v_i \Phi(\boldsymbol{x}_i) \right\|^2 \geq 0 \text{, as required.}$$

□

Theorem 2.2 summarises the extension of Theorem 2.1 to a result that can be used to characterise kernel functions.

THEOREM 2.2: ***CHARACTERISATION OF KERNELS***

A function $k : \aleph \times \aleph \to \Re$, which is either continuous or has finite domain, can be decomposed into a feature transformation function Φ (into a Hilbert space $\Im$) applied to both its arguments, followed by the evaluation of the inner product in $\Im$, *i.e.* k can be decomposed into

$$k(\boldsymbol{x}, \boldsymbol{z}) = \left\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \right\rangle$$

if and only if it satisfies the finitely positive semi-definite property. □

For a proof of Theorem 2.2, see Shawe-Taylor and Cristianini (2004, *pp*. 61-63). Note that Theorem 2.2 provides a relatively simple method to confirm that a given function $k : \aleph \times \aleph \to \Re$ is a kernel function: for any value of n and any selection of input vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, the resulting kernel matrix must be positive semi-definite.

Construction of new kernels is facilitated by several closure properties possessed by the class of kernel functions. We provide the next theorem as a summary of these properties. The reader is referred to Shawe-Taylor and Cristianini (2004, *p*. 75) for a proof.

THEOREM 2.3: ***CLOSURE PROPERTIES***

Consider kernel functions $k_1, k_2 : \aleph \times \aleph \to \Re$, $\aleph \subseteq \Re^p$, and $k_3 : \Re^N \times \Re^N \to \Re$, where N denotes the dimension of the feature space $\Im$. Furthermore, let f be a real-valued function on $\aleph$, with $a \in \Re^+$, and $\boldsymbol{B}$ a symmetric positive semi-definite $p \times p$ matrix. Then the function k resulting from the following operations will also be a kernel function:

i. $k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) + k_2(\boldsymbol{x}, \boldsymbol{z})$

ii. $k(\boldsymbol{x}, \boldsymbol{z}) = ak_1(\boldsymbol{x}, \boldsymbol{z})$

iii. $k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z})k_2(\boldsymbol{x}, \boldsymbol{z})$

iv. $k(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})f(\boldsymbol{z})$

v. $k(\boldsymbol{x}, \boldsymbol{z}) = k_3(\Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}))$

vi. $k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}'\boldsymbol{B}\boldsymbol{z}$ □

A frequently used example of constructing a new kernel from an existing one is by *normalising* it. Let k be a kernel function with associated feature mapping Φ, then the *normalised* kernel $\hat{k}$ corresponds to the normalised feature mapping $\Phi(\boldsymbol{x}) / \|\Phi(\boldsymbol{x})\|$. It can be shown that the relationship between the given kernel k and its normalised version $\hat{k}$ is given by

$$\hat{k}(\boldsymbol{x}, \boldsymbol{z}) = \left\langle \frac{\Phi(\boldsymbol{x})}{\|\Phi(\boldsymbol{x})\|}, \frac{\Phi(\boldsymbol{z})}{\|\Phi(\boldsymbol{z})\|} \right\rangle = \frac{k(\boldsymbol{x}, \boldsymbol{z})}{\sqrt{k(\boldsymbol{x}, \boldsymbol{x})k(\boldsymbol{z}, \boldsymbol{z})}}. \tag{2.1}$$

Note that the Cauchy-Schwarz inequality implies that $-1 \le \hat{k}(\boldsymbol{x}, \boldsymbol{z}) \le 1$.

Appropriate specification of the kernel function is an important and difficult aspect when a kernel procedure is applied: understanding the effect of alternative kernel functions on the properties of the resulting kernel methods is hardly a trivial matter. In some cases there may be very specific properties of the data which should be incorporated into specification of the kernel function. Certain classes of kernel functions can be used to accommodate such data-specific requirements, and in many practical applications even further fine-tuning of the kernel function may be required.

As an example, consider a binary classification problem. Recall that an inner product between two vectors can be interpreted as a measure of their similarity. Hence $k(\boldsymbol{x},\boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$ reflects the similarity between $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{z})$. Ideally a kernel function should measure similarities between feature patterns in a way that is meaningful in the particular application. In the context of this example we intuitively desire the measured similarity to be relatively large when two patterns belong to the same group, and relatively small otherwise. This suggests that an appropriate kernel function should yield a kernel matrix with the following structure: an upper left $n_1 \times n_1$ sub-matrix with large entries, a similar $n_2 \times n_2$ sub-matrix in the lower right position, and all remaining entries relatively small.

There are many contributions in the literature providing important principles for incorporating such application-specific *a priori* information into kernel functions. Schölkopf *et al.* (1996), for example, consider the construction of kernels which take into account prior knowledge for optical character recognition problems. Other important references concerning appropriate specification of kernel functions include Amari and Wu (1999), Genton (2001), Cristianini *et al.* (2000 and 2002), and Shawe-Taylor and Cristianini (2004). Methods for obtaining tailor-made kernel functions certainly enhance the versatility of kernel methods, but this specialised topic falls outside our scope and will therefore not receive further attention in the thesis.

There are several examples of kernel functions occurring frequently in the literature. These kernels are often applied because of their good performance in many situations. Two examples are the *polynomial* and *sigmoidal* kernels:

- The polynomial kernel $k(\boldsymbol{x},\boldsymbol{z}) = (\langle \boldsymbol{x},\boldsymbol{z} \rangle + c)^d$, where d is a natural number and $c \geq 0$, yields a discriminant function which is a p^{th} order polynomial in the data.

- The sigmoidal kernel $k(\boldsymbol{x},\boldsymbol{z}) = tanh(a\langle \boldsymbol{x},\boldsymbol{z} \rangle - c)$, where a and c are positive real numbers, leads to a classifier with attributes similar to a three-layer neural network.

A kernel function which also performs well in many contexts (see for example Schölkopf *et al*., 1997) and which has become extremely popular with practitioners is the so-called *Gaussian kernel*. It is given by

$$k(\boldsymbol{x},\boldsymbol{z}) = exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|^2\right) \tag{2.2}$$

where γ is a so-called kernel hyperparameter which has to be specified (*cf*. for example Cristianini *et al*., 1998). The Gaussian kernel function belongs to the family of *radial basis function* (*RBF*) kernels, which are kernels that can be written in the form $k(\boldsymbol{x},\boldsymbol{z}) = h(\delta(\boldsymbol{x},\boldsymbol{z}))$, where δ is a metric on $\aleph$ and h is a function defined on positive real numbers. Usually the metric δ is a function of the inner product, *i.e.* $\delta(\boldsymbol{x},\boldsymbol{z}) = \|\boldsymbol{x} - \boldsymbol{z}\| = \sqrt{\langle \boldsymbol{x} - \boldsymbol{z},\ \boldsymbol{x} - \boldsymbol{z} \rangle}$. In this case the resulting RBF kernel function will be unitary invariant. We will restrict attention to the Gaussian kernel in this thesis.

The following theorem confirms that (2.2) does in fact define a valid kernel.

THEOREM 2.4: ***VALID KERNEL FUNCTIONS***

Let $k_1 : \aleph \times \aleph \to \Re$ be an existing kernel. Let also $p(x)$ denote a polynomial with positive coefficients. The following functions are valid kernels:

i. $k(\boldsymbol{x}, \boldsymbol{z}) = p(k_1(\boldsymbol{x}, \boldsymbol{z}))$

ii. $k(\boldsymbol{x}, \boldsymbol{z}) = exp(k_1(\boldsymbol{x}, \boldsymbol{z}))$

iii. $k(\boldsymbol{x}, \boldsymbol{z}) = exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|^2\right)$

PROOF

i. Part *i.* follows directly from operations *i.* to *iv.* in Theorem 2.3, where $f(\boldsymbol{x})$ in *iv.* assumes a constant value to provide for a constant term in the polynomial.

ii. Polynomials with positive coefficients may be used to approximate exponential functions, implying that exponential functions are limits of kernels. Since taking point-wise limits preserves the positive semi-definite property, the second result follows.

iii. Clearly from part *ii*, $k(\boldsymbol{x}, \boldsymbol{z}) = exp\left(\tilde{\gamma} \langle \boldsymbol{x}, \boldsymbol{z} \rangle\right)$ with $\tilde{\gamma} \in \Re^+$ will be a valid kernel. We now normalise k to obtain

$$\begin{aligned} \hat{k}(\boldsymbol{x}, \boldsymbol{z}) &= \frac{exp\left(\tilde{\gamma} \langle \boldsymbol{x}, \boldsymbol{z} \rangle\right)}{\sqrt{exp\left(\tilde{\gamma} \langle \boldsymbol{x}, \boldsymbol{x} \rangle\right) exp\left(\tilde{\gamma} \langle \boldsymbol{z}, \boldsymbol{z} \rangle\right)}} \\ &= exp\left(\tilde{\gamma} \langle \boldsymbol{x}, \boldsymbol{z} \rangle - \tfrac{\tilde{\gamma}}{2} \langle \boldsymbol{x}, \boldsymbol{x} \rangle - \tfrac{\tilde{\gamma}}{2} \langle \boldsymbol{z}, \boldsymbol{z} \rangle\right) \\ &= exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|^2\right) \end{aligned}$$

where $\gamma = \tilde{\gamma}/2$.

□

Note also that the Gaussian kernel is an example of a *translation invariant* kernel: in (2.2) both input vectors may be translated by the same vector without causing any change in the Gaussian kernel function value, *i.e.* $k(\boldsymbol{x},\boldsymbol{z}) = k(\boldsymbol{x}+\boldsymbol{a},\boldsymbol{z}+\boldsymbol{a})$ for all $\boldsymbol{a} \in \aleph$.

Since for the Gaussian kernel, $\|\Phi(\boldsymbol{x})\|^2 = k(\boldsymbol{x},\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in \aleph$, all feature vectors are at a fixed distance from the origin, *i.e.* they lie on the surface of a hypersphere with radius 1. Furthermore, consider two feature vectors, $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{z})$, and denote the angle between these vectors by θ. Then it follows that $0 \le k(\boldsymbol{x},\boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = \cos\theta \le 1$, with the implication that the enclosed angle between any two features will be at most $\pi/2$. We can therefore choose the feature space so that the mapped values all lie in a single orthant. This is illustrated informally in Figure 2.3 below.

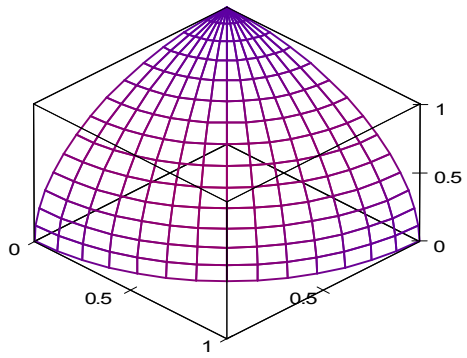


Figure 2.3: *Using the Gaussian kernel function, all mapped values lie on the surface of a single orthant of a hypersphere with radius* 1

From the above figure it may seem as if the Gaussian kernel maps data into a rather restricted area of the feature space. The following proposition however shows that features induced by the Gaussian kernel function actually occupy a space which is in some sense as large as possible.

THEOREM 2.5: Gaussian kernel matrices have full rank

Given distinct input patterns $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ and a hyperparameter value $\gamma > 0$, the kernel matrix with entries $k_{ij} = exp\left(- \gamma \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2 \right)$ will have full rank. □

Provided that no two input patterns are the same, note that the features $\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2), \ldots, \Phi(\boldsymbol{x}_n)$ will be linearly independent, and hence they span an *n*-dimensional subspace of the induced Hilbert space. In this sense the Gaussian kernel yields features which inhabit a part of the feature space which is as large as possible.

As seen in Equation 2.2, application of the Gaussian kernel requires specification of a kernel hyperparameter value γ. Similarly, *c* and *d* in the polynomial kernel, and *a* and *c* in the sigmoidal kernel are hyperparameter values that need to be specified. This is an important step in the application of kernel techniques, since inappropriate specification of kernel hyperparameter values typically has a detrimental effect on their generalisation performance. Possible approaches to specification of kernel hyperparameter values are given in Müller *et al*. (2001). Contributions regarding the specification of γ in the Gaussian kernel function include Cristianini *et al*. (1998), Chapelle *et al*. (2004), Keerthi (2002) and Wang *et al*. (2004). Although requiring expensive computations, cross-validation (or hold-out testing) seems thus far to be the most popular approach for determining kernel hyperparameter values. In a cross-validation approach, an upper bound on the generalisation error is calculated on hold-out sections of the data for several hyperparameter values. The hyperparameter value yielding the best performance across hold-out data sets is then selected. In kernel functions with large numbers of hyperparameter values that need to be specified, cross-validation procedures soon become computationally intractable. Since in the context of SVMs a number of upper bounds on the 1-fold cross-validation (or leave-one-out) error can be derived (*cf*. Vapnik and Chapelle, 2000, and Duan *et al*., 2001), many of the proposals regarding kernel hyperparameter specification in SVMs involve the calculation of (fast) approximations to

the SVM leave-one-out error. These procedures are however often based on solutions to non-convex optimisation problems and are therefore prone to yield only locally optimum kernel hyperparameter values.

2.2.3 *CONSTRUCTING A KERNEL CLASSIFIER*

Having provided some detail regarding kernel functions, we proceed with a description of the actual construction of a kernel discriminant function in the context of binary classification. The requirement of a linear classifier in $\Im$ implies that the classifier will be of the form $sign\{\langle \Phi(\boldsymbol{x}), \boldsymbol{w} \rangle + b\}$ where $\boldsymbol{w} \in \Im$ and $b \in \Re$ have to be determined from the training data. The feature vector $\boldsymbol{w}$ is usually obtained by solving a constrained optimisation problem. It is a well known result in the theory of constrained optimisation that the dual formulation of such an optimisation problem is easier to deal with than the primal formulation (see for example Shawe-Taylor and Cristianini, 2004). The optimal $\boldsymbol{w}$ typically turns out to be a linear combination of the transformed input vectors, *viz.* $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$, where the vector of Lagrange multipliers, *viz.* $\boldsymbol{\alpha}' = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ (required to deal with the constraints), are obtained from the data. Hence the kernel classifier becomes

$$sign\left\{\left\langle \Phi(\boldsymbol{x}), \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i) \right\rangle + b\right\} = sign\left\{\sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b\right\}, \tag{2.3}$$

with $\boldsymbol{x}$ the (new) input pattern to be classified.

The specific form of the objective function which has to be optimised and the constraints required in determining the $\boldsymbol{\alpha}$-vector depend on the type of kernel classifier implemented. For example, the optimisation problem specifications for SVMs differ from those required for constructing kernel Fisher discriminant functions or kernel logistic regression functions.

Further details in this regard are given in Chapter 4, in particular for SVMs and kernel Fisher discriminant functions.

Returning to Equation (2.3), we see that different kernel classifiers display two common characteristics: at a first level the discriminant function depends on the training data only via a kernel function, and at a second level the discriminant function depends on the training data via weights $\alpha_1, \alpha_2, \ldots, \alpha_n$ which are typically obtained through solving an optimisation problem.

Shawe-Taylor and Cristianini (2004) make use of shared and differentiating aspects in kernel methods to describe an effective framework for conceptualising these techniques. This framework has a modular structure, emphasising that the aspects in kernel methods (for example, specifying the kernel function and its hyperparameter values, and specifying and finding a solution to the optimisation problem) can be treated independently. This independence between kernel components induces flexibility in constructing a kernel technique. Any (sensible) combination of the many possibilities regarding each of the kernel aspects may be used, and will yield a different kernel procedure.

2.2.4 *A REGULARISATION PERSPECTIVE*

In the previous section we saw that (binary) classification may be viewed as a function estimation problem: the finite set of training data, $\mathcal{T} = \{(\boldsymbol{x}_i, y_i), i = 1, 2, \ldots, n\}$, where $y_i \in \{-1, +1\}$, is used to estimate a function f such that $y = sign\{f(\boldsymbol{x})\}$ can be used to predict class membership of newly observed cases. In regression contexts, Y is a continuous response variable, and the objective is to use $\mathcal{T}$ to determine a regression function $f(\boldsymbol{x}) = E(Y \mid \boldsymbol{x})$ estimating (or predicting) the response of a newly observed input pattern as accurately as possible. Clearly therefore also regression may be viewed as a function estimation problem.

In any given function estimation problem, let F denote the class of candidate functions from which the function f will be obtained, *i.e.* F is the class of candidate functions $f : \aleph \to \Re$. For example, in binary classification contexts, F represents the class of all possible binary discriminant functions. In function estimation problems, $f \in F$ is often obtained by minimising a measure of error on the training data, *e.g.* $\frac{1}{n}\sum_{i=1}^{n} L\left(f(\boldsymbol{x}_i), y_i\right)$. Here $L\left(f(\boldsymbol{x}_i), y_i\right)$ denotes a loss function used to measure in some appropriate way the closeness of the estimated or predicted response $f\left(\boldsymbol{x}\right)$ to the true response y. Popular loss functions in classification and regression will be given during the course of our discussion below. Note that $\frac{1}{n}\sum_{i=1}^{n} L\left(f(\boldsymbol{x}_i), y_i\right)$ is the average or mean loss in the *training data* when the estimate $f\left(\boldsymbol{x}_i\right)$ is used instead of the true response y_i. Hence this term reflects the performance of a candidate function only on the training patterns, and is called the *empirical risk*. It is important to note that using the function $f \in F$ which minimises the empirical risk would in most cases lead to overfitting. The result would be an f which precisely follows the training data, *i.e.* $f\left(\boldsymbol{x}_i\right) = y_i, \quad i = 1, 2, \ldots, n$. Such a function would almost certainly generalise poorly to classification of unseen cases.

Another complicating factor is that F is often a class of functions defined on a high-dimensional space. For example, in the variable selection application domains discussed in Chapter 1 the data sets often consist of relatively few observations on a large number of variables. In such cases the data soon become very sparse, rendering function estimation an *ill-posed* problem. This means that small perturbations of the data can induce large changes in the estimated functions. Stated differently, the data provide too little information to accurately estimate f. Irrespective of the dimension of the problem in input space, in kernel methods the transformed feature space $\Im$ typically has a very high dimension. Therefore ill-posed function estimation is a particularly relevant problem in this context.

A widely used approach for solving ill-posed function estimation problems is known as *regularisation* (see Tikhonov and Arsenin, 1977; Bertero, 1986, and Bertero *et al.*, 1988, for early contributions in this regard). Regularisation theory formalises classification and regression problems as so-called variational problems, *viz.* where one aims to find the function $f \in F$ which minimises a *regularised* risk functional. This implies that f is obtained by solving the following optimisation problem:

$$\min_{f \in \mathcal{H}} \left\{ R(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \Omega(f) \right\}. \tag{2.4}$$

In (2.4), $\mathcal{H}$ is typically a Hilbert space of candidate functions. Therefore minimisation of the regularised risk is carried out over all possible functions contained in an appropriately defined Hilbert space.

Effectively, regularisation restricts the complexity of the function class F. This restriction is implemented via the term $\Omega(f)$ in (2.4), which penalises functions that are complex or non-smooth. Hence regularisation purposely induces bias in estimating f: when comparing functions with similar empirical risk values, the simpler function will be favoured. Obviously the desire for a simple function must not be taken to extremes: there are data sets requiring fairly complex discriminant functions. The trade-off between minimisation of the empirical risk, and complexity or non-smoothness of the candidate functions is controlled by the *regularisation* or *smoothing parameter* $\lambda \geq 0$. In (2.4), appropriate specification of a value for λ is crucial. With $\lambda = 0$, we find ourselves back in the empirical risk minimisation setup. Larger values of λ implies stricter penalisation of complex functions, generally yielding simpler solutions. For more details regarding specification of λ, the reader is referred to Wahba (1990).

The idea behind regularisation is well illustrated in the next example which presents some details on ridge regression.

EXAMPLE 2.3

Let F represent the class of functions of the form $f(\boldsymbol{x}) = E(Y \mid \boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$, where $\boldsymbol{\beta}$ denotes a p-dimensional vector of regression coefficients. In addition, assume that $\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim (\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$, for a sample of n cases. The ordinary least squares (OLS) regression function is obtained by assuming a *squared error loss function*, *i.e.* $L(f(\boldsymbol{x}_i), y_i) = (y_i - f(\boldsymbol{x}_i))^2$, and finding $f \in F$ which minimises the empirical risk. Hence, in order to perform OLS we simply need to solve the following optimisation problem:

$$\min_{\boldsymbol{\beta}} \left\{ R(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) = \frac{1}{n} \| \boldsymbol{y} - \boldsymbol{X\beta} \|^2 \right\}. \tag{2.5}$$

This yields the OLS regression estimate $\hat{\boldsymbol{\beta}} = (\boldsymbol{X'X})^{-1} \boldsymbol{X'y}$. It is clear that if two or more of the input variables are highly correlated, $\boldsymbol{X'X}$ will become ill-conditioned, and in extreme cases the inverse of $\boldsymbol{X'X}$ may not exist. The result would be a poorly determined OLS estimate which exhibits high variance (see also Hastie *et al*., 2001, *pp*. 59-60). This limitation of OLS estimates formed the initial rationale for introducing ridge regression into statistics (Hoerl and Kennard, 1970): to ensure non-singularity of $\boldsymbol{X'X}$, a positive constant λ is added to the diagonal of $\boldsymbol{X'X}$. Called the *pseudo* (or *generalised*) *inverse*, $(\boldsymbol{X'X} + \lambda \boldsymbol{I}_p)^{-1}$ is then substituted for $(\boldsymbol{X'X})^{-1}$, yielding the *ridge regression least squares estimate* $\tilde{\boldsymbol{\beta}} = (\boldsymbol{X'X} + \lambda \boldsymbol{I}_p)^{-1} \boldsymbol{X'y}$.

In applications with many correlated input variables, ridge regression generally performs better than OLS regression functions. This result is not immediately evident from the formulation above, but becomes clear when one realises that substitution of the generalised inverse for $(\boldsymbol{X'X})^{-1}$ in $\hat{\boldsymbol{\beta}}$ yields a ridge regression least squares estimate $\tilde{\boldsymbol{\beta}}$ which can actually be shown to solve the regularised (or penalised least squares) optimisation problem

$$\min_{\boldsymbol{\beta}} \left\{ R(\boldsymbol{\beta}) = \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \langle \boldsymbol{\beta}, \boldsymbol{\beta} \rangle \right\} \tag{2.6}$$

Note that the regularisation term $\lambda\langle \boldsymbol{\beta}, \boldsymbol{\beta} \rangle$ in (2.6) restricts the size of the estimated regression coefficients, thereby guarding against potential high variance.

□

How does one go about solving the regularised optimisation problem in (2.4)? We have seen that minimisation of the regularised risk should be performed over all functions in a Hilbert space $\mathcal{H}$. Thus (2.4) is an infinite dimensional problem, and solving it is generally not a trivial matter. A surprisingly elegant solution can however be found for an important subclass of the problem in (2.4). This subclass of problems is generated by *positive semi-definite* kernel functions, and their solutions yield a wide spectrum of statistical techniques, including for example the entire family of smoothing splines (*cf*. Wahba, 1990). Moreover, we will soon see that they also provide an alternative view on the way in which kernel methods can be defined.

One might ask why positive semi-definite kernel functions enable us to solve the optimisation problem in (2.4). Firstly, it can be shown that for a positive semi-definite kernel function we can define a Hilbert space $\mathcal{H}$ such that it possesses the so-called reproducing property, *i.e*. such that $\mathcal{H}$ will be a *reproducing kernel Hilbert space* (*RKHS*). Thus in kernel-induced regularisation problems the candidate functions reside in an RKHS, and it is the properties of an RKHS which yield a relatively simple way for obtaining a solution to the optimisation problem. For more details on RKHSs, see for example Hastie *et al.* (2001, Chapter 5). Secondly, if we consider (2.4) in an RKHS, the celebrated *Representer Theorem* of Kimeldorf and Wahba (1971) (presented in Theorem 2.6 to follow) implies that the optimisation problem becomes finite-dimensional.

In the previous paragraph we indicated that with every positive semi-definite kernel function an RKHS can be associated. In the next definition and the brief section to follow,

we start with a Hilbert space $\mathcal{H}$ and provide the conditions required for $\mathcal{H}$ to be an RKHS. The conclusion in the end is that with every RKHS we can associate a unique positive semi-definite kernel function.

DEFINITION 2.3: ***A REPRODUCING KERNEL HILBERT SPACE***

Let $\aleph$ be a non-empty set and $\mathcal{H}$ a Hilbert space of functions which map $\aleph$ into $\Re$. Then $\mathcal{H}$ is a reproducing kernel Hilbert space endowed with a dot product $\langle .,. \rangle_{\mathcal{H}}$ and a norm $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$ if there exists a function $k : \aleph \times \aleph \to \Re$ with the following properties:

i. The reproducing property, *viz*.

$$f(\boldsymbol{x}) = \langle f, k(\boldsymbol{x},.) \rangle_{\mathcal{H}} \quad \forall f \in \mathcal{H} \tag{2.7}$$

and specifically, $\langle k(\boldsymbol{x},.), k(\boldsymbol{z},.) \rangle_{\mathcal{H}} = k(\boldsymbol{x}, \boldsymbol{z})$.

ii. The property that k spans $\mathcal{H}$.

It can easily be shown that if the kernel function in Definition 2.3 exists, it will always be uniquely defined (*cf*. Schölkopf and Smola, 2002). Note therefore that an RKHS implied by a kernel function k will henceforth be denoted by $\mathcal{H}_k$. To verify that the kernel function corresponding to an RKHS is a positive semi-definite function, consider

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i,j=1}^{n} \alpha_i \alpha_j \left\langle k(\boldsymbol{x}_i, .), k(\boldsymbol{x}_j, .) \right\rangle_{\mathcal{H}_k}$$

$$= \left\langle \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, .), \sum_{j=1}^{n} \alpha_j k(\boldsymbol{x}_j, .) \right\rangle_{\mathcal{H}_k} = \left\| \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, .) \right\|_{\mathcal{H}_k}^2 \geq 0 . \qquad (2.8)$$

For a clearer understanding of an RKHS, we present the following example (see Hastie *et al.*, 2001, *p.* 144) which illustrates the explicit construction of an RKHS from a given positive semi-definite kernel. The properties of an RKHS which are important to us in this section will also be given along the way.

EXAMPLE 2.4

Let $\gamma_1, \gamma_2, \ldots$ and $\phi_1, \phi_2, \ldots$ respectively denote sequences of positive numbers and independent functions, with $\sum_{i=1}^{\infty} \gamma_i^2 < \infty$, and consider the symmetric function $k(\boldsymbol{x}, \boldsymbol{z})$ defined by $k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{\infty} \gamma_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{z})$. Now consider the Hilbert space $\mathcal{H}$ consisting of functions of the form $f(\boldsymbol{x}) = \sum_{i=1}^{\infty} c_i \phi_i(\boldsymbol{x})$, where the coefficients $c_1, c_2, \ldots$ are real numbers. Also define the scalar product in $\mathcal{H}$ to be $\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \frac{c_i d_i}{\gamma_i}$, where $f(\boldsymbol{x}) = \sum_{i=1}^{\infty} c_i \phi_i(\boldsymbol{x})$ and $g(\boldsymbol{x}) = \sum_{i=1}^{\infty} d_i \phi_i(\boldsymbol{x})$, $d_1, d_2, \ldots \in \Re$. It follows that the squared norm in $\mathcal{H}_k$ is

$$\|f\|_{\mathcal{H}_k}^2 = \langle f, f \rangle_{\mathcal{H}_k} = \sum_{i=1}^{\infty} \frac{c_i}{\gamma_i} \qquad (2.9)$$

and we assume this to be finite.

Now since

$$\left\langle f, k(\boldsymbol{x},.)\right\rangle_{\mathcal{H}_k} = \sum_{i=1}^{\infty} \frac{c_i \gamma_i \phi_i(\boldsymbol{x})}{\gamma_i} = \sum_{i=1}^{\infty} c_i \phi_i(\boldsymbol{x}) = f(\boldsymbol{x}), \tag{2.10}$$

k clearly satisfies Equation (2.7). Consequently, as seen in (2.8), k is a positive semi-definite kernel function. Therefore the Hilbert space $\mathcal{H}_k$ is an RKHS, with k the corresponding positive semi-definite reproducing kernel.

□

Summarising: for any given RKHS we can identify a unique positive semi-definite kernel function, and vice versa. In practical applications we start with a positive semi-definite kernel function, and perform function estimation in the associated RKHS. However, it will become clear that we need not explicitly construct the associated RKHS since we can solve the optimisation problem given in (2.4) simply in terms of the kernel function. We now return to this aspect.

Performing function estimation in an RKHS involves the use of a functional measuring the complexity of candidate functions. We do this in terms of $\Omega(f) = \|f\|^2_{\mathcal{H}_k}$. Substituting (2.9) into (2.4) we obtain

$$\min_{f \in \mathcal{H}_k} \left\{ R(f) = \tfrac{1}{n} \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_k} \right\} \tag{2.11}$$

$$= \min_{\{c_j\}_{j=1}^{\infty}} \left\{ \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, \sum_{j=1}^{\infty} c_j \phi_j(\boldsymbol{x}_i) \right) + \lambda \sum_{j=1}^{\infty} c_j^2 / \gamma_j \right\}. \tag{2.12}$$

Even after this simplification, (2.12) remains an infinite dimensional problem: there are an infinite number of coefficients $c_1, c_2, \ldots$ which have to be determined. Fortunately the Representer Theorem shows that a finite dimensional solution to (2.12) does exist. We provide the version given in Schölkopf and Smola (2002).

THEOREM 2.6: ***THE REPRESENTER THEOREM***

Let $\aleph$ be a set, and as before, use $L\left(y_i, f(\boldsymbol{x}_i)\right)$ to denote an arbitrary loss function. Then each minimiser $f \in \mathcal{H}_k$ of the regularised risk $R(f) = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f(\boldsymbol{x}_i)\right) + \lambda \|f\|_{\mathcal{H}_k}^2$ admits a representation of the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) \tag{2.13}$$

□

Using (2.13) we obtain

$$\begin{aligned} \|f\|_{\mathcal{H}_k}^2 &= \langle f, f \rangle_{\mathcal{H}_k} = \sum_{i,j=1}^{n} \alpha_i \alpha_j \left\langle k(\boldsymbol{x}_i, \boldsymbol{x}), k(\boldsymbol{x}_j, \boldsymbol{x}) \right\rangle_{\mathcal{H}k} \\ &= \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\alpha}' \boldsymbol{K} \boldsymbol{\alpha}. \end{aligned} \tag{2.14}$$

Hence the Representer Theorem guarantees an explicit form for the solution to (2.12). Moreover, this representation indicates that instead of having to find an infinite number of coefficients as in (2.12), we actually only have to find n coefficients, *viz.* $\alpha_1, \alpha_2, \ldots, \alpha_n$. This phenomenon whereby the infinite-dimensional optimisation problem in (2.11) or (2.12) reduces to a finite-dimensional optimisation problem, is referred to in the literature as the *kernel property*. The functions $h_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$, $i = 1, 2, \ldots, n$, are also known as the *representers of evaluation* at $\boldsymbol{x}_i$ in $\mathcal{H}_k$.

Finally therefore we see that the kernel-induced regularised optimisation problem in (2.12) may also be written in matrix form as

$$\min_{\boldsymbol{\alpha}} \left\{ R(\boldsymbol{\alpha}) = L(\boldsymbol{y}, \boldsymbol{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}' \boldsymbol{K} \boldsymbol{\alpha} \right\}. \tag{2.15}$$

Note that if $L(\boldsymbol{y}, \boldsymbol{K\alpha})$ is a convex function of $\alpha_1, \alpha_2, \ldots, \alpha_n$, (2.15) will be a convex optimisation problem which does not suffer from local minima, and can thus be solved easily.

Through different specifications of the convex loss function and the regularisation term, different regularised statistical procedures can be obtained, including several of the modern methods which originated in machine learning contexts. In this section we keep the regularisation term fixed, *i.e.* we specify the regulariser to be $\boldsymbol{\alpha' K\alpha}$, and proceed by providing a few examples of the various statistical techniques obtained by different specifications of the loss function.

We start with a brief description of loss functions that are frequently applied in classification setups. Perhaps the most basic loss function used in classification is the 0-1 loss, or the *misclassification loss function*, where a loss of 1 is incurred for each misclassification. Since the signed function value $y\, f(\boldsymbol{x})$ will be positive if f correctly classifies the input pattern $\boldsymbol{x}$, and negative otherwise, the misclassification loss function can also be given in terms of $y\, f(\boldsymbol{x})$. That is, we may write $L(y_i, f(\boldsymbol{x}_i)) = L(y_i f(\boldsymbol{x}_i)) = \theta(-y_i f(\boldsymbol{x}_i))$, where θ is the Heaviside function, *i.e.*

$$\theta(x) = \begin{cases} 0, & \text{if } x < 0 \\ \frac{1}{2}, & \text{if } x = 0 \\ 1, & \text{if } x > 0 \end{cases}.$$

The misclassification loss function is shown in Figure 2.4.

Loss functions used in classification contexts are often specified in terms of the quantity $y\, f(\boldsymbol{x})$, which is frequently referred to as a classification *margin*. We have seen that the sign of the margin indicates whether an input pattern is misclassified or not, but the same information could be derived from the quantity $y\, sign\{f(\boldsymbol{x})\}$, which attains a value of 1 if f correctly classifies the input pattern $\boldsymbol{x}$, and -1 otherwise.

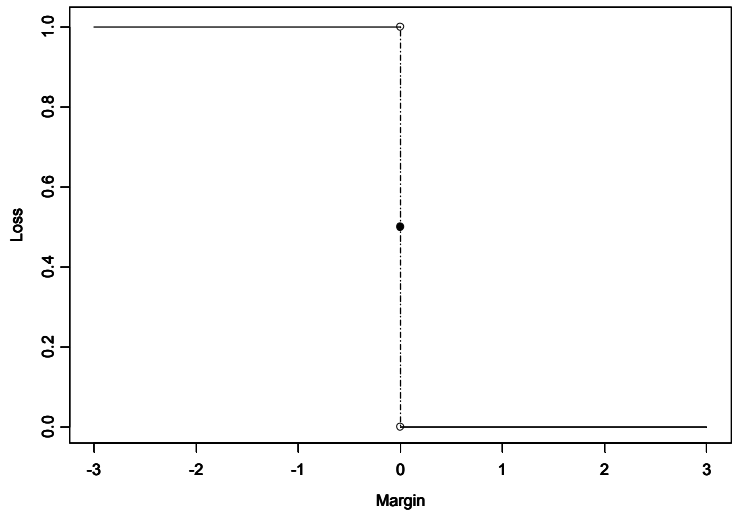


Figure 2.4: *The misclassification loss function*

What do we gain by using the real-valued function $f(\boldsymbol{x})$ in the margin instead of $sign\{f(\boldsymbol{x})\}$ in $y\, sign\{f(\boldsymbol{x})\}$? In order to see why the margin in particular is an important quantity in classification, note first of all that the value of $f(\boldsymbol{x})$ can be regarded as a measure of the confidence with which f estimates (or predicts) the binary response: a large positive value for $f(\boldsymbol{x})$ indicates f predicting with virtual certainty that $\boldsymbol{x}$ belongs to group 1, whereas a large negative value indicates a strong belief that $\boldsymbol{x}$ belongs to group 2. Conversely, small values of $|f(\boldsymbol{x})|$ indicate uncertain classification of $\boldsymbol{x}$. Now consider the margin and note that the relative sizes of positive values of $y\, f(\boldsymbol{x})$ provide an indication of how well f predicts the group membership of $\boldsymbol{x}$. A large positive margin indicates that a prediction is far from incorrect, and is thus preferred to a small positive value indicating a prediction which is closer to being a misclassification. Thus in a sense a large positive margin indicates a smaller margin for error. Similarly, the relative sizes of negative margin values can be used as a measure of how badly f predicts the group membership of $\boldsymbol{x}$ in cases where this is done incorrectly. A small negative margin signifies that the input pattern $\boldsymbol{x}$ is nearly classified correctly, and this is therefore preferred to a large negative margin. In summary, large values for $y\, f(\boldsymbol{x})$ are preferred.

A second loss function which can be used in classification setups is the so-called *hard margin loss function*, *viz.* $L\left(y_i f(\boldsymbol{x}_i)\right) = \theta\left(1 - y_i f(\boldsymbol{x}_i)\right)$. This loss function is similar to misclassification loss: it assigns either a zero penalty, or a penalty of 1, but it is stricter in the sense that it distinguishes between correct classifications with a margin larger than 1, correct classifications with a margin smaller than 1, and misclassified cases. The latter two categories both incur a loss of 1. Therefore the misclassification and hard margin loss functions operate differently when $y f(\boldsymbol{x}) \in [0,1]$. The reason for the threshold value of 1 for the margin will become clear in Chapter 4. Figure 2.5 below depicts a hard margin loss function.

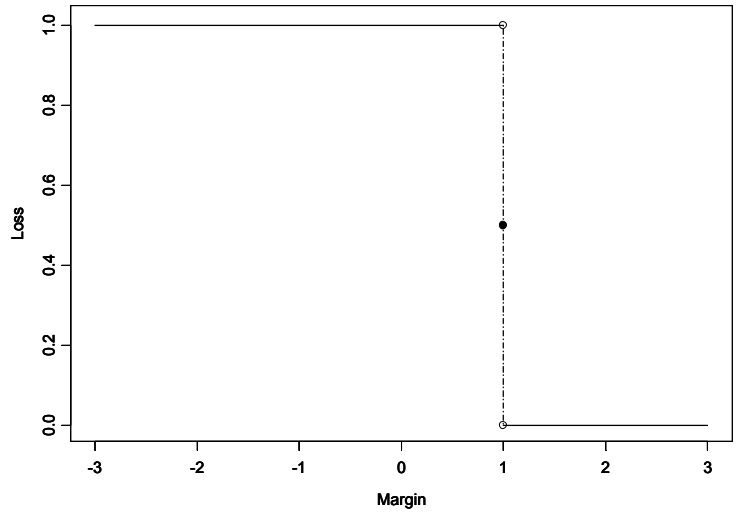


Figure 2.5: *The hard margin loss function*

A slight modification to the hard margin loss function yields the *soft margin loss function* (also called the *hinge loss function*). The soft margin loss is defined as $L\left(y_i f(\boldsymbol{x}_i)\right) = \left|1 - y_i f(\boldsymbol{x}_i)\right|_+$, where $\left|a\right|_+ = a$ if a is positive, and zero otherwise. Figure 2.6 depicts the soft margin loss function.

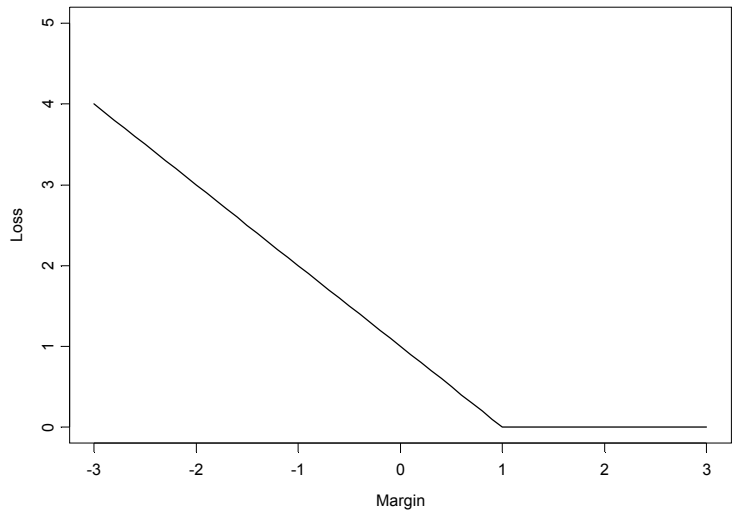


Figure 2.6: *The soft margin loss function*

As is the case with hard margin loss, the soft margin loss function also penalises correct classifications that are made with a margin less than 1. The soft margin loss is however more forgiving when $y\,f(\boldsymbol{x}) \in [0,1]$. On the other hand, for $y\,f(\boldsymbol{x}) < 0$, the soft margin loss function assigns a larger penalty than the hard margin loss: as $y\,f(\boldsymbol{x})$ decreases, the penalty increases linearly.

Two related loss functions in classification are the *exponential* and the *deviance* (or *logistic*) loss functions, given by $L\left(y_i f(\boldsymbol{x}_i)\right) = exp\left(-y_i f(\boldsymbol{x}_i)\right)$ and $L\left(y_i, f(\boldsymbol{x}_i)\right) = log\left(1 + exp\left(-y_i f(\boldsymbol{x}_i)\right)\right)$ respectively. Graphical representations of these loss functions are given in Figure 2.7.

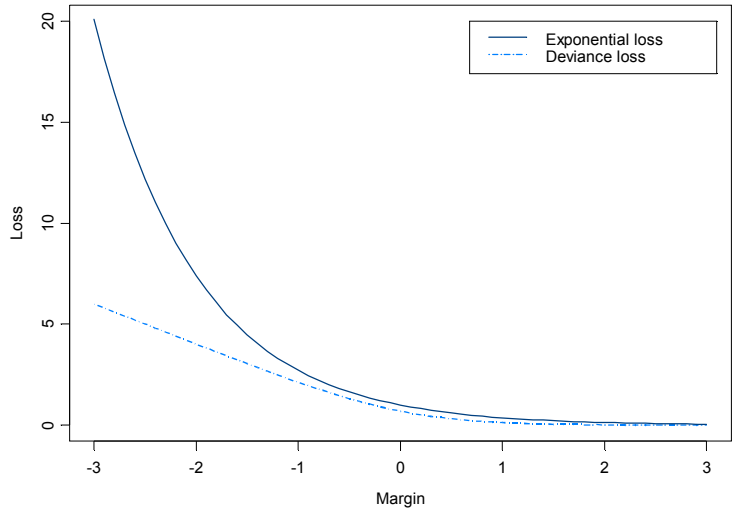


Figure 2.7: *The exponential and deviance loss functions*

From Figure 2.7 we see that both the exponential and the deviance loss functions are continuous approximations to the soft margin loss: as $y\,f(\boldsymbol{x})$ decreases, the penalties assigned by both of these loss functions increase. The main difference between the exponential and deviance loss functions is the rate at which the penalties for $y\,f(\boldsymbol{x}) \in [-\infty, 0]$ increases. In this sense the deviance loss function more closely resembles the soft margin loss: for increasingly large negative margins, the penalty increases in an approximately linear fashion. Conversely, application of the exponential loss function implies that penalties will increase exponentially with an increase in negative margins, which leads to very heavy penalties for badly misclassified cases. This is a good idea only if these cases are true representations of the underlying populations. However, when working with noisy data, or with data in which input patterns might have been mislabelled, large negative margins are likely to correspond to noisy or mislabelled inputs. In such settings assigning too heavy penalties to badly misclassified cases may yield a classifier that inappropriately concentrates on correctly classifying noisy or mislabelled inputs. Since the deviance loss function penalises cases with large negative margins less

harshly, it is more robust than the exponential loss function and therefore the preferred loss function in noisy setups.

A reasonable question at this stage is how one should decide which loss function to use, and what the effect of different loss function specifications on the solution of the problem in (2.15) will be. Below we summarise results in Zhu and Hastie (2005) which clarify the statistical properties of classifiers constructed to minimise the risk $E_{Y|\boldsymbol{x}}\left[L(Yf(\boldsymbol{x}))\right]$ when $L\left(y\,f(\boldsymbol{x})\right)$ is specified as either the soft margin-, exponential- or deviance loss function. We will see that particular loss function specifications result in well known kernel classifiers. For example, the solution to the minimisation problem in (2.15) using $L\left(y_i f(\boldsymbol{x}_i)\right)=\left|1-y_i f(\boldsymbol{x}_i)\right|_+$ is a support vector classifier (*cf.* Girosi *et al.*, 1995; Wahba, 1998; Evgeniou *et al.*, 2000, and Hastie *et al.*, 2001). Although, as we will see in Chapter 4, the support vector classifier was not originally derived as the solution to a regularised optimisation problem, viewing the support vector discriminant function as the solution to a regularisation problem using the loss function $L\left(y_i f(\boldsymbol{x}_i)\right)=\left|1-y_i f(\boldsymbol{x}_i)\right|_+$ provides additional insight into its structure. In order to appreciate this point, note that the function which minimises $E_{Y|\boldsymbol{x}}\left[\left|1-Yf(\boldsymbol{x})\right|_+\right]$ is

$$\begin{aligned} f^*(\boldsymbol{x}) &= \arg\min_{f(\boldsymbol{x})}\left\{E_{Y|\boldsymbol{x}}\left[\left|1-Yf(\boldsymbol{x})\right|_+\right]\right\} \\ &= P\left(Y=1|\boldsymbol{x}\right)-\tfrac{1}{2}, \end{aligned} \tag{2.16}$$

indicating that it is not unreasonable to conclude that a support vector discriminant function will be a good estimator of the posterior probability $P\left(Y=1|\boldsymbol{x}\right)-\frac{1}{2}$.

Now compare the form of the solution in (2.16) with the function which minimises $E_{Y|\boldsymbol{x}}\left[exp\left(-Yf(\boldsymbol{x})\right)\right]$ and $E_{Y|\boldsymbol{x}}\left[log\left(1+exp\left(-Yf(\boldsymbol{x})\right)\right)\right]$ in the case of the exponential and the deviance loss functions respectively. In both cases we obtain

$$
\begin{aligned}
f^*(\boldsymbol{x}) &= \arg\min_{f(\boldsymbol{x})} \left\{ E_{Y|\boldsymbol{x}}\left[exp\left(-Yf(\boldsymbol{x})\right)\right]\right\} \\
&= \arg\min_{f(\boldsymbol{x})} \left\{ E_{Y|\boldsymbol{x}}\left[log\left(1+exp\left(-Yf(\boldsymbol{x})\right)\right)\right]\right\} \\
&= \tfrac{1}{2} log\left\{ P\left(Y=1|\boldsymbol{x}\right)/P\left(Y=-1|\boldsymbol{x}\right)\right\}. \qquad (2.17)
\end{aligned}
$$

Estimating $f^*(\boldsymbol{x})$ in (2.17) by $\hat{f}^*(\boldsymbol{x})$, we may obtain a classifier $sign\left\{\hat{f}^*(\boldsymbol{x})\right\}$. Based on the deviance loss function, this classifier is known as a *kernel logistic regression* (*KLR*) function (*cf*. Zhu and Hastie, 2005).

The form of the risk minimising functions in (2.16) and (2.17) confirms the soft margin-, exponential- and deviance loss functions to be appropriate in classification setups. Knowing what functions are being estimated by solving the regularisation problem in (2.15) may also aid in a decision regarding which loss function to use.

Also from Expression (2.17), an important connection between the exponential and deviance loss functions becomes apparent: the corresponding minimising functions of $E_{Y|\boldsymbol{x}}\left[L\left(Yf(\boldsymbol{x})\right)\right]$ have the same form. In addition, note that (2.17) may be rewritten so that we obtain

$$
P\left(Y=1|\boldsymbol{x}\right)=\frac{e^{\hat{f}^*(\boldsymbol{x})}}{e^{-\hat{f}^*(\boldsymbol{x})}+e^{\hat{f}^*(\boldsymbol{x})}}=\frac{1}{1+e^{-2\hat{f}^*(\boldsymbol{x})}}, \qquad (2.18)
$$

which points to a definite advantage of kernel logistic discriminant functions compared to support vector classifiers. In the former case, probabilities of group membership can easily be estimated. That is, $\hat{P}\left(Y=1|\boldsymbol{x}\right)=\frac{1}{1+e^{-2\hat{f}^*_{KLR}(\boldsymbol{x})}}$, where $\hat{f}^*_{KLR}$ denotes the solution to (2.15) based on the deviance loss. A similar result can however not be obtained for a support vector classifier. Furthermore, kernel logistic regression generalises naturally to

kernel multi-logit regression, offering a simple way of handling more than two groups in classification. Here the estimators of group probabilities will be

$$\hat{P}(Y = j|\boldsymbol{x}) = e^{\hat{f}_j^*(\boldsymbol{x})} \Big/ \sum_{m=1}^{M} e^{\hat{f}_m^*(\boldsymbol{x})}, \tag{2.19}$$

where M denotes the number of groups, and $\sum_{m=1}^{M} \hat{f}_m^* = 0$.

Although there are ways to extend binary SVMs to classification for multiple groups, these procedures are intuitively less appealing compared to the natural way in which kernel logistic regression handles classification into more than two groups.

Since support vector and kernel logistic regression discriminant functions are solutions to the regularised optimisation problem in (2.15), we know from Theorem 2.6 that both may be written in the form $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$. Importantly, in the case of a support vector discriminant function, the truncation property of the soft margin loss typically causes a sizeable fraction of $\alpha_1, \alpha_2, \ldots, \alpha_n$ to be zero. Therefore a support vector classifier will only depend on the training patterns with non-zero coefficient values, called *support vectors*. Replacing the soft margin loss with the deviance loss function unfortunately compromises the aforementioned *support vector property*. Hence in kernel logistic regression all of $\alpha_1, \alpha_2, \ldots, \alpha_n$ will typically be non-zero values. In this sense computation of $\hat{f}_{KLR}^*$ will in general be more expensive than computation of $\hat{f}_{SVM}^*$. In Zhu and Hastie (2005) the authors propose a classification procedure (called an *import vector machine*) which can be obtained by substituting the loss function $L(y_i, f(\boldsymbol{x}_i)) = log(1 + exp(-f(\boldsymbol{x}_i))) - y_i f(\boldsymbol{x}_i)$ in problem (2.15). This particular form of the loss function is meant to induce a classifier exhibiting the positive attributes of both a support vector and kernel logistic regression classifier, and it succeeds: an import vector machine typically has few non-zero $\alpha_1, \alpha_2, \ldots, \alpha_n$ coefficients, allows simple calculation

of posterior group membership probabilities, and can easily be extended to multi-group classification contexts.

We now briefly turn attention to expression (2.6) in a regression setup, and illustrate how the ridge regression estimator may be obtained as the solution to the regularised optimisation problem in (2.15). We have seen in Example 2.3 that the ridge regression estimator is the minimiser of a regularised risk functional, *viz.* $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \{ R(\boldsymbol{\beta}) \}$, where

$$R(\boldsymbol{\beta}) = \tfrac{1}{n} \| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \|^2 + \lambda \langle \boldsymbol{\beta}, \boldsymbol{\beta} \rangle . \tag{2.20}$$

Minimising (2.20) is straightforward: differentiating $R(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ and equating to zero yields the normal equations $\left[\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p \right] \boldsymbol{\beta} = \boldsymbol{X}'\boldsymbol{y}$ with solution $\tilde{\boldsymbol{\beta}} = \left[\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p \right]^{-1} \boldsymbol{X}'\boldsymbol{y}$. Alternatively, we can rewrite the normal equations in the form $\boldsymbol{\beta} = \boldsymbol{X}'\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})/\lambda$. Writing $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{K}'\boldsymbol{\alpha}$, where $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}'$, we see that $R(\boldsymbol{\beta})$ in (2.20) becomes (2.15). Note that $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}'$ essentially implies that we are using a linear kernel: the ij^{th} element of the matrix $\boldsymbol{X}\boldsymbol{X}'$ is simply the inner product between the i^{th} and j^{th} rows of $\boldsymbol{X}$. Note also that positive semi-definiteness of the kernel follows immediately from the fact that

$$\boldsymbol{v}'\boldsymbol{X}\boldsymbol{X}'\boldsymbol{v} = \left\langle \sum_{i=1}^{n} v_i \boldsymbol{x}_i, \sum_{j=1}^{n} v_j \boldsymbol{x}_j \right\rangle = \left\| \sum_{i=1}^{n} v_i \boldsymbol{x}_i \right\|^2 = \| \boldsymbol{X}'\boldsymbol{v} \|^2 \geq 0 . \tag{2.21}$$

We also see that

$$f(\boldsymbol{x}) = \langle \boldsymbol{x}, \tilde{\boldsymbol{\beta}} \rangle = \boldsymbol{\alpha}'\boldsymbol{X}\,\boldsymbol{x} = [\alpha_1 \ \alpha_2 \ldots \ \alpha_n] \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x} \rangle \\ \langle \boldsymbol{x}_2, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle \end{bmatrix} \tag{2.22}$$

which is in line with (2.13). Finally note that implementing the solution $\tilde{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$ requires determination of n quantities, $\alpha_1, \alpha_2, \ldots, \alpha_n$, rather than the p quantities $\beta_1, \beta_2, \ldots, \beta_p$. This is a definite advantage in cases where p becomes much larger than n, as is typically the case if we work in a high-dimensional feature space. We refer to $\alpha_1, \alpha_2, \ldots, \alpha_n$ as the components of the *dual solution*, while $\beta_1, \beta_2, \ldots, \beta_p$ is called the *primal solution*.

We therefore see that the ridge regression optimisation problem presented in Example 2.3 can also be cast within the more general framework in (2.15). This also presents the possibility of formulating a generalised version of the ridge problem by replacing the linear kernel with an appropriate alternative.

Summarising, we have seen that kernel classifiers are solutions to regularised optimisation problems. This explains the ability of kernel classifiers to generalise well, despite the high-dimensional spaces in which they are defined. Focusing on support vector- and kernel logistic discriminant functions, we have also illustrated that different classes of kernel classifiers may be obtained through appropriate specification of the loss function. Although kernel classifiers were not originally motivated through a regularised risk minimisation approach, the underlying theory provides insight regarding the function that each kernel classifier class aims to estimate, and facilitates comparisons across kernel techniques.

Having introduced the most important concepts required in training kernel classifiers, we now turn our attention to the important problem of variable selection in binary classification applications.

2.3 *VARIABLE SELECTION IN BINARY CLASSIFICATION: IMPORTANT ASPECTS*

The remainder of this chapter is devoted to variable selection in binary classification contexts. We therefore briefly reconsider the two-group classification problem and notation introduced in Chapter 1, where a data set $\mathcal{T}$ (consisting of n measurements on p input variables) is used to estimate a classification function which can subsequently be used to predict the group membership of new data cases. In many such scenarios little or no prior knowledge is available for a decision regarding how many and which input variables to observe and include in $\mathcal{T}$. A consequence of this is that the original set of input variables will usually include some variables which do not contribute any useful information. Before estimating the classifier, a data-dependent decision regarding possible reduction in the initial set of input variables is therefore often required. Potential advantages of such a reduction have already been discussed in Section 2.1: reducing a (large) set of input variables to a smaller set which contains only the important variables leads to more accurate classifiers, while simultaneously identifying the variables which separate the two groups. Although these two advantages often go hand in hand, different approaches to variable selection tend to focus more sharply on one of the two. In this regard, different strategies and criteria for variable selection may be described as *allocatory* (emphasising more accurate classification) or *separatory* (emphasising identification of the variables separating the groups). This aspect is discussed in more detail in Section 2.3.1.

It should also be noted that one can distinguish two levels of decision making in variable selection. Firstly, a decision regarding the number of variables to retain has to be made, and secondly, we have to decide which variables these should be. More detail in this regard will be given in Chapter 6, where we specifically address the problem of specifying a value for the number of input variables to retain (m). Before proceeding to a detailed discussion of other important aspects of variable selection for classification, we provide a brief historical perspective on the topic. Variable selection has for many years received considerable attention in the statistics literature. Traditionally selection techniques were designed in the context of small to medium sized data sets (*i.e.* data sets usually having

fewer than 1000 observations on usually no more than 100 input variables). These techniques often relied on assumptions regarding the underlying distribution generating the data. For example, many classical selection criteria are based on the *F*-test for *no additional information*. For a comprehensive overview of classical variable selection procedures, the reader is referred to Louw (1997) and references cited there, including McKay and Campbell (1982 *a* and *b*), and McLachlan (1992).

Applications such as image and text analysis have traditionally been areas of expertise in mainly the computer and information sciences, and typically involve very large data sets. Large input dimensions create a strong need for variable selection. Kernel methods belong to the class of procedures known to perform well in applications characterised by large numbers of input variables. But even these methods benefit from an initial variable selection step, as is evidenced by the growing literature on variable selection for kernel methods, and also by an investigation regarding the importance of variable selection for kernel methods which is reported in Section 2.4.1.

It seems as if during the last decade much of the research in kernel variable selection has taken place in the computer science and machine learning fields, and considerable scope still exists for contributions from the statistics and data mining disciplines. In this regard, it should however be borne in mind that modern variable selection applications pose challenges not encountered in classical variable selection. Therefore it comes as no surprise that approaches to variable selection for kernel methods are quite different to those in more traditional selection. We will elaborate more on this point in Section 2.4.

2.3.1 *THE RELEVANCE OF VARIABLES*

In order to further discuss variable selection in classification contexts, we first require clarity regarding the concept of *variable relevance*. Relatively few papers in the literature contain formal definitions of the relevance of variables. In this section we present some informal definitions regarding degrees and types of variable relevance. Several subtleties come into play when one considers the relevance of variables. We indicate important

points to bear in mind, and provide examples illustrating some intricacies involved in determining the relevance of variables.

In our opinion, a limitation of many papers on variable selection is their failure to distinguish between the different objectives commonly pursued in classification contexts. In some cases, the most important objective may be correct classification of new cases, *i.e.* the emphasis is placed on the *predictive* (*allocatory*) aspects of a classifier. In other cases, the greater interest may lie in a clear understanding of the nature of separation between groups. In such classifiers, the *descriptive* (*separatory*) aspects of the resulting decision functions are more important. This distinction is made by McLachlan (1992). Variable selection procedures should be clear regarding the intended purpose of selection: is the selection process derived to focus more sharply on improving the allocatory or separatory properties of a classifier? Most variable selection proposals in the literature assume that the allocatory aspects of the classifier are more important. In this way selection procedures often neglect the importance of classifiers explaining the separation among groups. McKay and Campbell (1982 *a* and *b*) and Louw (1997) share this sentiment and categorise variable selection criteria in discriminant analysis according to whether they primarily incorporate the allocatory or separatory aspect of a discriminant function. There is however still much scope for variable selection procedures that concentrate more on enhancing the separatory properties of a classifier.

A similar distinction is possible regarding the relevance of variables: a variable may be helpful as far as the correct allocation of a new case to one of the groups is concerned, and useless regarding concise description of the separation between groups, or vice versa. In other cases, a variable may be relevant in both an allocatory and separatory sense. Therefore *allocatory relevance* rates variables according to the extent to which their inclusion yields smaller $Err(f)$, whereas the importance of variables in terms of how well they explain differences between groups may be referred to as *separatory relevance*.

Regarding variable relevance in both an allocatory and separatory sense, a further distinction is possible. A variable may either be considered *strongly* or *weakly* relevant in

allocating new cases, or in describing group separation. John *et al*. (1994) provide informal definitions of strong and weak relevance in an allocatory sense. Note however that similar definitions can also be given to further distinguish among variables which are relevant with regard to the separation between groups. See also Blum and Langley (1997).

DEFINITION 2.4 ***STRONG RELEVANCE***

Let $\mathcal{J} = \{1,2,\ldots,p\}$ index the set of available variables $X_1, X_2, \ldots, X_p$. Also, let G represent the superset containing all possible subsets of $\mathcal{J}$, and use $Err[\Gamma]$ to generically denote the generalistion error of a model based on the variables with indices in Γ. Now consider an arbitrary variable X_j, $j \in \mathcal{J}$, in this set, and let $G_{(+j)}$ and $G_{(-j)}$ (both $\subseteq G$) represent all possible subsets of $\mathcal{J}$, with and without X_j respectively, *i.e.* $G_{(+j)} = G_{(-j)} \cup \{j\}$. Variable X_j is said to be *strongly relevant* if $Err[G_{(+j)}]$ is always smaller than $Err[G_{(-j)}]$.

From Definition 2.4 it is clear that a strongly relevant variable can never be removed without negatively affecting generalisation performance.

DEFINITION 2.5 ***WEAK RELEVANCE***

A variable X_j is considered *weakly relevant* if it is not strongly relevant, and if there exists a subset of variables not containing X_j, *i.e.* $\widetilde{G}_{(-j)}$, such that $Err[\widetilde{G}_{(+j)}]$ is smaller than $Err[\widetilde{G}_{(-j)}]$.

Therefore inclusion of a weakly relevant variable can in some cases yield higher prediction accuracy.

The main point to be made in this section concerns the importance of not considering variables individually when determining their relevance. In variable selection one is typically interested in obtaining the subset of variables which, when used in combination with one another, will be the most relevant. In the next example we will show that the relevance of individual variables can be much different when their use is considered in combination with other sets of variables. In particular, we will indicate that a variable which by itself does not contribute to the separation between two groups may indeed provide helpful information regarding the classification of future cases.

EXAMPLE 2.5

Reconsider the binary classification problem as stated in Chapter 1, and let the correlation between observations on the pair of variables X_1 and X_2 be denoted by ρ. Assume that ρ has a fairly large positive value. Furthermore, let variable X_1 provide relatively clear separation between the two groups in terms of location, but with measurements on X_2 for the two groups following roughly the same distribution. Specifically, conditioning on group membership, let the distribution of the data cases be

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \Big| Y = +1 \sim N\left(\begin{bmatrix} 5 \\ 5 \end{bmatrix}; \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right) \text{ and } \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \Big| Y = -1 \sim N\left(\begin{bmatrix} 12.5 \\ 5 \end{bmatrix}; \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right). \qquad (2.23)$$

We present the marginal distributions of X_1 and X_2 in Figures 2.8 and 2.9 respectively.

It would seem from Figure 2.9 that an observed value of X_2 cannot in any way assist us in allocating a data case to one of the two groups. However, consider a new case $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ with the measurement x_1 falling in the region of overlap between the two groups (in Figure 2.10). Clearly, on its own x_1 does not seem to be of much help in classifying the new observation, since x_1 can either be considered an exceptionally large value from group 1, or an exceptionally small value from group 2.

However, consider the observed X_2-value for a new case. From the assumed correlation structure between X_1 and X_2, an above (or below) average value for X_2 may

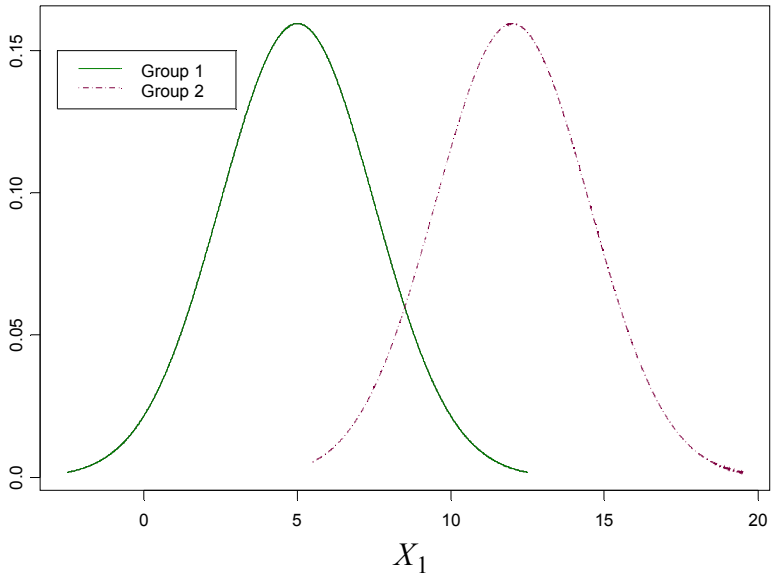


Figure 2.8: *X_1 provides relatively clear separation between the two groups*

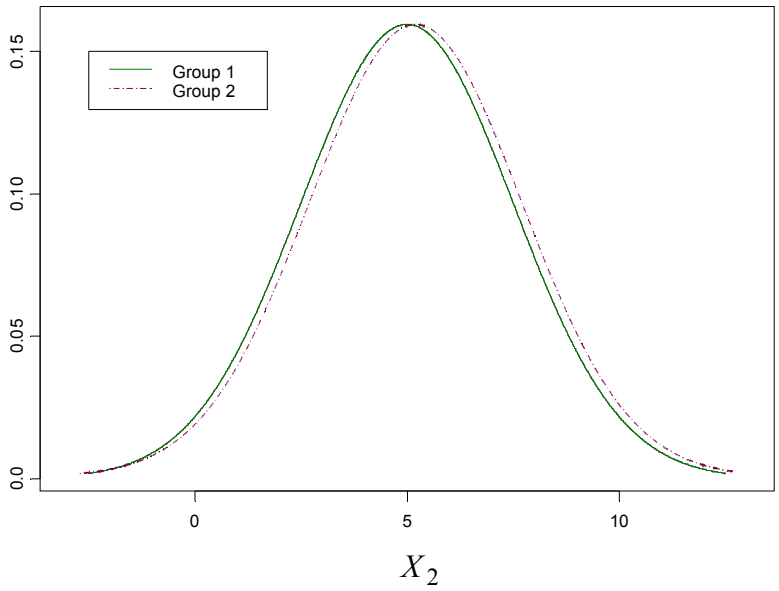


Figure 2.9: *X_2 provides no separation between the two groups*

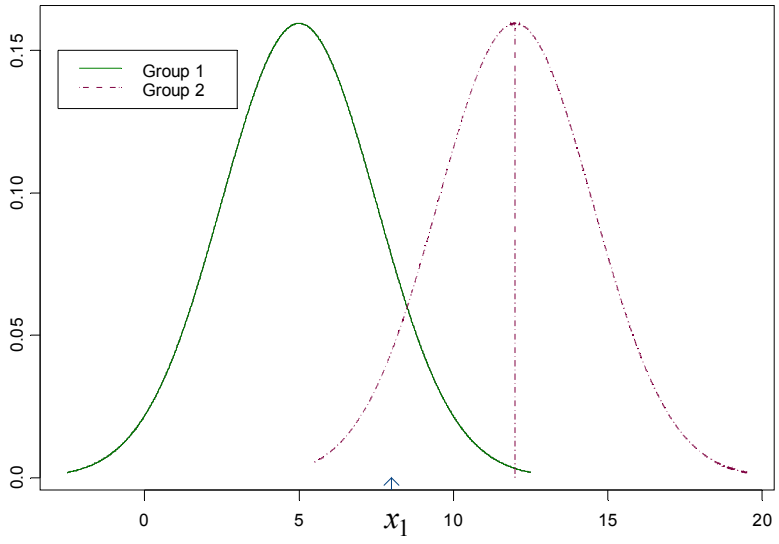


Figure 2.10: *Suppose* x_1 *in* $\boldsymbol{x}$ *falls in the region of overlap – therefore on its own* x_1 *is not useful in the classification of* $\boldsymbol{x}$

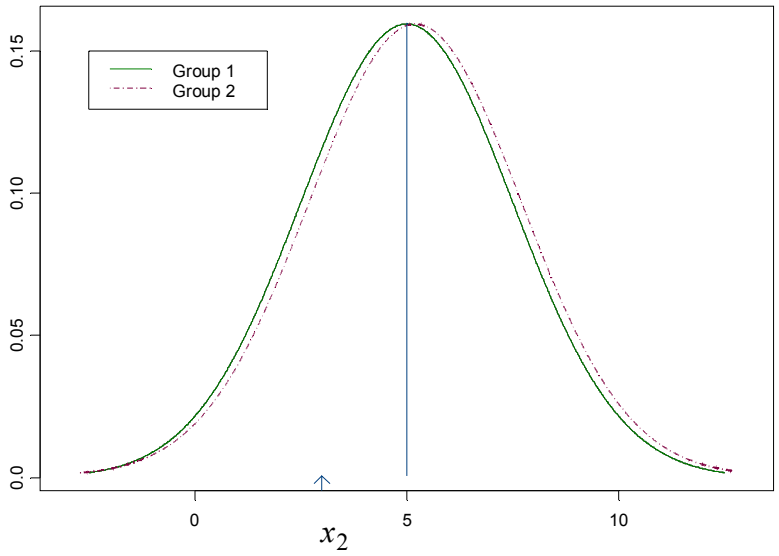


Figure 2.11: *Here* x_2 *lies below the average of the* x_2 *values and is indicative of* $\boldsymbol{x}$ *belonging to group* 2

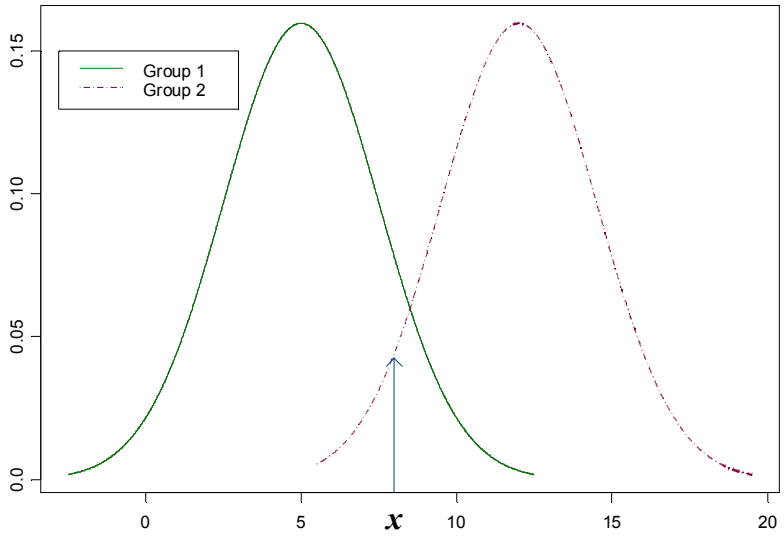


Figure 2.12: *The new case is allocated to group* 2

be considered indicative of the x_1 value lying above (or below) the average of X_1. Hence if x_2 is below the average of the x_2 values, the observed x_1 value is most likely to lie below the average of the x_1 values as well, implying that the new case should be allocated to group 2 (see Figures 2.11 and 2.12).

A similar argument is valid for the case where x_2 is above the average of the x_2 values. In this way, knowledge of the x_2 value assists in the allocation of the new case, *i.e.* including X_2 in the classifier most probably will result in a smaller $Err(f)$. Note however in this example that if there were no correlation between X_1 and X_2, inclusion of X_2 would not improve $Err(f)$. In this example variable X_2 may therefore be considered weakly relevant. □

From Example 2.5 we see that it is important to determine the relevance of a variable when it is used in combination with (ideally) the rest of the variables which will be contained in

the model: variables that are completely irrelevant on their own can often, when used in combination with another (set of) variable(s), provide important auxiliary information and thus be considered relevant. If the relevance of variable X_2 in Example 2.5 were determined by only considering the contribution of X_2 when it is used on its own, its (weak) relevance (when used in combination with variable X_1) would not have been revealed.

We now provide two further examples from Guyon and Ellisseeff (2003) which illustrate that determining the relevance of input variables is not a trivial matter.

EXAMPLE 2.6

Let X_1 and X_2 be two independent and identically distributed variables. In particular, let

$$\boldsymbol{X}|Y=+1 \sim N\left(\begin{bmatrix}1\\1\end{bmatrix};\boldsymbol{\Sigma}\right) \text{ and } \boldsymbol{X}|Y=-1 \sim N\left(\begin{bmatrix}-1\\-1\end{bmatrix};\boldsymbol{\Sigma}\right), \tag{2.24}$$

where $\boldsymbol{X}'=\begin{bmatrix}X_1 & X_2\end{bmatrix}'$ and $\boldsymbol{\Sigma}=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$.

Since X_1 and X_2 provide exactly the same separation between the two groups, it might at first seem unnecessary to make use of both these variables in order to discriminate between the two groups.

Consider however the orthogonal transformation $\boldsymbol{U}=\boldsymbol{A}\boldsymbol{X}$ where

$$\boldsymbol{A}=\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{bmatrix}. \tag{2.25}$$

Hence, conditioning on group membership, the distribution of the transformed data cases will be

$$\boldsymbol{U}|Y=+1 \sim N\left(\begin{bmatrix}\sqrt{2}\\ 0\end{bmatrix};\boldsymbol{\Sigma}\right) \text{ and } \boldsymbol{U}|Y=-1 \sim N\left(\begin{bmatrix}-\sqrt{2}\\ 0\end{bmatrix};\boldsymbol{\Sigma}\right), \tag{2.26}$$

with $\boldsymbol{U}'=\begin{bmatrix}U_1 & U_2\end{bmatrix}'$. From (2.26) we see that, although the separation provided by $U_1/\sqrt{2}$, and the separation provided by X_1 (or X_2) are both equal to 2, the standard deviation of $U_1/\sqrt{2}$ is $1/\sqrt{2}$, as opposed to the unit standard deviation of X_1.

In this case much is gained by using U_1 (and therefore including also variable X_2) instead of only making use of variable X_1 (or vice versa). The separation between two groups can be improved by adding variables that are identically and independently distributed.

□

The following example illustrates that a significant correlation between input variables does also not necessarily indicate one of the highly correlated input variables to be irrelevant.

EXAMPLE 2.7

Consider variables X_1 and X_2, and suppose their marginal distributions, conditioning on group membership, are once again

$$\begin{bmatrix}X_1\\ X_2\end{bmatrix}|Y=+1 \sim N\left(\begin{bmatrix}1\\ 1\end{bmatrix};\boldsymbol{\Sigma}\right) \text{ and } \begin{bmatrix}X_1\\ X_2\end{bmatrix}|Y=-1 \sim N\left(\begin{bmatrix}-1\\ -1\end{bmatrix};\boldsymbol{\Sigma}\right), \tag{2.27}$$

but now $\boldsymbol{\Sigma}=\begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}$. In this case the transformation $U=(X_1+X_2)/\sqrt{2}$ implies

$$U|Y=+1 \sim N\left(\sqrt{2};1+\rho\right) \text{ and } U|Y=-1 \sim N\left(-\sqrt{2};1+\rho\right). \tag{2.28}$$

Hence we see that X_1, X_2 and $U/\sqrt{2}$ provide a separation in location of 2. The standard deviation of the distribution of $U/\sqrt{2}$ is however $\sqrt{(1+\rho)/2}$, a value that will be smaller than the standard deviation of the distributions of X_1 and X_2 if $-1 \leq \rho < 1$.

□

2.3.2 *SELECTION STRATEGIES AND CRITERIA*

Given that we may assume a fixed value for m, the number of variables to be selected, different strategies exist for identifying which m out of p variables this should be. We proceed with a description of these strategies, once again placing emphasis on selection in binary classification. The most natural strategy for variable selection is an *all possible subsets* approach. This approach is based on an appropriate selection criterion and entails calculating the criterion for all $\binom{p}{m}$ possible subsets and selecting the subset corresponding to an optimum value of the criterion. If the criterion emphasises separation of the two groups, the optimum value will typically be a maximum, while in the case of an allocatory criterion optimisation usually corresponds to finding a minimum value. In application domains for kernel methods, note however that m may still be a very large number, rendering such an all possible subsets approach impossible from a computational point of view. Alternative classical variable selection strategies (*viz*. backward elimination, forward selection, and stepwise algorithms) restrict the optimisation process to a much smaller number of variable subsets. These selection strategies are iterative: at each step a variable is added to and/or eliminated from a current set of variables, until a sequentially constructed final subset of size m is reached. Whether a variable is added to and/or removed from a current set of variables is the aspect which differentiates the aforementioned selection strategies. Before we discuss implementation of these selection strategies, the following additional notation is first required.

Let the set of indices corresponding to the entire set of initially available input variables be denoted by $\mathcal{J} = \{1,2,\ldots,p\}$, and the corresponding full set of input variables by $\mathcal{V}$. Furthermore, let subsets of $\mathcal{J}$ be denoted generically by J (and input variable subsets by V), and the complement of J in $\mathcal{J}$ be $\bar{J}$, *i.e.* $\bar{J} = \mathcal{J} - J$. Also, let the number of variables in a subset J be equal to $card(J) = j$. Recall that in Chapter 1 we defined $\boldsymbol{X}$ as the data matrix consisting of n measurements on all variables in $\mathcal{V}$. Using the indices contained in a subset J we now construct a data matrix $\breve{\boldsymbol{X}}_J$ with rows equal to $\breve{\boldsymbol{x}}_i'$, $i = 1,2,\ldots,n$, where $\breve{\boldsymbol{x}}_i$ is a j-component $(0 < j \leq p)$ column vector containing the values of input variables X_h, $h \in J$, for case i in the training data.

Since algorithms for backward elimination, forward selection and stepwise procedures usually involve several steps, note that we will write J_k to indicate a set of input variable indices obtained after the k^{th} step of the algorithm. Moreover, we will use the notation $J_{k(-h)}$ to indicate a particular subset of J_k, *viz.* the set J_k without variable index h. That is, $J_{k(-h)} \subset J_k$ where $h \in J_k$. We will also need notation for a specific superset of J_k, *viz.* the set J_k after adding index h. Let this superset be denoted by $J_{k(+h)}$, *i.e.* $J_{k(+h)} \supset J_k$ where $h \notin J_k$. We will use similar notation to refer to the corresponding subsets of input variables, *viz.* $V_{k(\pm h)} \supset V_k$, where V_k is the subset of input variables included during the k^{th} step of the selection algorithm. Finally, we require notation for the selection criterion calculated by using only the training sample observations in $\breve{\boldsymbol{X}}_J$. Hence during the k^{th} step in the selection algorithm, let c_k denote the value of the selection criterion based on $\breve{\boldsymbol{X}}_{J_k}$. Similarly $c_{k(-h)}$ and $c_{k(+h)}$ represent the criteria using only the observations in $\breve{\boldsymbol{X}}_{J_{k(-h)}}$ and $\breve{\boldsymbol{X}}_{J_{k(+h)}}$ respectively.

Having introduced the necessary variable subset notation, we now proceed with a discussion of a backward elimination strategy. Backward elimination starts with the full

set of available variables, therefore $J_0 = \mathcal{J}$. Each step in backward selection requires the elimination of a variable, and subsequent updating of the current set of variables. During for example the first step in the selection algorithm, one needs to determinine which of the variables in V_0 to eliminate. For this purpose the effect of omitting each of the variables in V_0 one at a time, is required, *i.e.* the estimated generalisation performances of the variable subsets $V_{0(-1)}, V_{0(-2)}, \ldots, V_{0(-p)}$ have to be compared. Hence step one in backward selection involves calculation of the values $c_{0(-1)}, c_{0(-2)}, \cdots, c_{0(-p)}$ of the selection criterion, followed by identification of the optimum criterion value. Suppose the optimum criterion value is $c_{0(\hat{h})}$, then $X_{\hat{h}}$ is the variable indicated as the least relevant variable in $\mathcal{V}$ and we proceed by eliminating $X_{\hat{h}}$ from V_0. The current current set of variable indices is then updated, yielding $J_1 = J_{0(-\hat{h})} = J_0 - \{\hat{h}\}$ and variable subsets $\{V_{1(-h)}, h \in J_1\}$ to consider during the next step in the algorithm. The above process continues until only *m* variables remain, therefore terminating after step $k = p - m$, and sequentially generating nested subsets $V_0 \supset V_1 \supset V_2 \supset \ldots \supset V_{p-m}$. With *p* typically very large, eliminating variables one at a time is still a costly procedure. Hence it may often be necessary to remove groups of variables at each step, or at least during the initial stages of the process. Letting *r* denote the number of repetitions employed in the selection, note therefore that in general $r \leq p - m$, with equality holding if each step amounts to elimination of only a single variable.

Forward selection proceeds along the same lines as backward elimination, except that at each step, variables may potentially be added instead of eliminated. Also in the case of forward selection, more than one variable may be added at a time. After *r* steps, the nested subsets $V_0 \subset V_1 \subset V_2 \subset \ldots \subset V_r$ are returned. Of course, *r* will depend on *m*, *p* and the number of variables simultaneously entering the model during each step of the algorithm.

Guyon and Elisseeff (2003) indicate the important aspects to consider in deciding whether to implement a backward elimination or a forward selection procedure. From their

discussion, backward elimination seems to be the more appropriate choice in most cases. A disadvantage of backward elimination is that in cases of a small number of variables to be retained (small m), selection may take significantly longer than would be the case with forward selection. Backward elimination strategies do however beat forward selection in more than one respect. Firstly, incorrect inclusions in forward selection often cause the importance of other variables not to be detected. Thus forward selection may yield a completely suboptimal subset of variables as a consequence of misleading inclusion of irrelevant variables early on. Secondly, in forward selection strategies, the full set of available variables is never assessed. For a small value of m relative to the value of p, the forward selection process may in fact terminate far short from considering the full set of available variables. In backward elimination the full set of potential variables is considered at the first step.

Stepwise selection may be viewed as a strategy that tries to prevent wrong inclusions from blinding the procedure to correct inclusions later on. It may essentially be described as a combination of forward selection and backward elimination: at step k, after an input variable (say $X_{\hat{h}_k}$) has been *added* to V_k, possible *elimination* of each of the input variables in V_k is considered. In regression contexts, forward stagewise- and forward stepwise-selection are forward selection strategies based on the correlation between an input variable and the response, as selection criterion. A new development, which may be considered a more cautious (less greedy) version of forward stepwise regression, is the least angle regression and shrinkage (LARS) algorithm (*cf*. Efron *et al*., 2004). Application of the LARS proposal has since been extended to classification setups (specifically to selection in support vector classifiers using a linear kernel function, *cf*. Keerthi, 2005) as well. A brief discussion of the LARS algorithm and related selection procedures follows after the next paragraph regarding (classical) selection criteria.

Having discussed strategies which avoid consideration of every possible variable subset, the remaining difficult aspect is a decision regarding which criterion to use in rating the various subsets. In the context of ordinary discriminant analysis and linear regression,

several approaches with regards to selection criteria and their application in the various selection strategies above, may be found in the literature. Important references in this regard include McKay and Campbell (1982 *a* and *b*), McLachlan (1992), Louw (1997) and George (2000). We have earlier on referred to the useful qualification regarding selection criteria made by these authors, *viz*. that selection criteria can be broadly categorised according to whether they are primarily of a separatory or allocatory nature. Examples of separatory criteria in classical discriminant analysis include the squared multiple correlation coefficient, Akaike's Information Criterion, and Mallows' C_p criterion. Also the *F*-statistic was derived to test for so-called *no additional information* provided by a variable. In classification problems where the correct allocation of future cases is the primary concern, selection criteria are typically simply estimators of the generalisation error rate.

As an alternative to the above approach to variable selection (involving specification of a selection- criterion and strategy), several more novel proposals in the literature implicitly perform variable selection as part of the learning algorithm. Examples are the non-negative garotte (Breiman, 1995), the lasso (Tibshirani, 1996) and the elastic net (Zou and Hastie, 2005) – all derived in a linear regression context, but also been shown to work well in classification problem domains: Ghosh and Chinnaiyan (2004) apply the lasso in cancer classification problems, and Zou and Hastie (2005) state that '*the elastic net penalty can be used in classification problems*'. With a view to our focus on binary classification, we start by briefly considering the lasso in regression contexts, and then switch over to its equivalent form for generalised linear models (as proposed by Park and Hastie, 2006).

The non-negative garotte, lasso and elastic net algorithms in regression are examples of regularised (or shrinkage) estimation, and are thus connected to ridge regression (*cf*. Friedman *et al*., 2007) for a summary of these techniques. The regularised optimisation problem in the lasso, for example, is:

$$\min_{\boldsymbol{\beta}} \left\{ R(\boldsymbol{\beta}) = \tfrac{1}{n} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \right\|^2 + \lambda \sum_{j=1}^{p} \left| \beta_j \right| \right\} \quad . \tag{2.29}$$

By substituting the regulariser $\sum_{j=1}^{p}\left|\beta_j\right|$ for $\sum_{j=1}^{p}\beta_j^2$ in the case of ridge regression, some of the regression coefficients may be shrunk to zero. Therefore the lasso alters ridge regression to implement variable selection. For small enough values of λ, the regularisation constraint will have no effect, and the estimated regression coefficients will be the usual least squares estimates. As with ridge regression, large values of λ induce bias in the estimation, and an estimate with smaller variance. Appropriate specification of λ often ensures a trade-off between the larger bias and smaller variances that result in a smaller generalisation error. In variable selection, smaller subsets of input variables may also yield larger bias, but fewer coefficients to estimate and therefore hopefully also a reduction in variance. Accurate estimation of the true model dimension can be thought of as being equivalent to correct specification of λ. Cross validation is often considered a good tool for specifying λ.

The optimisation problem in (2.29) is a quadratic optimisation problem that can, for any positive value of λ, be solved by standard numerical procedures. It is however shown in Efron *et al*. (2004) that lasso estimates for all values of λ can simultaneously be obtained via the LARS algorithm. Osborne *et al*. (2000) have also devised an algorithm for efficiently solving (2.29).

Despite first-class performances in many scenarios, Zhu and Hastie (2005) list three situations in which the lasso does not perform as well as one would have hoped. The first of these occurs in the case of wide data sets characterised by (many) more variables than observations. In this case, the lasso can select a maximum of only n input variables. Also, the lasso is undefined for λ larger than a certain value. The second unfavourable scenario involves groups of input variables for which correlations between pairs of variables are high. In such situations, the lasso often selects only a single variable from the highly correlated group, causing the importance of the variables with which it is correlated to be masked. Thirdly, in the case of $p < n$ and highly correlated input variables, it was shown by Tibshirani (1996) that ridge regression outperforms the lasso.

The elastic net was introduced with the aim of finding a regularised estimation technique that '*works as well as the lasso whenever the lasso does the best, and can fix the problems*…' given above (Zou and Hastie, 2005, *p*. 302).

The regularised optimisation problem for the elastic net makes use of both the lasso and ridge regression regularisers:

$$\min_{\boldsymbol{\beta}}\left\{R(\boldsymbol{\beta})=|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}|+\lambda_1\sum_{j=1}^{p}|\beta_j|+\lambda_2\sum_{j=1}^{p}\beta_j^2\right\}\ . \tag{2.30}$$

In the same way that LARS efficiently finds $\boldsymbol{\beta}$ for all possible λ values, Zou and Hastie (2005) developed an efficient algorithm (LARS-EN) for finding $\boldsymbol{\beta}$ for all possible non-negative values of λ_1 and λ_2. In fact, the computational cost of determining $\boldsymbol{\beta}$ for all possible $(\lambda_1;\lambda_2)$ combinations, is shown to be the same as for a single ordinary least squares fit. Since LARS and LARS-EN can be used to efficiently find the lasso and elastic net estimates for a sequence of λ, or λ_1 and λ_2 parameter values, they are examples of so-called *regularisation path-following* algorithms. Once again a cross-validation scheme over a $(\lambda_1;\lambda_2)$-grid is proposed for determining λ_1 and λ_2.

In Park and Hastie (2006), LARS and LARS-EN are extended to regularisation path-following algorithms for generalised linear models. Recall that in generalised linear models, the objective function to be maximised is $R(\boldsymbol{\beta})=\ell(\boldsymbol{y},\boldsymbol{\beta})$, where ℓ is the likelihood function with respect to the given data set. Through the introduction of a complexity penalisation term of the form $\sum_{j=1}^{p}|\beta_j|$, the usual estimation process is modified to a variable selection technique. The resulting regularised optimisation problem is

$$\min_{\boldsymbol{\beta}}\left\{-\ell(\boldsymbol{y},\boldsymbol{\beta})+\lambda\sum_{j=1}^{p}|\beta_j|\right\}. \tag{2.31}$$

Park and Hastie (2006) devise an algorithm for efficiently solving (2.31) for all positive values of the λ regularisation parameter, and demonstrate its use in the application of both penalised logistic regression (Lockhorst, 1999) and the Cox proportional hazards model (Cox, 1972).

There are statistical techniques (*e.g.* classification and regression trees (CART), multivariate adaptive regression splines (MARS) and random forests (RFs)) that are often considered to more naturally lend themselves to variable selection. References relevant to tree methodology are Breiman *et al*. (1984), De'ath and Fabricius (2000), and Hastie *et al*. (2001), and MARS was popularised by Friedman (*cf*. Friedman, 1991). Since some of our further investigations make use of RFs, we will conclude this section on selection strategies and criteria by briefly highlighting some important aspects with regards to the ranking of variables using output provided by most RF software.

Many positive characteristics can be attributed to classification and regression trees. As mentioned above, the relatively simple yet effective way in which variables can be ranked, is certainly one of the much acclaimed advantages in using CART. Unfortunately however, it is well known that trees are very sensitive to randomness in the data. Small perturbations in the data typically yield considerable changes in tree structures. Different data sets generated from the same distribution may result in completely different rankings of the input variables. Note that the same can be said of most other classical subset selection methods. Regularisation (shrinkage) approaches to selection are however much more stable. See Breiman (1996) for an extensive comparative study regarding instabilities and stabilisation of input variable selection in linear regression.

Random forests (RFs) for classification and regression is a class of methods which was designed partly in answer to the above shortcoming of CART. Again in our discussion we will assume the basic theory underlying RFs to be known (*cf*. for example Bauer and Kohavi, 1999; Breiman, 1996 and 2001; Dietterich, 1998; and Ho, 1998), and we therefore mainly focus on important aspects regarding the calculation of relative variable importance measures in RFs. Hence briefly, random forests are ensembles of single classification or

regression trees, each constructed after some form of randomness has been added to the originally available training data set. The final set of predictions is obtained as the majority vote over the forest of trees (in classification), or the average prediction across trees (in regression).

DEFINITION 2.6: ***A RANDOM FOREST***

A random forest is a classifier or regression function consisting of a collection of tree classifiers or tree regression functions

$$\left\{T\left(\boldsymbol{x},\boldsymbol{\theta}_k\right), k=1,2,\ldots,K\right\} \tag{2.32}$$

where $\boldsymbol{x}$ is an input vector and $\boldsymbol{\theta}_k$ is a random vector, generated independently from $\boldsymbol{\theta}_{k-1},\boldsymbol{\theta}_{k-2},\ldots,\boldsymbol{\theta}_1$, and the $\left\{\boldsymbol{\theta}_k, k=1,2,\ldots,K\right\}$ index the individual trees.

Breiman (2001) reports significant improvements in accuracy for RFs over CART. Moreover, it has already been mentioned that variable rankings may change considerably after even only slight perturbation of the original data set. RFs purposely inject randomness and after averaging results across individual trees, obtain more reliable variable importance values than those reported per individual tree.

There are numerous ways in which a random element may be incorporated into the growth of each individual tree in an RF. We mention only a few examples. The reader is referred to Breiman (1996 and 2001), Amit and Geman (1997), Dietterich (1998) and Ho (1998) for more details and remarks on how the different variations compare. Firstly, one may randomly select a set of training sample cases which is then used to grow each individual tree. A second proposal is to randomly select a split among a number of best splits. Thirdly, one could add a random element to the response values of the original training set. A fourth option is to use a random selection of the total number of available features to grow each tree.

In order to facilitate a discussion regarding variable importance in classification and regression random forests, the following background and notation is required. Let $f_1, f_2, \ldots, f_n$ be the set of response values to be estimated by a regression tree ensemble, and likewise, let $y_1, y_2, \ldots, y_n$ denote the binary outcomes to be estimated by a classification random forest. Now consider a classification or regression RF which incorporates a random element on two levels: firstly, a bootstrap sample $\mathcal{T}_t = \{(\boldsymbol{x}_i, y_i),\ i = 1, 2, \ldots, n\}$ is selected randomly and used to grow tree T_t, $t = 1, 2, \ldots, K$, where K denotes the total number of trees in the forest; and secondly, at each node in an individual tree, a random selection of m out of the total number of p input variables $(m < p)$, is made. Let V_R denote the set of randomly selected variables at a particular node, then at that node, a decision regarding which variable to split on is restricted to the m randomly selected variables in V_R. Note that V_R at distinct nodes in the tree will typically differ.

Each of the K trees is trained on a different bootstrap sample of cases from the original training data set and yield estimated values for $f_1, f_2, \ldots, f_n$. Thus the process of drawing bootstrap samples and training individual trees on them, yields $\hat{f}_{t1}, \hat{f}_{t2}, \ldots, \hat{f}_{tn}$, $t = 1, 2, \ldots, K$. Now let $\overline{T}_i$ indicate the set of trees that was trained on a bootstrap sample which did not contain training data case i. Then the random forest estimate of f_i is $\breve{f}_i = \sum_{t \in \overline{T}_i} \hat{f}_{ti} \Big/ card(\overline{T}_i)$, $i = 1, 2, \ldots, n$, where $card(\overline{T}_i)$ is the number of trees in $\overline{T}_i$.

The performance of random forests is typically evaluated via an out-of-bag error estimate, *viz*.

$$\frac{1}{n\left(card(\overline{T}_t)\right)} \sum_{i=1}^{n} \sum_{t \in \overline{T}_i} \left(\hat{f}_{ti} - \breve{f}_i\right)^2 . \tag{2.33}$$

It can be shown that the random forest error rate depends on the correlation between any two trees, and the strength of each individual tree in the forest (*cf*. Breiman, 2001). Low correlations and individual trees with low error rate yield random forests with low error rates. Specifying a lower value for m reduces the correlation between pairs of trees, but increases the error rate of individual trees. We of course need to specify a value for m which would lead to a favourable trade-off. Luckily, the range of acceptable values for m has been found to be quite wide. In the literature $m \cong \sqrt{p}$ yields good results in most cases.

Quantification of the relative importance of input variables typically forms part of random forest software. We used the following approach, as implemented in the software provided by Breiman and Cutler (2002). Consider tree T_t, and let $\bar{J}_t$ denote the set of cases which were not used in training this tree. We start by randomly permuting the entries of variable X_1, and let the resulting training data set be denoted by $\mathcal{T}_t^1$. We then use T_t to classify the training patterns in $\mathcal{T}_t^1$, hence obtaining $\hat{f}_{t1}^1, \hat{f}_{t2}^1, \ldots, \hat{f}_{tn}^1$, and the corresponding out-of-bag error estimate $\frac{1}{card(\bar{J}_t)} \sum_{i \in \bar{J}_t} \left(f_i - \hat{f}_{ti}^1\right)^2$ when the entries in variable X_1 was permuted. This process of obtaining out-of-bag error estimates is repeated for variables X_2 up to X_p. The ratio

$$r_j = \frac{1}{card(\bar{J}_t)} \sum_{i \in \bar{J}_t} \left(f_i - \hat{f}_{ti}^j\right)^2 \Bigg/ \frac{1}{n(card(\bar{T}_t))} \sum_{i=1}^{n} \sum_{t \in \bar{T}_i} \left(\hat{f}_{ti} - \breve{f}_i\right)^2 \tag{2.34}$$

is used to quantify the importance of variable X_j. Relatively large r-values indicate the more important input variables.

2.4. *VARIABLE SELECTION FOR KERNEL METHODS*

In contrast to proposals regarding feature selection, contributions with respect to input variable selection for kernel methods have been relatively few, and since feature selection and input variable selection are only equivalent when linear kernels are used, little of the feature selection literature is relevant in our context. (See Chapter 1 for the distinction between input variable and feature selection). Ideas in the literature on feature selection for kernel techniques have varied much. A distinction often made in the machine learning literature on feature selection, is between so-called *filter*, *wrapper* and *embedded* methods for selection (*cf*. Kohavi and John, 1997; and Guyon and Elisseef, 2003). We will say more on filter and wrapper selection criteria in Chapter 3. Papers on variable filtering include Liu *et al*. (2002), Stoppiglia and Dreyfus (2003), and Niijima and Kuhara (2006). A category of selection procedures involves generalisation of adaptive tuning of multiple kernel hyperparameter values. Adaptive shrinkage methods are examples of embedded methods. Proposals regarding shrinkage for selection in SVMs are found in Grandvalet and Canu (2002), Weston *et al*. (2001 and 2003), and Chapelle *et al*. (2004). An input variable selection approach which may also be regarded as an embedded method is the more recently proposed procedure by Keerthi (2005). Closely connected to the regularised optimisation-based selection procedures in the previous section, application of this procedure is however restricted to SVMs using only linear kernel functions.

2.4.1 *THE NEED FOR VARIABLE SELECTION*

In this section we illustrate the advantageous effect of correctly eliminating irrelevant variables on the performance of both classical and kernel classification procedures. For this purpose we assimilate results from simulation experiments which will be more thoroughly discussed in Section 3.4.1. In this section we therefore only briefly discuss aspects of the simulation study needed to interpret the reported results.

One of several factors considered in the simulation study was the distribution used to generate input variable values in the simulation data sets. We investigated normal and

lognormal distributions. Each simulation run started by generating a data set (in a binary classification setup) where only a portion of the initially available variables were relevant (in a separatory sense). Let the subset of separating variables be denoted by V_S, of variables not separating the two groups by $V_{\bar{S}}$, and let $\mathcal{V}$ refer to the full set of available input variables. Depending on the training sample size, there were either ten or sixty input variables in total. Each of the variables in V_S contributed equally to the separation between groups, and we considered a correlation of either $\rho = 0$ or $\rho = 0.7$ between pairs of variables in V_S. Pairs of variables from $V_{\bar{S}}$ and V_S were either uncorrelated, or equicorrelated, with – in the case of normal data – a correlation of $\rho_{S\bar{S}} = 0.9$ between them. In lognormal configurations, we were compelled to set $\rho_{S\bar{S}} = 0.2$ or 0.25 (a motivation in this regard will be given in Appendix A.1). Furthermore, we generated data groups to be separated either in terms of location, or with respect to their variance-covariance structure. Thus in terms of the data distribution and type of group separation, we considered four configurations: normal data and group separation with respect to either location (denoted *NL* cases), or spread (denoted *NS* cases); and lognormal data with group separation in terms of location (*LL* cases), or spread (*LS* cases). We also varied the training sample sizes. Small and wide samples consisted of 15 cases in each group; mixed samples had 25 cases in the first group, and 75 cases in the second; and large samples consisted of 100 cases in each group.

We divided the data into a training and test set, and applied LDA and KFDA and fitted an SVM on the training set using measurements on

i. the total number of available variables (in $\mathcal{V}$)

ii. only the fraction of relevant variables (in V_S)

Note here that the percentage of relevant variables referred to in *ii*. where either $\pi = 0.1$ or $\pi = 0.4$. The linear and kernel Fisher discriminant functions, as well as the support vector classifier obtained in *i*. and *ii*. were then evaluated on the test sample. For each of the classifiers considered, the classification test error rate obtained in *ii*., relative to the

classification test error rate in *i*. was used to measure the effect of irrelevant variables on classification performance. Denote the test error in *i*. by $Err(\mathcal{V})$ and the test error in *ii*. by $Err(V_S)$. The effect of the irrelevant variables (henceforth called *irrelevance effect*) was quantified in terms of $irr = Err(\mathcal{V})/Err(V_S)$. Hence an irrelevance effect value greater than one indicates the omission of irrelevant variables to yield an improved classification performance. In turn, irrelevance effect values close to 0 indicate the elimination of irrelevant variables to have a negative effect on classification accuracy and would be indicative of situations where seemingly irrelevant variables are actually weakly relevant.

Thus each simulation run yielded an irrelevance effect value for LDA, KFDA and the fitted SVM. An average of the irrelevance effect values for each classifier was calculated over 1000 simulation runs.

The obtained irrelevance values are reported for normal data and $\pi = 0.1$ in Table 2.1; and for normal data and $\pi = 0.4$ in Table 2.2. Similarly, irrelevance effect values for lognormal data and $\pi = 0.1$ are given in Table 2.3, followed by the results for lognormal data and $\pi = 0.4$ in Table 2.4. The type of distribution and group separation, sample sizes and various correlation values considered, are indicated across rows. Configurations that could not be handled by LDA are indicated by a *-symbol.

From the vast majority of configurations in Tables 2.1-2.4, the improvement in the performance of LDA, KFDA and SVMs after omitting the variables in $V_{\bar{S}}$, is evident. (We highlighted irrelevance values greater than 2.) Note especially the considerable improvement in generalisation accuracy realised in wide samples: irrelevance (*irr*) values as high as 29 and 45 were observed in *NS*, $\pi = 0.4$ configurations (Table 2.2). Fewer of the *NL* configurations benefit from the omission of variables in $V_{\bar{S}}$: compare 82% of the *NL* configurations that showed improvements in accuracy with 89% in the *NS* case, and 95% in both the *LL* and *LS* cases.

In agreement with our discussion in Example 2.5, we see that in some of the cases where pairs of relevant and irrelevant variables were highly correlated, the model fitted to the full set of available variables outperformed the one fitted to only the subset of truly relevant variables. (We underlined all irrelevance values between 0 and 1.) In these cases, the variables in $V_{\bar{S}}$ therefore actually turned out to be weakly relevant. Note that the highest incidence (30%) of weakly relevant variables occurred in the *NL*, $\pi = 0.4$ configurations (Table 2.2), and the lowest (only 0.2%) in *LL*, $\pi = 0.1$.

The KFDA *NS* results for mixed samples in Table 2.1 require some explanation: all the irrelevance values equal 1. This would seem to suggest that in these cases excluding the irrelevant variables has no effect on the error rate. Closer inspection of the simulation results, however, shows that these irrelevance values are obtained because we found that for both $Err(\mathcal{V})$ and $Err(V_S)$ the smallest error rate obtainable by varying the cost parameter over the values which we considered, equalled 0.25. A more refined search in terms of specification of the cost parameter would seem to be necessary in these cases. Similar remarks are applicable to the results in Table 2.3 and to several tables appearing in later chapters.

For a comparison of the generally positive effect of eliminating irrelevant variables among classification techniques, first consider the case of normally distributed data (reported in Tables 2.1 and 2.2). Here smaller test errors after elimination of irrelevant variables are typically less prominent in the case of LDA. When the two groups are separated with respect to location and $\pi = 0.1$, the number of configurations in which SVMs benefit more than KFDA from eliminating variables in $V_{\bar{S}}$ is more or less equal to the number of configurations where the converse is true. Hence SVMs and KFDA seem to benefit to the same extent from eliminating variables in $V_{\bar{S}}$. When the groups are however separated with respect to spread (and $\pi = 0.1$), KFDA seems to benefit slightly more: in 13 out of 16 configurations the KFDA *irr* values are larger than the *irr* values obtained for SVMs, albeit by not that large a margin.

Table 2.1 : Ratios of test errors: $\pi = 0.1$, normal training data

			NORMAL LOCATION (NL)			***NORMAL SPREAD (NS)***		
	ρ_S	$\rho_{S\bar{S}}$	***SVM***	***KFDA***	***LDA***	***SVM***	***KFDA***	***LDA***
SMALL	0	0	**2.754**	1.307	1.263	1.562	1.593	1.071
	0.7	0	**2.754**	1.307	1.263	1.575	1.591	1.073
	0	0.9	1.224	1.310	1.131	1.561	1.598	1.073
	0.7	0.9	1.229	1.303	1.128	1.571	1.602	1.071
MIXED	0	0	**2.662**	1.083	1.150	0.997	1.000	1.362
	0.7	0	**2.677**	1.088	1.158	0.996	1.000	1.362
	0	0.9	1.255	1.088	1.029	1.303	1.000	1.358
	0.7	0.9	1.255	1.088	1.029	1.312	1.000	1.362
LARGE	0	0	1.784	1.091	1.062	1.533	1.561	1.029
	0.7	0	1.800	1.091	1.058	1.533	1.565	1.029
	0	0.9	1.048	1.052	0.909	1.502	1.565	1.027
	0.7	0.9	1.047	1.055	0.909	1.498	1.561	1.031
WIDE	0	0	1.862	2.320	*	**3.375**	**4.767**	*
	0.7	0	0.871	1.192	*	**2.263**	**2.119**	*
	0	0.9	0.752	**2.820**	*	1.445	**3.034**	*
	0.7	0.9	0.953	1.531	*	0.805	0.870	*

For both types of group separation and $\pi = 0.4$ (Table 2.2), again KFDA profits more from the omission of variables in $V_{\bar{S}}$ than SVMs (10 out of 16 data setups yield larger KFDA *irr* values compared to the *irr* values obtained for SVMs in both the *NS* and *NL* configurations).

Table 2.2 : Ratios of test errors: $\pi = 0.4$, normal training data

			NORMAL LOCATION (NL)			***NORMAL SPREAD (NS)***		
	ρ_S	$\rho_{S\bar{S}}$	***SVM***	***KFDA***	***LDA***	***SVM***	***KFDA***	***LDA***
SMALL	0	0	1.258	1.311	1.271	1.947	**2.485**	1.129
	0.7	0	1.110	1.108	1.145	1.764	1.740	1.129
	0	0.9	0.445	1.246	0.000	1.932	**2.495**	1.124
	0.7	0.9	1.052	1.101	1.025	1.729	1.709	1.126
MIXED	0	0	1.201	1.231	1.099	**2.250**	**2.477**	1.189
	0.7	0	1.265	1.055	1.088	**2.422**	1.472	1.920
	0	0.9	0.146	1.178	0.000	**2.151**	**2.465**	1.197
	0.7	0.9	1.106	1.014	0.967	**2.358**	1.433	1.191
LARGE	0	0	1.099	1.056	1.043	0.607	**2.056**	1.058
	0.7	0	1.061	1.017	1.038	1.764	1.740	1.129
	0	0.9	0.018	0.494	0.000	1.921	**2.094**	1.056
	0.7	0.9	0.952	0.941	0.897	**2.319**	1.646	1.056
WIDE	0	0	1.316	**3.167**	*	**14.33**	**45.00**	*
	0.7	0	0.997	1.017	*	1.654	1.922	*
	0	0.9	1.684	**8.833**	*	**18.80**	**29.33**	*
	0.7	0.9	0.922	1.108	*	1.041	1.449	*

Also note that we observe an interaction effect with regard to *irr* values, depending on the percentage of irrelevant input variables, and the type of group separation. SVMs, KFDA and LDA generally benefit more after eliminating variables in $V_{\bar{S}}$ when $\pi = 0.4$ instead of 0.1 and when groups are separated with respect to their variance-covariance structure, and less when $\pi = 0.4$ instead of 0.1 and when groups are separated in terms of location. (Compare for example $irr = 2.754$ for an SVM in an *NL*, $\pi = 0.1$ scenario in Table 2.1 with $irr = 1.258$ for an SVM in an *NL*, $\pi = 0.4$ scenario in Table 2.2; and $irr = 1.562$ for an SVM in an *NS*, $\pi = 0.1$ setup in Table 2.1 with $irr = 1.947$ for an SVM in an *NS*, $\pi = 0.4$ configuration in Table 2.2.)

We now turn our attention to results on the set of configurations when the data followed a lognormal distribution (reported in Tables 2.3 and 2.4). First note that we mostly observe greater improvements in classification error after omitting irrelevant variables here than when the data were normally distributed. Secondly, LDA once again benefits far less from the elimination of all irrelevant variables. If once again one compares the positive impact of an accurate reduction of the full set of available variables on the accuracies of KFDA and SVMs, one finds that when the two groups are separated with respect to location and $\pi = 0.1$, KFDA benefit more from the elimination of irrelevant variables than SVMs. (The improvement in accuracy is greater for KFDA compared to SVMs in 15 out of 16 configurations). We however observe the opposite when the groups are separated with respect to their variance-covariance structure. Here we observe greater improvements in the classification accuracy of SVMs in 11 out of 16 configurations.

In the case of group separation with respect to location and $\pi = 0.4$, SVMs benefit much more (4 out of 12 SVM *irr* values are larger than KFDA *irr* values). The same can be said in the *LS* case: here SVMs profit more in 2 out of 12 configurations).

Much of the interaction effect between the percentage of irrelevant input variables and the type of group separation reported in the normal case, is also observed in the lognormal data setup. The *LL* configuration is an exception in this regard.

In summary, the above simulation results indicate that successful variable selection frequently yields significant improvement in the accuracy of both more traditional classifiers (for example LDA) and kernel classifiers (for example SVMs and KFDA). Variable selection was shown to be less important in *NL* configurations, and can perhaps in that sense be regarded as a less important data setup to consider in subsequent simulation investigations regarding variable selection prior to kernel analyses.

Table 2.3 : Ratios of test errors: $\pi = 0.1$, lognormal training data

			LOGNORMAL LOCATION (LL)			***LOGNORMAL SPREAD (LS)***		
	ρ_S	$\rho_{S\bar{S}}$	***SVM***	***KFDA***	***LDA***	***SVM***	***KFDA***	***LDA***
SMALL	0	0	1.246	**2.355**	1.519	**2.095**	1.782	1.080
	0.7	0	1.249	**2.310**	1.519	**2.100**	1.788	1.087
	0	0.2	**2.041**	**2.369**	1.033	1.943	1.794	1.031
	0.7	0.2	**2.041**	**2.347**	1.033	**2.162**	1.797	1.040
MIXED	0	0	1.336	**2.050**	**2.044**	1.491	1.000	1.247
	0.7	0	1.332	**2.071**	**2.044**	1.500	1.000	1.247
	0	0.2	1.862	**2.071**	1.756	1.320	1.000	1.237
	0.7	0.2	1.875	**2.020**	1.778	1.327	1.000	1.232
LARGE	0	0	1.117	1.885	1.129	**2.034**	**2.053**	1.023
	0.7	0	1.117	1.885	1.124	**2.034**	**2.053**	1.021
	0	0.2	1.276	1.549	0.686	1.750	**2.005**	1.002
	0.7	0.2	1.735	1.995	1.004	1.750	1.995	1.004
WIDE	0	0	**4.306**	**5.877**	*	**3.351**	**2.280**	*
	0.7	0	**2.680**	**2.640**	*	**2.584**	**2.067**	*
	0	0.2	1.174	**3.172**	*	1.729	1.960	*
	0.7	0.2	1.374	1.894	*	1.936	1.869	*

In the configurations considered, the need for variable selection was generally much more significant in the case of SVMs and KFDA, than for LDA. It is not easy to say whether variable selection is more important for SVMs than for KFDA. Putting the *NL* data setups aside, it seems as if SVMs benefit more, *viz*. in the *LL*, $\pi = 0.1$, and *LS*, $\pi = 0.1$ and $\pi = 0.4$ cases. Taking the *NL* configurations into consideration, a comparison of the importance of variable selection in SVMs and KFDA is however much less conclusive.

Table 2.4 : Ratios of test errors: $\pi = 0.4$, lognormal training data

			LOGNORMAL LOCATION (LL)			***LOGNORMAL SPREAD (LS)***		
	ρ_S	$\rho_{S\bar{S}}$	***SVM***	***KFDA***	***LDA***	***SVM***	***KFDA***	***LDA***
SMALL	0	0	**2.013**	**2.012**	1.586	**2.133**	1.968	1.164
	0.7	0	1.727	1.659	1.339	1.932	1.785	1.203
	0.7	0.25	1.146	1.359	0.772	1.702	1.918	1.103
MIXED	0	0	1.738	1.459	1.614	**2.420**	1.209	1.125
	0.7	0	1.457	1.275	1.873	2.598	1.064	1.123
	0.7	0.25	0.986	1.099	1.457	1.927	1.093	1.069
LARGE	0	0	2.333	1.662	1.161	2.145	2.537	1.056
	0.7	0	1.805	1.305	1.061	2.419	1.522	1.071
	0.7	0.25	1.111	1.009	0.538	1.968	1.475	1.021
WIDE	0	0	**4.286**	**3.921**	*	**4.544**	**3.683**	*
	0.7	0	1.690	1.775	*	**2.199**	**2.156**	*
	0.7	0.25	1.102	1.244	*	1.727	1.576	*

2.4.2 *COMPLICATING FACTORS AND POSSIBLE APPROACHES*

Variable selection for kernel methods is in many respects a more complex problem than for standard statistical analyses. In this section we point out several of the factors which complicate the selection process when kernel techniques are used, and broadly discuss possible approaches towards overcoming these difficulties.

A first difficulty is that most classical selection approaches are based on one or more assumptions which are typically not made in kernel methods. Here for example we have in mind the assumption of a normal distribution for the input data. In many applications of kernel methods such a strong assumption is invalid. Whereas the assumption of normal data considerably simplifies a decision regarding the value of m in classical approaches, this is not possible in the case of kernel methods. Note that we will investigate a proposal for a data-dependent decision regarding m in Chapter 6 of the thesis.

A second complicating factor comes into play when patterns in the data are non-linear. In such cases non-linear kernels are needed, resulting in the end in kernel solutions which are not (weighted) sums over *input variables* (or even over transformed input variables) as we would like, *i.e.* $f(\boldsymbol{x}) = \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j + b$ or $f(\boldsymbol{x}) = \sum_{j=1}^{p} \beta_j \Phi(\boldsymbol{x}_j) + b$, but sums over *features*, *viz.*

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b = \sum_{j=1}^{N} w_j \phi_j + b, \tag{2.34}$$

In a classifier written in the form $\sum_{j=1}^{p} \beta_j \boldsymbol{x}_j + b$ or $\sum_{j=1}^{p} \beta_j \Phi(\boldsymbol{x}_j) + b$, where $x_1, x_2, \ldots, x_p$ are the components of the new case to be classified, we may directly interpret $|\beta_j|$ as an indicator of the relative importance of the j^{th} variable (provided the data have been appropriately scaled). In Expression (2.26) the $|w_j|$, $j = 1,2,\ldots,N$ however indicate the relative importance of the j^{th} feature (which does not correspond to the j^{th} variable transformed to $\Im$). Hence the $|w_j|$ can be used to guide *feature selection*, but do not provide information regarding the relative importance of the original variables in $\aleph$. Therefore, since a kernel classification function is only of the form $f(\boldsymbol{x}) = \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j + b$ when a linear kernel is used, we see that the use of a non-linear kernel inevitably obscures the contribution of each of the input variables to the kernel solution.

In order to further illustrate the point above, recall that in Section 2.2.3 it was shown that a kernel classification function can be written in the form

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i \boldsymbol{x}) + b \tag{2.35}$$

where the N-vector $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\varphi}_i$ is a linear combination over *data cases* in $\Im$. This stands in contrast to more traditional classifiers, which are usually expressed as (linear) combinations over the *input variables*. Hence the dilemma when attempting variable selection in a kernel analysis again becomes apparent: $|\alpha_i|$ reflects the relative importance of the i^{th} *data case* in the kernel classifier, rendering $\boldsymbol{w}$ (or its associated $f(\boldsymbol{x})$) not directly suitable as a basis for variable selection.

It is possible to handle the above problem by progressively taking the transformation to feature space into account. In our opinion, there are three approaches in this regard. The most naïve approach is to simply ignore the transformation. Here one assumes the influence of individual predictors in input and feature space to be approximately the same, and one performs variable selection by simply using classical selection criteria in input space. A second approach is to perform the calculation of selection criteria in feature space, and then to find a connection between the N coordinates of the criterion in $\Im$, and the p coordinates of the criterion in $\aleph$. Once the selection criterion has been embedded back into $\aleph$, it is of course possible to continue the selection process in input space. In this way part of the selection process is carried out in feature space, and part of it in input space. A third possibility is to calculate selection criteria and carry out the selection process in feature space. This however promises to be a computationally more expensive approach. Note that the first and second approaches (selection in input space) will be investigated in Chapter 3, followed by a discussion of the third approach (selection in feature space) in Chapter 4 to 6.

In the second and third approaches to variable selection for kernel methods, one is compelled to at least at some stage work in feature space. This however confronts us with a third problem: working in feature space is typically difficult. As we have seen before, we are restricted to the calculation of inner products via the kernel trick. Hence any selection criterion which we would like to propose in feature space has to depend on the data only via inner products between features. Although this may seem very restrictive, we will see

in Chapters 4 and 5 that there are still several informative ways of utilising the kernel function to summarise the clustering of the two groups of input data points in feature space.

As pointed out earlier, any kernel method has to use the information provided by the data via entries in the kernel matrix. Therefore the kernel matrix $\boldsymbol{K}$ plays a central role when one attempts selection in $\Im$. We will see in Section 4.6, that several selection criteria may be defined explicitly in terms of the structure of $\boldsymbol{K}$. Since different kernel classification procedures arise because of different algorithms used to construct the linear classifiers, it follows that a selection criterion based only on $\boldsymbol{K}$ will *ipso facto* be applicable to any kernel method. We can however also take the specific algorithm into account when proposing a selection criterion. This then leads to selection criteria only applicable in SVMs, or only in KFDA, as the case may be. These criteria, although not explicitly based on the structure of $\boldsymbol{K}$, also utilise entries in the kernel matrix. We will give further attention to such possibilities in Section 4.7.

The need to specify values for the hyperparameters used in a kernel method further complicates variable selection. We will treat these quantities largely as nuisance factors, *i.e*. we will not undertake a detailed study of the possible interaction between different sets of variables and hyperparameter values.

We have stated in the introduction to this chapter that in Chapters 3 to 5 the true model dimension (true value of m) is assumed to be known. In classical applications this usually leads to a substantial simplification. For example, in least squares multiple regression analysis, if the value of m is fixed, almost all selection criteria (including amongst others AIC, BIC and C_p) will identify the same subset of variables. The reason for this is that all these criteria measure accuracy in the same way, *viz*. in terms of the sum of squared residuals, and differ only with respect to the manner in which accuracy is traded off against complexity. In kernel methods, assuming the value of m known does not nearly lead to the same degree of simplification. To explain this subtle point, suppose we have a fixed value of m and taking our cue from multiple regression analysis, we try to select variables by minimising $err(f)$. This will almost certainly lead to overfitting, since even with a

fixed value for m, the feature transformation which characterises kernel methods implies that we will be optimising over a very large class of discriminant functions. Consequently, even the apparently simple case of a known value of m poses formidable challenges regarding specification of an appropriate selection criterion.

Although the different strategies discussed earlier in this chapter can also be applied in variable selection for kernel methods, our task is often complicated by the size of the data sets under investigation. Applying a one-variable-at-a-time forward selection or backward elimination approach to a data set containing thousands of variables is infeasible. As will be seen in Chapter 5, the traditional strategies can be suitably modified to make them practicable.

2.5 *SUMMARY*

In this chapter we provided an introduction to variable selection for kernel methods, with application to binary classification problems. This was done in two parts. We started by discussing the various components required to construct a kernel classifier, indicating that kernel procedures are regularised techniques. We then introduced the variable selection problem, and briefly discussed some of the more recent selection approaches based on regularised estimation. Since kernel procedures implement regularised estimation, selection procedures that are generalised estimation algorithms can be extended to selection in (special cases of) kernel techniques as well. Application of such an approach, as devised by Keerthi (2005), is unfortunately limited to a subset of kernel procedures. There are also several properties of kernel techniques which further complicate input variable selection. We concluded the chapter with a discussion of such characteristics, and provided some pointers regarding possible approaches to be followed in the remaining thesis chapters.

CHAPTER 3

KERNEL VARIABLE SELECTION IN INPUT SPACE

3.1 *INTRODUCTION*

The focus in this and the following two chapters is primarily on defining criteria for variable selection in kernel classifiers. Recall that in Chapter 2 mention was made of different classes of selection procedures depending on the nature of the selection criterion, *viz*. a filter- or wrapper selection criterion.

Filter criteria may essentially be viewed as pre-processors, independent of the choice of predictor (*cf*. Kira and Rendell, 1992, and Guyon and Elisseeff, 2003). A simple example of a filter criterion is the correlation coefficient between the response and each variable. There are in fact many proposals toward the derivation of selection criteria based on correlation coefficients. (See for example Hall, 1998). Filter criteria may be calculated to rank the relevance of individual input variables or subsets of variables in predicting the response. Depending on whether the rankings refer to a single variable or a particular variable subset, the corresponding filter criterion is referred to as either univariate or multivariate.

Wrapper criteria are determined by choices regarding the predictor to be used. In kernel methods wrapper criteria may be calculated in one of three ways. The first possibility only requires a decision to be made regarding the kernel function and its hyperparameter values; the wrapper criterion in kernel selection is therefore based on the kernel matrix only. We view such criteria as *semi-wrappers*. Therefore semi-wrapper selection criteria are derived with any kernel method in mind. The second type of wrapper criterion additionally depends on the values of the estimated parameters appearing in the predictor. In the

context of kernel techniques, such selection criteria may only be applied in context of the specific class of kernel methods for which they are intended and they additionally require the specific kernel method to be trained. A third alternative is to use as selection criterion the estimated generalisation error when the appropriate predictor is applied. One of the first such wrapper feature selection proposals was by Kohavi and John (1997), *viz*. to simply use the cross-validation errors calculated for each variable subset to obtain a ranking of variable subsets. Finally, note that both semi-wrapper and wrapper selection criteria may be univariate or multivariate as is the case with filter criteria.

To illuminate the different types of wrapper criteria, consider any kernel predictor. Suppose we decide to use a Gaussian kernel function with a specified value for the kernel hyperparameter γ. A semi-wrapper criterion will now only use the information contained in the kernel matrix, *i.e*. it depends only on the similarities between the data points as measured in feature space via the kernel function. Note that the variables selected using such a criterion will therefore remain the same, irrespective of the specific type of kernel predictor to be used. Application of the second type of wrapper criterion entails extraction of additional information from the data. We now for example have to decide that a support vector classifier will be fitted to the data and at each stage of the selection process the selection criterion may depend on entries in the kernel matrix, the coefficients $\alpha_1,\alpha_2,\ldots,\alpha_n$ and the intercept b. If instead of a support vector classifier we decide on using KFDA, the selection criterion may once again depend on the quantities listed above and the resulting set of selected variables need obviously not be the same. Note in this regard also that in the case of an SVM some of the coefficients $\alpha_1,\alpha_2,\ldots,\alpha_n$ will typically be zero and may therefore be described as not having an effect on the selection. Generally, when in the literature reference is made to wrapper criteria the above dependence on the type of predictor used as well as on the parameter estimates is implied. In order to speedup the selection process, parameter values are often not re-estimated for different variable subsets. Although parameter estimates may thus be kept fixed irrespective of the variable subset evaluated, the above selection techniques are still considered wrappers as opposed to semi-wrappers since the post-selection feature subset is potentially different depending on the type of predictor used.

In summary, semi-wrapper selection criteria are more cost effective than wrapper criteria and take the new higher dimensional configuration of data points into account in performing feature selection for kernel methods. Wrapper selection criteria rely more heavily on the specific class of kernel method in deciding on the relevant variables and are mostly also more costly. Filter criteria base selection solely on attributes of the data and are usually very fast to compute. In the remainder of the thesis, we will also refer to filter- and semi-wrapper criteria as algorithm-independent criteria, while wrapper criteria will be called algorithm-dependent criteria. Algorithm-independent and algorithm-dependent variable selection for kernel classifiers will be discussed in Chapter 4.

This chapter may be regarded as an introduction to Chapter 4 in the sense that it first considers a more basic question before attempting to define selection criteria for kernel methods, *viz*. in which space, input- or feature space, should a kernel variable selection criterion be defined? Whereas filter criteria operate in input space, and semi-wrapper and wrapper criteria are defined in feature space, we consider a new approach to variable selection for kernel methods: define the selection criterion in $\mathfrak{I}$, and then complete the variable selection process in $\aleph$.

In the following section we illustrate that selection in $\aleph$ may be sufficient in some data scenarios, but inadequate in others. In Section 3.3 we then propose the new variable selection approach, which is evaluated in the Monte Carlo simulation study described in Section 3.4. A summary of the chapter is given in Section 3.5.

3.2 *NAÏVE SELECTION IN INPUT SPACE*

In the final section of Chapter 2 we introduced three approaches towards variable selection for kernel methods. One of these approaches is based on the assumption that the influence of individual predictors in input and feature space remains approximately the same, and involves selection via the use of classical selection criteria in input space. The above assumption is of course valid in the case of a linear kernel function, but much less likely to

hold in the case of a non-linear relation between input- and feature space. Therefore in some cases selection in input space might turn out to be sufficient, but in general the influence of input variables may change considerably from input to feature space, causing selection in feature space to yield better results than classical selection. The following example illustrates exactly this: in an *NL* scenario in which LDA can be expected to perform well, classical selection outperforms selection in feature space. However in *NS*, *LL*, and *LS* data configurations which are less ideal setups for LDA, the results of selection in feature space are superior.

EXAMPLE 3.1 *Naïve selection in input and feature space*: *correlations and alignments*

The following excerpt from the results of a simulation experiment are now presented in order to compare the post-selection properties of (binary) classifiers after selection in input space, with those of classifiers following selection in feature space. Both selection approaches were quite naïve: we used ordinary Pearson correlation coefficients to perform selection in input space, and analogous to this, for selection in feature space we used *alignments* (see Section 3.6 in Chapter 4 for a definition and more details). The full set of configurations considered is given in Section 3.4. We only report on the subset of these configurations which was described in Section 2.4.1. The perspective from which we look at the simulation described there now changes: instead of reporting the effect of correctly eliminating irrelevant input variables, we now want to compare the effect of eliminating input variables using naïve selection procedures in input space, with the effect when naïve selection is carried out in feature space. Hence once again data sets were generated so that differences in location or spread were contributed only by a smaller (size m) set of relevant input variables (in V_S).

Each simulation repetition started with newly generated training and test data sets, both consisting of measurements on all variables contained in $\mathcal{V}$. To perform selection in input space, we calculated absolute values of the correlation coefficients between the response and each of the variables contained in $\mathcal{V}$, and assuming m as specified in the simulation setup to be known, selected the m most highly correlated variables. Let the selected subset

of variables be denoted by $V_{\aleph}$. To perform selection in feature space we calculated absolute values of the alignment between the response and each of the variables in $\mathcal{V}$, and once again selected the m variables corresponding to the largest alignments. The resulting subset of variables will be denoted by $V_{\beth}$. In this section we only report the results obtained for the case $m/p = 0.4$.

After selection, we fitted an SVM and performed LDA and KFDA on the training data using the variables contained in $\mathcal{V}$, V_S, $V_{\aleph}$ and $V_{\beth}$ respectively. Hence in total there were twelve combinations of procedures and variable (sub)-sets to evaluate at each data configuration. For each of the twelve combinations we recorded an average test error and percentage of misclassifications obtained on the test data. We will refer to the average test errors based on $\mathcal{V}$, V_S, $V_{\aleph}$ and $V_{\beth}$ as the average no selection-, oracle- and input- and feature space selection errors respectively. Tables 3.1 *a* and *b*, 3.2 *a* and *b*, and 3.3 to 3.5 contain these test errors for LDA, KFDA and SVMs, in scenarios *NL*, *NS*, *LL*, and *LS*. Note however that in the large p, small n setups, a linear discriminant function cannot be obtained: the covariance matrix of the $\boldsymbol{x}_{ij}$ observations, $i = 1,2,\ldots,n;\ j = 1,2,\ldots,p;\ p > n$, is singular, rendering the use of LDA inappropriate in such cases. This is indicated by *-symbols in the last four rows of the LDA columns in each table.

In Tables 3.1 *a* and 3.2 *a* we report on normal data where irrelevant and relevant input variables were generated independently ($\rho_{S\bar{S}} = 0$), whereas Tables 3.1 *b* and 3.2 *b* summarise the results obtained when relevant and irrelevant input variables were highly correlated. In each table the different configurations in terms of the correlation between pairs of variables in V_S are indicated across columns. The various types of errors are presented row-wise – in groups of four – with the first, second and third group of rows referring to small, mixed, large ($n > p$) and wide ($p > n$) sample sizes. We used 1000 simulation repetitions throughout. Standard errors of the average test errors are given in brackets.

Importantly, note that the SVM, KFDA and LDA test errors were calculated for a series of values for the SVM and KFDA cost parameter. In all cases we trained an SVM and performed KFDA using an RBF kernel function, and kept the kernel hyperparameter $\gamma = 1/p$ throughout. A motivation for this particular choice will be given in Section 3.4.4.

For each kernel technique we report only the smallest test error achieved across cost parameter values. Therefore it cannot be argued that the performance of any of the techniques was at an unfair disadvantage due to misspecification of the cost parameter value. This allows one to compare the performances of the various techniques. In this regard, the following can be noted. As expected, LDA performs relatively well in *NL* (in Tables 3.1 *a* and *b*) and in *LL* (in Tables 3.3 and 3.5) data scenarios. Compare for example LDA test error rates for mixed samples and $\rho_S = 0$ in the *NL* configuration in Table 3.1 *a*, *viz*. .156, .142, .143, and .155, with SVM test errors observed in the same setup, *viz*. .173, .144, .145 and .157. Also, when $\rho_S = 0$ in the *NL* configuration reported in Table 3.1 *b*, LDA based on the full set of available input variables correctly classifies all test sample cases. Although generally in *LL* scenarios LDA performs worse than SVMs, we observe smaller LDA test errors than KFDA in most configurations. For example, LDA test error rates for mixed samples and $\rho_S = 0$ in the *LL* configuration in Table 3.3 were .071, .044, .046 and .044. The corresponding KFDA test errors were .089, .061, .063 and .061. However, as soon as the data are not separated with respect to location, but with respect to spread, the classification performance of LDA is no longer adequate: for example, the minimum LDA test error observed for all the large *NS* data sets is .466 (see Tables 3.2 *a* and *b*), whereas for large *LS* data sets it is .437 (see Tables 3.4 and 3.5).

Table 3.1 *a*: Average test errors for the *NL* case ($\pi = 0.4$, $\rho_{S\bar{S}} = 0$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.249 (.001)	.352 (.001)	.240 (.001)	.329 (.001)	.239 (.001)	.371 (.001)
$Err(V_S)$	.198 (.001)	.317 (.001)	.183 (.001)	.297 (.001)	.188 (.001)	.324 (.001)
$Err(V_\aleph)$	.219(.001)	.324 (.001)	.207 (.001)	.304 (.001)	.210 (.001)	.329 (.001)
$Err(V_\beth)$	.236 (.002)	.330 (.001)	.224 (.001)	.311 (.001)	.222 (.001)	.329 (.001)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.173 (.001)	.286 (.002)	.165 (.001)	.232 (.000)	.156 (.000)	.260 (.001)
$Err(V_S)$	.144 (.000)	.226 (.001)	.134 (.000)	.220 (.000)	.142 (.000)	.239 (.001)
$Err(V_\aleph)$	.145 (.001)	.226 (.001)	.136 (.000)	.220 (.000)	.143 (.000)	.240 (.000)
$Err(V_\beth)$	.157 (.001)	.229 (.001)	.148 (.001)	.221 (.000)	.155 (.001)	.240 (.000)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.188 (.000)	.312 (.000)	.171 (.000)	.291 (.000)	.170 (.000)	.302 (.000)
$Err(V_S)$	.171 (.000)	.294 (.000)	.162 (.000)	.286 (.000)	.163 (.000)	.291 (.000)
$Err(V_\aleph)$	.171 (.000)	.294 (.000)	.162 (.000)	.286 (.000)	.163 (.000)	.291 (.000)
$Err(V_\beth)$	.171 (.000)	.294 (.000)	.162 (.000)	.286 (.000)	.163 (.000)	.291 (.000)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.025 (.000)	.308 (.001)	.038 (.001)	.292 (.001)	*	*
$Err(V_S)$	.019 (.000)	.309 (.002)	.012 (.000)	.287 (.001)	*	*
$Err(V_\aleph)$	.032 (.001)	.311 (.001)	.023 (.000)	.286 (.001)	*	*
$Err(V_\beth)$	.040 (.001)	.313 (.002)	.030 (.000)	.286 (.001)	*	*

Table 3.1 *b*: Average test errors for the *NL* case continued ($\pi = 0.4$, $\rho_{S\bar{S}} = 0.9$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.089 (.001)	.333 (.001)	.228 (.001)	.326 (.001)	.000 (.000)	.331 (.001)
$Err(V_S)$	.200 (.001)	.316 (.001)	.183 (.001)	.296 (.001)	.187 (.001)	.323 (.001)
$Err(V_\aleph)$	.216 (.001)	.322 (.001)	.201 (.001)	.302 (.001)	.200 (.001)	.325 (.001)
$Err(V_\beth)$	.225 (.001)	.325 (.001)	.218 (.001)	.310 (.001)	.207 (.001)	.324 (.001)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.021 (.000)	.253 (.001)	.159 (.000)	.223 (.000)	.000 (.000)	.231 (.001)
$Err(V_S)$	.144 (.000)	.226 (.001)	.135 (.000)	.220 (.000)	.143 (.000)	.239 (.000)
$Err(V_\aleph)$	.145 (.000)	.226 (.001)	.136 (.000)	.220 (.000)	.143 (.000)	.240 (.000)
$Err(V_\beth)$	.153 (.001)	.227 (.001)	.146 (.001)	.220 (.000)	.150 (.000)	.239 (.000)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.003 (.000)	.279 (.001)	.080 (.001)	.269 (.000)	.000 (.000)	.262 (.000)
$Err(V_S)$	.171 (.000)	.295 (.000)	.162 (.000)	.286 (.000)	.163 (.000)	.292 (.000)
$Err(V_\aleph)$	.171 (.000)	.295 (.000)	.162 (.000)	.286 (.000)	.163 (.000)	.292 (.000)
$Err(V_\beth)$	.171 (.000)	.295 (.000)	.162 (.000)	.286 (.000)	.163 (.000)	.292 (.000)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.032 (.001)	.285 (.001)	.106 (.003)	.318 (.001)	*	*
$Err(V_S)$	.019 (.000)	.309 (.001)	.012 (.000)	.287 (.001)	*	*
$Err(V_\aleph)$	.018 (.001)	.308 (.001)	.021 (.002)	.290 (.001)	*	*
$Err(V_\beth)$	.022 (.001)	.310 (.001)	.032 (.002)	.298 (.001)	*	*

Table 3.2 *a*: Average test errors for the *NS* case ($\pi = 0.4$, $\rho_{S\bar{S}} = 0$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.259 (.001)	.321 (.001)	.256 (.001)	.308 (.001)	.482 (.001)	.483 (.001)
$Err(V_S)$	.119 (.001)	.182 (.001)	.103 (.001)	.177 (.001)	.427 (.001)	.428 (.001)
$Err(V_N)$	.348 (.003)	.363 (.003)	.341 (.003)	.367 (.003)	.479 (.001)	.486 (.001)
$Err(V_3)$	.177 (.002)	.223 (.002)	.160 (.002)	.222 (.002)	.449 (.001)	.447 (.001)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.207 (.001)	.264 (.001)	.213 (.000)	.237 (.000)	.359 (.001)	.360 (.001)
$Err(V_S)$	.092 (.000)	.109 (.001)	.086 (.000)	.161 (.001)	.302 (.002)	.302 (.002)
$Err(V_N)$	.307 (.002)	.320 (.002)	.250 (.000)	.250 (.000)	.337 (.001)	.329 (.001)
$Err(V_3)$	.367 (.002)	.366 (.002)	.250 (.000)	.250 (.000)	.299 (.001)	.302 (.001)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.156 (.001)	.321 (.001)	.220 (.000)	.308 (.001)	.493 (.000)	.483 (.001)
$Err(V_S)$	.257 (.000)	.182 .001)	.107 (.000)	.177 (.001)	.466 (.001)	.428 (.001)
$Err(V_N)$	.475 (.002)	.363 (.003)	.305 (.003)	.367 (.003)	.492 (.000)	.486 (.001)
$Err(V_3)$	.257 (.000)	.223 (.002)	.107 (.000)	.222 (.002)	.466 (.001)	.447 (.001)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.086 (.001)	.253 (.001)	.135 (.001)	.223 (.001)	*	*
$Err(V_S)$	.006 (.000)	.153 (.001)	.003 (.000)	.116 (.002)	*	*
$Err(V_N)$	.298 (.002)	.356 (.002)	.166 (.003)	.273 (.003)	*	*
$Err(V_3)$	.046 (.001)	.201 (.001)	.014 (.000)	.145 (.002)	*	*

Table 3.2 *b*: Average test errors for the *NS* case continued ($\pi = 0.4$, $\rho_{S\bar{S}} = 0.9$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.255 (.001)	.315 (.001)	.257 (.001)	.306 (.001)	.480 (.001)	.483 (.001)
$Err(V_S)$	.118 (.001)	.180 (.001)	.103 (.001)	.179 (.001)	.427 (.001)	.429 (.001).
$Err(V_{\aleph})$	.338 (.003)	.360 (.003)	.344 (.003)	.368 (.003)	.479 (.001)	.485 (.001)
$Err(V_{\beth})$	.174 (.002)	.218 (.002)	.158 (.002)	.222 (.002)	.448 (.001)	.450 (.001)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.200 (.001)	.256 (.001)	.212 (.000)	.235 (.000)	.359 (.001)	.361 (.001)
$Err(V_S)$	.092 (.000)	.110 (.001)	.086 (.000)	.164 (.001)	.300 (.002)	.303 (.002)
$Err(V_{\aleph})$	.305 (.002)	.317 (.003)	.250 (.000)	.250 (.000)	.335 (.001)	.329 (.001)
$Err(V_{\beth})$	.365 (.002)	.364 (.002)	.250 (.000)	.250 (.000)	.303 (.001)	.302 (.001)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.145 (.001)	.210 (.001)	.222 (.000)	.270 (.000)	.492 (.000)	.492 (.000)
$Err(V_S)$	.076 (.000)	.088 (.000)	.106 (.000)	.164 (.000)	.466 (.001)	.466 (.001)
$Err(V_{\aleph})$	.244 (.003)	.258 (.003)	.298 (.003)	.327 (.003)	.492 (.000)	.493 (.000)
$Err(V_{\beth})$	.076 (.000)	.088 (.000)	.106 (.000)	.164 (.000)	.466 (.001)	.466 (.001)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.094 (.001)	.151 (.001)	.088 (.003)	.171 (.002)	*	*
$Err(V_S)$	.005 (.000)	.145 (.001)	.003 (.000)	.118 (.002)	*	*
$Err(V_{\aleph})$	.185 (.003)	.260 (.002)	.193 (.005)	.282 (.003)	*	*
$Err(V_{\beth})$	.169 (.002)	.191 (.001)	.183 (.003)	.173 (.002)	*	*

Table 3.3: Average test errors for the *LL* case ($\pi = 0.4$, $\rho_{S\bar{S}} = 0$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.157 (.001)	.221 (.002)	.173 (.001)	.219 (.002)	.157 (.001)	.253 (.002)
$Err(V_S)$	.078 (.001)	.128 (.001)	.086 (.000)	.132 (.001)	.099 (.001)	.189 (.002)
$Err(V_\aleph)$	.095 (.001)	.147 (.002)	.104 (.001)	.149 (.002)	.109 (.001)	.203 (.003)
$Err(V_\beth)$	.080 (.001)	.131 (.001)	.087 (.001)	.135 (.001)	.100 (.001)	.192 (.002)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.073 (.001)	.102 (.001)	.089 (.000)	.102 (.001)	.071 (.001)	.133 (.001)
$Err(V_S)$	.042 (.000)	.070 (.000)	.061 (.000)	.080 (.000)	.044 (.000)	.071 (.001)
$Err(V_\aleph)$	.044 (.000)	.071 (.000)	.063 (.000)	.082 (.001)	.046 (.000)	.076 (.001)
$Err(V_\beth)$	.043 (.000)	.070 (.000)	.061 (.000)	.080 (.001)	.044 (.000)	.071 (.001)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.077 (.000)	.148 (.001)	.108 (.000)	.154 (.001)	.101 (.000)	.210 (.001)
$Err(V_S)$	.033 (.000)	.082 (.000)	.065 (.000)	.118 (.000)	.087 (.000)	.198 (.001)
$Err(V_\aleph)$	.033 (.000)	.082 (.000)	.065 (.000)	.118 (.000)	.087 (.000)	.198 (.001)
$Err(V_\beth)$	.033 (.000)	.082 (.000)	.065 (.000)	.118 (.000)	.087 (.000)	.198 (.001)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.090 (.001)	.218 (.001)	.149 (.003)	.229 (.002)	*	*
$Err(V_S)$	.021 (.000)	.129 (.001)	.038 (.001)	.129 (.001)	*	*
$Err(V_\aleph)$	.032 (.001)	.149 (.001)	.046 (.001)	.146 (.002)	*	*
$Err(V_\beth)$	.023 (.000)	.134 (.001)	.042 (.001)	.132 (.001)	*	*

Table 3.4: Average test errors for the *LS* case ($\pi = 0.4$, $\rho_{S\bar{S}} = 0$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.273 (.002)	.313 (.002)	.307 (.002)	.332 (.002)	.439 (.002)	.433 (.002)
$Err(V_S)$	.128 (.001)	.162 (.001)	.156 (.002)	.186 (.002)	.377 (.002)	.360 (.003)
$Err(V_{\aleph})$	.296 (.003)	.316 (.004)	.330 (.003)	.365 (.004)	.427 (.002)	.446 (.002)
$Err(V_{\beth})$	.157 (.002)	.189 (.002)	.182 (.002)	.220 (.003)	.398 (.002)	.388 (.003)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.196 (.001)	.252 (.001)	.237 (.001)	.248 (.000)	.342 (.001)	.339 (.001)
$Err(V_S)$	.081 (.001)	.097 (.001)	.196 (.002)	.233 (.001)	.304 (.001)	.302 (.001)
$Err(V_{\aleph})$	.276 (.003)	.290 (.003)	.250 (.000)	.250 (.000)	.319 (.001)	.306 (.001)
$Err(V_{\beth})$	.290 (.002)	.309 (.003)	.250 (.000)	.250 (.000)	.312 (.001)	.300 (.001)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.163 (.001)	.225 (.001)	.340 (.001)	.271 (.000)	.473 (.001)	.468 (.001)
$Err(V_S)$	.076 (.000)	.093 (.000)	.134 (.001)	.178 (.000)	.448 (.001)	.437 (.001)
$Err(V_{\aleph})$	.203 (.003)	.238 (.004)	.293 (.003)	.318 (.003)	.470 (.001)	.477 (.002)
$Err(V_{\beth})$	.072 (.000)	.093 (.000)	.135 (.001)	.179 (.001)	.449 (.001)	.438 (.001)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$	$\rho_S = 0$	$\rho_S = 0.7$
$Err(\mathcal{V})$	.259 (.001)	.332 (.002)	.302 (.001)	.345 (.002)	*	*
$Err(V_S)$	.057 (.000)	.151 (.002)	.082 (.001)	.160 (.002)	*	*
$Err(V_{\aleph})$	.270 (.002)	.352 (.003)	.269 (.003)	.344 (.005)	*	*
$Err(V_{\beth})$	.102 (.001)	.213 (.002)	.132 (.001)	.200 (.003)	*	*

Table 3.5: Average test errors for the *LL* and *LS* cases ($\pi = 0.4$, $\rho_S = 0.7$, $\rho_{S\bar{S}} = 0.25$)

	SMALL SAMPLES					
	SVM		**KFDA**		**LDA**	
$\rho_S = 0.7$	***LL***	***LS***	***LL***	***LS***	***LL***	***LS***
$Err(\mathcal{V})$	.148 (.001)	.278 (.002)	.178 (.002)	.322 (.003)	.142 (.002)	.396 (.003)
$Err(V_S)$	.129 (.001)	.161 (.002)	.131 (.001)	.184 (.002)	.184 (.003)	.359 (.004)
$Err(V_\aleph)$	.139 (.002)	.299 (.005)	.143 (.002)	.358 (.005)	.186 (.003)	.444 (.003)
$Err(V_\beth)$	.130 (.001)	.187 (.003)	.133 (.001)	.218 (.004)	.185 (.003)	.386 (.004)
	MIXED SAMPLES					
	SVM		**KFDA**		**LDA**	
$\rho_S = 0.7$	***LL***	***LS***	***LL***	***LS***	***LL***	***LS***
$Err(\mathcal{V})$	.069 (.000)	.212 (.001)	.089 (.001)	.249 (.001)	.102 (.001)	.328 (.001)
$Err(V_S)$	.069 (.000)	.096 (.001)	.081 (.001)	.233 (.001)	.070 (.001)	.300 (.002)
$Err(V_\aleph)$	.070 (.001)	.267 (.004)	.082 (.001)	.250 (.000)	.074 (.001)	.303 (.001)
$Err(V_\beth)$	.069 (.001)	.304 (.004)	.081 (.001)	.250 (.000)	.071 (.001)	.296 (.001)
	LARGE SAMPLES					
	SVM		**KFDA**		**LDA**	
$\rho_S = 0.7$	***LL***	***LS***	***LL***	***LS***	***LL***	***LS***
$Err(\mathcal{V})$	.089 (.000)	.182 (.001)	.118 (.001)	.264 (.002)	.107 (.001)	.446 (.003)
$Err(V_S)$	.082 (.001)	.092 (.001)	.117 (.000)	.179 (.001)	.199 (.002)	.437 (.002)
$Err(V_\aleph)$	.083 (.001)	.220 (.005)	.117 (.000)	.308 (.005)	.199 (.002)	.473 (.003)
$Err(V_\beth)$	.082 (.001)	.092 (.001)	.117 (.000)	.180 (.001)	.199 (.002)	.437 (.002)
	WIDE SAMPLES					
	SVM		**KFDA**		**LDA**	
$\rho_S = 0.7$	***LL***	***LS***	***LL***	***LS***	***LL***	***LS***
$Err(\mathcal{V})$	.141 (.001)	.278 (.002)	.158 (.001)	.249 (.003)	*	*
$Err(V_S)$	.128 (.001)	.161 (.001)	.127 (.001)	.158 (.002)	*	*
$Err(V_\aleph)$	.134 (.001)	.302 (.005)	.134 (.002)	.300 (.006)	*	*
$Err(V_\beth)$	.129 (.001)	.187 (.003)	.127 (.001)	.174 (.003)	*	*

Interestingly, KFDA outperforms SVMs in the case of normal data (in the *NL* and *NS* scenarios), while SVMs yield lower error rates than KFDA when the data are lognormally distributed (in the *LL* and *LS* scenarios). The above observations are summarised in Table 3.6 below. First (or tied first) positions are indicated by 1, second (or tied second) positions are indicated by 2, and worst performances are indicated by 3.

Table 3.6: Relative performances of LDA, KFDA and SVMs

	NL	***NS***	***LL***	***LS***
LDA	1	3	2	3
KFDA	1	1	2	2
SVMs	2	2	1	1

The set of results reported in Table 3.1 *b* were obtained for *NL* data when the variables in $\mathcal{V}$ and V_S were highly correlated ($\rho_{S\bar{S}} = 0.9$ in all cases). As was also seen in Chapter 2, Section 3.4.1, Table 3.1 *b* clearly reflects that whenever pairs of relevant (those in V_S) and (seemingly) irrelevant variables (those in $V_{\bar{S}}$) are highly correlated, classification performance may improve with the inclusion of seemingly irrelevant (hence called weakly relevant) input variables: use of the full set of available input variables is preferred to use of the reduced set of truly relevant variables in 10 out of the 24 data configurations. Table 3.2 *b* contains the simulation test errors obtained when $\rho_{S\bar{S}} = 0.9$ for *NS* data. Since for lognormally distributed data we were compelled to only investigate scenarios in which $\rho_S = 0.7$ (and not $\rho_S = 0$) whenever we wanted to set $\rho_{S\bar{S}} > 0$ (and then we had to use $\rho_{S\bar{S}} = 0.25$ when $\pi = 0.4$), the compound set of results obtained for *LL* and *LS* data scenarios and the correlated case is given in Table 3.5 (A motivation for these particular configurations will be given in Appendix A.1). In both Tables 3.2 *b* and 3.5 we see that correct reduction of $\mathcal{V}$ to V_S proved beneficial in all configurations under consideration –

once again bearing evidence to the importance of correct elimination of irrelevant input variables.

We now compare the performance of naïve selection in input space via correlations, with that of naïve selection in feature space using alignments. Consider first the *NL* case where $\rho_{S\bar{S}} = 0$ (in Table 3.1 *a*). Here both selection approaches succeed in improving the average no selection misclassification rate, with selection in input space being generally more efficient. Selection in input and feature space does however perform more or less the same in tall and wide training samples. Consider next the *NL* case where $\rho_{S\bar{S}} = 0.9$ (in Table 3.1 *b*). We have already seen that, due to the high correlation between relevant and irrelevant input variables, in this scenario classifiers based on the correct set of input variables fare worse than the no selection classifier. Hence in this particular configuration, in terms of a possible improvement in classification accuracy, variable selection does not seem to be a good idea.

Next consider the results reported for *NS* data (in Table 3.2 *a*). In all cases, selection in input space yielded worse results than in the case of no selection. Selection in feature space consistently outperformed selection in input space, yielding in the case of large samples error rates equal to the error rates obtained when the separating variables are known. This is also in line with the results pertaining to $\rho_{S\bar{S}} = 0.9$ configurations reported in Table 3.2 *b*.

In the *LL* data scenario (in Table 3.3 and part of Table 3.5), the post-selection errors based on calculating alignments in $\Im$ are typically smaller than those based on correlations in $\aleph$. Compare for example the SVM, KFDA and LDA test errors obtained for small and wide training sample sizes. A similar pattern is even more evident in the *LS* data configurations (in Table 3.4 and part of Table 3.5).

Finally in this section, we comment on the percentage of times that each variable was selected. Since similar selection percentages were obtained for the various training sample sizes considered, we restrict attention to small training data sets. With regards to the correlation structure, we also only consider configurations where $\rho_S = 0.7$ and $\rho_{S\bar{S}} = 0$ or 0.9. The selection percentages pertaining to *NL-LS* data scenarios are depicted in Figures 3.1 to 3.4.

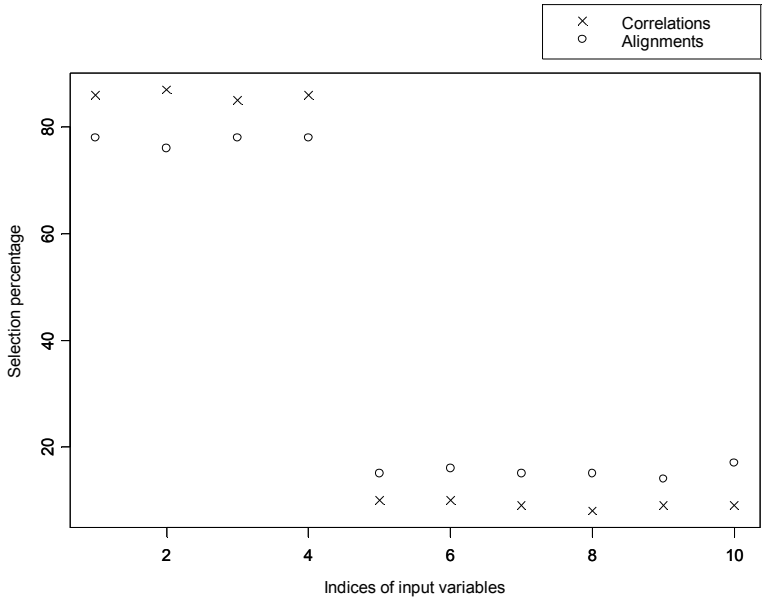


Figure 3.1: *Selection percentages obtained in the NL case*

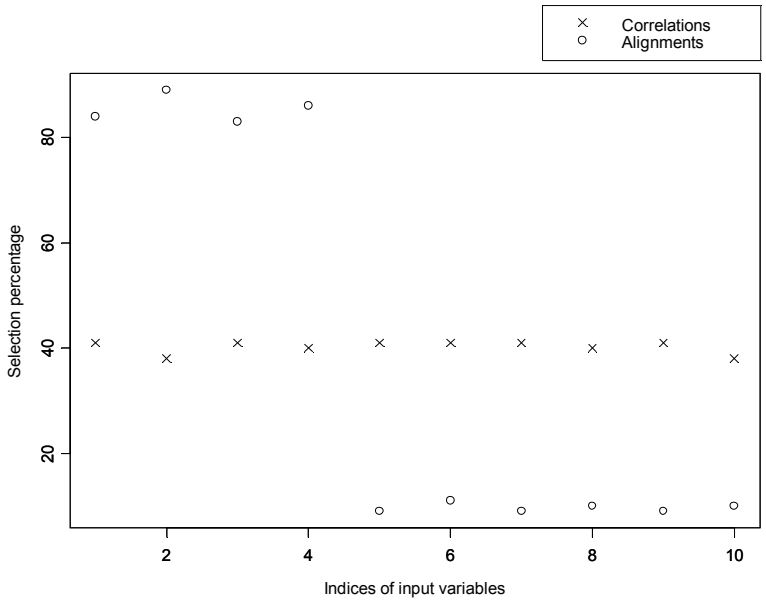


Figure 3.2: *Selection percentages obtained in the NS case*

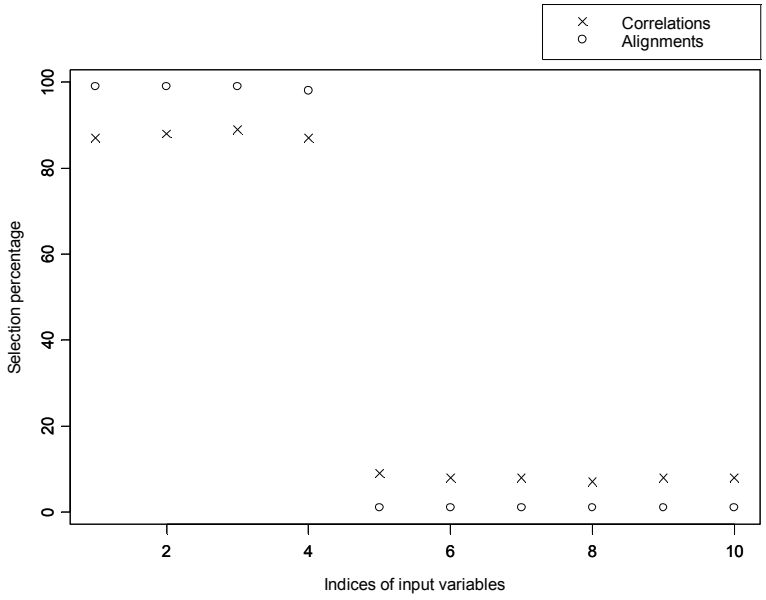


Figure 3.3: *Selection percentages obtained in the LL case*

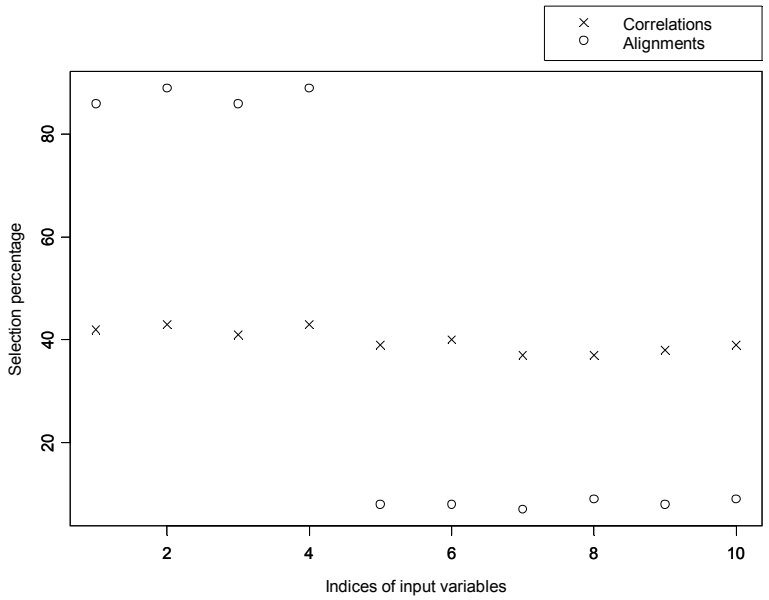


Figure 3.4: *Selection percentages obtained in the LS case*

In Figures 3.1 and 3.3 it is clear that in the *NL* and *LL* data setups correlation coefficients in $\aleph$ and alignments in $\Im$ does approximately equally well in selecting the separating variables. In the *NS* and *LS* cases reported in Figures 3.2 and 3.4 however, selection in $\aleph$ performs terribly, while selection in $\Im$ once again accurately identifies the relevant variable subset.

□

Of course correlation coefficients and alignments are very simple selection criteria. In the remainder of the chapter we propose and evaluate the use of other selection criteria defined in $\aleph$ and $\Im$.

3.3 *KERNEL VARIABLE SELECTION: FEATURE-TO-INPUT SPACE*

We saw in the previous section that variable selection is a worthwhile enterprise when kernel methods are used. In the *NL* case the simple use of traditional selection criteria in input space, without taking into account properties of the kernel method to be used, yielded adequate results. More generally however, we saw that the post-selection performance of kernel methods could be enhanced by incorporating algorithm-specific information in the selection procedure. Simply making use of the fact that the final classification of cases will take place in feature space, and therefore using correlations or alignments in $\mathfrak{I}$ rather than ordinary correlations in $\aleph$, lead to significant reductions in the observed test errors.

Notwithstanding the relatively poor performance of naïve selection in input space, conceptually there are reasons why one would prefer selection to take place in input space rather than in feature space. In terms of simplicity and interpretability the ideal would be to associate a single coefficient with each input variable, and base selection on the absolute sizes of these variable coefficients, as is for example frequently done in a multiple linear regression model. Although this ideal seems unattainable, we will see in this section that there are several approaches towards variable selection in input space using information from feature space. More specifically, we propose and evaluate variable selection approaches implemented in input space but taking into account the fact that kernel methods operate in feature space. This so-called *feature-to-input space* approach may thus be viewed as an intermediate approach bridging the gap between naïve selection approaches in input space, and more sophisticated selection procedures performed entirely in feature space.

Consider therefore the p-dimensional input vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, belonging to an input space $\aleph$, and consisting of n_j observations from group j, $j = 1,2$. The transformed input vectors, $\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n$ where $\boldsymbol{\varphi}_i = \Phi(\boldsymbol{x}_i)$, $i = 1,2,\ldots,n$, belong to a (possibly infinite dimensional) feature space $\mathfrak{I}$. Variable selection in $\mathfrak{I}$ is considerably more difficult than selection in $\aleph$ mainly because calculations in $\mathfrak{I}$ are restricted to evaluation of inner products between elements of $\mathfrak{I}$. The quantities in feature space which are amenable to

manipulation are linear combinations of the elements in $\Im$, in particular linear combinations of the form

$$\boldsymbol{\xi} \equiv \boldsymbol{\xi}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\varphi}_i \tag{3.1}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are known scalars. Our discussion focuses on the Gaussian kernel. For this kernel,

$$\|\boldsymbol{\varphi}_i\|^2 = \langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_i \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_i) = 1, \tag{3.2}$$

for $i = 1, 2, \ldots, n$, *i.e.* the observed data vectors in $\Im$ are all of length 1. From (3.1) we see that

$$\|\boldsymbol{\xi}\|^2 = \langle \boldsymbol{\xi}, \boldsymbol{\xi} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k_{ij} = \boldsymbol{\alpha}' \boldsymbol{K} \boldsymbol{\alpha} \tag{3.3}$$

and this clearly need not equal 1. It will prove useful in the following discussion to have $\|\boldsymbol{\xi}\|^2 = 1$. This can be guaranteed by restricting $\alpha_1, \alpha_2, \ldots, \alpha_n$ to satisfy

$$\boldsymbol{\alpha}' \boldsymbol{K} \boldsymbol{\alpha} = 1 \tag{3.4}$$

and we implement this restriction without loss of generality throughout the remainder of our discussion.

The important role played by quantities such as (3.1) in kernel classification is confirmed by the following observation. A binary kernel classifier is of the form

$$sign\left\{ \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \right\}, \tag{3.5}$$

where b represents an intercept and $\boldsymbol{x}$ the p-vector corresponding to a new case which has to be classified. We see that (3.5) equals

$$sign\left\{\left\langle \Phi(\boldsymbol{x}), \boldsymbol{\xi} \right\rangle + b\right\} \tag{3.6}$$

for some $\boldsymbol{\xi}$ as in (3.1). Hence, we can easily associate a kernel classifier with every $\boldsymbol{\xi}$, and vice versa. The quality of such a classifier will of course largely depend on the manner in which $\alpha_1, \alpha_2, \ldots, \alpha_n$ are determined or specified.

We require the following further notation in our discussion. Let

$$f(\boldsymbol{x}) = \left\langle \Phi(\boldsymbol{x}), \boldsymbol{\xi} \right\rangle = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}). \tag{3.7}$$

If we add an intercept to $f(\boldsymbol{x})$ we obtain a quantity which can be used to assign $\boldsymbol{x}$ to one of the two groups. Also, we write

$$f_j = f(\boldsymbol{x}_j) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j), \; j = 1,2,\ldots,n, \tag{3.8}$$

i.e. f_j is the value of f evaluated at the j^{th} input data vector. Finally, $\boldsymbol{f} = [f_1, f_2, \ldots, f_n]'$ denotes the vector with $f_1, f_2, \ldots, f_n$ as components. Combined once again with an intercept, the quantities $f_1, f_2, \ldots, f_n$ are used to classify the training data cases based on the kernel classifier associated with $f(\boldsymbol{x})$.

Three proposals for feature-to-input space selection are discussed below. The first and simplest of these is based on the idea of approximating $f(\boldsymbol{x})$ in (3.7) with an expression of the form $\sum_{j=1}^{p} \beta_j x_j$, where now we may directly interpret $|\beta_j|$ as an indicator of the relative

importance of the j^{th} variable (provided the data have been appropriately scaled). The second proposal is based on the concept of a *pre-image*: the pre-image of a vector $\boldsymbol{\xi} \in \Im$ is defined to be a vector $\boldsymbol{x} \in \aleph$ such that $\Phi(\boldsymbol{x}) = \boldsymbol{\xi}$. We will see that not all vectors $\boldsymbol{\xi} \in \Im$ have pre-images, and in such cases we are compelled to work in terms of approximations to the pre-image. The third proposal considers the quantities $f_1, f_2, \ldots, f_n$ and tries to identify the input variables which in some sense best explain the observed variation in these values.

3.3.1 *RE-EXPRESSING THE DISCRIMINANT FUNCTION*

Consider $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$, where the scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ have been determined by application of a kernel algorithm. With b denoting an intercept, $sign\{f(\boldsymbol{x}) + b\}$ is a kernel classifier. For some kernel functions it is possible to approximate $f(\boldsymbol{x})$ by an expression of the form $\sum_{j=1}^{p} \beta_j h(x_j)$, for scalars $\beta_1, \beta_2, \ldots, \beta_p$ depending on the input data and the values of $\alpha_1, \alpha_2, \ldots, \alpha_n$, and some function $h(.)$. The somewhat naïve idea behind such a re-expression is to interpret $|\beta_j|$ as an indicator of the relative importance of the j^{th} variable. Consider for example the kernel discriminant function when a quadratic kernel function is used. Ignoring the intercept term in the kernel discriminant function we have

$$
\begin{aligned}
f(\boldsymbol{x}) &= \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \\
&= \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle^2 \\
&= \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{p} x_{ij} x_j \right)^2
\end{aligned}
$$

$$= \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{p} x_{ij}^2 x_j^2 + \sum_{k \neq j}^{p} x_{ij} x_{ik} x_j x_k \right)$$

$$= \sum_{j=1}^{p} x_j^2 \left(\sum_{i=1}^{n} \alpha_i x_{ij}^2 \right) + \sum_{k \neq j}^{p} x_j x_k \left(\sum_{i=1}^{n} \alpha_i x_{ij} x_{ik} \right). \tag{3.9}$$

From this it would seem that we could use $\sum_{i=1}^{n} \alpha_i x_{ij}^2$ as an indication of the importance of variable x_j. In the case of a kernel classifier based on the Gaussian kernel function we can use a first-order Taylor expansion as follows:

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=1}^{n} \alpha_i e^{-\gamma \|\boldsymbol{x} - \boldsymbol{x}_i\|^2} \\ &= \sum_{i=1}^{n} \alpha_i \left\{ 1 - \gamma \|\boldsymbol{x} - \boldsymbol{x}_i\|^2 + \gamma^2 \|\boldsymbol{x} - \boldsymbol{x}_i\|^4 / 2 - \ldots \right\} \\ &\cong \sum_{i=1}^{n} \alpha_i - \gamma \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{p} (x_j - x_{ij})^2 \\ &= \sum_{i=1}^{n} \alpha_i - \gamma \sum_{j=1}^{p} x_j^2 \sum_{i=1}^{n} \alpha_i + 2\gamma \sum_{j=1}^{p} x_j \left(\sum_{i=1}^{n} \alpha_i x_{ij} \right) - \gamma \sum_{j=1}^{p} \sum_{i=1}^{n} \alpha_i x_{ij}^2 . \end{aligned} \tag{3.10}$$

From this it would seem that the quantity $\sum_{j=1}^{n} \alpha_i x_{ij}$ should provide an indication of the importance of variable x_j.

3.3.2 *PRE-IMAGE APPROXIMATIONS IN INPUT SPACE*

It has already been stated several times that application of a kernel method entails non-linear transformation of input data $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ to feature vectors $\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n$ using a feature mapping $\Phi : \aleph \to \Im$. The kernel trick is then used to compute inner products in the feature space $\Im$. In many applications, for example kernel classification, this approach is

sufficient and there is no need for a closer examination of quantities in $\Im$. Variable selection, however, seems to be an example of a problem where it would be useful to examine certain vectors in $\Im$ more closely. This is often a formidable problem, especially since $\Im$ may be infinite dimensional. The concept of the *pre-image* in $\aleph$ of a vector in $\Im$ seems to be worth investigating in this regard.

Consider therefore a linear expansion $\boldsymbol{\xi} \equiv \boldsymbol{\xi}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\varphi}_i$ in $\Im$, where $\boldsymbol{\varphi}_i = \Phi(\boldsymbol{x}_i)$, $i = 1,2,\ldots,n$, with $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, belonging to an input space $\aleph$. The pre-image of $\boldsymbol{\xi}$ is a vector $\boldsymbol{z} \equiv \boldsymbol{z}(\boldsymbol{\xi}) \in \aleph$ such that $\Phi(\boldsymbol{z}) = \boldsymbol{\xi}$. Schölkopf and Smola (2002, *p*. 544) provides the following result which can be used to calculate this pre-image *if it exists*.

LEMMA 3.1: ***COMPUTING THE PRE-IMAGE OF A LINEAR COMBINATION***

Consider $\boldsymbol{\xi} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$. If there exists a vector $\boldsymbol{z} \in \aleph$ such that $\Phi(\boldsymbol{z}) = \boldsymbol{\xi}$, and an invertible function $f_k(.)$, depending on the kernel function, such that $k(\boldsymbol{u}, \boldsymbol{w}) = f_k(\langle \boldsymbol{u}, \boldsymbol{w} \rangle)$, then we can compute the pre-image from

$$\boldsymbol{z} = \sum_{j=1}^{p} f_k^{-1}\left(\sum_{i=1}^{n} \alpha_i \, k(\boldsymbol{x}_i, \boldsymbol{e}_j) \right) \boldsymbol{e}_j , \tag{3.11}$$

where $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$ is any orthonormal basis of $\aleph$.

This is obviously a useful result for computing a pre-image if it exists. Unfortunately, in many cases a pre-image for quantities $\boldsymbol{\xi} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$ does not exist. In fact, for the

Gaussian kernel which we focus on, Schölkopf and Smola (2002, pp. 545-546) argue that the pre-image of $\boldsymbol{\xi}$ will only exist in trivial cases consisting of a single term. We are therefore forced into considering the somewhat less satisfactory option of finding an approximate pre-image for a given linear combination $\boldsymbol{\xi}$. The vector $\tilde{\boldsymbol{z}} \in \aleph$ is called an *approximate pre-image* of $\boldsymbol{\xi}$ if

$$\rho(\tilde{\boldsymbol{z}}) = \left\| \boldsymbol{\xi} - \Phi(\tilde{\boldsymbol{z}}) \right\|^2 \tag{3.12}$$

is small. As pointed out by Schölkopf and Smola (2002, *p*. 546), the meaning of small in this definition will depend on the particular application. How can an approximate pre-image be calculated? We describe an approach discussed by Schölkopf and Smola (2002, pp. 547-548). Consider the slightly more general problem of finding $\tilde{\boldsymbol{z}} \in \aleph$ such that $\beta\,\Phi(\tilde{\boldsymbol{z}})$ provides a good approximation to $\boldsymbol{\xi} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$. For $\beta = 1$ this is exactly the pre-image problem. Allowing $\beta \neq 1$ makes sense since the length of $\tilde{\boldsymbol{z}}$ is not crucial. A simple geometric argument (illustrated in Figure 3.3) confirms that this is equivalent to finding $\tilde{\boldsymbol{z}}$ which minimises the distance between $\boldsymbol{\xi}$ and its orthogonal projection onto $span(\Phi(\boldsymbol{z}))$, *i.e.*

$$\left\| \frac{\langle \boldsymbol{\xi}, \Phi(\boldsymbol{z}) \rangle}{\langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle} \Phi(\boldsymbol{z}) - \boldsymbol{\xi} \right\|^2 = \|\boldsymbol{\xi}\|^2 - \frac{\langle \boldsymbol{\xi}, \Phi(\boldsymbol{z}) \rangle^2}{\langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle}. \tag{3.13}$$

Our problem is therefore to find $\tilde{\boldsymbol{z}}$ to maximise $\dfrac{\langle \boldsymbol{\xi}, \Phi(\boldsymbol{z}) \rangle^2}{\langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle}$. For the Gaussian kernel this simplifies to finding $\tilde{\boldsymbol{z}}$ to maximise

$$\langle \boldsymbol{\xi}, \Phi(\boldsymbol{z}) \rangle^2 = \left(\sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{z}) \right)^2 = \left(\sum_{i=1}^{n} \alpha_i \, exp\left[-\gamma \left\| \boldsymbol{x}_i - \boldsymbol{z} \right\|^2 \right] \right)^2. \tag{3.14}$$

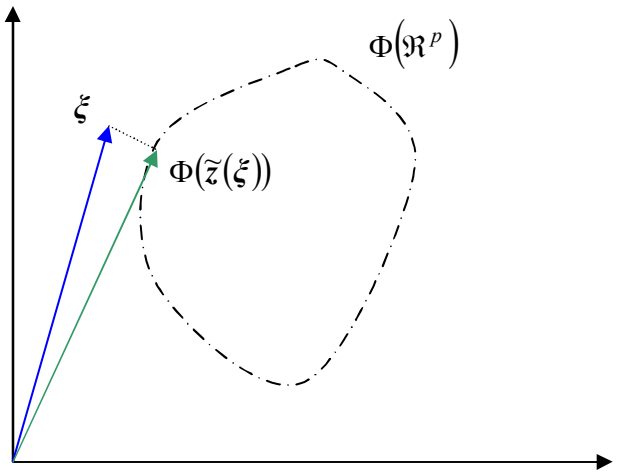


Figure 3.3: *We try to find* $\tilde{z}$ *which minimises the distance between* ξ *and its orthogonal projection onto* $span(\Phi(z))$

In our numerical study we used standard unconstrained optimisation software to find $\tilde{z}$. The corresponding optimal value of β can then be computed from

$$\tilde{\beta} = \frac{\langle \xi, \Phi(\tilde{z}) \rangle}{\langle \Phi(\tilde{z}), \Phi(\tilde{z}) \rangle} = \langle \xi, \Phi(\tilde{z}) \rangle . \tag{3.15}$$

How can the concept of (approximate) pre-images be utilised for variable selection? We investigated various possibilities empirically and found the following two options to be most promising. Since our focus is on binary classification, it seems a sensible idea to select those variables maximising some measure of the difference between the groups of data points corresponding to the two populations. Consider in this regard therefore the respective group means in $\Im$, *viz.*

$$\overline{\boldsymbol{\Phi}}_j = \frac{1}{n_j} \sum_{i \in I_j} \Phi(\boldsymbol{x}_i), \; j = 1,2 . \tag{3.16}$$

The two pre-images corresponding to these feature space mean vectors, namely $\tilde{z}(\overline{\boldsymbol{\Phi}}_1)$ and $\tilde{z}(\overline{\boldsymbol{\Phi}}_2)$ respectively, can be computed as explained above. Our first proposal for variable selection based on pre-images is to select the variables corresponding to the largest absolute components of the difference vector $\boldsymbol{\delta} = \tilde{z}(\overline{\boldsymbol{\Phi}}_1) - \tilde{z}(\overline{\boldsymbol{\Phi}}_2)$, which is equivalent to selecting the variables maximising the quantity $\left\|\tilde{z}(\overline{\boldsymbol{\Phi}}_1) - \tilde{z}(\overline{\boldsymbol{\Phi}}_2)\right\|^2$. This proposal is based on the assumption that variables which maximally separate the two groups in terms of the pre-images in $\aleph$ of their mean vectors in feature space, will also be the appropriate variables for separating the groups in feature space. Note that this proposal is independent of the specific kernel classifier being used. This variable selection proposal is investigated further in the simulation study discussed later, and we refer to it as a feature-to-input space proposal based on mean vectors, abbreviated to $FI(M)$.

It is also possible to propose a variable selection criterion based on pre-images and using information derived from the specific kernel classifier. Consider in this regard the kernel classifier weight vector $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$, where the scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ are determined from the training data according to some kernel algorithm. We can write this weight vector as

$$\boldsymbol{w} = \boldsymbol{w}_1 + \boldsymbol{w}_2 = \sum_{i \in I_1} \alpha_i \Phi(\boldsymbol{x}_i) + \sum_{i \in I_2} \alpha_i \Phi(\boldsymbol{x}_i). \tag{3.17}$$

In line with our first proposal we now consider the vector $\boldsymbol{\delta} = \tilde{z}(\boldsymbol{w}_1) - \tilde{z}(\boldsymbol{w}_2)$ and propose to select the variables corresponding to the largest absolute components of this vector. Here, $\tilde{z}(\boldsymbol{w}_1)$ and $\tilde{z}(\boldsymbol{w}_2)$ are once again the pre-images of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ respectively. This proposal, called feature-to-input space variable selection based on the weight vector and abbreviated to $FI(W)$, is also investigated more thoroughly in the later simulation study.

3.3.3 *SELECTING VARIABLES TO EXPLAIN THE VARIATION IN* $f_1, f_2, \ldots, f_n$

In this section we discuss the possibility of selecting input variables to explain the observed variation in the values $f_1, f_2, \ldots, f_n$, where $f_j = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $j = 1,2,\ldots,n$. Note that $sign\{f_j + b\}$, with b an intercept, is used to assign case j in the training data to one of the two groups.

A first possibility in this regard is to use multiple linear regression analysis. Note first of all that $f_j = \langle \boldsymbol{\alpha}, \boldsymbol{k}_j \rangle$, where $\boldsymbol{k}_j$ is the j^{th} column of the kernel matrix $\boldsymbol{K}$. Consequently, if we write $\boldsymbol{K} = [\boldsymbol{k}_1 \quad \boldsymbol{k}_2 \quad \ldots \quad \boldsymbol{k}_n]$, we have

$$\boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \boldsymbol{K}'\boldsymbol{\alpha} = \boldsymbol{K}\boldsymbol{\alpha} \tag{3.18}$$

(since the kernel matrix is symmetric). Suppose our objective is to approximate $f_1, f_2, \ldots, f_n$ as well as possible by using a function of the form $g(\boldsymbol{x}) = \sum_{k=1}^{p} \beta_k x_k$, with $\beta_1, \beta_2, \ldots, \beta_p$ constants which have to be determined. Applying this function to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ gives the values

$$g_j = g(\boldsymbol{x}_j) = \sum_{k=1}^{p} \beta_k x_{jk} = \langle \boldsymbol{\beta}, \boldsymbol{x}_j \rangle, j = 1,2,\ldots,n, \tag{3.19}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_p]'$ and $\boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_n'$ are the rows of the data matrix $\boldsymbol{X}$. We can therefore write

$$\boldsymbol{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = \boldsymbol{X}\boldsymbol{\beta}\,. \tag{3.20}$$

We would now like to determine $\beta_1, \beta_2, \ldots, \beta_p$ to minimise the discrepancy between the values in $\boldsymbol{f}$ and $\boldsymbol{g}$. If we measure discrepancy in a least squares sense, we look for $\beta_1, \beta_2, \ldots, \beta_p$ to minimise

$$\sum_{i=1}^{n}\left(f_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 . \tag{3.21}$$

This is an ordinary least squares optimisation problem, with solution

$$\hat{\boldsymbol{\beta}} \equiv \left[\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p\right]' = \left(\boldsymbol{X}'\boldsymbol{X}\right)^{-1}\boldsymbol{X}'\boldsymbol{f} = \left(\boldsymbol{X}'\boldsymbol{X}\right)^{-1}\boldsymbol{X}'\boldsymbol{K}\boldsymbol{\alpha}\,. \tag{3.22}$$

It follows that $\hat{\boldsymbol{g}} \equiv \left[\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_n\right]' = \boldsymbol{X}\hat{\boldsymbol{\beta}}$, and the accuracy of the least squares approximation is reflected in the value of the residual sum of squares,

$$err\left(\hat{\boldsymbol{\beta}}\right) = \left\|\boldsymbol{f} - \boldsymbol{g}\right\|^2 = \sum_{i=1}^{n}\left(f_i - g_i\right)^2 . \tag{3.23}$$

There is probably little reason to believe that this approximation will be accurate. However, the motivation for investigating $\hat{\boldsymbol{\beta}}$ is different: we hope that the relative importance of the input variables will be reflected in the relative sizes of $\left|\hat{\beta}_1\right|, \left|\hat{\beta}_2\right|, \ldots, \left|\hat{\beta}_p\right|$. The following heuristic argument lends support to this hope.

Consider the special case of a linear kernel function, *i.e.* $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left\langle \boldsymbol{x}_i, \boldsymbol{x}_j \right\rangle$. In this case the kernel matrix becomes $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}'$ and (3.22) simplifies to

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{\alpha}\text{, } i.e.\ \hat{\beta}_j = \sum_{i=1}^{n} \alpha_i x_{ij}\text{, } j = 1,2,\ldots,p\text{.} \tag{3.24}$$

Now suppose input variable X_1 separates the two groups well. Since we work with our data in centred form, this implies that x_{i1} will be (relatively) large positive for $i \in \{1,2,\ldots,n_1\}$ and (relatively) large negative for $i \in \{1,2,\ldots,n_2\}$ (or, vice versa). In addition, for any reasonable kernel classifier the (signed) α-coefficients corresponding to the $Y = +1$ group (the first n_1 data cases) will differ in sign from the α-coefficients corresponding to the $Y = -1$ group (the remaining n_2 data cases). This implies that $\hat{\beta}_1$ will have a relatively large absolute value, thereby reflecting the importance of variable X_1.

Variable selection based on multiple linear regression analysis of $f_1, f_2, \ldots, f_n$ is investigated further in the simulation study discussed later, and we refer to it as feature-to-input space least squares selection, abbreviated to $FI(LS)$.

The above approach may be viewed from a broader perspective. Apart from an intercept, the values $f_1, f_2, \ldots, f_n$ contain all the relevant information for classifying the training data cases. Naturally these values exhibit a certain degree of variation. It seems a reasonable strategy to select the input variables which in some sense best explain this variation. Such a strategy can be implemented in a variety of ways, of which multiple linear regression analysis as described above is a first possibility. More generally, at a first level a distinction can be made between a regression and a classification approach to analysing the information provided by $f_1, f_2, \ldots, f_n$. In the regression approach we view $f_1, f_2, \ldots, f_n$ as the values of a response variable and wish to identify the subset of input variables best explaining the variation in $f_1, f_2, \ldots, f_n$. In the classification approach, there are two options: we can either use $\hat{y}_i = sign(f_i)$ or y_i, $i = 1,2,\ldots,n$, as training data labels. Once again we seek the subset of input variables best explaining the variation in $\hat{y}_i$ or y_i,

$i = 1,2,\ldots,n$. At a second level, we have to decide whether to proceed parametrically or non-parametrically when modelling the variation in the response values. A parametric approach seems to be the less desirable option. Firstly, it makes rigid assumptions, which may be too restrictive, and secondly, it is difficult to surmise which parametric form to use. A non-parametric approach does not suffer from these disadvantages. Since classification and regression trees as well as random forests offer a natural way to quantify the importance of input variables, we propose their use in this regard.

The above remarks are summarised in Table 3.7. We propose linear discriminant analysis and multiple linear regression analysis as examples of parametric classification and regression techniques, although it must be noted that any other parametric classification or regression procedure may also be used.

Table 3.7: Possible approaches toward modelling the variation in response values

	CLASSIFICATION	***REGRESSION***
PARAMETRIC	Linear discriminant analysis	Multiple linear regression analysis
NON-PARAMETRIC	Classification trees or random forests	Regression trees or random forests

In an initial simulation study regression random forests were found to perform quite well as a variable selection strategy based on explaining the variation in $f_1, f_2, \ldots, f_n$. We therefore provide a more detailed discussion of random forests and the way in which they measure variable importance. A random forest consists of an ensemble of classification or regression trees (regression trees in our application) grown on bootstrap samples of the original training data and featuring random selections of input variables to determine the split at a particular node. More specifically, consider bootstrap samples $\mathcal{T}_b = \{(\boldsymbol{x}_i(b), y_i(b)),\ i = 1,2,\ldots,n\}$, drawn with replacement from the training data $\mathcal{T} = \{(\boldsymbol{x}_i, y_i),\ i = 1,2,\ldots,n\},\ b = 1,2,\ldots,B$. The tree grown on $\mathcal{T}_b$ is denoted by T_b, and

if this tree is applied to the original training data we obtain fitted values $\hat{f}_{b1}, \hat{f}_{b2}, \ldots, \hat{f}_{bn}$. Of course, some of the original data points will not be present in bootstrap sample b. Let $\overline{T}(i)$ represent the subset of trees from $T_1, T_2, \ldots, T_B$ that were trained on bootstrap samples not containing case i, $i = 1, 2, \ldots, n$. Then the random forest fitted value for case i is defined to be

$$\hat{f}_i(RF) = \sum_{b \in \overline{T}_i} \hat{f}_{bi} \Big/ card\left(\overline{T}_i\right), \tag{3.25}$$

where $card\left(\overline{T}_i\right)$ is the number of trees in $\overline{T}(i)$. Clearly, $\hat{f}_{bi}$, $b \in \overline{T}_i$, is an estimate of f_i based on a tree constructed from data which did not contain case i. In this sense it is to be expected that the squared error $\left(f_i - \hat{f}_{bi}\right)^2$ will provide a more realistic indication of the accuracy with which f_i can be estimated than a squared error based on an estimate of f_i calculated from data containing case i. Quantities such as $\hat{f}_{bi}$ are called *out-of-bag*, calculated for data cases not included in the sample from which the tree T_b was constructed. The performance of a random forest can be evaluated in terms of the out-of-bag estimates, *viz.* $\frac{1}{n}\sum_{i=1}^{n}\left[f_i - \hat{f}_i(RF)\right]^2$.

It is important to note that a random form of variable selection takes place when tree T_b in a random forest is grown. At each node in T_b a subset S_d of $d < p$ input variables is randomly selected from the total set of p available input variables. At that specific node the variable on which to split is then determined in the usual tree-growing fashion from the variables in S_d. Naturally the set S_d will typically differ from node to node. It is of course necessary to specify a value for d. In this regard it has been shown that the random forest error rate depends on the correlation between any two trees, as well as the strength of each individual tree in the forest (*cf.* Breiman, 2001). Small correlations between trees, combined with individual trees with low error rates, yield random forests with low error

rates. Now specifying a small value for d reduces the correlation between pairs of trees, but tends to increase the error rate of individual trees. We therefore need to specify a value for d which leads to a favourable trade-off between these two conflicting requirements. Fortunately the range of *d*-values leading to good random forest error rates has been found to be quite large. A recommendation which is frequently implemented in practice is to use $d = p/3$.

Of particular interest to us is quantification of the relative importance of the input variables in a random forest. The following approach, implemented in the software provided by Breiman and Cutler (http://stat-www.berkeley.edu/users/breiman/RandomForests/), was implemented in our simulation study. Suppose we wish to quantify the importance of variable X_j, $j = 1,2,\ldots,p$. Consider tree T_b and let $\bar{J}_b$ denote the set of cases which were not used in training this tree. We randomly permute the values of variable X_j for all the data cases in $\bar{J}_b$. Using the X_j - permuted out-of-bag training patterns corresponding to $\bar{J}_b$ we calculate estimates for all the f_i - values in $\bar{J}_b$. If this is done for all B trees, we can calculate out-of-bag random forest estimates $\hat{f}_i^{(j)}(RF)$, $i = 1,2,\ldots,n$. A measure of the importance of variable X_j is then obtained by calculating the relative increase in out-of-bag error caused by permutation of the X_j -values, *i.e.* the measure of variable importance for variable X_j is

$$\sum_{i=1}^{n}\left[f_i - \hat{f}_i^{(j)}(RF)\right]^2 \Big/ \sum_{i=1}^{n}\left[f_i - \hat{f}_i(RF)\right]^2 . \tag{3.26}$$

Large values of this measure would be an indication that X_j is an important input variable.

At this stage one might rightfully ask which of the above pre-image, regression and classification approaches should be used. Such a recommendation can only be made once the properties of kernel methods constructed from the selected input variables, yielded by

each selection approach, have been evaluated. Two properties are especially important in this regard. Firstly, in terms of the generalisation error of the classifier based on the selected variables, we would naturally like our procedure to be able to classify new cases accurately. Secondly, the probability with which a given selection procedure includes an input variable into the set of selected variables should be considered. Here one would hope that variables which contribute to separation between the groups will be selected with high probability while pure noise variables would appear fairly rarely amongst the selected variables. The complicated dependence on the data implied by a selection procedure, combined with the complex form of (some of) the selection criteria, make it impossible to investigate these post-selection classification properties of kernel methods analytically. Investigating the relative merits of the different selection procedures therefore necessitates one to conduct a simulation study.

In the remainder of this chapter we describe a fairly extensive simulation study for evaluating the different variable selection proposals based on pre-images and regression and classification analyses of $f_1, f_2, \ldots, f_n$, as described above. Details regarding the experimental design, the steps entailed by each simulation repetition, and the generation of training and test data sets are provided in Sections 3.4.1 to 3.4.3. In Section 3.4.4 we comment on the problem of specifying values for the hyperparameters in a kernel classifier, and in Section 3.4.5 we summarise the specific procedures which were investigated. Results of the study are discussed in Section 3.4.6.

3.4 *MONTE CARLO SIMULATION STUDY*

3.4.1 *EXPERIMENTAL DESIGN*

Naturally there are many factors influencing the post-selection properties of kernel classifiers, making it virtually impossible to conduct an exhaustive investigation in this regard. In our simulation studies we attempted to investigate the influence of the most important of these factors, which we now proceed to describe.

i. The underlying distribution from which the input variables arise. Two cases were considered: a multivariate normal, and a multivariate lognormal distribution. These distributions were selected as representative of symmetric and asymmetric distributions respectively.

ii. The manner in which the two groups differ. Firstly, we looked at situations where the groups differ with respect to location, and then only for a subset of m of the available input variables (see *v* below). We will refer to this subset of variables as the set of *separating* or *relevant* input variables, whose indices are contained in $V_S \subset \mathcal{V}$, with $card(V_S) = m$. The set of input variables which do not contribute to differences between the two groups will be denoted by $V_{\bar{S}}$ (with $card(V_{\bar{S}}) = p - m$). Secondly, situations were investigated where the two groups differ only with respect to spread (variance-covariance structure) – once again only for the variables contained in a subset V_S of the set of available input variables.

iii. The dimension of the problem, *i.e.* the total number p of input variables in $\mathcal{V}$, which was 10 in cases where $p < n$ and 60 whenever $p > n$.

iv. We varied the training sample size, investigating three cases. Firstly, small equal group sample sizes, *viz.* $n_1 = n_2 = 15$; secondly, relatively large equal

group sample sizes, *viz.* $n_1 = n_2 = 100$. Thirdly, to provide for scenarios where the prevalence of positive cases and negative cases differs (as is often the case), mixed sample sizes, *viz.* $n_1 = 25; n_2 = 75$. Combining these different sample sizes with the different values of p specified in *iii*, gives us the following cases which were investigated: $n_1 = n_2 = 15$ and $p = 10$ (referred to as *small samples* in the further discussion), $n_1 = 25; n_2 = 75$ and $p = 10$ (referred to as *mixed samples* in the further discussion), and $n_1 = n_2 = 100$ combined with $p = 10$ (referred to as *large samples* in the further discussion). In addition we also investigated the scenario $n_1 = n_2 = 15$ and $p = 60$, referred to as *wide samples* in the further discussion, since the standard 'data case' by 'variables' input matrix has a 'wide' appearance for such data sets. Note that we can also describe these scenarios in terms of the ratio of total sample size n to the number of input variables p. This gives n/p- values of 3, 10, 20 and 0.5 for the scenarios described above.

v. The fraction of relevant input variables contributing towards separation between the two groups, *i.e.* $\pi = m/p$. We used two fractions: 0.1 (thus, 1 out of 10, and 6 out of 60), and 0.4 (thus, 4 out of 10, and 24 out of 60).

vi. Finally, we varied the dependence amongst the input variables as reflected in their correlation coefficients. Consider two distinct input variables, X_j and X_k. If $X_j, X_k \in V_S$, we denote the correlation between X_j and X_k by ρ_S, and we assume that all such pairs exhibit this correlation. If $X_j, X_k \in V_{\bar{S}}$, we use the symbol $\rho_{\bar{S}}$ for the common correlation between all such pairs, and if $X_j \in V_S, X_k \in V_{\bar{S}}$, the common correlation coefficient is denoted by $\rho_{S\bar{S}}$. For *normal* input data we consistently assumed that all irrelevant variables were independent, *i.e.* we used $\rho_{\bar{S}} = 0$ throughout. For the relevant variables we used $\rho_S = 0$ and $\rho_S = 0.7$, combining this with two values for the correlation

between relevant and irrelevant variables, *viz.* $\rho_{S\bar{S}} = 0$ and $\rho_{S\bar{S}} = 0.9$. For *lognormal* input data we also fixed $\rho_{\bar{S}} = 0$ and investigated the following cases: for $\pi = 0.1$, we combined each of $\rho_S = 0$ and $\rho_S = 0.7$ with each of $\rho_{S\bar{S}} = 0$ and $\rho_{S\bar{S}} = 0.2$; for $\pi = 0.4$, we combined $\rho_S = 0$ and $\rho_S = 0.7$ with $\rho_{S\bar{S}} = 0$, and $\rho_S = 0.7$ with $\rho_{S\bar{S}} = 0.25$. See Appendix A.1, for a detailed motivation.

We require notation to describe the cases arising from combination of the factors described above at their different levels. Firstly, we write *NL* to denote cases where the training data were generated from two normal distributions differing with respect to location, *NS* for the cases where the two normal distributions differed with respect to spread, and *LL* and *LS* for the corresponding lognormal distribution cases. For the *NL* scenarios, Table 3.8 summarises the 32 different configurations that were investigated. In this table, ρ_1 denotes the cases where $\rho_S = 0, \rho_{S\bar{S}} = 0$; ρ_2 the cases where $\rho_S = 0.7, \rho_{S\bar{S}} = 0$; ρ_3 the cases where $\rho_S = 0, \rho_{S\bar{S}} = 0.9$; and ρ_4 the cases where $\rho_S = 0.7, \rho_{S\bar{S}} = 0.9$. As noted above, we used $\rho_{\bar{S}} = 0$ throughout. In later tables we refer to *NL*1-*NL*4, *NL*17-*NL*20 collectively as *small sample* cases, to *NL*5-*NL*8, *NL*21-*NL*24 collectively as *mixed sample* cases, to *NL*9-*NL*12, *NL*25-*NL*28 collectively as *large sample* cases, and to *NL*13-*NL*16, *NL*29-*NL*32 collectively as *wide sample* cases. Similar notation and conventions are used for the *NS* scenarios, with the obvious replacement of *NL* by *NS*, thereby giving rise to cases *NS*1 to *NS*32.

Table 3.8: Notation for *NL* (normal location differences) cases

			SAMPLE SIZE		
			$n_1 = n_2 = 15$	$n_1 = 25,\ n_2 = 75$	$n_1 = n_2 = 100$
$p = 10$	ρ_1	$\pi = 0.1$	*NL*1	*NL*5	*NL*9
		$\pi = 0.4$	*NL*2	*NL*6	*NL*10
	ρ_2	$\pi = 0.1$	*NL*3	*NL*7	*NL*11
		$\pi = 0.4$	*NL*4	*NL*8	*NL*12
	ρ_3	$\pi = 0.1$	*NL*17	*NL*21	*NL*25
		$\pi = 0.4$	*NL*18	*NL*22	*NL*26
	ρ_4	$\pi = 0.1$	*NL*19	*NL*23	*NL*27
		$\pi = 0.4$	*NL*20	*NL*24	*NL*28
$p = 60$	ρ_1	$\pi = 0.1$	*NL*13		
		$\pi = 0.4$	*NL*14		
	ρ_2	$\pi = 0.1$	*NL*15		
		$\pi = 0.4$	*NL*16		
	ρ_3	$\pi = 0.1$	*NL*29		
		$\pi = 0.4$	*NL*30		
	ρ_4	$\pi = 0.1$	*NL*31		
		$\pi = 0.4$	*NL*32		

For lognormal input data with location differences between the two groups, Table 3.9 provides a similar summary of the different configurations which were investigated. In this table, ρ_1 denotes the cases where $\rho_S = 0,\ \rho_{S\bar{S}} = 0$; ρ_2 the cases where $\rho_S = 0.7,\ \rho_{S\bar{S}} = 0$; ρ_3 the cases where $\rho_S = 0,\ \rho_{S\bar{S}} = 0.2$; ρ_4 the cases where $\rho_S = 0.7,\ \rho_{S\bar{S}} = 0.2$; and ρ_5 the cases where $\rho_S = 0.7,\ \rho_{S\bar{S}} = 0.25$. Once again we used $\rho_{\bar{S}} = 0$ throughout.

Table 3.9: Notation for *LL* (lognormal location differences) cases

			SAMPLE SIZE		
			$n_1 = n_2 = 15$	$n_1 = 25,\ n_2 = 75$	$n_1 = n_2 = 100$
$p = 10$	ρ_1	$\pi = 0.1$	*LL*1	*LL*5	*LL*9
		$\pi = 0.4$	*LL*2	*LL*6	*LL*10
	ρ_2	$\pi = 0.1$	*LL*3	*LL*7	*LL*11
		$\pi = 0.4$	*LL*4	*LL*8	*LL*12
	ρ_3	$\pi = 0.1$	*LL*17	*LL*20	*LL*23
	ρ_4	$\pi = 0.1$	*LL*18	*LL*21	*LL*24
	ρ_5	$\pi = 0.4$	*LL*19	*LL*22	*LL*25
$p = 60$	ρ_1	$\pi = 0.1$	*LL*13		
		$\pi = 0.4$	*LL*14		
	ρ_2	$\pi = 0.1$	*LL*15		
		$\pi = 0.4$	*LL*16		
	ρ_3	$\pi = 0.1$	*LL*26		
	ρ_4	$\pi = 0.1$	*LL*27		
	ρ_5	$\pi = 0.4$	*LL*28		

In later tables we refer to *LL*1-*LL*4, *LL*17-*LL*19 collectively as *small sample* cases, to *LL*5-*LL*8, *LL*20-*LL*22 collectively as *mixed sample* cases, to *LL*9-*LL*12, *LL*23-*LL*25 collectively as *large sample* cases, and to *LL*13-*LL*16, *LL*26-*LL*28 collectively as *wide sample* cases. Similar notation and conventions are used for the *LS* scenarios, with the obvious replacement of *LL* by *LS*, thereby giving rise to cases *LS*1 to *LS*28.

3.4.2 *STEPS IN EACH SIMULATION REPETITION*

For each of the configurations described in the previous section we performed 1000 Monte Carlo repetitions, each repetition entailing generation of a new training data set according to the specifications of the case. Now consider any given relevant selection procedure. This procedure was applied to the training data, thereby obtaining a selected subset of m variables. After mc Monte Carlo repetitions we therefore ended up with mc (potentially) different sets of input variables identified by this particular selection technique. For each selection technique we are interested in two measures of its performance:

i. The probability with which an input variable is selected;
ii. The generalisation error if the particular selection technique is applied.

The probability with which a given procedure selects variable X_j for inclusion in the model is estimated by

$$\hat{p}_j = \hat{P}\left(variable\ X_j\ is\ selected\right) = \frac{1}{mc}\sum_{k=1}^{mc} I_k\ , \tag{3.27}$$

where mc is the number of Monte Carlo simulation repetitions, and I_k is the indicator of the event that variable X_j is selected for inclusion at Monte Carlo repetition k. For estimating the generalisation error of a given selection technique, we have to use test data cases which are independent of the training data. Hence, as part of every Monte Carlo repetition we also generated a new test data set according to the specifications for the training data. The number of cases in the test data set was 2000 throughout, with the percentages corresponding to the two groups the same as in the training data. These test data cases were classified using the relevant kernel classifier based only on the input variables selected from the corresponding training data set. The misclassification error rate was calculated as the proportion of incorrectly classified test cases, and we used the average over the 1000 Monte Carlo repetitions of these misclassification error rates as an estimate of the generalisation error associated with the given selection proposal.

3.4.3 *GENERATING THE TRAINING AND TEST DATA*

For a more detailed description of the way in which the *NL*, *NS*, *LL* and *LS* data sets were generated, consider the p - component mean vector $\boldsymbol{\mu}_j$ and the $p \times p$ variance-covariance matrix $\boldsymbol{\Sigma}_j$ of the multivariate distribution for group j, $j = 1,2$. Partition these vectors and matrices as follows:

$$\boldsymbol{\mu}_j = \begin{bmatrix} \boldsymbol{\mu}_{S_j} \\ \boldsymbol{\mu}_{\bar{S}_j} \end{bmatrix}, \text{ and } \boldsymbol{\Sigma}_j = \begin{bmatrix} \boldsymbol{\Sigma}_{S_j} & \boldsymbol{\Sigma}_{S\bar{S}} \\ \boldsymbol{\Sigma}_{\bar{S}S} & \boldsymbol{\Sigma}_{\bar{S}} \end{bmatrix}, \tag{3.28}$$

where the quantities indexed by the subscript S correspond to the m relevant input variables in V_S, and the quantities indexed by $\bar{S}$ correspond to the $p-m$ irrelevant input variables in $V_{\bar{S}}$. Note therefore that $\boldsymbol{\Sigma}_{S\bar{S}}$ contains the covariances between pairs of relevant and irrelevant variables.

The following further notation is also required. We assumed the same variance for all relevant variables and we denote this value for group j by $\sigma^2_{S_j}$, $j = 1,2$. Similarly, the same variance was used for all irrelevant variables and for group j this value is denoted by $\sigma^2_{\bar{S}_j}$, $j = 1,2$. In group j we therefore have $var(X_k) = \sigma^2_{S_j}$ for all $X_k \in V_S$, and $var(X_k) = \sigma^2_{\bar{S}_j}$ for all $X_k \in V_{\bar{S}}$. Similarly, for $j = 1,2$, we write $\tau_{S_j} = covar(X_i, X_k)$ for all $X_i, X_k \in V_S$, $i \neq k$, and $\tau_{\bar{S}_j} = covar(X_i, X_k)$ for all $X_i, X_k \in V_{\bar{S}}$, $i \neq k$. Finally, let the common covariance in group j between any pair of relevant and irrelevant variables, *viz.* $covar(X_i, X_k)$, $X_i \in V_S$, $X_k \in V_{\bar{S}}$, be denoted by $\tau_{S\bar{S}_j}$, $j = 1,2$.

We are now ready to describe the parameter settings for the different scenarios which were investigated. In all cases where the two groups differed with respect to location, *i.e.* all the *NL* and *LL* cases, we used $\boldsymbol{\mu}_1 = \left[\mathbf{0}', \mathbf{0}'\right]'$ and $\boldsymbol{\mu}_2 = \left[\mathbf{1}', \mathbf{0}'\right]'$. This implies that the mean

vectors corresponding to the irrelevant variables were identical for both groups, while the relevant variables each had an expected value of 0 in the first group, and an expected value of 1 in the second group. In these cases the two groups had identical variance-covariance matrices and we set $\sigma^2_{S_j} = 1$ and $\sigma^2_{\bar{S}_j} = 20$ for $j = 1, 2$. This implies that we had $\tau_{S_1} = \tau_{S_2} = \rho_S$, $\tau_{\bar{S}_1} = \tau_{\bar{S}_2} = 20\,\rho_{\bar{S}}$ and $\tau_{S\bar{S}_1} = \tau_{S\bar{S}_2} = \sqrt{20}\,\rho_{S\bar{S}}$, with ρ_S, $\rho_{\bar{S}}$ and $\rho_{S\bar{S}}$ as discussed in Section 3.4.1.

In data scenarios where the two groups differed with respect to their variance-covariance structure, *i.e.* the *NS* and the *LS* cases, we set $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0}$, and generated differences between the two groups by assuming larger variation for the relevant variables in cases belonging to the second group, than for cases belonging to the first group. More specifically, we set $\sigma^2_{S_1} = 1$ and $\sigma^2_{S_2} = 10$, while keeping $\sigma^2_{\bar{S}_j} = 20$ for $j = 1, 2$. In these cases we therefore had $\tau_{S_1} = \rho_S$, $\tau_{S_2} = 10\,\rho_S$, $\tau_{\bar{S}_1} = \tau_{\bar{S}_2} = 20\,\rho_{\bar{S}}$, $\tau_{S\bar{S}_1} = \sqrt{20}\,\rho_{S\bar{S}}$ and $\tau_{S\bar{S}_2} = \sqrt{200}\,\rho_{S\bar{S}}$, with ρ_S, $\rho_{\bar{S}}$ and $\rho_{S\bar{S}}$ once again as discussed in Section 3.4.1.

Note finally that the Johnson translation system (Johnson, 1986) was used to generate lognormal data (*cf.* also Louw, 1997, and Section A.1 in this thesis).

3.4.4 *HYPERPARAMETER SPECIFICATION*

We saw in Chapter 2 that application of a kernel classifier requires specification of values for two types of parameters, *viz.* parameters used in the kernel function (γ in the Gaussian kernel), and a regularisation parameter (λ). Viewed in different frameworks, kernel procedures typically involve solving constrained optimisation problems which contain a penalisation term. Different formulations of these optimisation problems use either λ or, in the place of λ, a so-called *cost parameter* value (C), with λ inversely proportional to C.

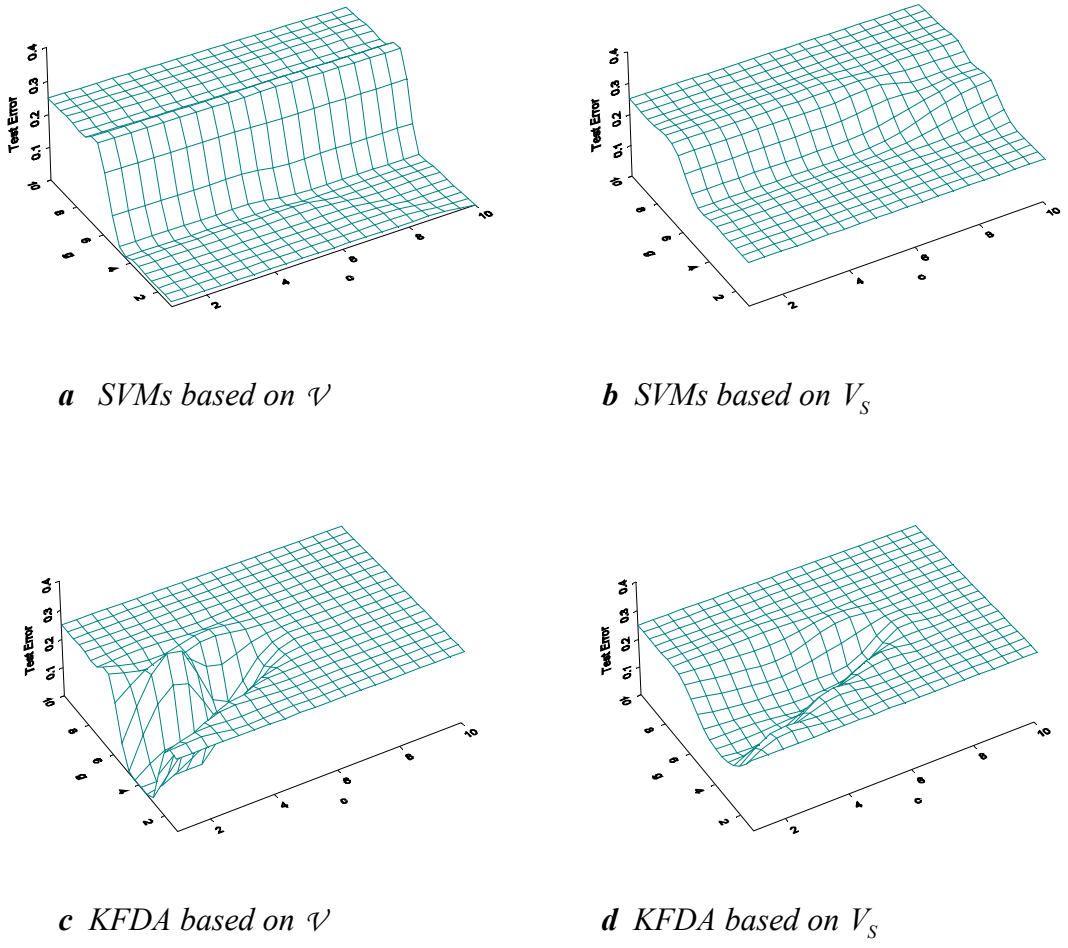
In this section we briefly discuss the problem of appropriately specifying values for C and γ. We start by exploring the influence of different specifications of C and γ on the accuracy of kernel classifiers, and end the section with a few pointers to the relevant literature with regard to hyperparameter specification.

We investigated the effect of hyperparameter values in kernel analyses via a numerical study. We were particularly interested in knowing whether the influence of hyperparameter specification differs for kernel procedures based on different subsets of variables. For this purpose we made use of the experimental design described in Sections 3.4.1 and 3.4.3. Each simulation repetition involved the following two steps. We generated a training sample according to the data scenario under consideration, and used it to train an SVM and perform KFDA using

i. the total number of available variables (in $\mathcal{V}$)
ii. only the fraction of relevant variables (in V_S).

We then generated a test data set according to the same data configuration, and calculated the test error achieved by each of the four classifiers. We repeated this process using C and γ which we varied over a grid, from $\left(C = 10^{-5}\,;\, \gamma = 10^{-5}\right)$ to $\left(C = 10^{3}\,;\, \gamma = 10^{3}\right)$. We used 1000 Monte Carlo repetitions, and for each classifier calculated average test errors over the 1000 repetitions.

Generally similar patterns were found for the different data scenarios considered. As an example, surface plots for the test errors obtained in the *NL* data, *viz*. *NL*18, is presented below.



a *SVMs based on* $\mathcal{V}$ ***b*** *SVMs based on* V_S

c *KFDA based on* $\mathcal{V}$ ***d*** *KFDA based on* V_S

Figure 3.4: *Surface plots of average test errors in NL18 data*

Two observations are important in Figure 3.4. Firstly, in kernel analyses the use of different subsets of variables have a significant impact on the effect of the specification of hyperparameter values. Secondly, the effect of hyperparameters changes when different kernel procedures are used. In the *NL*18 data configuration, for example, it seems as if the region of unsuitable hyperparameter specifications is somewhat smaller when SVMs or KFDA is based on V_S instead of on $\mathcal{V}$. Surface plots of the errors in SVMs are also quite different from those obtained for KFDA.

Many contributions in the literature focus on the problem of optimising hyperparameter values in kernel analyses. Cross-validation is often used, although it is computationally expensive. This may become problematic in cases where a large number of hyperparameters have to be specified.

Many authors investigate specification of multiple hyperparameter values. References in this regard include Amari and Wu (1999), Chapelle *et al*. (2004), Duan *et al*. (2001), Grandvalet and Canu (2002), Keerthi (2002), and Keerthi and Lin (2003). Of particular interest is the paper by Hastie *et al*. (2004), in which it is shown that the regularised optimisation problem for SVMs (refer to Chapter 2) can be solved simultaneously for all values of the regularisation parameter λ. It should also be noted that some of the ideas in the above papers have been extended to variable selection: input variables with relatively small associated kernel hyperparameter values are considered unimportant and are therefore discarded from the model.

In the numerical investigations presented in the thesis, our concern with regard to the specification of hyperparameters was that it should not influence our recommendations regarding which selection procedures perform better.

The way in which we specified γ was based on the results of an empirical study in Steel and Louw (2004), which strongly suggest using the reciprocal of the number of variables in the model. In this chapter, as well as in Chapters 4 and 5, our aim is to rank different variable subsets of the same size (m). Allowing kernel analyses based on $\mathcal{V}$ to be favoured, we therefore specified $\gamma = 1/p$. According to the proposal by Steel and Louw (2004) this will not be the best choice for any of the post-selection models (a better specification would have been $\gamma = 1/m$). Since we are only interested in the relative performance of selection procedures, we do not consider this to be a drawback. The $\gamma = 1/p$ specification places models based on $\mathcal{V}$ at an advantage. Hence the error of a selection procedure X relative to the no selection error (*i.e.* $Err(X)/Err(\mathcal{V})$) can be regarded as a conservative estimate of improvements in accuracy gained via the use of X.

In numerical evaluations presented in the remainder of this chapter, we varied the cost parameter between 10^{-4} and 10^{4}. The values for C that were used in Chapters 4 to 6 are given in the sections where the Monte Carlo simulation studies are described.

3.4.5 *THE VARIABLE SELECTION PROCEDURES*

Several variable selection proposals were introduced in Section 3.3. All of these proposals were evaluated empirically in a simulation study along the lines described in Sections 3.4.1 to 3.4.3. It became evident, however, that not all of the proposed approaches are worthwhile. Consequently the results presented and discussed in Section 3.4.6 are only for the following selection procedures.

- Feature-to-input space pre-image approximations of the group mean vectors. We abbreviate this to *FI*(*M*) – see the discussion in Section 3.3.2.
- Feature-to-input space pre-image approximation of the kernel classifier weight vector. We abbreviate this to *FI*(*W*) – see once again the discussion in Section 3.3.2.
- Using least squares multiple linear regression analysis to explain the variation in the values $f_1, f_2, \ldots, f_n$, where $f_j = \sum_{i=1}^{n} \alpha_i k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right),\ j = 1,2,\ldots,n$. We abbreviate this to *FI*(*LS*) – see the discussion in Section 3.3.3.
- Using regression random forests to explain the variation in the values $f_1, f_2, \ldots, f_n$. We abbreviate this to *FI*(*RF*) – see once again the discussion in Section 3.3.3.
- Using alignments to perform selection in feature space. Since the selection in this case is performed entirely in feature space, we abbreviate this to *F*(*A*).

The selection strategies discussed in Sections 3.4.1 to 3.4.3 that are excluded from the simulation study include the strategies based on re-expressing the kernel function

(discussed in Section 3.3.1), and several of the approaches for explaining the variation in $f_1, f_2, \ldots, f_n$ discussed in Section 3.3.3.

3.4.6 *RESULTS AND CONCLUSIONS*

In this section we present and briefly discuss the results of the simulation study described above. We focus on two aspects: the post-selection classification accuracy of SVMs after using the variable selection procedures in Section 3.4.5, and the probabilities with which each of the variables was selected.

Consider first the post-selection classification accuracy of the selection procedures considered. For easier comparison of the procedures, we calculated the size of the average test error obtained for a procedure *X* relative to that of the SVM based only on the relevant subset of variables, *i.e.* $Err(X)/Err(V_S)$. We will refer to these relative errors as *selection values*. The obtained selection values are reported in two parts. Firstly, in Tables 3.10-3.13, we report all values pertaining to the first 16 configurations for each data scenario (*i.e.* the configurations where $\rho_{S\bar{S}} = 0$, *viz.* *NL*1-*NL*16, *NS*1-16*NS*16, *LL*1-*LL*16, and *LS*1-*LS*16). Secondly, the selection values for the configurations where $\rho_{S\bar{S}} > 0$, (*viz.* *NL*17-*NL*32, *NS*17-*NS*32, *LL*17-*LL*28, and *LS*1-*LS*28), are given in Tables 3.14-3.17.

The selection values in Tables 3.10-3.13 reveal that it is difficult to make an overall recommendation regarding the best performing variable selection procedures: the relative performances of the procedures seem to depend on the data scenario (*NL*, *NS*, *LL* or *LS*), as well as on the sample size.

Table 3.10: Selection values for *NL* data

	ρ_S	$\rho_{S\bar{S}}$	π	*F*(*A*)	*FI*(*M*)	*FI*(*W*)	*FI*(*LS*)	*FI*(*RF*)
SMALL SAMPLES								
***NL*1**	0.0	0.0	0.1	1.19	1.16	1.17	1.16	1.16
***NL*2**	0.0	0.0	0.4	1.19	1.15	1.18	1.23	1.16
***NL*3**	0.7	0.0	0.1	1.20	1.15	1.16	1.15	1.16
***NL*4**	0.7	0.0	0.4	1.04	1.03	1.07	1.09	1.05
MIXED SAMPLES								
***NL*5**	0.0	0.0	0.1	1.03	1.00	1.02	1.00	1.01
***NL*6**	0.0	0.0	0.4	1.09	1.02	1.06	1.05	1.03
***NL*7**	0.7	0.0	0.1	1.03	1.00	1.02	1.00	1.01
***NL*8**	0.7	0.0	0.4	1.01	1.00	1.04	1.05	1.02
LARGE SAMPLES								
***NL*9**	0.0	0.0	0.1	1.00	1.00	1.00	1.00	1.00
***NL*10**	0.0	0.0	0.4	1.00	1.00	1.00	1.00	1.00
***NL*11**	0.7	0.0	0.1	1.00	1.00	1.00	1.00	1.00
***NL*12**	0.7	0.0	0.4	1.00	1.00	1.02	1.03	1.01
WIDE SAMPLES								
***NL*13**	0.0	0.0	0.1	1.75	1.55	1.57	2.44	3.51
***NL*14**	0.0	0.0	0.4	2.11	1.74	1.74	1.74	25.4
***NL*15**	0.7	0.0	0.1	1.08	1.05	1.09	1.05	1.55
***NL*16**	0.7	0.0	0.4	1.01	1.01	1.03	1.00	1.58

The *NL* scenario (in Table 3.10) leads us to somewhat different conclusions than in the case of *NS*, *LL* and *LS* data. Therefore first consider this setup. In small sample cases, the selection procedures perform very similar – still *FI*(*M*) seems to perform the best, yielding slightly smaller average test errors than those obtained for *FI*(*RF*). In mixed and large

samples the selection values corresponding to the different selection procedures all lie very close to 1, indicating that practically all the selection procedures consistently identify the correct variables. This shows that the selection criteria perform well, especially in large samples. The *FI*(*M*) and *FI*(*W*) criteria perform relatively well in wide sample cases, followed by *F*(*A*); whereas *FI*(*RF*) performs rather poorly.

The relatively high selection values obtained for *FI*(*RF*) in wide *NL* samples lead us to also closely consider the *FI*(*RF*)-selection values obtained for *NS*, *LL* and *LS* data. In this regard, we see that *FI*(*RF*) performs poorly only in wide samples from a normal distribution. The *NL*14 and *NS*14 configurations are by far the most problematic: compare for example the *FI*(*RF*) selection value of 25.4 to the value of 1.74 achieved by *FI*(*M*) in *NL*14, and similarly, 83.17 with 48.33 in *NS*14. The performance of *FI*(*RF*) is affected less in *NL* data.

We now consider the *NS*, *LL* and *LS* data setups (in Tables 3.11 to 3.13 respectively). Here *F*(*A*) and *FI*(*RF*) definitely stand out, consistently yielding the lowest selection values, and doing so by a relatively large margin. The relative performance of *F*(*A*) and *FI*(*RF*) changes when the different data scenarios and sample sizes are considered. In small *NS* and *LL* samples, *F*(*A*) outperforms *FI*(*RF*), whereas in mixed *NS*, *LL* and *LS* data the reverse is true. Once again in wide *NS*, *LL* and *LS* data, *F*(*A*) performs best. In large samples it is not clear which of the two procedures should be favoured.

Concerning the performance of the remaining selection procedures, we again see that in data sets with different sizes, the relative performance of *FI*(*M*), *FI*(*W*) and *FI*(*LS*) vary. We have seen that in small *NL* data, *FI*(*M*) performs best. In mixed *NS*, *LL* and *LS* data, *FI*(*W*) is favoured after *FI*(*RF*) and *F*(*A*), whereas in wide *NS* and *LS* samples, *FI*(*LS*) comes in third.

Turning our attention to the configurations where $\rho_{S\bar{S}} > 0$ (in Tables 3.14-3.17), we see that in essence, our conclusions and recommendations in the case of $\rho_{S\bar{S}} = 0$ also hold for

these data setups. We therefore only indicate instances where the remarks in the previous

Table 3.11: Selection values for *NS* data

	ρ_S	$\rho_{S\bar{S}}$	π	$F(A)$	$FI(M)$	$FI(W)$	$FI(LS)$	$FI(RF)$
SMALL SAMPLES								
***NS*1**	0.0	0.0	0.1	1.19	1.64	1.63	1.64	1.26
***NS*2**	0.0	0.0	0.4	1.44	2.82	2.58	2.87	1.63
***NS*3**	0.7	0.0	0.1	1.18	1.66	1.65	1.67	1.26
***NS*4**	0.7	0.0	0.4	1.23	1.89	1.84	1.78	1.37
MIXED SAMPLES								
***NS*5**	0.0	0.0	0.1	1.06	1.04	1.04	1.05	1.04
***NS*6**	0.0	0.0	0.4	3.98	3.34	2.66	2.91	1.05
***NS*7**	0.7	0.0	0.1	1.06	1.04	1.04	1.04	1.04
***NS*8**	0.7	0.0	0.4	3.35	2.81	2.28	2.29	1.07
LARGE SAMPLES								
***NS*9**	0.0	0.0	0.1	1.00	1.83	1.82	1.72	1.00
***NS*10**	0.0	0.0	0.4	1.00	3.11	2.34	2.83	1.00
***NS*11**	0.7	0.0	0.1	1.00	1.84	1.80	1.72	1.00
***NS*12**	0.7	0.0	0.4	1.00	2.88	2.18	2.17	1.00
WIDE SAMPLES								
***NS*13**	0.0	0.0	0.1	2.02	3.73	3.74	3.26	3.96
***NS*14**	0.0	0.0	0.4	7.67	48.33	45.67	5.50	83.2
***NS*15**	0.7	0.0	0.1	1.50	2.38	2.44	1.80	2.51
***NS*16**	0.7	0.0	0.4	1.31	2.19	2.38	1.19	3.26

paragraphs cannot be extended to $\rho_{S\bar{S}} > 0$ data setups, and further comment on some aspects that we feel is important to emphasise. Firstly, in the small sample *NL* setup, we

see that *FI*(*M*) is no longer the best performer, and that the *FI*(*LS*) procedure should

Table 3.12: Selection values for *LL* data

	ρ_S	$\rho_{S\bar{S}}$	π	*F*(*A*)	*FI*(*M*)	*FI*(*W*)	*FI*(*LS*)	*FI*(*RF*)
SMALL SAMPLES								
***LL*1**	0.0	0.0	0.1	1.05	1.42	1.33	1.70	1.07
***LL*2**	0.0	0.0	0.4	1.03	1.15	1.15	1.71	1.01
***LL*3**	0.7	0.0	0.1	1.06	1.32	1.32	1.73	1.09
***LL*4**	0.7	0.0	0.4	1.02	1.09	1.10	1.28	1.05
MIXED SAMPLES								
***LL*5**	0.0	0.0	0.1	1.00	1.05	1.05	1.08	1.00
***LL*6**	0.0	0.0	0.4	1.02	1.02	1.07	1.17	1.00
***LL*7**	0.7	0.0	0.1	1.02	1.05	1.03	1.12	1.00
***LL*8**	0.7	0.0	0.4	1.00	1.00	1.00	1.16	1.00
LARGE SAMPLES								
***LL*9**	0.0	0.0	0.1	1.00	1.03	1.02	1.03	1.01
***LL*10**	0.0	0.0	0.4	1.00	1.06	1.06	1.00	1.00
***LL*11**	0.7	0.0	0.1	1.00	1.04	1.03	1.03	1.01
***LL*12**	0.7	0.0	0.4	1.00	1.02	1.30	1.21	1.00
WIDE SAMPLES								
***LL*13**	0.0	0.0	0.1	1.10	1.56	1.67	3.61	1.05
***LL*14**	0.0	0.0	0.4	1.10	1.48	1.48	1.48	1.48
***LL*15**	0.7	0.0	0.1	1.07	1.36	1.36	1.37	1.11
***LL*16**	0.7	0.0	0.4	1.04	1.13	1.06	1.06	1.22

rather be used. In wide *NL* data sets, where formerly *FI*(*W*) came second to *FI*(*M*), *FI*(*W*) now performs best, followed by *FI*(*LS*). Secondly, as was seen in the $\rho_{S\bar{S}} = 0$ data setups,

the use of *FI*(*RF*) can also in the case of $\rho_{S\bar{S}} > 0$ not be recommended for wide *NL* and *NS*

Table 3.13: Selection values for *LS* data

	ρ_S	$\rho_{S\bar{S}}$	π	*F*(*A*)	*FI*(*M*)	*FI*(*W*)	*FI*(*LS*)	*FI*(*RF*)
SMALL SAMPLES								
***LS*1**	0.0	0.0	0.1	1.24	1.97	1.92	2.16	1.12
***LS*2**	0.0	0.0	0.4	1.23	1.88	1.74	2.24	1.13
***LS*3**	0.7	0.0	0.1	1.26	1.98	1.92	2.14	1.15
***LS*4**	0.7	0.0	0.4	1.17	1.55	1.65	1.53	1.16
MIXED SAMPLES								
***LS*5**	0.0	0.0	0.1	1.21	1.17	1.13	1.15	1.00
***LS*6**	0.0	0.0	0.4	3.58	2.28	1.84	2.56	1.02
***LS*7**	0.7	0.0	0.1	1.21	1.17	1.14	1.15	1.01
***LS*8**	0.7	0.0	0.4	3.19	1.65	2.04	2.03	1.10
LARGE SAMPLES								
***LS*9**	0.0	0.0	0.1	1.01	1.66	1.12	2.02	1.00
***LS*10**	0.0	0.0	0.4	0.95	1.33	1.07	2.03	0.95
***LS*11**	0.7	0.0	0.1	1.01	1.65	1.12	2.12	1.00
***LS*12**	0.7	0.0	0.4	1.00	1.12	1.52	1.59	1.00
WIDE SAMPLES								
***LS*13**	0.0	0.0	0.1	1.51	2.98	2.95	2.83	1.41
***LS*14**	0.0	0.0	0.4	1.79	4.39	4.30	1.93	1.98
***LS*15**	0.7	0.0	0.1	1.42	2.34	2.42	1.67	1.43
***LS*16**	0.7	0.0	0.4	1.41	1.97	2.14	1.23	1.78

Samples. For example, the *NL*14 selection value for *FI*(*RF*) is 25.89, compared to a value of 1.42 in the case of *FI*(*M*). Thirdly, in the *NS*, *LL* and *LS* scenarios once again *F*(*A*) and *FI*(*RF*) dominate.

Table 3.14: Selection values for *NL* data

	ρ_S	$\rho_{S\bar{S}}$	π	*F*(*A*)	*FI*(*M*)	*FI*(*W*)	*FI*(*LS*)	*FI*(*RF*)
SMALL SAMPLES								
***NL*17**	0.0	0.9	0.1	1.2	1.15	1.16	1.08	1.16
***NL*18**	0.0	0.9	0.4	1.13	1.11	1.07	1.04	1.13
***NL*19**	0.7	0.9	0.1	1.19	1.15	1.15	1.08	1.16
***NL*20**	0.7	0.9	0.4	1.03	1.03	1.04	1.04	1.03
MIXED SAMPLES								
***NL*21**	0.0	0.9	0.1	1.04	1.00	1.01	1.00	1.01
***NL*22**	0.0	0.9	0.4	1.06	1.01	1.02	1.01	1.03
***NL*23**	0.7	0.9	0.1	1.04	1.01	1.01	1.00	1.01
***N*24**	0.7	0.9	0.4	1.00	1.00	1.02	1.02	1.00
LARGE SAMPLES								
***NL*25**	0.0	0.9	0.1	1.00	1.00	1.00	1.00	1.00
***NL*26**	0.0	0.9	0.4	1.00	1.00	1.01	1.00	1.01
***NL*27**	0.7	0.9	0.1	1.00	1.00	1.00	1.00	1.00
***NL*28**	0.7	0.9	0.4	1.00	1.00	1.00	1.00	1.00
WIDE SAMPLES								
***NL*29**	0.0	0.9	0.1	1.18	1.28	1.05	1.21	3.53
***NL*30**	0.0	0.9	0.4	1.16	1.42	0.11	0.05	25.89
***NL*31**	0.7	0.9	0.1	1.07	1.09	1.07	1.07	1.64
***NL*32**	0.7	0.9	0.4	1.00	1.01	0.87	0.97	1.59

Our conclusions with regard to the relative performances of $F(A)$ and $FI(RF)$ in small, mixed and large samples when $\rho_{S\bar{S}} = 0$ carry directly over to cases where $\rho_{S\bar{S}} > 0$. In wide samples, however, a distinction between the two procedures is not that clear, except in *LL* data, where $F(A)$ performs best.

Table 3.15: Selection values for *NS* data (continued)

	ρ_S	$\rho_{S\bar{S}}$	π	$F(A)$	$FI(M)$	$FI(W)$	$FI(LS)$	$FI(RF)$
SMALL SAMPLES								
***NS*17**	0.0	0.9	0.1	1.2	1.66	1.64	1.67	1.27
***NS*18**	0.0	0.9	0.4	1.47	2.81	2.58	2.83	1.65
***NS*19**	0.7	0.9	0.1	1.19	1.67	1.64	1.66	1.26
***NS*20**	0.7	0.9	0.4	1.21	1.91	1.85	1.77	1.36
MIXED SAMPLES								
***NS*21**	0.0	0.9	0.1	1.06	1.04	1.04	1.04	1.04
***NS*22**	0.0	0.9	0.4	3.97	3.27	2.57	2.85	1.05
***NS*23**	0.7	0.9	0.1	1.07	1.05	1.04	1.05	1.04
***NS*24**	0.7	0.9	0.4	3.31	2.83	2.22	2.23	1.25
LARGE SAMPLES								
***NS*25**	0.0	0.9	0.1	1.00	1.83	1.80	1.70	1.00
***NS*26**	0.0	0.9	0.4	1.00	3.13	2.29	2.76	1.00
***NS*27**	0.7	0.9	0.1	1.00	1.82	1.79	1.70	1.00
***NS*28**	0.7	0.9	0.4	1.00	2.78	2.13	2.16	1.00
WIDE SAMPLES								
***NS*29**	0.0	0.9	0.1	1.74	2.22	2.26	1.90	3.93
***NS*30**	0.0	0.9	0.4	33.8	40.8	17.8	15.0	56.4
***NS*31**	0.7	0.9	0.1	1.31	1.66	2.17	1.56	0.94
***NS*32**	0.7	0.9	0.4	1.32	1.73	1.91	1.23	1.69

Table 3.16: Selection values for *LL* data (continued)

	ρ_S	$\rho_{S\bar{S}}$	π	$F(A)$	$FI(M)$	$FI(W)$	$FI(LS)$	$FI(RF)$
SMALL SAMPLES								
***LL*17**	0.0	0.20	0.1	1.03	1.36	1.14	1.26	1.10
***LL*18**	0.7	0.20	0.1	1.03	1.31	1.17	1.23	1.08
***LL*19**	0.7	0.25	0.4	1.01	1.04	1.08	1.15	1.05
MIXED SAMPLES								
***LL*20**	0.0	0.20	0.1	1.00	1.00	1.00	0.96	0.94
***LL*21**	0.7	0.20	0.1	1.02	1.06	1.05	1.03	1.00
***LL*22**	0.7	0.25	0.4	1.00	1.03	1.03	1.18	1.06
LARGE SAMPLES								
***LL*23**	0.0	0.20	0.1	1.00	1.01	1.00	1.04	1.04
***LL*24**	0.7	0.20	0.1	1.00	1.03	1.02	1.04	1.01
***LL*25**	0.7	0.25	0.4	1.00	1.07	1.40	1.43	1.01
WIDE SAMPLES								
***LL*26**	0.0	0.20	0.1	1.00	1.19	1.10	1.10	1.00
***LL*27**	0.7	0.20	0.1	1.02	1.19	1.20	1.19	1.10
***LL*28**	0.7	0.25	0.4	1.01	1.12	1.02	0.98	1.10

Consider now the selection percentages with which each of the input variables was selected. In Figure 3.5 we present the percentages obtained for $F(A)$, $FI(RF)$ and $FI(M)$ in the second configuration of each data scenario.

Table 3.17: Selection values for *LS* data (continued)

	ρ_S	$\rho_{S\bar{S}}$	π	$F(A)$	$FI(M)$	$FI(W)$	$FI(LS)$	$FI(RF)$
SMALL SAMPLES								
***LS*17**	0.0	0.20	0.1	1.28	1.95	1.58	1.95	1.07
***LS*18**	0.7	0.20	0.1	1.27	1.88	1.50	1.92	1.12
***LS*19**	0.7	0.25	0.4	1.16	1.42	1.50	1.50	1.16
MIXED SAMPLES								
***LS*20**	0.0	0.20	0.1	1.21	1.13	1.03	1.15	1.00
***LS*21**	0.7	0.20	0.1	1.20	1.14	1.11	1.15	1.00
***LS*22**	0.7	0.25	0.4	3.17	1.43	1.94	2.01	1.08
LARGE SAMPLES								
***LS*23**	0.0	0.20	0.1	0.99	1.25	1.02	2.02	0.99
***LS*24**	0.7	0.20	0.1	1.01	1.26	1.01	2.12	1.00
***LS*25**	0.7	0.25	0.4	1.00	1.05	1.62	1.62	1.00
WIDE SAMPLES								
***LS*26**	0.0	0.20	0.1	1.07	1.41	1.16	1.34	1.08
***LS*27**	0.7	0.20	0.1	1.15	1.71	1.58	1.51	1.21
***LS*28**	0.7	0.25	0.4	1.16	1.43	1.51	1.48	1.14

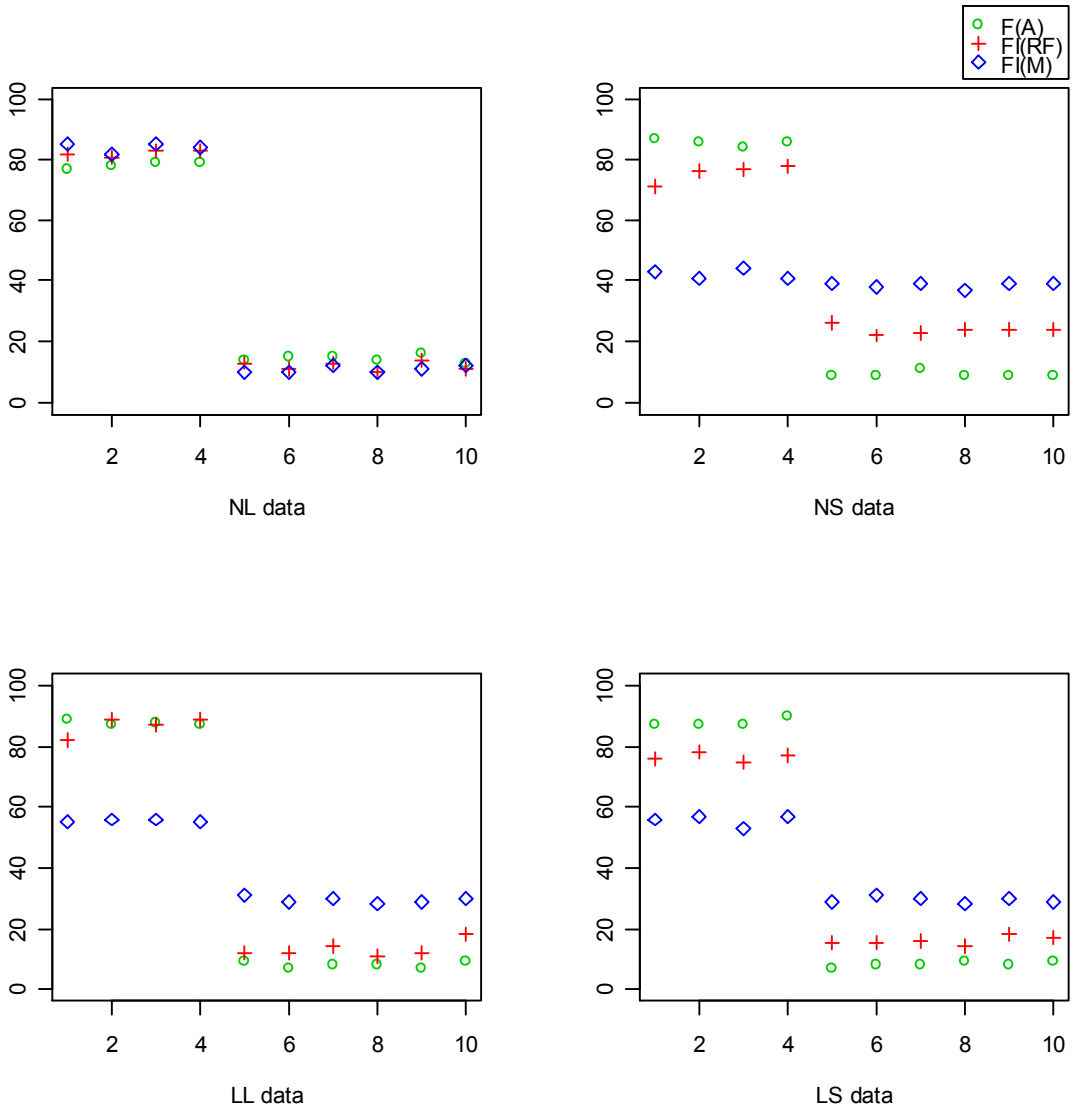


Figure 3.5: *Selection percentages for the F(A), FI(RF) and FI(M) procedures*

Figure 3.5 substantiates our findings based on selection values. In the small sample *NL* setup, *FI*(*M*) performs relatively well, whereas in *NS*, *LL* and *LS* scenarios, the use of *FI*(*RF*) and *FI*(*A*) is preferred.

3.5 *SUMMARY*

The basic question addressed in this chapter concerned the space in which selection should be performed. We found that although selection in input space is sufficient for *NL* data, this is definitely not the case for the other data configurations. It is therefore necessary to utilise information arising from feature space in the selection process.

In this regard we proposed a new approach: so-called feature-to-input space selection. The basic idea underlying this approach is to combine the information obtained from feature space computations with the easy interpretation in input space. On a technical level implementation of such an approach requires calculation of pre-images of quantities in feature space. We discussed this aspect in some detail. This led to the definition of several new *FI* selection criteria. From another angle we also investigated the possibility of selecting variables in input space which best explain the variation in the discriminant function values obtained by applying the kernel algorithm to the input patterns.

The results of an empirical study investigating the properties of the new selection procedures were reported and discussed. Although no clear winner emerged in all cases, the following general conclusions can be made. One of the *FI* criteria, *viz*. *FI*(*RF*), performed very well, with the exception of wide normal data sets. This suggests that the new *FI* approach is promising, deserving of further exploration. As expected, the alignment, being the only criterion computed entirely in feature space, also did well. The other three criteria performed very similar.

CHAPTER 4

ALGORITHM-INDEPENDENT AND ALGORITHM-DEPENDENT SELECTION IN FEATURE SPACE

4.1 *INTRODUCTION*

Different approaches towards variable selection for kernel methods were introduced in Chapter 3. We saw that selection could be performed in input space, using for example a naïve criterion such as the correlations between the response and the input variables; in feature space, using a criterion such as the alignment, or based on feature-to-input space criteria such as pre-images. The empirical study discussed in Chapter 3 showed that the merits of these different approaches depend on the distribution generating the training patterns and the nature of the separation between the groups. For example, for data from normal populations differing with respect to location, it may well be sufficient to restrict attention to selection strategies in input space. This changes, however, when we are faced with *NS*, *LL* or *LS* data setups, when selection in feature space or selection using feature-to-input space concepts seems to be preferable.

In this chapter we therefore present a more thorough discussion of variable selection based on feature space criteria. We start by providing some details regarding the two kernel classification procedures which play a role in the remainder of the thesis: the support vector machine (SVM) in Section 4.2 and kernel Fisher discriminant analysis (KFDA) in Section 4.3. In Section 4.4 we make a distinction between criteria which may be applied in any kernel classification problem (so-called algorithm-independent criteria), and criteria which depend on output from the specific kernel algorithm and which are therefore only applicable in cases where this algorithm is employed. The important role played by the kernel matrix will become clear during this discussion. Section 4.5 is devoted to a fairly

brief discussion of feature space geometry. This enables us to define several variable selection criteria which are suitable for application in any kernel classification context. We define these criteria in Section 4.6.1, and we report and discuss the results of a simulation study investigating the properties of the criteria in Section 4.6.2. Section 4.7 is devoted to the class of algorithm-dependent selection criteria. Definitions of such criteria are given in Section 4.7.1, followed by numerical evaluations of their properties and performance in Section 4.7.2. Section 4.8 contains a summary of the chapter.

4.2 *SUPPORT VECTOR MACHINES*

The introduction of support vector machines in a paper delivered at the 1992 *Annual Workshop on Computational Learning Theory* is widely acknowledged as one of the first contributions to kernel procedures (*cf*. Boser *et al*., 1992). Interestingly, the mathematics which underlies SVM algorithms had already been available since the early 1960s. It took however approximately three decades to bring together all the required mathematical results to produce an SVM methodology which could be applied to real-world problems. This is perhaps understandable, since the SVM requires synthesis of concepts from a wide range of fields: from generalisation theory, regularisation and the control of algorithm complexity, to functional analysis, reproducing kernel Hilbert spaces and optimisation theory. An algorithm which is equivalent to that of SVMs, but which is restricted to fitting hyperplanes in input space, was in fact introduced by Vapnik and Lerner (1963). Other important early contributions include Cover (1965), Duda and Hart (1973), and Anlauf and Biehl (1989). Aronszajn (1950) provided an early discussion of the use of kernel functions, while Aizerman *et al*. (1964) contributed interpretational insight to the use of kernel functions, *viz*. that they provide a way of calculating inner products in a feature space. Cristianini and Shawe-Taylor (2000) is the first comprehensive introduction to SVMs. At this time research on SVMs had reached a maturity which made many view SVMs as a subfield of machine learning and statistics in its own right. Currently, SVMs are still a fast developing area of research, and have also become popular as a powerful tool in practice.

The SVM for classification may be explained and viewed from several different perspectives. For example, an SVM discriminant function is usually written as $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$, where $y_1, y_2, \ldots, y_n$ are the class labels of the training patterns, but we can of course incorporate the labels into the α -parameters, thereby obtaining

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b, \tag{4.1}$$

which is the general form of kernel discriminant functions given in Section 2.2.3. Hence the first perspective of support vector classifiers is that they belong to a more general class of procedures having the form in (4.1), with a specific algorithm for determining values for the parameters from the training data.

Another quite different perspective is provided by regularisation theory. In Section 2.2.4 we saw that different kernel methods may be viewed as solutions to different regularisation problems of the form

$$\min_{f \in \mathcal{H}} \left\{ R(f) = \tfrac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \Omega(f) \right\}, \tag{4.2}$$

where L is a loss function and $\Omega(f)$ denotes the regularisation penalty applied to a function f. If we take L to be the soft margin loss function (see Figure 2.6) and use an L_2 penalty functional, the SVM is obtained as the solution to the optimisation problem in (4.2) (see also Hastie *et al*., 2001, *p*. 380). From this perspective one would tend to conclude that the SVM effectively avoids the danger of overfitting training data in high dimensional spaces. Such a conclusion is however based on the assumption that the regularisation parameter λ is specified 'optimally'. (For a thorough discussion of SVMs in a regularisation framework, the reader is referred to the paper by Evgeniou *et al*., 2000).

Although the regularisation perspective on SVMs is very useful, the original SVM proposal was from an entirely different point of view. In the next three sections we explain the original motivation for SVMs. We start in Section 4.2.1 by considering the case where the training patterns corresponding to the two groups are linearly separable when viewed in input space $\aleph$. It will become clear in this section that the SVM is based on a simple and intuitively acceptable argument for finding a *separating hyperplane*. The basic idea of a separating hyperplane is extended in Section 4.2.2 to the scenario where the training patterns are not linearly separable in input space, but in fact only in a feature space corresponding to some non-linear transformation Φ. In this section the use of a kernel function elegantly solves the problem of computing inner products in a feature space. The most general case is discussed in Section 4.2.3: the training patterns from the two groups are now no longer linearly separable even in feature space.

The concept of a hyperplane plays an important role when discussing SVMs. A hyperplane or affine set is an extension of the concept of a line to higher dimensional spaces. Hyperplanes can be defined in input or in feature space. In input space a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is the set $L_{\aleph}(\boldsymbol{w}, b) = \{\boldsymbol{x} \in \Re^p : \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\}$. A similar definition holds in feature space, with a hyperplane $L_{\Im}(\boldsymbol{w}, b)$ now defined to be the set $L_{\Im}(\boldsymbol{w}, b) = \{\boldsymbol{z} \in \Re^N : \langle \boldsymbol{w}, \boldsymbol{z} \rangle + b = 0\}$. The reader is referred to Appendix B for a more detailed discussion of hyperplanes and their properties.

4.2.1 *THE TRAINING DATA ARE LINEARLY SEPARABLE IN INPUT SPACE*

Reconsider the binary classification problem described in Chapter 1. We observe training data $\mathcal{T} = \{(\boldsymbol{x}_i, y_i), i = 1, 2, \ldots, n\}$ on a response variable $Y \in \{-1, +1\}$ and p classification variables $X_1, X_2, \ldots, X_p$ for $n = n_1 + n_2$ sample cases. Our objective is to use $\mathcal{T}$ to find a discriminant function $f(\boldsymbol{x})$ so that the classifier $sign\{f(\boldsymbol{x})\}$ can be used to assign a new

case with observed values of the classification variables in the vector $\boldsymbol{x}$ to one of the two groups. In this section we assume that the training patterns are linearly separable in input space, *i.e.* that there is at least one hyperplane $L_{\aleph}(\boldsymbol{w},b)$ which perfectly separates the two groups of training patterns. Intuitively this means that all the input patterns to the 'left' of $L_{\aleph}(\boldsymbol{w},b)$ will belong to one of the groups and all the patterns to the 'right' will belong to the second group. Of course, in such scenarios there will typically be many separating hyperplanes for the training data, and this is illustrated in Figure 4.1 for the two-dimensional case. The question to be answered in such cases is: which one of the many separating hyperplanes should we use? We now describe the answer to this question provided by the SVM.

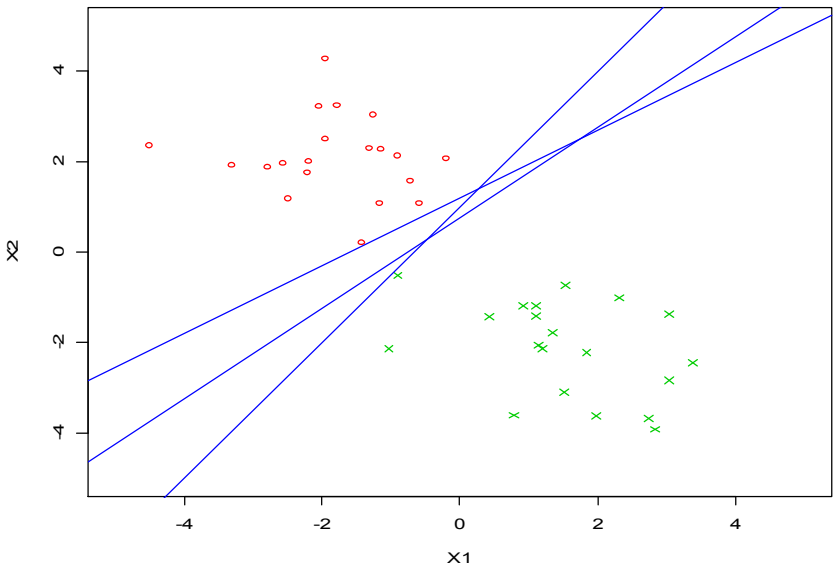


Figure 4.1: *Many separating linear functions in two dimensions*

It is clear that for every separating hyperplane $L_{\aleph}(\boldsymbol{w},b)$ we have $y_i\left(\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b\right) > 0$, $i = 1,2,\ldots,n$. By appropriate rescaling of the hyperplane parameters $\boldsymbol{w}$ and b we can therefore find $\boldsymbol{w}$ and b such that $y_i\left(\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b\right) \geq 1$, $i = 1,2,\ldots,n$, with equality for at least one value of i. This shows that we can restrict attention to hyperplanes for which

$$min\left\{\left|\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b\right|,\ i = 1, 2, \ldots, n\right\} = 1\ . \tag{4.3}$$

A hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ satisfying (4.3) is called a canonical hyperplane for the training patterns $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. Note that if $L_{\aleph}(\boldsymbol{w}, b)$ is a canonical hyperplane for $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, then $L_{\aleph}(-\boldsymbol{w}, -b)$ will also be canonical. Since the unsigned distance from a point $\boldsymbol{x} \in \Re^p$ to a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is given by

$$d_{\boldsymbol{w},b}(\boldsymbol{x}) = \left|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b\right| / \|\boldsymbol{w}\|, \tag{4.4}$$

it follows from (4.3) that the (unsigned) distance from a canonical hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ to the point closest to the hyperplane is simply $1/\|\boldsymbol{w}\|$.

Consideration of the distances between training patterns and a hyperplane leads to the concept of the margin of a data set with respect to a hyperplane. The *geometric margin* of a point $\boldsymbol{x} \in \Re^p$ with respect to a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is simply $d_{\boldsymbol{w},b}(\boldsymbol{x})$, while the geometric margin of the training patterns $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with respect to this hyperplane is defined to be $min\left\{d_{\boldsymbol{w},b}(\boldsymbol{x}_i),\ i = 1, 2, \ldots, n\right\}$. The geometric margin of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with respect to a canonical hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is therefore $1/\|\boldsymbol{w}\|$. (For more details on margins the reader is referred to Appendix B). In the discussion below we will simply refer to the margin, meaning the geometric margin of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with respect to a specified hyperplane, and we will denote this quantity by g. Figure 4.2 illustrates the geometric margin of a canonical hyperplane.

The SVM separating hyperplane for a linearly separable set of training patterns $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ is now defined to be the hyperplane with maximum margin. There are several arguments which can be presented to support the sensibility of such an approach. Consider first the following motivation, based on Schölkopf and Smola, 2002, *pp*. 192-193.

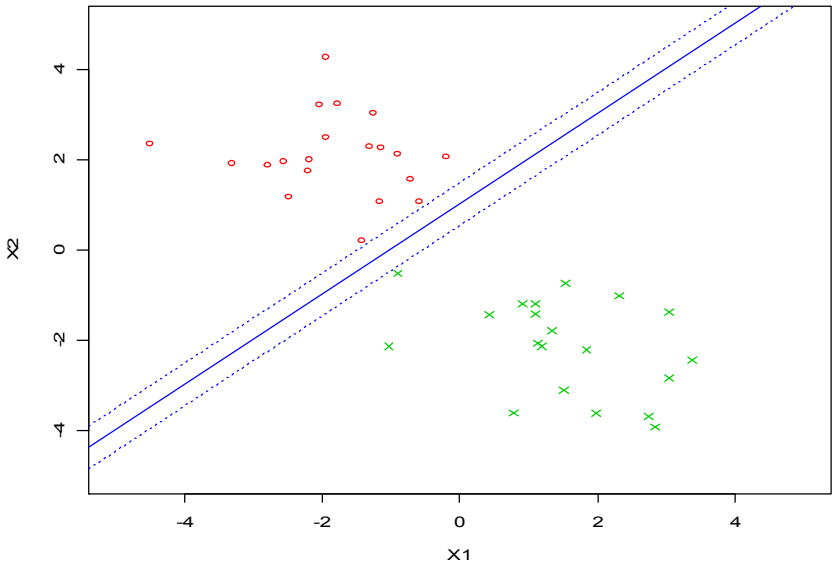


Figure 4.2: *The geometric margin of a canonical separating hyperplane*

It is reasonable to assume that the test and training patterns are generated by the same underlying distribution. More specifically, imagine the test data to be generated according to some distribution in circular regions around each of the training patterns. We would obviously like to classify as many as possible of the test data cases correctly. For test data cases to be classified correctly the circular region around a training pattern into which these test cases fall has to be on the same side of the separating hyperplane as the training pattern itself. The probability of this occurring is increased by maximising the margin of the training patterns with respect to the hyperplane. Figure 4.2 illustrates this argument. Suppose the nearest training case to a hyperplane $L_{\aleph}(\boldsymbol{w},b)$ is $\tilde{\boldsymbol{x}}_i$, with corresponding training label $\tilde{y}_i = +1$. Now consider an unseen pattern $\boldsymbol{x}_i^*$ with $y_i^* = +1$. Clearly, a larger distance between $L_{\aleph}(\boldsymbol{w},b)$ and $\tilde{\boldsymbol{x}}_i$ leaves more room for $\boldsymbol{x}_i^*$ to lie between $L_{\aleph}(\boldsymbol{w},b)$ and $\tilde{\boldsymbol{x}}_i$ and still be classified correctly. In Figure 4.3 the star indicates the unseen input pattern which would not have been classified correctly were the margin any smaller.

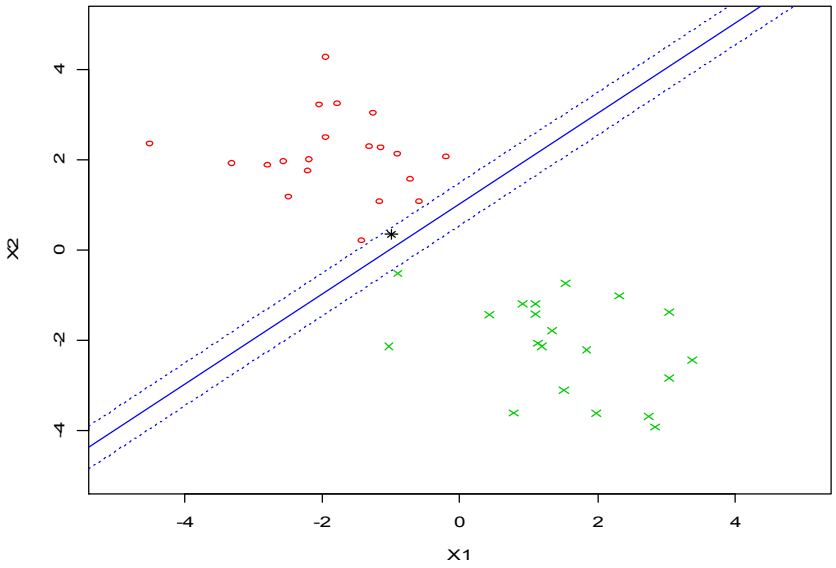


Figure 4.3: *Maximising the geometric margin*

A second more technical argument supporting maximisation of the margin is based on expressions providing upper bounds for the generalisation error of a separating hyperplane. The reader is referred to Schölkopf and Smola (2002, *p*. 194) for an example of such a bound. In essence these bounds, which hold with some prescribed probability calculated with respect to the distribution generating the data, are of the form $Err\left(L_{\aleph}(\boldsymbol{w},b)\right) \leq h(g)$, where $Err\left(L_{\aleph}(\boldsymbol{w},b)\right)$ is the generalisation error of the hyperplane classifier $L_{\aleph}(\boldsymbol{w},b)$, and $h(g)$ is a decreasing function of the margin g. Hence, increasing the margin will decrease the bound $h(g)$, implying that we obtain a classifier with better generalisation error behaviour and thereby supporting maximisation of the margin.

The above discussion explains why the SVM is frequently referred to as a *maximum margin* classifier. Another term frequently used for the hyperplane having maximum margin is the *optimal separating hyperplane* (*OSH*). We see that for the simple case where the training patterns are linearly separable in input space the SVM is equivalent to the OSH classifier.

Having described and motivated the concept underlying the SVM, we now focus on the technical problem of actually finding the OSH, assuming that it exists. Since the margin of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with respect to a canonical hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is simply $1/\|\boldsymbol{w}\|$, we can formulate the following optimisation problem.

PROBLEM 4.1

Consider a linearly separable training data set $\mathcal{T}$. Find the hyperplane $L_{\aleph}(\tilde{\boldsymbol{w}}, \tilde{b})$ which solves the optimisation problem

$$\min_{\boldsymbol{w}, b} \left\{ \tfrac{1}{2} \|\boldsymbol{w}\|^2 \right\} \text{ subject to } y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) \geq 1, \; i = 1, 2, \ldots, n. \tag{4.5}$$

The hyperplane $L_{\aleph}(\tilde{\boldsymbol{w}}, \tilde{b})$ solving Problem 4.1 will be the OSH. Note that we consider $\|\boldsymbol{w}\|^2/2$ rather than $\|\boldsymbol{w}\|$ merely for mathematical convenience. Since the quantity to be minimised in Problem 4.1 is quadratic, and since minimisation takes place under linear constraints, we see that this problem is a convex quadratic optimisation problem having a global minimiser. This is a particularly attractive property of the SVM: we obtain the SVM as the unique solution to an optimisation problem, without having to worry about the possibility of getting stuck at a local minimum. In order to solve Problem 4.1 we import a Lagrange multiplier for each restriction. This enables us to write down an expression for the *primal* Lagrangian of the problem:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \tfrac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) - 1 \right], \tag{4.6}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the non-negative Lagrange multipliers. The Lagrangian in (4.6) has to be minimised with respect to $\boldsymbol{w}$ and b, and maximised with respect to

$\alpha_1, \alpha_2, \ldots, \alpha_n$. In optimisation theory it is well known (see for example Rockafellar, 1970) that it is easier to solve the so-called *dual Lagrangian* formulation of Problem 4.1, which is obtained by differentiating $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{w}$ and b, setting the derivatives equal to zero and substituting the resulting equations back into the primal Lagrangian, $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$. In our case the resulting equations are

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i = \boldsymbol{0}, \text{ and} \tag{4.7}$$

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0. \tag{4.8}$$

Substituting $\boldsymbol{w} = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i$ and $\sum_{i=1}^{n} y_i \alpha_i = 0$ back into the primal Lagrangian in (4.6) yields the dual form of the optimisation problem:

PROBLEM 4.2

$$\max_{\boldsymbol{\alpha}} \left\{ W(\boldsymbol{\alpha}) = \tfrac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle \boldsymbol{x}_i, \boldsymbol{x}_j \right\rangle - \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle \boldsymbol{x}_i, \boldsymbol{x}_j \right\rangle + \sum_{i=1}^{n} \alpha_i \right\} \text{ or}$$

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle \boldsymbol{x}_i, \boldsymbol{x}_j \right\rangle \right\} \tag{4.9}$$

$$\text{subject to } \sum_{i=1}^{n} y_i \alpha_i = 0 \text{ and } \alpha_i \geq 0,\ i = 1,2,\ldots,n.$$

This is a quadratic programming problem and it can be solved by using standard software. Denote the maximising $\boldsymbol{\alpha}$ by $\tilde{\boldsymbol{\alpha}}$, then the optimal weight vector $\tilde{\boldsymbol{w}}$ can easily be obtained from (4.7), *viz.* $\tilde{\boldsymbol{w}} = \sum_{i=1}^{n} y_i \tilde{\alpha}_i \boldsymbol{x}_i$. This provides us with a computable expression for the weight vector of the OSH.

There are various ways of determining a value for the intercept $\widetilde{b}$ of the maximum margin classifier. The most commonly used method is based on the so-called Karush-Kuhn-Tucker (KKT) optimality conditions, *viz*.

$$\alpha_i\left[y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b\right) - 1\right] = 0, \quad i = 1, 2, \ldots, n. \tag{4.10}$$

It follows from the theory of quadratic programming that the KKT conditions are satisfied at the optimal solution. This implies that whenever $\widetilde{\alpha}_i > 0$, $y_i\left(\langle \widetilde{\boldsymbol{w}}, \boldsymbol{x}_i\rangle + \widetilde{b}\right) = 1$ from which it follows that $\widetilde{b} = y_i - \langle \widetilde{\boldsymbol{w}}, \boldsymbol{x}_i\rangle$ (since in a dichotomous classification setup $y_i = \pm 1$). For the sake of increased numerical stability it is common practice to average $\widetilde{b} = y_i - \langle \widetilde{\boldsymbol{w}}, \boldsymbol{x}_i\rangle$ over all training patterns for which $\widetilde{\alpha}_i > 0$. In situations where the two classes overlap, and especially also in cases where the sample number of observations from the two classes are unbalanced, contributions in the literature have shown that the above method for determining $\tilde{b}$ can be improved to yield smaller generalisation errors. Alternatively, one could use the value $\tilde{b}$ causing $err(f)$ to be a minimum.

How do we use the OSH for classification of new cases? The OSH is the set $L_{\lesssim}\left(\widetilde{\boldsymbol{w}}, \widetilde{b}\right) = \left\{\boldsymbol{x} \in \Re^p : \langle \widetilde{\boldsymbol{w}}, \boldsymbol{x}\rangle + \widetilde{b} = 0\right\}$, and this defines the decision boundary between the two groups. We classify a new case with training pattern $\boldsymbol{x}$ into one of the two groups by using the classifier

$$sign\{f(\boldsymbol{x})\} = sign\left\{\sum_{i=1}^{n} \widetilde{\alpha}_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x}\rangle + \widetilde{b}\right\}. \tag{4.11}$$

Note that in terms of the training patterns computation of (4.11) only entails evaluation of an inner product. The classifier in (4.11) is the simplest example of a support vector classifier: because the discriminant function $\sum_{i=1}^{n} \widetilde{\alpha}_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x}\rangle + \widetilde{b}$ is a linear function of $\boldsymbol{x}$, it is also known as a linear support vector classifier.

Before moving on to a discussion of the case where the two groups are linearly separable only in feature space, note from (4.7) that the weight vector of the OSH is a linear function of the training patterns, $\boldsymbol{x}_i$, $i = 1,2,\ldots,n$. We also see from (4.8) that the sum of the Lagrange coefficients α_i corresponding to group 1 is equal to the sum of these quantities corresponding to group 2. Perhaps most importantly, Equation (4.10) shows that $\tilde{\alpha}_i > 0$ only if $y_i\left(\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle + \tilde{b}\right) = 1$. That is, only the training patterns closest to the hyperplane (realising the minimum canonical margin value of 1) can have non-zero Lagrange multipliers. These training patterns, referred to as *support vectors* or *support points*, typically form a fairly small fraction of the total number of input patterns in the training data. It follows that we can calculate the OSH weight vector from $\tilde{\boldsymbol{w}} = \sum_{i=1}^{n} y_i \tilde{\alpha}_i \boldsymbol{x}_i$ by summing only over the (relatively small number of) support vectors – the support vectors contain all the information required to compute $\tilde{\boldsymbol{w}}$. This *sparseness* property of SVMs contributes to their popularity: once having trained an SVM, it is often quite fast to use.

4.2.2 *THE TRAINING DATA ARE LINEARLY SEPARABLE IN FEATURE SPACE*

In this section we extend the formulation of a linear support vector classifier to the case where the training patterns from the two groups are not linearly separable in input space, *i.e*. the case where the support vector classifier turns out to be a non-linear decision function in input space. This is the context in which the use of the kernel trick (Aronszajn, 1950) was originally modernised, and consequently also the context in which SVMs were proposed by Boser *et al*. (1992). Recall from Chapter 2 that any algorithm in which the training patterns appear only in the form of inner products can easily be extended to an algorithm which is linear in feature space, and typically non-linear in input space, by substituting a kernel function for the inner product. Such a modification of the linear support vector classifier in (4.11) is straightforward: we simply replace the inner product

$\langle \boldsymbol{x}_i, \boldsymbol{x} \rangle$ by an appropriate kernel function $k(.,.)$ evaluated on the pair $(\boldsymbol{x}_i, \boldsymbol{x})$. This yields the general form of the support vector classifier, *viz*.

$$sign\{f(\boldsymbol{x})\} = sign\left\{\sum_{i=1}^{n} \tilde{\alpha}_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \tilde{b}\right\} \quad . \tag{4.12}$$

This support vector classifier corresponds to the hyperplane which we obtain by solving the following optimisation problem.

PROBLEM 4.3

$$\max_{\boldsymbol{\alpha}}\left\{W(\boldsymbol{\alpha}) = \tfrac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i\right\} \text{ or}$$

$$\max_{\boldsymbol{\alpha}}\left\{\sum_{i=1}^{n} \alpha_i - \tfrac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle\right\} \tag{4.13}$$

$$\text{subject to } \sum_{i=1}^{n} y_i \alpha_i = 0 \text{ and } \alpha_i \geq 0,\ i = 1,2,\ldots,n\,.$$

Implicit to the above formulation is the concept of transforming the training patterns originally observed in an input space $\aleph$ to a higher dimensional feature space $\Im$, and then finding an optimal separating hyperplane in $\Im$. The final classifier (4.12) is therefore a linear function in $\Im$, but typically highly non-linear in $\aleph$.

The weight vector of the separating hyperplane in $\Im$ has the same form as in the linear case – the original input patterns are merely replaced by their feature space counterparts, *i.e*. we have $\tilde{\boldsymbol{w}} = \sum_{i=1}^{n} y_i \tilde{\alpha}_i \Phi(\boldsymbol{x}_i)$, where Φ introduces the non-linearity of $\tilde{\boldsymbol{w}}$ with respect to the observed input patterns. As pointed out in Chapter 2, $\Phi(\boldsymbol{x}_i)$ is usually unobtainable, since for example the associated feature space $\Im$ may have infinite dimension, or Φ may

be unknown or difficult to determine. Fortunately this does not prevent us from computing the resulting support vector classifier. For a new case with training pattern $\boldsymbol{x}$ we have

$$\begin{aligned} sign\{f(\boldsymbol{x})\} &= sign\left\{\langle \tilde{\boldsymbol{w}}, \Phi(\boldsymbol{x})\rangle + \tilde{b}\right\} \\ &= sign\left\{\left\langle \sum_{i=1}^{n} \tilde{\alpha}_i y_i \Phi(\boldsymbol{x}_i),\ \Phi(\boldsymbol{x})\right\rangle + \tilde{b}\right\} \\ &= sign\left\{\sum_{i=1}^{n} \tilde{\alpha}_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \tilde{b}\right\} \end{aligned} \tag{4.14}$$

Note that the intercept $\tilde{b}$ is calculated as in the previous section; we only replace the inner product by a kernel function.

4.2.3 *HANDLING NOISY DATA*

The previous two sections were devoted to a discussion of the support vector classifier when the training patterns are linearly separable: in Section 4.2.1 we assumed the patterns to be linearly separable in input space, and in Section 4.2.2, in feature space. In practice we frequently encounter scenarios where the training patterns are not linearly separable even in feature space. Moreover, if one cannot assume the data to be free of noise, care should be taken against possible overfit if a support vector classifier with a very small training error is used. In such cases we have to accommodate training patterns falling on the 'wrong' side of the hyperplane decision boundary. Vapnik and Cortes (1995) presented an algorithm for fitting hyperplanes in such situations. The method introduces a vector $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_n]'$ to the optimisation problems discussed thus far, enabling us to allow for training patterns to violate the optimisation constraints $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1,\ i = 1,2,\ldots,n$. In fact, the constraints are relaxed to

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1 - \xi_i,\ i = 1,2,\ldots,n, \tag{4.15}$$

where $\xi_1, \xi_2, \ldots, \xi_n$ are called *slack variables* and we assume $\xi_i \geq 0$. The slack variable ξ_i measures the degree to which training input pattern $\Phi(\boldsymbol{x}_i)$ violates the former optimisation constraint, *i.e.* the extent to which $\Phi(\boldsymbol{x}_i)$ falls on the 'wrong' side of the hyperplane decision boundary in feature space. Since we would obviously not like to have transgressions which are too large, we have to incorporate the ξ_i into the objective function as well. This can be done in different ways, a common approach being to modify the objective function in Problem 4.1 to $\min_{\boldsymbol{w},b}\left\{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\,h\left(\sum_{i=1}^{n}\xi_i\right)\right\}$. In this expression C is a non-negative *cost parameter*, controlling the trade-off between maximising the margin and minimising the penalty associated with training patterns on the wrong side of the decision boundary. Also, $h(.)$ is a monotonically increasing function on $\Re^+$. Very often we take $h(x) = x$, so that the basic optimisation problem which has to be solved can be formulated as follows.

PROBLEM 4.4

$$\min_{\boldsymbol{w},b}\left\{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right\}$$

subject to $y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, \ldots, n.$

It is possible to show that the solution to Problem 4.4 is once again of the form $\tilde{\boldsymbol{w}} = \sum_{i=1}^{n} y_i \tilde{\alpha}_i \boldsymbol{x}_i$, with only a subset of $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_n$ being positive. In fact, the positive $\tilde{\alpha}_i$ correspond to the data cases where the constraint $y_i\left(\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i\rangle + b\right) \geq 1 - \xi_i$ is exactly met, and these points are once again called support vectors. It can also be shown that we can find $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_n$ by solving the following quadratic optimisation problem.

PROBLEM 4.5

$$\max_{\boldsymbol{\alpha}}\left\{W(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\tfrac{1}{2}\sum_{i,j=1}^{n}y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j)\right\}$$

subject to $\sum_{i=1}^{n} y_i \alpha_i = 0$ and $0 \le \alpha_i \le C$, $i = 1,2,\ldots,n$.

It is interesting to observe that the dual objective function in Problem 4.5 is exactly the same as that in Problem 4.3, Equation (4.13). The only difference between Problem 4.3 and Problem 4.5 is a change in the restriction placed on the α_i's: the restriction $\alpha_i \ge 0$ in Problem 4.3 is replaced by the restriction $0 \le \alpha_i \le C$, $i = 1,2,\ldots,n$, in Problem 4.5. Determining a value for the intercept is also analogous to the previous cases: a commonly used option is to determine $\tilde{b}$ as the average of $b = y_j - \sum_{i=1}^{n} y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ over all cases satisfying $0 < \alpha_j < C$.

Specification of the cost parameter, C, is an important aspect. If C is too large, it leads to a decision boundary which follows the training data too closely, allowing too few training errors, and possibly yielding an overfit of the training data. Specifying the value of C too small implies that the term $\|\boldsymbol{w}\|^2/2$ in the objective function is dominant, which can easily lead to underfitting.

4.3 *KERNEL FISHER DISCRIMINANT ANALYSIS*

In this section we turn attention to a second procedure for classification in feature space, *viz*. kernel Fisher discriminant analysis. KFDA was first proposed in Mika *et al*. (1999) as a direct generalisation of Fisher's well known linear discriminant analysis. As such it seems that from a statistical perspective KFDA is an intuitively simpler technique than support vector machines. Although perhaps less well known than support vector machines, the performance in terms of generalisation error of KFDA compares well to that of SVMs (*cf*. Mika et al., 1999). An advantage offered by KFDA compared to SVMs is that the former provides a natural option for calculating posterior probabilities of group membership (*cf*. Schölkopf and Smola, 2002, *p*. 464).

Since KFDA is a kernel classification procedure, kernel Fisher discriminant functions exhibit all the characteristics of this class of discriminant functions. In particular, the discriminant function in KFDA is linear in feature space, but typically highly non-linear in input space. As we will see, KFDA basically entails the well known transformation of training patterns $\boldsymbol{x}_i \in \aleph$ to a feature space $\Im$, followed by application of linear discriminant analysis in $\Im$.

We start our discussion of KFDA in Section 4.3.1 with a brief revision of LDA. We then show in Section 4.3.2 how the LDA algorithm can be generalised to feature space to obtain the kernel Fisher discriminant function. Several further aspects of KFDA are also discussed in this section. Parts of our discussion are based on Louw and Steel (2006).

4.3.1 *LINEAR DISCRIMINANT ANALYSIS*

Linear discriminant analysis is one of the best known and most frequently used statistical procedures for solving classification problems, being especially successful in cases where the training patterns arise from (approximate) normal distributions. In such scenarios it can for example be derived by applying Bayes' theorem, provided we also assume equal

variance-covariance matrices in the two groups. Another possibility for deriving the LDA classifier is to seek the direction which corresponds to the linear projection providing the best possible separation between the two groups. Group separation is measured in terms of two aspects: firstly, in terms of the distances between the two projected group means (this should be as large as possible), and secondly, in terms of the variance of the data within a specific group (which should be as small as possible). From this perspective the linear discriminant function is defined to be $sign\{u_0 + \langle \tilde{\boldsymbol{u}}, \boldsymbol{x} \rangle\}$, where $\tilde{\boldsymbol{u}}$ maximises the ratio

$$J(\boldsymbol{u}) = \frac{\boldsymbol{u}'\boldsymbol{B}\boldsymbol{u}}{\boldsymbol{u}'\boldsymbol{W}\boldsymbol{u}} = \frac{\langle \boldsymbol{B}\boldsymbol{u}, \boldsymbol{u} \rangle}{\langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{u} \rangle}. \tag{4.16}$$

In (4.16), $\boldsymbol{B} = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'$ is the between group scatter matrix, while $\boldsymbol{W}$ is the pooled sample covariance matrix for the two groups defined by the expression $(n_1 + n_2 - 2)\boldsymbol{W} = \sum_{i \in I_1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_1)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_1)' + \sum_{i \in I_2} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_2)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_2)'$, with $\bar{\boldsymbol{x}}_1$ and $\bar{\boldsymbol{x}}_2$ the respective group sample mean vectors. The quantity $J(\boldsymbol{u})$ in (4.16) is often referred to as the *Rayleigh quotient*. Note also that $\boldsymbol{W}$ is typically assumed to be a non-singular matrix.

How do we find $\tilde{\boldsymbol{u}}$ maximising the ratio in (4.16)? A well known extension of the Cauchy-Schwarz inequality implies that $\tilde{\boldsymbol{u}}$ is proportional to $\boldsymbol{W}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$, *i.e.* $\tilde{\boldsymbol{u}} = c\boldsymbol{W}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$ for some constant c. Taking this constant equal to 1 we find that $\tilde{\boldsymbol{u}} = \boldsymbol{W}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$. The intercept traditionally used in LDA is given by $u_0 = \frac{1}{2}\left(\bar{\boldsymbol{x}}_2'\boldsymbol{W}^{-1}\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1'\boldsymbol{W}^{-1}\bar{\boldsymbol{x}}_1\right) + \log(n_1/n_2)$, and the LDA rule to classify a new case with training pattern $\boldsymbol{x}$ therefore becomes

$$sign\left\{\tfrac{1}{2}\left(\bar{\boldsymbol{x}}_2'\boldsymbol{W}^{-1}\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1'\boldsymbol{W}^{-1}\bar{\boldsymbol{x}}_1\right) + \log(n_1/n_2) + \left\langle \boldsymbol{W}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2), \boldsymbol{x} \right\rangle\right\}. \tag{4.17}$$

Obviously the linear classifier in (4.17) will not be adequate in all cases. Several attempts have therefore been made to broaden the applicability of LDA, leading to extensions of the basic technique such as quadratic discriminant analysis (QDA), flexible discriminant analysis, penalised discriminant analysis, and regularised discriminant analysis. A quadratic discriminant function is obtained when the assumption of equal variance-covariance matrices for the two groups is relaxed. This leads to a significant increase in the number of parameters which have to be estimated: whereas LDA in a binary classification problem requires estimation of $p+1$ parameters, quadratic discriminant analysis requires $p(p+3)/2+1$ parameters. For large values of p this represents a substantial increase, which may lead to worse than expected classification performance in the quadratic case. In an attempt to gain the advantages offered by the greater flexibility of quadratic discriminant analysis while retaining at least some of the simplicity of LDA, Friedman (1989) proposed an approach known as regularised discriminant analysis. For a discussion of this and other generalisations of LDA, the interested reader is referred to Hastie *et al.*, 2001.

LDA and QDA perform well in many practical applications. As stated in Hastie *et al.* (2001), the reason for this is unlikely to be that the assumptions of equal variance-covariance matrices (in the linear case) and multivariate normal distributions (in the linear and the quadratic cases) hold. A more likely explanation is to be found in the relative simplicity of these procedures, requiring relatively few parameters to be estimated (especially in the linear case). Sometimes this is all that can reasonably be achieved on small data sets.

There are however many examples of data sets where more complex non-linear functions are required to provide adequate separation between cases belonging to different groups. In order to provide techniques for such scenarios without having to relinquish the positive attributes of a linear classifier, Mika (2002) extends LDA to a classification technique operating linearly in feature space, called kernel Fisher discriminant analysis, which we now discuss.

4.3.2 *THE KERNEL FISHER DISCRIMINANT FUNCTION*

It was indicated in Chapter 2 that the kernel Fisher discriminant function is of the form $sign\left\{\left\langle \Phi(\boldsymbol{x}), \sum_{i=1}^{n}\alpha_i\Phi(\boldsymbol{x}_i)\right\rangle + b\right\} = sign\left\{\sum_{i=1}^{n}\alpha_i k(\boldsymbol{x},\boldsymbol{x}_i)+b\right\}$, where $\alpha_1,\alpha_2,\ldots,\alpha_n$ and b are determined from the training data by applying the KFDA algorithm. In this section we provide more detail on this process.

Since KFDA is a direct extension to feature space of LDA, the KFDA algorithm requires one to maximise a Rayleigh quotient in $\Im$. Hence, let $\boldsymbol{B}_\Phi = (\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2)(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2)'$ denote the between group scatter matrix in feature space, and similarly, let $(n_1+n_2-2)\boldsymbol{W}_\Phi = \sum_{i\in I1}(\varphi_i - \overline{\boldsymbol{\Phi}}_1)(\varphi_i - \overline{\boldsymbol{\Phi}}_1)' + \sum_{i\in I2}(\varphi_i - \overline{\boldsymbol{\Phi}}_2)(\varphi_i - \overline{\boldsymbol{\Phi}}_2)'$ denote the within group scatter matrix in $\Im$. In the KFDA algorithm one needs to solve the following optimisation problem:

$$\max_{\boldsymbol{v}\in\Im}\left\{J(\boldsymbol{v}) = \frac{\boldsymbol{v}'\boldsymbol{B}_\Phi\boldsymbol{v}}{\boldsymbol{v}'\boldsymbol{W}_\Phi\boldsymbol{v}} = \frac{\langle \boldsymbol{B}_\Phi\boldsymbol{v},\boldsymbol{v}\rangle}{\langle \boldsymbol{W}_\Phi\boldsymbol{v},\boldsymbol{v}\rangle}\right\}. \tag{4.18}$$

In principle, if we could assume $\boldsymbol{W}_\Phi$ to be non-singular we could, similar to the LDA solution $\tilde{\boldsymbol{u}} = \boldsymbol{W}^{-1}(\overline{\boldsymbol{x}}_1 - \overline{\boldsymbol{x}}_2)$, find $\tilde{\boldsymbol{v}} = \boldsymbol{W}_\Phi^{-1}(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2)$ in feature space and construct the linear discriminant rule

$$sign\{v_0 + \langle \tilde{\boldsymbol{v}}, \boldsymbol{\varphi}\rangle\}. \tag{4.19}$$

However, in many cases $\boldsymbol{B}_\Phi$ and $\boldsymbol{W}_\Phi$ have infinite dimension and are therefore not directly obtainable. Solving the optimisation problem in (4.18) is therefore impracticable, and we require an alternative formulation.

For this purpose, consider the linear space $\mathcal{D}$ spanned by $\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n$, and an arbitrary vector in feature space, denoted in this section by $\boldsymbol{z}$. Now $\boldsymbol{z}$ can be decomposed into $\boldsymbol{z} = \boldsymbol{z}_1 + \boldsymbol{z}_2$, with $\boldsymbol{z}_1 \in \mathcal{D}$ and $\boldsymbol{z}_2 \in \mathcal{D}^{\perp}$, and where $\mathcal{D}^{\perp}$ denotes the orthogonal complement of $\mathcal{D}$, so that $\langle \boldsymbol{A}\boldsymbol{z}, \boldsymbol{z} \rangle = \langle \boldsymbol{A}\boldsymbol{z}_1, \boldsymbol{z}_1 \rangle$ for any symmetric matrix $\boldsymbol{A}$ whose rows (or columns) belong to $\mathcal{D}$. Since it can be shown that the rows of both $\boldsymbol{B}_\Phi$ and $\boldsymbol{W}_\Phi$ belong to $\mathcal{D}$, we can write $\langle \boldsymbol{B}_\Phi \boldsymbol{z}, \boldsymbol{z} \rangle = \langle \boldsymbol{B}_\Phi \boldsymbol{z}_1, \boldsymbol{z}_1 \rangle$ and $\langle \boldsymbol{W}_\Phi \boldsymbol{z}, \boldsymbol{z} \rangle = \langle \boldsymbol{W}_\Phi \boldsymbol{z}_1, \boldsymbol{z}_1 \rangle$ for any $\boldsymbol{z}$ in $\mathfrak{I}$. Substituting these expressions into (4.18), we see that our optimisation problem is equivalent to maximising $J(\boldsymbol{z}_1) = \dfrac{\boldsymbol{z}_1' \boldsymbol{B}_\Phi \boldsymbol{z}_1}{\boldsymbol{z}_1' \boldsymbol{W}_\Phi \boldsymbol{z}_1} = \dfrac{\langle \boldsymbol{B}_\Phi \boldsymbol{z}_1, \boldsymbol{z}_1 \rangle}{\langle \boldsymbol{W}_\Phi \boldsymbol{z}_1, \boldsymbol{z}_1 \rangle}$ over $\boldsymbol{z}_1 \in \mathcal{D}$, and it is therefore sufficient to restrict attention to vectors $\boldsymbol{z}$ belonging to $\mathcal{D}$, *i.e.* vectors which can be written in the form $\boldsymbol{z} = \sum_{i=1}^{n} \alpha_i \boldsymbol{\varphi}_i = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$ for given scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$. Defining the n elements of $\boldsymbol{m}_j$ by $\dfrac{1}{n_j} \sum_{k=1}^{n_j} k(\boldsymbol{x}_i, \boldsymbol{x}_k)$, $j = 1, 2$, if we let $\boldsymbol{\alpha}$ be the vector containing $\alpha_1, \alpha_2, \ldots, \alpha_n$, we may therefore write

$$\langle \boldsymbol{z}_j, \overline{\boldsymbol{\Phi}}_j \rangle = \frac{1}{n_j} \sum_{i=1}^{n} \sum_{k=1}^{n_j} \alpha_i \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_k) \rangle = \frac{1}{n_j} \sum_{i=1}^{n} \sum_{k=1}^{n_j} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_k) = \langle \boldsymbol{\alpha}, \boldsymbol{m}_j \rangle, \; j = 1, 2 .$$

If we now define $\boldsymbol{M} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)'$, it readily follows that the numerator of $J(\boldsymbol{v})$ in (4.18) is given by $\boldsymbol{z}' \boldsymbol{B}_\Phi \boldsymbol{z} = \boldsymbol{\alpha}' \boldsymbol{M} \boldsymbol{\alpha}$. One can also obtain the denominator of (4.18) in a tractable form. Define an $n \times n_1$ kernel matrix $\boldsymbol{K}_1$ for class 1 to be the matrix with elements $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n_1$, and write $\boldsymbol{1}_{n_1}$ for the $n_1 \times n_1$ matrix with all elements equal to $1/n_1$. Define $\boldsymbol{K}_2$ and $\boldsymbol{1}_{n_2}$ similarly. If we now set $\boldsymbol{N} = \sum_{j=1}^{2} \boldsymbol{K}_j (\boldsymbol{I} - \boldsymbol{1}_{n_j}) \boldsymbol{K}_j'$, it follows that the denominator of $J(\boldsymbol{v})$ in (4.18) can be written as $\boldsymbol{z} \boldsymbol{W}_\Phi \boldsymbol{z} = \boldsymbol{\alpha}' \boldsymbol{N} \boldsymbol{\alpha}$. We conclude that maximising (4.18) is equivalent to maximising the ratio

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}'\boldsymbol{M}\boldsymbol{\alpha}}{\boldsymbol{\alpha}'\boldsymbol{N}\boldsymbol{\alpha}} = \frac{\langle \boldsymbol{M}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle}{\langle \boldsymbol{N}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle}. \tag{4.20}$$

While solving (4.18) will generally be impossible (because maximisation has to be carried out over possibly infinite dimensional vectors $\boldsymbol{z}$ in feature space), maximisation of (4.20) is in this regard simple, since it only involves vectors $\boldsymbol{\alpha} \in \Re^n$. Now suppose $\tilde{\boldsymbol{\alpha}}$ maximises (4.20), and let $\tilde{\boldsymbol{z}} = \sum_{i=1}^{n} \tilde{\alpha}_i \Phi(\boldsymbol{x}_i)$. Since $\langle \boldsymbol{z}, \boldsymbol{\varphi} \rangle = \sum_{i=1}^{n} \tilde{\alpha}_i \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) \rangle = \sum_{i=1}^{n} \tilde{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x})$, the KFDA classification rule in feature space becomes

$$sign\left\{b + \langle \tilde{\boldsymbol{u}}, \boldsymbol{z} \rangle\right\} = sign\left\{\tilde{b} + \sum_{i=1}^{n} \tilde{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x})\right\}, \tag{4.21}$$

which is a practically useful result once the coefficients $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_n$ and the intercept $\tilde{b}$ have been determined. We will refer to (4.21) as the KFDA classification rule. Note how application of the kernel trick once again obviates explicit specification or use of the feature mapping Φ.

The above approach does however leave us with one problem: the matrix $\boldsymbol{N}$ is singular and consequently we cannot find the maximising $\tilde{\boldsymbol{\alpha}}$ by simply calculating $\tilde{\boldsymbol{\alpha}} = \boldsymbol{N}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$. Mika *et al*. (1999) propose and motivate the use of regularisation to overcome this difficulty. In the present context regularisation entails replacing $\boldsymbol{N}$ by a matrix $\boldsymbol{N}_\lambda = \boldsymbol{N} + \lambda \boldsymbol{I}$, for some (small) positive scalar λ. This yields a solution $\tilde{\boldsymbol{\alpha}} = \boldsymbol{N}_\lambda^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$, depending on λ, which can be used in (4.21).

The intercept $\tilde{b}$ in (4.21) can be specified in different ways. Müller *et al*. (2001) mention the mean of the average projections of the two groups. A popular choice is $\tilde{b} = \frac{1}{2}\left(\boldsymbol{m}_2' \boldsymbol{N}_\lambda^{-1} \boldsymbol{m}_2 - \boldsymbol{m}_1' \boldsymbol{N}_\lambda^{-1} \boldsymbol{m}_1\right) + \log(n_1/n_2)$, which shows a clear similarity to the intercept used in LDA.

There are also other approaches which may be used to derive the KFDA classification rule. Mika *et al*. (2001) show how this can be done by solving a convex quadratic optimisation problem, while Van Gestel *et al*. (2002) exploit a link between least squares support vector machines and LDA in feature space. Finally, Shawe-Taylor and Cristianini (2004) present an argument which enables one to find the KFDA classification function by solving a ridge like problem in feature space. In our empirical work we used the approach which was presented above, based on regularising the matrix $\boldsymbol{N}$.

4.4 *ALGORITHM-INDEPENDENT VERSUS ALGORITHM-DEPENDENT SELECTION*

In this section we discuss the difference between variable selection criteria which may be applied in any kernel classification problem (so-called algorithm-independent criteria), and those that depend on the output of a specific kernel classification algorithm and which are therefore only applicable when this specific algorithm is used (so-called algorithm-dependent criteria). The following definition is relevant in this regard.

DEFINITION 4.1: ***ALGORITHM-INDEPENDENT AND -DEPENDENT CRITERIA***

An input variable selection criterion which can be used for any kernel classifier, for example the SVM as well as the kernel Fisher discriminant classifier, is called *algorithm-independent*, while a criterion which depends on the output from a specific kernel classification algorithm is called *algorithm-dependent*.

This distinction is now discussed in greater detail. We saw in Chapter 2, and again in the discussion of the SVM in Section 4.2 and KFDA in Section 4.3, that the general form of a kernel classifier is given by

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b, \tag{4.22}$$

where $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i) \in \Im$, and where $\alpha_1, \alpha_2, \ldots, \alpha_n$ and b are output from the particular kernel learning algorithm. Although different kernel classifiers are based on different algorithms to calculate $\alpha_1, \alpha_2, \ldots, \alpha_n$ and b, all of these algorithms use the information provided by the training patterns via the output from a kernel function. We can therefore view an algorithm-independent criterion as a criterion which depends only on the information provided by the kernel function applied to all pairs of training patterns, together with the training labels. For an algorithm-dependent criterion there is additional dependence on the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$. Also, in selection based on an algorithm-specific criterion the kernel algorithm used in derivation of the criterion and the post-selection algorithm should be the same. In contrast, algorithm independent criteria may be applied as part of the pre-processing step to any kernel analysis.

It should be clear that the kernel matrix plays an important role in both algorithm-independent and algorithm-dependent criteria. The kernel matrix $\boldsymbol{K}$ is the $n \times n$ symmetric matrix with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j = 1, 2, \ldots, n$. An algorithm-independent selection criterion uses properties of $\boldsymbol{K}$ and patterns in its entries to select a subset of the available input variables. For example, if we keep in mind that k_{ij} may be viewed as a measure of the similarity between training patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, it seems reasonable to postulate the following 'ideal' structure for $\boldsymbol{K}$: in the top-left $n_1 \times n_1$ sub-matrix, as well as the bottom-right $n_2 \times n_2$ sub-matrix, we would like to observe entries reflecting a high degree of similarity between training patterns, while in the two off-diagonal sub-matrices a high degree of dissimilarity should ideally be reflected. Such considerations imply that we can perform variable selection to end up with a matrix $\boldsymbol{K}$ which is in some sense as 'close as possible' to this 'ideal' structure. In our further discussions we will use the notation $\boldsymbol{K}_I$ to denote a kernel matrix exhibiting such an 'ideal' structure.

Another aspect regarding use of $\boldsymbol{K}$ in variable selection is that $\boldsymbol{K}$ obviously has to be re-computed every time a new set of input variables is considered. In this respect using the Gaussian kernel offers somewhat of a computational advantage. We require some notation. Let $\boldsymbol{K}_j$ be the kernel matrix based on variable X_j only, $j = 1, 2, \ldots, p$, *i.e.* $\boldsymbol{K}_j$ has elements $exp\{-\gamma(x_{ij} - x_{kj})\}$, $i, k = 1, 2, \ldots, n;\ j = 1, 2, \ldots, p$. Then the kernel matrix corresponding to a subset variable subset V with indices in $J \subset \mathcal{J} = \{1, 2, \ldots, p\}$ can be formed by element-wise multiplication of the elements of the kernel matrices $\boldsymbol{K}_j$, $j \in J$. Having once computed and stored the single variable kernel matrices $\boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_p$ we are therefore able to subsequently compute the kernel matrix corresponding to any subset of variables without the need for re-computing exponentials.

Finally, an exhaustive all possible subsets approach to variable selection requires a kernel matrix to be computed for every possible subset of variables. Even using the computational savings for the Gaussian kernel referred to in the previous paragraph, this is a daunting computational problem except in cases where p is very small. It is therefore no surprise that stepwise variable selection approaches are popular options in practical applications. An important example of a stepwise elimination approach, *viz.* recursive feature elimination, is discussed in Chapter 5.

4.5 *FEATURE SPACE GEOMETRY*

In this section we examine data properties and operations in feature space more closely. We start in Section 4.5.1 with several results pertaining to individual points in $\Im$. This is followed in Section 4.5.2 with results applicable to sets of feature vectors and linear combinations in feature space. We consistently emphasise the form of the results when a Gaussian kernel function is used.

4.5.1 *INDIVIDUAL POINTS*

Consider a typical feature vector denoted by $\Phi(\boldsymbol{x})$ or $\Phi(\boldsymbol{z})$, $\boldsymbol{x}, \boldsymbol{z} \in \aleph$, or simply by $\boldsymbol{\varphi}$. The following result summarises some useful operations on one (or between two) feature vector(s).

RESULT 4.1

i. Inner products

It has already been pointed out that $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$. In the case of the Gaussian kernel this is also the cosine of the angle between $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{z})$ (since $\|\Phi(\boldsymbol{x})\| = \sqrt{k(\boldsymbol{x}, \boldsymbol{x})} = 1$ – see point *ii.* below). Thus, when one uses a Gaussian kernel, inner products in feature space will always lie in the interval $[0,1]$. Also, $k(\boldsymbol{x}, \boldsymbol{z})$ is large when the enclosed angle between $\boldsymbol{x}$ and $\boldsymbol{z}$ is small. Clearly we may view $k(\boldsymbol{x}, \boldsymbol{z})$ as a measure of similarity between the training patterns $\boldsymbol{x}$ and $\boldsymbol{z}$.

ii. Norms

The norm of a feature vector $\Phi(\boldsymbol{x})$ is $\|\Phi(\boldsymbol{x})\| = \sqrt{\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}) \rangle} = \sqrt{k(\boldsymbol{x}, \boldsymbol{x})}$. For the Gaussian kernel function the feature vectors will have unit norm, *i.e.* $\|\Phi(\boldsymbol{x})\| = \sqrt{k(\boldsymbol{x}, \boldsymbol{x})} = 1$. Hence, as also shown in Section 2.2.2, when one uses a Gaussian kernel all feature vectors lie on the surface of a hypersphere with radius 1, which implies that the enclosed angle between any two features will be at most $\pi/2$. Therefore we can view input patterns mapped to a feature space via use of a Gaussian kernel to all lie in a single orthant of a hypersphere with radius 1 in $\Im$. This was illustrated in Figure 2.3.

iii. Distances

The Euclidian distance between feature vectors $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{z})$ is

$$\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{z})\|^2 = \langle \Phi(\boldsymbol{x}) - \Phi(\boldsymbol{z}), \Phi(\boldsymbol{x}) - \Phi(\boldsymbol{z}) \rangle$$

$$= \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}) \rangle - 2\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle + \langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle$$
$$= k(\boldsymbol{x}, \boldsymbol{x}) - 2k(\boldsymbol{x}, \boldsymbol{z}) + k(\boldsymbol{z}, \boldsymbol{z}). \quad (4.23)$$

This reduces to $2[1 - k(\boldsymbol{x}, \boldsymbol{z})]$ when using a Gaussian kernel function. In this case we know that with $\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{z})\|^2 \cong 0$, $k(\boldsymbol{x}, \boldsymbol{z}) \cong 1$, whereas with $\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{z})\|^2 \cong \infty$, $k(\boldsymbol{x}, \boldsymbol{z}) \cong 0$. Hence, when the feature space is induced via use of a Gaussian kernel, the Euclidian distance between any two feature vectors will be confined to the interval $[0,1]$.

□

4.5.2 *SETS OF POINTS*

We now proceed with several definitions and results summarising properties of and operations on a set of two or more feature vectors (for example, the feature vectors corresponding to a data group in classification or clustering). We will use the notation $F = \{\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n\}$ to represent such a set.

DEFINITION 4.2: ***THE CENTRE OF MASS***

Define the *centre of mass* (sample mean vector, or centroid) of a set of points in feature space as $\overline{\boldsymbol{\Phi}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\varphi}_i$. Also, let $\overline{\boldsymbol{\Phi}}_j = \frac{1}{n_j}\sum_{i \in I_j} \boldsymbol{\varphi}_i$ represent the centre of mass of the feature vectors belonging to data group j.

Note that $\overline{\boldsymbol{\Phi}}$ may be a vector such that no pattern in $\aleph$ will yield $\overline{\boldsymbol{\Phi}}$ after being transformed to $\Im$, *i.e.* there does not necessarily exist an input pattern $\boldsymbol{x} \in \aleph$ such that $\Phi(\boldsymbol{x}) = \overline{\boldsymbol{\Phi}}$.

The following proposition clarifies the interpretation of the centre of mass in feature space.

PROPOSITION 4.1

The centre of mass of a set of feature vectors $\left(\bar{\boldsymbol{\Phi}}\right)$ is the solution to the optimisation problem

$$\min_{\boldsymbol{\mu}}\left\{\frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{\varphi}_i-\boldsymbol{\mu}\right\|^2\right\}. \tag{4.24}$$

The above proposition can easily be verified by keeping in mind that $\sum_{i=1}^{n}\left(\boldsymbol{\varphi}_i-\bar{\boldsymbol{\Phi}}\right)=0$.

Despite the inaccessibility of the coordinates of $\bar{\boldsymbol{\Phi}}\in\Im$, the norm of the centre of mass can easily be calculated.

RESULT 4.2

Consider a set of feature vectors $F=\{\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\ldots,\boldsymbol{\varphi}_n\}$ with $\bar{\boldsymbol{\Phi}}$ the centre of mass.

i. Norms

Calculation of the norm of $\bar{\boldsymbol{\Phi}}$ is straightforward:

$$\begin{aligned}\left\|\bar{\boldsymbol{\Phi}}\right\|=\sqrt{\left\langle\bar{\boldsymbol{\Phi}},\bar{\boldsymbol{\Phi}}\right\rangle}&=\sqrt{\left\langle\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\varphi}_i,\frac{1}{n}\sum_{j=1}^{n}\boldsymbol{\varphi}_j\right\rangle}\\&=\frac{1}{n}\sqrt{\sum_{i,j=1}^{n}\left\langle\boldsymbol{\varphi}_i,\boldsymbol{\varphi}_j\right\rangle}\\&=\frac{1}{n}\sqrt{\sum_{i,j=1}^{n}k_{ij}}\ .\end{aligned} \tag{4.25}$$

ii. Distances

The mean sample Euclidian distance from feature vectors in F to $\bar{\boldsymbol{\Phi}}$ is

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{\varphi}_i-\bar{\boldsymbol{\Phi}}\right\|^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{\varphi}_i-\bar{\boldsymbol{\Phi}}\right)'\left(\boldsymbol{\varphi}_i-\bar{\boldsymbol{\Phi}}\right)\\ &= \frac{1}{n}\sum_{j=1}^{N}\sum_{i=1}^{n}\left(\varphi_{ij}-\bar{\phi}_{\vec{T}j}\right)^2\\ &= \frac{1}{n}\sum_{i=1}^{n}k_{ii}+\frac{1}{n^2}\sum_{i,j=1}^{n}k_{ij}-\frac{2}{n^2}\sum_{i,j=1}^{n}k_{ij}\\ &= \frac{1}{n}\sum_{i=1}^{n}k_{ii}-\frac{1}{n^2}\sum_{i,j=1}^{n}k_{ij}\ .\end{aligned} \tag{4.26}$$

The Euclidian distance between two mean vectors in feature space is

$$\begin{aligned}\left\|\bar{\boldsymbol{\Phi}}_1-\bar{\boldsymbol{\Phi}}_2\right\|^2 &= \left\langle \tfrac{1}{n_1}\sum_{i\in I_1}\boldsymbol{\varphi}_i, \tfrac{1}{n_1}\sum_{j\in I_1}\boldsymbol{\varphi}_j\right\rangle - 2\left\langle \tfrac{1}{n_1}\sum_{i\in I_1}\boldsymbol{\varphi}_i, \tfrac{1}{n_2}\sum_{j\in I_2}\boldsymbol{\varphi}_j\right\rangle + \left\langle \tfrac{1}{n_2}\sum_{i\in I_2}\boldsymbol{\varphi}_i, \tfrac{1}{n_2}\sum_{j\in I_2}\boldsymbol{\varphi}_j\right\rangle\\ &= \sum_{i\in I_1}\sum_{j\in I_1}\left\langle\boldsymbol{\varphi}_i,\boldsymbol{\varphi}_j\right\rangle\Big/n_1^2+\sum_{i\in I_2}\sum_{j\in I_2}\left\langle\boldsymbol{\varphi}_i,\boldsymbol{\varphi}_j\right\rangle\Big/n_2^2-2\sum_{i\in I_1}\sum_{j\in I_2}\left\langle\boldsymbol{\varphi}_i,\boldsymbol{\varphi}_j\right\rangle\Big/n_1n_2\\ &= \sum_{i\in I_1}\sum_{j\in I_1}k_{ij}\Big/n_1^2+\sum_{i\in I_2}\sum_{j\in I_2}k_{ij}\Big/n_2^2-2\sum_{i\in I_1}\sum_{j\in I_2}k_{ij}\Big/n_1n_2\ .\end{aligned} \tag{4.27}$$

□

Of course, when one uses a Gaussian kernel function, the mean sample distance in (4.26) reduces to $1-\frac{1}{n^2}\sum_{i,j=1}^{n}k_{ij}$.

4.6 *ALGORITHM-INDEPENDENT SELECTION*

4.6.1 *SELECTION CRITERIA*

THE ALIGNMENT

Reconsider the so-called 'ideal' kernel matrix, $\boldsymbol{K}_I$, in binary classification. It seems natural to propose selecting the subset of input variables yielding a kernel matrix with the 'closest' agreement to the structure in $\boldsymbol{K}_I$. How can one measure the agreement of matrices $\boldsymbol{K}_j$, $j \in J \subset \mathcal{J}$, with $\boldsymbol{K}_I$? Cristianini *et al.* (2002) define a measure of agreement between kernel matrices $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ as

$$A(\boldsymbol{K}_1, \boldsymbol{K}_2) = \frac{\langle \boldsymbol{K}_1, \boldsymbol{K}_2 \rangle_F}{\sqrt{\langle \boldsymbol{K}_1, \boldsymbol{K}_1 \rangle_F \langle \boldsymbol{K}_2, \boldsymbol{K}_2 \rangle_F}}, \tag{4.28}$$

where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F = \sum_{i,j} a_{ij} b_{ij} = tr(\boldsymbol{AB})$ is the so-called *Frobenius inner product* between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. In (4.28) the resemblance of the alignment with a correlation coefficient is evident: if the elements of $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are stacked into n^2-dimensional vectors $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$, the alignment can be viewed as the cosine of the angle between $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$. Hence it follows that $-1 \leq A(\boldsymbol{K}_1, \boldsymbol{K}_2) \leq 1$. The alignment was initially introduced in the context of finding the 'optimal' hyperparameter values in kernel analyses. In this context, kernel matrices corresponding to alternative hyperparameter values are measured against $\boldsymbol{K}_I = \boldsymbol{yy}'$, where

$$\boldsymbol{y}' = \left[\underbrace{+1, +1, \ldots, +1}_{n_1 \text{ terms}}, \underbrace{-1, -1, \ldots, -1}_{n_2 \text{ terms}} \right], \tag{4.29}$$

and a hyperparameter specification yielding a kernel matrix closer to $\boldsymbol{K}_I = \boldsymbol{yy}'$ is considered to be the more 'optimal' specification.

Hence $\boldsymbol{K}_I$ (as proposed in the paper by Cristianini *et al*., 2002) contains an $n_1 \times n_1$ sub-matrix of 1's in the top left position, a similar $n_2 \times n_2$ sub-matrix in the bottom right position, all other elements equal to -1, thereby clearly reflecting 'perfect similarity' between cases belonging to the same group, and 'perfect dissimilarity' between cases belonging to opposite groups.

Returning to the variable selection problem, it seems reasonable to propose the use of $A(\boldsymbol{K}, \boldsymbol{K}_I)$ to measure the agreement of the structure in kernel matrices (based on different subsets of input variables) with the structure in $\boldsymbol{K}_I = \boldsymbol{yy}'$. That is, we propose using

$$A(\boldsymbol{K}, \boldsymbol{K}_I) = A(\boldsymbol{K}, \boldsymbol{yy}') = \frac{\langle \boldsymbol{K}, \boldsymbol{yy}' \rangle_F}{\sqrt{\langle \boldsymbol{K}, \boldsymbol{K} \rangle_F \langle \boldsymbol{yy}', \boldsymbol{yy}' \rangle_F}}, \tag{4.30}$$

which simplifies to

$$A = \frac{\sum_{i \in I_1} \sum_{j \in I_1} k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i \in I_2} \sum_{j \in I_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{i \in I_1} \sum_{j \in I_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{n \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}} \tag{4.31}$$

to rate the different variable subsets of a given size, and then to select the subset of input variables corresponding to the highest value for the alignment criterion (A) above.

In our evaluation of the alignment criterion later on, we make use of the Gaussian kernel function to obtain the kernel matrices corresponding to the variable subsets considered. Of course, using the Gaussian kernel function, 'perfect similarity' between input patterns belonging to the same class is reflected by 1's in the $n_1 \times n_1$ and $n_2 \times n_2$ sub-matrices of $\boldsymbol{K}$, and 'perfect dissimilarity' is reflected by all the remaining entries being equal to 0. One might now ask whether $\boldsymbol{K}_I = \boldsymbol{yy}'$ should not be redefined: should we not rather make

use of $\boldsymbol{y}' = \left[\underbrace{1,1,\ldots,1}_{n_1 \text{ terms}}, \underbrace{0,0,\ldots,0}_{n_2 \text{ terms}} \right]$ in (4.30)? If we decide on the latter definition of $\boldsymbol{K}_I$, note that (4.31) would simply become

$$A = \frac{\sum_{i \in I_1} \sum_{j \in I_1} k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i \in I_2} \sum_{j \in I_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{n_1 \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}}. \tag{4.32}$$

In a binary classification setup, one expects the $\sum_{i \in I_1} \sum_{j \in I_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j)$-term to provide valuable information regarding how well the two groups can be separated, and therefore we believe that the form of the criterion in (4.31) should be preferred to the one in (4.32). If one uses the Gaussian kernel function to calculate $\boldsymbol{K}$, a problem however arises with the criterion in (4.31): a kernel matrix reflecting 'perfect' separation between the two groups will have 0's instead of -1's in its $n_1 \times n_2$ sub-matrix. In order to ensure comparability of $\boldsymbol{K}$ with $\boldsymbol{K}_I = \boldsymbol{y}\boldsymbol{y}'$, we therefore suggest making use of the following translation of the Gaussian kernel function:

$$\begin{aligned} \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= 2k(\boldsymbol{x}_i, \boldsymbol{x}_j) - 1 \\ &= 2exp\left(-\gamma \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2 \right) - 1. \end{aligned} \tag{4.33}$$

Based firstly on the use of (4.33) instead of the usual Gaussian kernel function (in (2.2)) to calculate the kernel matrices (pertaining to each variable subset under consideration), and secondly, on using the alignment criterion (in (4.31)) to decide between input variable subsets of a fixed size, an alternative selection procedure therefore arises. We will refer to this proposal as the *translated alignment* (or the *AT*) procedure in subsequent sections.

THE SUM OF GROUP SIMILARITIES

Often in a two-group classification problem it is more important to discern significant differences between cases belonging to opposite groups than establishing similarities among sample cases in the same group. Arguing that perhaps the first two terms (which accumulate similarities between cases in the same group) in the numerator of the alignment may overshadow the third term (which quantifies the correspondence between input patterns from distinct groups), a possible criterion for selecting input variables may therefore simply be the sum of the kernel matrix entries in the $n_1 \times n_2$ submatrix, *viz.* $SS = \sum_{i \in I_1} \sum_{j \in I_2} k_{ij}$.

If the two groups are well separated, similarities between pairs of input patterns observed for sample cases belonging to opposite groups should be small. Therefore, using *SS*, our selection strategy is to select the subset of variables which minimises the sum of similarities contained in the $n_1 \times n_2$ submatrix of the kernel matrix. If the Gaussian kernel function is used, note that $0 \leq SS \leq n_1 n_2$. Naturally one could also consider using the translated Gaussian kernel function in (4.33). In this case, $-n_1 n_2 \leq SS \leq n_1 n_2$.

THE DIFFERENCE IN GROUP MEANS

When the two groups are separated mainly with respect to their mean vectors (in $\Im$), a logical choice for a selection criterion is the distance between the respective group means, where the means are calculated in feature space. Hence we propose selecting the subset of variables which maximises the length of the difference vector $\bar{\boldsymbol{\Phi}}_1 - \bar{\boldsymbol{\Phi}}_2$ in $\Im$. A possible selection criterion is therefore $M = \left\| \bar{\boldsymbol{\Phi}}_1 - \bar{\boldsymbol{\Phi}}_2 \right\|^2$, and we propose selection of the variable subset which yields a maximum value for *M*. Recall that in general the coordinates of $\bar{\boldsymbol{\Phi}}_j \in \Im$, $j = 1,2$, cannot be obtained. Fortunately, we have seen in (4.27) that a value for *M* can be calculated via use of a kernel function.

THE VARIANCE RATIO

Wang *et al*. (2004) propose the following criterion for determining a value for the hyperparameter γ in the Gaussian kernel function:

$$VA(\gamma) = \frac{s_1^2 + s_2^2}{\left\| \overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_1 \right\|^2}. \tag{4.34}$$

In this expression, $\overline{\boldsymbol{\Phi}}_j = \frac{1}{n_j} \sum_{i \in I_j} \boldsymbol{\varphi}_i$ and $s_j^2 = \sum_{i \in I_j} \left\| \boldsymbol{\varphi}_i - \overline{\boldsymbol{\Phi}}_j \right\|^2$, $j = 1, 2$. Note that $VA(\gamma)$ can easily be calculated: the denominator is given in (4.27), and the terms in the numerator are

$$s_j^2 = \sum_{i \in I_j} \left\| \boldsymbol{\varphi}_i - \overline{\boldsymbol{\Phi}}_j \right\|^2 = n_j - \sum_{i \in I_j} \sum_{h \in I_j} k(\boldsymbol{x}_h, \boldsymbol{x}_i), \; j = 1,2. \tag{4.35}$$

The numerator in $VA(\gamma)$ may be interpreted as a measure of the total within-group sample variation in feature space. Similarly, the denominator is a measure of the between-group variation, once again in feature space. Given these interpretations of the numerator and denominator, we will refer to $VA(\gamma)$ as a *variation ratio* criterion.

The variation ratio resembles the reciprocal of the Rayleigh quotient in feature space (given in Section 4.3.2), which is maximised to determine the KFD classifier. In KFDA we work with measures of this variation after projecting the feature vectors onto a line in feature space, and our purpose is to find the classification rule maximising the between-to-within group variation measured along the projection line. The variation ratio criterion also considers between- and within-group variation after projecting the points in feature space onto some line. In order to see this, consider the average (centre) of the projections of all group j cases in $\Im$ onto a line spanned by $\boldsymbol{u}$ in $\Im$, *viz*.

$$P_{\boldsymbol{u}}\overline{\boldsymbol{\Phi}}_j = \sum_{i\in I_j} \frac{\langle \boldsymbol{u}, \boldsymbol{\varphi}_i \rangle}{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} \boldsymbol{u} \Big/ n_j \,,\ j = 1,2\,. \tag{4.36}$$

The line connecting the post-projection centres $P_{\boldsymbol{u}}\overline{\boldsymbol{\Phi}}_j$, $j = 1,2$, has length

$$\begin{aligned}
\left\| P_{\boldsymbol{u}}\overline{\boldsymbol{\Phi}}_1 - P_{\boldsymbol{u}}\overline{\boldsymbol{\Phi}}_2 \right\|^2 &= \left\| \left\langle \boldsymbol{u}, \tfrac{1}{n_1} \sum_{i\in I_1} \boldsymbol{\varphi}_i \right\rangle \boldsymbol{u} \Big/ \langle \boldsymbol{u}, \boldsymbol{u} \rangle - \left\langle \boldsymbol{u}, \tfrac{1}{n_2} \sum_{i\in I_2} \boldsymbol{\varphi}_i \right\rangle \boldsymbol{u} \Big/ \langle \boldsymbol{u}, \boldsymbol{u} \rangle \right\|^2 \\
&= \left\{ \frac{\left\| \left(\left\langle \boldsymbol{u}, \overline{\boldsymbol{\Phi}}_1 \right\rangle - \left\langle \boldsymbol{u}, \overline{\boldsymbol{\Phi}}_2 \right\rangle \right) \boldsymbol{u} \right\|}{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} \right\}^2 \\
&= \left\{ \frac{\left\langle \boldsymbol{u}, \overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2 \right\rangle \|\boldsymbol{u}\|}{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} \right\}^2 \\
&= \frac{\boldsymbol{u}' \left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2 \right) \left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2 \right)' \boldsymbol{u}}{\boldsymbol{u}'\boldsymbol{u}}\,.
\end{aligned} \tag{4.37}$$

Substituting $\boldsymbol{u} = \overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2$ in (4.37), the length of the line connecting the post-projection centres $P_{\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2}\overline{\boldsymbol{\Phi}}_j$, $j = 1,2$, turns out to be $\left\| \overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2 \right\|^2$, the denominator in the variation ratio. Hence, if we project all data cases in $\mathfrak{I}$ onto the line connecting the group centres in feature space, *i.e.* onto $\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2$, the length of the line connecting the post-projection group centres in $\mathfrak{I}$ is exactly the length of the line connecting the original group centres in $\mathfrak{I}$. Although the denominator in the variation ratio seems to simply measure between-group variation in $\mathfrak{I}$, it can therefore also be interpreted as a measure of between group variation after projecting all sample points onto $\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2$. From the above observation, the similarity of the denominator in $VA(\gamma)$ and the numerator in the Rayleigh quotient becomes more apparent.

From a classification perspective, a small value for the variation ratio is desirable and Wang *et al*. (2004) consequently propose determining γ to minimise $VA(\gamma)$. It seems

natural to consider using the variation ratio as a criterion in variable selection: for a given variable subset size m, select those variables minimising this criterion. In a selection setup it seems reasonable to keep γ fixed, since we are not primarily interested in the variation ratio as a function of γ, but in obtaining relative ranks for the variable subsets of equal size. We propose using $\gamma = 1/p$ in $VA(\gamma)$. Since we will be using $VA(\gamma)$ as selection criterion and not to determine a value for γ, we will suppress its dependence on γ and simply denote it by VA.

4.6.2 *MONTE CARLO SIMULATION STUDY*

We now describe an empirical study which was conducted to evaluate the relative performance of the algorithm-independent selection criteria in the previous section (*A*, *AT*, *M*, *SS* and *VA*), and follow this with a presentation and brief discussion of the results that were obtained.

In essence we made use of the same simulation configurations as described in Section 3.4. Recall that the post-selection simulation results presented in Chapter 3 indicated *NL* configurations to be more suited for selection in input space. Therefore note that all *NL* configurations were omitted in the simulation study discussed in this section, and will also not be investigated in any of the simulation studies described in the remainder of the thesis. We also only generated data sets where the relevant and irrelevant subsets of input variables were uncorrelated (*i.e.* where $\rho_{S\bar{S}} = 0$). The resulting experimental design amounted to 16 configurations per data scenario, *viz.* the *NS*, *LL* and *LS* setups. In this section, note that we make use of the following numbering of data configurations: a '1' refers to $\rho_S = 0$, $\pi = 0.1$ cases; a '2' refers to $\rho_S = 0$, $\pi = 0.4$ cases; we use a '3' to denote all $\rho_S = 0.7$, $\pi = 0.1$ cases; and a '4' is used in cases where $\rho_S = 0.7$ and $\pi = 0.4$.

Each simulation repetition was structured as follows. Consider any selection criterion. We firstly generated a training sample according to the data scenario under consideration, and

used the training sample values on each of the input variables $X_1, X_2, \ldots, X_p$ to obtain a value for the criterion. Depending on the selection criterion considered, we used the m smallest (or largest) absolute values of a selection criterion to indicate the m input variables to select, and kept record of the number of times each input variable was selected. We then generated a test sample, and calculated the test error after performing KFDA or after fitting an SVM based on the selected subset of input variables only. We calculated the average of the selection frequencies and test errors over the simulation repetitions (we made use of 500 repetitions throughout), and repeated the process for all selection criteria.

Note that we treated specification of the kernel hyperparameter and the cost hyperparameter (in this case we used $C = 10^{-5}, 10^{-3}, 10^{-1}, 10^{1}, 10^{3}, 10^{5}$) in the same manner as in the simulation studies carried out in Chapters 2 and 3. The average test errors reported are the minimum average test errors obtained over the various *C*-values. The value of the kernel hyperparameter was once again specified as $\gamma = 1/p$ throughout.

The average test errors are reported in a set of six tables. The first pair of tables (Tables 4.1 and 4.2 below) report *NS* average test errors, the second pair of tables correspond to the *LL* data scenarios, and the third pair summarise the *LS* results. Furthermore, the first table of each pair reports the average KFDA test errors, while the second table of each pair summarises SVM generalisation performances. As was the case in Chapters 2 and 3, note that we included rows which report on the average KFDA and SVM test errors when no selection is performed (in the $Err(\mathcal{V})$-rows), as well as the average KFDA and SVM errors when only the set of relevant input variables are used (in the $Err(V_S)$-rows). Standard errors ranged between 0.000 and 0.007.

Since our conclusions based on the *NS* scenario were representative of the conclusions emanating in the *LL* and *LS* scenarios, note that we only present Tables 4.1 and 4.2 below. The four remaining tables are given in Appendix A.2.1.

Table 4.1: Average test errors in the *NS* case

		KFDA						
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(VA)$	$Err(M)$	$Err(SS)$	$Err(A)$	$Err(AT)$
***NS*1**	***SMALL***	.454	.282	.407	.414	.461	.345	.336
	MIXED	.250	.250	.250	.250	.250	.250	.250
	LARGE	.379	.258	.259	.259	.436	.258	.258
	WIDE	.410	.085	.462	.463	.468	.207	.194
***NS*2**	***SMALL***	.225	.102	.188	.193	.291	.154	.149
	MIXED	.199	.086	.130	.132	.250	.250	.250
	LARGE	.204	.110	.110	.110	.243	.110	.110
	WIDE	.129	.003	.123	.125	.160	.014	.012
***NS*3**	***SMALL***	.455	.282	.410	.417	.465	.340	.332
	MIXED	.250	.250	.250	.250	.250	.250	.250
	LARGE	.379	.259	.259	.259	.427	.259	.259
	WIDE	.425	.138	.466	.467	.472	.281	.274
***NS*4**	***SMALL***	.309	.153	.221	.227	.313	.197	.191
	MIXED	.250	.250	.250	.250	.250	.250	.250
	LARGE	.135	.083	.083	.083	.218	.083	.083
	WIDE	.223	.019	.237	.238	.266	.110	.101

Firstly, we note that in mixed samples, the KFDA errors pertaining to the *NS*1, *NS*3 and *NS*4 cases, as well as the *LS*1, *LS*3 and *LS*4 cases, were consistently 0.25 – see the discussion on page 70. Secondly, we see that the *AT* criterion generally was the best performer, with the *A* criterion in second place, followed by the *VA*-, and *M* criteria. The *SS* criterion performed rather poorly throughout. Relative to differences between the post-selection errors when *AT* and *A* were used, as well as between the errors obtained in the case of *VA* and *M*, note that we typically observe a larger margin between errors using *A* and *VA*.

Table 4.2: Average test errors in the *NS* case (continued)

		SVM						
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(VA)$	$Err(M)$	$Err(SS)$	$Err(A)$	$Err(AT)$
***NS*1**	***SMALL***	.455	.290	.408	.412	.462	.345	.336
	MIXED	.332	.254	.259	.266	.267	.266	.260
	LARGE	.396	.258	.258	.258	.426	.258	.258
	WIDE	.430	.123	.463	.464	.468	.251	.240
***NS*2**	***SMALL***	.259	.129	.189	.194	.297	.203	.169
	MIXED	.206	.091	.127	.129	.365	.366	.365
	LARGE	.227	.090	.090	.090	.231	.090	.090
	WIDE	.086	.005	.246	.248	.287	.049	.043
***NS*3**	***SMALL***	.450	.291	.408	.415	.464	.346	.339
	MIXED	.334	.254	.259	.260	.268	.267	.266
	LARGE	.397	.257	.257	.257	.429	.257	.257
	WIDE	.430	.187	.470	.471	.476	.295	.285
***NS*4**	***SMALL***	.320	.180	.250	.255	.326	.227	.222
	MIXED	.265	.116	.144	.146	.363	.366	.367
	LARGE	.163	.077	.077	.077	.217	.077	.077
	WIDE	.250	.151	.321	.322	.341	.198	.194

The above conclusions hold irrespective of whether SVMs were trained, or KFDA were performed. Differences between the post-selection average errors were somewhat less prominent in the *LL* data scenarios. There *AT* and *A* perform very similarly, with the *AT* criterion doing only slightly better in wide sample cases. The *VA* criterion generally outperformed *M*, especially in small sample cases. A clear distinction between *VA* and *M* was however not possible for mixed, large and wide samples. In the *LS* data setups, once again the *AT* criterion emerged as the overall winner. As was the case with *NS* and *LL* data, *VA* consistently outperformed *M*, except in wide samples, where the opposite was found.

4.7 *ALGORITHM-DEPENDENT SELECTION*

In this section, we consider algorithm-dependent selection criteria. In addition to entries in the kernel matrix, algorithm-dependent criteria are allowed to also depend on the $\alpha_1, \alpha_2, \ldots, \alpha_n$ values obtained as output from the kernel algorithm. This allowance of course causes algorithm-dependent criteria to be computationally much more expensive. We start by considering a criterion for selection in SVMs, and then focus on selection in KFDA.

4.7.1 *SELECTION CRITERIA*

THE SQUARED NORM OF THE SVM WEIGHT VECTOR

Consider an SVM discriminant function, $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b$, where the weight vector $\boldsymbol{w}$ is given by $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\boldsymbol{x}_i)$, with $\alpha_1, \alpha_2, \ldots, \alpha_n$ of course output from the SVM algorithm. It follows that the squared norm of the SVM weight vector is $\|\boldsymbol{w}\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with k the kernel function. Since $\|\boldsymbol{w}\|^2$ plays such an intrinsic part in the SVM methodology, we consider the use of $\|\boldsymbol{w}\|^2$ as selection criterion for SVMs: firstly, can $\|\boldsymbol{w}\|^2$ be regarded as an appropriate selection criterion, and secondly, if so, how should $\|\boldsymbol{w}\|^2$ be used?

One tends to think that amongst variable subsets with the same size, if one performs variable selection based on $\|\boldsymbol{w}\|^2$, the variable subset of choice should be the subset leading to a minimum value for $\|\boldsymbol{w}\|^2$; after all, the idea underlying training of an SVM is to maximise the margin, which is inversely proportional to $\|\boldsymbol{w}\|^2$. This point of view turns out to be wrong. Substantial empirical evidence convinced us that the proper way of using

$\|\boldsymbol{w}\|^2$ as criterion in selection for SVMs, is to select the subset of variables which maximise the value of $\|\boldsymbol{w}\|^2$. The following heuristic argument may provide more insight. Consider

$$\|\boldsymbol{w}\|^2 = \left\| \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\|^2$$

$$= \left\| \sum_{i \in I_1, j \in I_1} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i \in I_2, j \in I_2} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{i \in I_1, j \in I_2} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\|^2 . \qquad (4.38)$$

This expression follows easily if we keep in mind that $y_i y_j = 1$ when sample cases i and j belong to the same group, and $y_i y_j = -1$ otherwise. Since the α_i's and the Gaussian kernel function values are non-negative, it follows that each of the three terms in (4.38) will also be non-negative. Suppose we compute (4.38) for a subset of variables which separate the two groups well. Then the cross-group term in (4.38) will tend to be small, since entities from groups which are well separated will look markedly different and therefore $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ will be close to zero if i and j correspond to cases from different groups. This should result in a larger value of (4.38) than the value obtained for a subset of variables which do not separate the two groups well. Note that this does not contradict the practice of computing $\alpha_1, \alpha_2, \ldots, \alpha_n$ for an SVM by maximising the margin, and therefore minimising $\|\boldsymbol{w}\|^2$. One should bear in mind that when we compute $\alpha_1, \alpha_2, \ldots, \alpha_n$, we do so for a fixed set of input variables, *i.e.* for fixed values of $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This is completely different from the situation in variable selection.

With the aim of empirically verifying the above conclusion, we conducted a very limited simulation experiment. We used the same data structure as in the *NS*, *LL* and *LS* data scenarios described in Chapter 3, and fixed the number of relevant input variables at $m = 1$ (out of $p = 5$). Regarding the correlation between pairs of input variables, we set $\rho_S = 0$ or 0.7. The correlation between pairs of relevant and irrelevant variables was $\rho_{S\bar{S}} = 0$ throughout. In cases where the two groups were separated with respect to location, we

investigated one extra scenario: in addition to a difference in group means $(\mu_{21} - \mu_{11} = \Delta)$ equal to 1, we also set $\Delta = 3$. When differences between the two groups were obtained via different variance-covariance structures, we used $s_1^2 = 1$ and $s_2^2 = 100$ in addition to $s_1^2 = 1$ and $s_2^2 = 10$.

Note that the experiment did not involve any selection steps: we simply trained five SVMs (each making use of the Gaussian kernel), based on the single input variables $X_1, X_2, \ldots, X_5$. For each of the five support vector classifiers, *i.e.* $SVM(X_1)$, $SVM(X_2)$, …, $SVM(X_5)$, we then calculated an average $\|\boldsymbol{w}\|^2$-value and, based on 2000 test cases, an average test error over 1000 simulation repetitions. The experiment was repeated using different cost parameter values, *viz*. C= 0.01, 0.1, 1, 10, 100 and 1000, and for both small ($n_1 = n_2 = 15$) and large ($n_1 = n_2 = 100$) sample sizes.

The results are summarised in Tables 4.3-4.5, for the *NS*, *LL* and *LS* data scenarios respectively. Each cell contains two rows. The entry in the first row of each cell is the average value of $\|\boldsymbol{w}\|^2$, and the figure in the second row is the corresponding test error value. Note that we do not report the average test errors at each of the six cost parameter values considered, but only at the C-value where a minimum average test error was obtained. For each configuration (in the table columns) we indicate the largest average $\|\boldsymbol{w}\|^2$-value in bold face.

Table 4.3: Average $\|w\|^2$ values and minimum average test errors for *NS* data

SMALL SAMPLES				
VARIABLE	$\rho_S = 0$, $\Delta = 1$	$\rho_S = 0.7$, $\Delta = 1$	$\rho_S = 0$, $\Delta = 3$	$\rho_S = 0.7$, $\Delta = 3$
X_1	0.0845 .2904	0.01658 .2885	**2.4876** .1330	0.7184 .1334
X_2	0.0252 .5001	0.0211 .4999	1.3631 .4994	0.4853 .5000
X_3	0.0156 .4997	0.0089 .4998	2.2004 .4999	**2.8824** .5003
X_4	**0.1662** .5003	**0.0256** .5002	1.7159 .5004	1.4332 .5001
X_5	0.1462 .5001	0.0176 .5002	0.3199 .5001	0.5859 .4998

LARGE SAMPLES				
VARIABLE	$\rho_S = 0$, $\Delta = 1$	$\rho_S = 0.7$, $\Delta = 1$	$\rho_S = 0$, $\Delta = 3$	$\rho_S = 0.7$, $\Delta = 3$
X_1	**.0413** .2570	**.0490** .2572	**.8939** .1069	.9536 .1072
X_2	.0151 .4998	.0089 .4995	.0477 .4995	1.5409 .4994
X_3	.0267 .5005	.0118 .4999	.0409 .5003	**5.2769** .4999
X_4	.0097 .5005	.0098 .4999	.4827 .5003	.1451 .5003
X_5	.0152 .4998	.0046 .5002	.0914 .4997	.1851 .5007

Table 4.4: Average $\|w\|^2$-values and minimum average test errors for *LL* data

SMALL SAMPLES				
VARIABLE	$\rho_S = 0$, $\Delta = 1$	$\rho_S = 0.7$, $\Delta = 1$	$\rho_S = 0$, $\Delta = 3$	$\rho_S = 0.7$, $\Delta = 3$
X_1	**.000003** .1237	**.000303** .1235	**.000020** .02624	**.002007** .02602
X_2	.000000 .4995	.000200 .4998	.000001 .4999	.000046 .4497
X_3	.000001 .5002	.000016 .5003	.000000 .5010	.000024 .5000
X_4	000001 .5005	.000100 .5000	.000000 .5001	.000047 .4998
X_5	.000000 .4998	.000206 .4995	.000001 .4998	.000048 .0260

LARGE SAMPLES				
VARIABLE	$\rho_S = 0$, $\Delta = 1$	$\rho_S = 0.7$, $\Delta = 1$	$\rho_S = 0$, $\Delta = 3$	$\rho_S = 0.7$, $\Delta = 3$
X_1	**.0003** .1137	**.000001** .1144	.1408 .0218	**.0103** .02213
X_2	.0000 .5000	.000000 .4999	.0580 .5004	.0038 .4998
X_3	.0000 .4996	.000000 .5005	.1553 .4999	.0062 .5005
X_4	.0000 .4999	.000000 .5001	.1010 .4999	.0035 .5001
X_5	.0000 .5000	.000000 .4996	**.1810** .4999	.0070 .4493

Table 4.5: Average $\|w\|^2$-values and minimum average test errors for *LS* data

SMALL SAMPLES				
VARIABLE	$\rho_S = 0$, $\Delta = 1$	$\rho_S = 0.7$, $\Delta = 1$	$\rho_S = 0$, $\Delta = 3$	$\rho_S = 0.7$, $\Delta = 3$
X_1	.0472 .2135	.1503 .2119	.2331 .1100	**4.7804** .1108
X_2	.2078 .4994	.0603 .5002	.1602 .5005	1.3880 .5002
X_3	**.2298** .5001	.02819 .4999	.1085 .5001	.1817 .4997
X_4	.1329 .5000	**.1735** .4996	**.2458** .4999	.1518 .4992
X_5	.0711 .4002	.0401 .5003	.01619 .4998	.1002 .5002

LARGE SAMPLES				
VARIABLE	$\rho_S = 0$, $\Delta = 1$	$\rho_S = 0.7$, $\Delta = 1$	$\rho_S = 0$, $\Delta = 3$	$\rho_S = 0.7$, $\Delta = 3$
X_1	.8006 .1805	**.2489** .1800	**1.7244** .0825	**7.7024** .0830
X_2	.1783 .5004	.0093 .5002	.6405 .5000	.6188 .5002
X_3	**.2281** .5001	.02202 .5004	.4727 .4998	.4940 .5002
X_4	.1481 .5002	.0240 .4998	.6134 .5001	.3220 .4999
X_5	.2342 .4999	.0141 .4992	.0433 .4999	3.8326 .4997

As in the numerical experiments reported in previous chapters, first note the significant improvement in the minimum (over *C*-values) average test error when the support vector classifier uses only the separating input variable (X_1), rather than any other single variable.

What can be said about suitability of the idea to select the set of input variables maximising $\|\boldsymbol{w}\|^2$? Consider first the small sample results, summarised in the first parts of Tables 4.3 to 4.5. Here the *LL* data sets do indeed yield maximum values of $\|\boldsymbol{w}\|^2$ for the SVM based on X_1, which is also the SVM having the smallest test error. In these cases it seems that using a selection procedure based on maximising $\|\boldsymbol{w}\|^2$ should work well. The results for the small sample *NS* and *LS* data scenarios are not as promising. In the four small sample *NS* cases, $\|\boldsymbol{w}\|^2$ is a maximum for $SVM(X_1)$ only once, with a similar result for the four small sample *LS* cases. The large sample results (presented in the lower parts of Tables 4.3 to 4.5) are decidedly more promising. For each of *NS*, *LL* and *LS* we find $\|\boldsymbol{w}\|^2$ to be a maximum for $SVM(X_1)$ in three out of the four scenarios investigated. Overall it seems that selecting variables to maximise $\|\boldsymbol{w}\|^2$ may indeed be a worthwhile proposal. Note that in our further discussions we will denote the $\|\boldsymbol{w}\|^2$ criterion by *N*.

THE RAYLEIGH QUOTIENT

In this section we turn attention to selection for KFDA. Considering algorithm-dependent selection, we now ask whether it is possible to propose a quantity for selection which, in addition to kernel matrix entries, also make use of the $\alpha_1, \alpha_2, \ldots, \alpha_n$ output from the KFDA algorithm. We follow our reasoning in proposing the *N* criterion for SVMs, and focus our attention on the key aspect in KFDA methodology.

The objective of KFDA in binary classification is to find the direction in feature space corresponding to the projection providing the best possible separation between the two groups. Separation is measured in terms of the ratio between the distance between the two

projected group means in $\Im$, and the variance of the projected data in $\Im$ within a specific group. We have seen in Section 4.3.2 that the KFDA objective is realised via maximisation of the Rayleigh quotient (given in (4.20)). The Rayleigh quotient is the criterion on which the KFDA algorithm is founded, and since its value depends on the KFDA α-coefficients, we propose using it to rate variable subsets of the same size: variable subsets corresponding to large values for the Rayleigh quotient are preferred.

Note that the Rayleigh quotient bears strong resemblance to the inverse of the variation ratio criterion in Section 4.6.1. A distinction between these two criteria is of course that, through additional dependence of the Rayleigh quotient on the KFDA $\alpha_1, \alpha_2, \ldots, \alpha_n$ output, the latter is computation-wise a much more expensive criterion.

4.7.2 *MONTE CARLO SIMULATION STUDY*

We compared the performances of the algorithm-dependent selection criteria that were proposed in the previous section, *viz*. the Rayleigh coefficient (*R*) in KFDA, and the norm of the weight vector in SVMs (*N*) by conducting a Monte Carlo simulation study. In our numerical evaluation we made use of the same experimental design and simulation configurations as those described in Section 4.6.2.

The obtained average test errors are presented in Tables 4.6-4.8 below. Each table contains KFDA test errors in the first three columns, followed by SVM test errors in columns 5 to 7. We once again report the *no selection* and the *oracle* average test errors (denoted as before by $Err(\mathcal{V})$ and $Err(V_S)$ respectively); and in the case of KFDA, the post-selection error based on *R* (denoted by $Err(R)$); or in the case of SVMs, the average error rate obtained after performing selection based on the *N* criterion (in the $Err(N)$ column).

Table 4.6: Average test errors in the *NS* case

		KFDA				SVM			
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(R)$	*sel*	$Err(\mathcal{V})$	$Err(V_S)$	$Err(N)$	*sel*
***NS*1**	***SMALL***	.454	.282	.326	1.16	.455	.290	.328	1.13
	MIXED	.250	.250	.250	1.00	.332	.254	.254	1.00
	LARGE	.379	.250	.258	1.03	.396	.258	.258	1.00
	WIDE	.410	.085	.204	2.40	.430	.123	.255	2.07
***NS*2**	***SMALL***	.255	.102	.138	1.35	.259	.129	.175	1.36
	MIXED	.199	.086	.092	1.07	.206	.091	.096	1.05
	LARGE	.204	.110	.110	1.00	.227	.090	.090	1.00
	WIDE	.129	.003	.009	3.00	.086	.005	.039	7.80
***NS*3**	***SMALL***	.455	.282	.324	1.15	.450	.291	.332	1.14
	MIXED	.250	.250	.250	1.00	.334	.254	.254	1.00
	LARGE	.379	.259	.259	1.00	.397	.257	.257	1.00
	WIDE	.425	.138	.278	2.01	.430	.187	.295	1.58
***NS*4**	***SMALL***	.309	.153	.180	1.18	.320	.180	.209	1.16
	MIXED	.250	.250	.250	1.00	.265	.116	.120	1.03
	LARGE	.135	.083	.083	1.00	.163	.077	.077	1.00
	WIDE	.223	.019	.097	5.11	.250	.151	.185	1.23

The *selection* (*sel*) column for KFDA contains the values $Err(R)/Err(V_S)$, whereas for SVMs, the selection column represents $Err(N)/Err(V_S)$. Since the $Err(V_S)$-values portray the scenario where we did know which of the m variables were relevant, they represent the best one could possibly do. Hence the values in the selection columns should be larger than 1. Of course smaller *sel*-values (*i.e.* values closer to 1) indicate higher post-selection classification accuracy. Care should again be taken when interpreting the mixed sample results – see the discussion on page 70.

Table 4.7: Average test errors in the *LL* case

		KFDA				SVM			
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(R)$	*Sel*	$Err(\mathcal{V})$	$Err(V_S)$	$Err(N)$	*sel*
***LL*1**	***SMALL***	.337	.135	.137	1.01	.331	.122	.124	1.02
	MIXED	.165	.079	.079	1.00	.171	.064	.064	1.00
	LARGE	.215	.122	.122	1.00	.232	.115	.115	1.00
	WIDE	.301	.058	.069	1.19	.265	.068	.071	1.04
***LL*2**	***SMALL***	.166	.078	.078	1.00	.157	.084	.084	1.00
	MIXED	.078	.042	.042	1.00	.071	.045	.046	1.02
	LARGE	.072	.032	.032	1.00	.077	.033	.034	1.03
	WIDE	.122	.039	.049	1.26	.089	.020	.022	1.10
***LL*3**	***SMALL***	.330	.137	.139	1.01	.334	.121	.124	1.02
	MIXED	.160	.079	.079	1.00	.171	.064	.064	1.00
	LARGE	.213	.121	.121	1.00	.229	.113	.113	1.00
	WIDE	.352	.117	.127	1.09	.324	.122	.190	1.56
***LL*4**	***SMALL***	.219	.124	.125	1.01	.216	.127	.127	1.00
	MIXED	.100	.079	.079	1.00	.100	.068	.069	1.01
	LARGE	.151	.084	.084	1.00	.143	.084	.085	1.01
	WIDE	.228	.099	.107	1.08	.218	.128	.133	1.04

Comparing the selection values obtained for the *R* criterion for KFDA with those of the *N* criterion for SVMs, we see that in the *NS* and *LL* data scenarios, selection using the *N* criterion is mostly preferred to selection using the *R* criterion. Two exceptions occur: in the *NS*2 and the *LL*3 wide sample cases the *R* criterion outperforms the *N* criterion. In the *NS*2 case selection using the *R* criterion yields significantly smaller average test errors than selection using the *N* criterion: compare a selection value of 3 in the case of *R* with a value of 7.8 in the case of *N*. Finally, in the *LS* data setups, note that we generally observe the opposite: there the *R* criterion outperforms the *N* criterion, with a single exception occurring in the *LS*4 wide sample case.

Table 4.8: Average test errors in the *LS* case

		KFDA				SVM			
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(R)$	*sel*	$Err(\mathcal{V})$	$Err(V_S)$	$Err(N)$	*sel*
***LS*1**	***SMALL***	.438	.225	.248	1.10	.441	.213	.368	1.73
	MIXED	.250	.250	.250	1.00	.335	.227	.242	1.07
	LARGE	.360	.201	.202	1.00	.361	.180	.194	1.08
	WIDE	.449	.119	.200	1.68	.449	.136	.332	2.44
***LS*2**	***SMALL***	.262	.126	.138	1.10	.269	.125	.158	1.26
	MIXED	.193	.096	.099	1.03	.195	.086	.245	2.85
	LARGE	.154	.085	.085	1.00	.163	.076	.078	1.03
	WIDE	.279	.054	.106	1.96	.260	.062	.128	2.06
***LS*3**	***SMALL***	.442	.223	.244	1.09	.442	.208	.266	1.28
	MIXED	.250	.250	.250	1.00	.333	.224	.245	1.09
	LARGE	.360	.201	.202	1.00	.361	.179	.193	1.08
	WIDE	.457	.141	.263	1.87	.449	.174	.356	2.05
***LS*4**	***SMALL***	.323	.149	.158	1.06	.315	.166	.223	1.34
	MIXED	.243	.129	.135	1.05	.251	.095	.146	1.54
	LARGE	.206	.126	.126	1.00	.221	.092	.095	1.03
	WIDE	.342	.082	.196	2.39	.327	.168	.250	1.49

Up to this point, the results of our numerical evaluation has lead us to conclude that, of the algorithm-independent selection criteria, the *AT* criterion generally is the selection criterion of choice, followed by the *A* criterion, and then by *VA*. Amongst the algorithm-dependent criteria, we concluded that typically in *NS* and *LL* data setups, the *N* criterion for SVMs performs best, but that in *LS* scenarios, use of the *R* criterion for KFDA should be recommended.

The remaining question is whether the extra computational burden required to calculate the algorithm-dependent criteria is justified: do the algorithm-dependent criteria tend to select better variable subsets?

In order to facilitate a recommendation regarding a choice between algorithm-independent and -dependent selection criteria, we calculated the selection values corresponding to best-perfoming algorithm-independent criteria (*i.e. AT*, *A* and *VA*), and report these values, along with the *N* and *R* selection values (provided earlier on) in Tables 4.9-4.11. Care should once again be taken when interpreting the mixed sample results – refer to the discussion on page 70.

Table 4.9: Selection values for algorithm-dependent and –independent criteria in *NS* data

		KFDA				**SVM**			
		AT	*A*	*VA*	*R*	*AT*	*A*	*VA*	*N*
***NS*1**	***SMALL***	1.19	1.22	1.44	1.16	1.16	1.19	1.41	1.13
	MIXED	1.00	1.00	1.00	1.00	1.02	1.05	1.02	1.00
	LARGE	1.00	1.00	1.00	1.03	1.00	1.00	1.00	1.00
	WIDE	2.28	2.44	5.44	2.40	1.95	2.04	3.76	2.07
***NS*2**	***SMALL***	1.46	1.51	1.84	1.35	1.31	1.57	1.47	1.36
	MIXED	2.91	2.91	1.51	1.07	4.01	4.02	1.40	1.05
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	WIDE	4.00	4.67	41.0	3.00	8.60	9.8	49.2	7.80
***NS*3**	***SMALL***	1.18	1.21	1.45	1.15	1.16	1.19	1.40	1.14
	MIXED	1.00	1.00	1.00	1.00	1.05	1.05	1.02	1.00
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	WIDE	1.99	2.04	3.38	2.01	1.52	1.58	2.51	1.58
***NS*4**	***SMALL***	1.25	1.29	1.44	1.18	1.23	1.26	1.39	1.16
	MIXED	1.00	1.00	1.00	1.00	3.16	3.16	1.24	1.03
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	WIDE	5.32	5.79	12.5	5.11	1.28	1.31	2.13	1.23

Table 4.10: Selection values for algorithm-dependent and –independent criteria in *LL* data

		KFDA				SVM			
		AT	*A*	*VA*	*R*	*AT*	*A*	*VA*	*N*
LL1	***SMALL***	1.03	1.03	1.15	1.01	1.03	1.03	1.14	1.02
	MIXED	1.01	1.01	1.00	1.00	1.02	1.00	1.00	1.00
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	WIDE	1.10	1.14	1.52	1.19	0.96	0.97	1.34	1.04
LL2	***SMALL***	1.01	1.01	1.06	1.00	1.01	1.02	1.06	1.00
	MIXED	1.00	1.00	1.00	1.00	0.93	0.93	0.93	1.02
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.03
	WIDE	1.08	1.10	1.15	1.26	1.10	1.15	1.50	1.10
LL3	***SMALL***	1.01	1.02	1.12	1.01	1.03	1.04	1.17	1.02
	MIXED	1.01	1.00	1.00	1.00	1.05	1.00	1.02	1.00
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	WIDE	1.06	1.09	1.40	1.09	1.07	1.07	1.33	1.56
LL4	***SMALL***	1.01	1.01	1.03	1.01	1.02	1.02	1.06	1.00
	MIXED	1.00	1.00	1.00	1.00	1.01	1.01	1.00	1.01
	LARGE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.01
	WIDE	1.07	1.09	1.33	1.08	1.03	1.03	1.13	1.04

We see that the pattern in all three tables is more or less the same. In KFDA the algorithm-dependent *R* criterion does very well. It is only in wide samples that the algorithm-independent *AT* criterion performs better, in many cases only slightly so. The same conclusion is largely applicable to the *N* criterion for SVMs, except in *LS* data, where the *AT* criterion more or less dominates *N*. Overall it is evident that the additional effort required to compute the algorithm-dependent criteria offers a substantial improvement in post-selection classification accuracy.

Table 4.11: Selection values for algorithm-dependent and –independent criteria in *LS* data

		KFDA				SVM			
		AT	*A*	*VA*	*R*	*AT*	*A*	*VA*	*N*
***LS*1**	***SMALL***	1.20	1.22	1.63	1.10	0.70	0.72	1.73	1.73
	MIXED	1.00	1.00	1.00	1.00	1.10	1.10	1.07	1.07
	LARGE	1.00	1.00	1.03	1.00	1.01	1.01	1.04	1.08
	WIDE	1.50	1.55	3.24	1.68	1.38	1.43	2.88	2.44
***LS*2**	***SMALL***	1.18	1.20	1.44	1.10	1.20	1.22	1.48	1.26
	MIXED	2.60	2.60	1.45	1.03	2.26	3.43	1.45	2.85
	LARGE	1.00	1.00	1.01	1.00	1.00	1.00	1.03	1.03
	WIDE	1.72	1.83	3.98	1.96	1.56	1.63	3.82	2.06
***LS*3**	***SMALL***	1.21	1.25	1.68	1.09	1.25	1.28	1.75	1.28
	MIXED	1.00	1.00	1.00	1.00	1.19	1.20	1.11	1.09
	LARGE	1.00	1.00	1.04	1.00	1.01	1.01	1.04	1.08
	WIDE	1.67	1.72	2.99	1.87	1.39	1.43	2.43	2.05
***LS*4**	***SMALL***	1.15	1.17	1.38	1.06	1.14	1.16	1.29	1.34
	MIXED	1.94	1.94	1.37	1.05	3.24	3.25	1.51	1.54
	LARGE	1.00	1.00	1.01	1.00	1.00	1.00	1.01	1.03
	WIDE	2.10	2.18	3.82	2.39	1.22	1.24	1.88	1.49

4.8 *SUMMARY*

It was found in Chapter 3 that at least some feature space information should be used in the variable selection process. In this chapter we therefore studied selection in feature space, focusing on different types of feature space information. This led to two classes of selection criteria, *viz*. algorithm-independent and algorithm-dependent criteria. Criteria in the first category only uses feature space information which is independent of the specific kernel algorithm being applied, whilst criteria in the second category also employ information derived from this algorithm.

This chapter started with a section in which a detailed description of SVMs and KFDA was provided. Several algorithm-independent criteria were then defined, and compared in an empirical study. The results indicated that the AT and A criteria performed best, followed by VA. Two algorithm-dependent criteria were also defined, and their properties studied empirically. No general preference for one of the two criteria was found. Finally, we compared the algorithm-independent and -dependent criteria. Although the algorithm-dependent criteria generally performed best, the AT and A criteria were still very much competitive, especially in wide samples. Given their computational simplicity, these two criteria are recommendable, especially when p is large.

CHAPTER 5

BACKWARD ELIMINATION FOR KERNEL CLASSIFIERS

5.1 *INTRODUCTION*

In order to address the variable selection problem in its entirety, three questions need to be answered, *viz*. which selection criterion to use, how to reduce the number of all possible variable subsets in an efficient manner to a manageable number, and how many input variables to include in the final model. Thus far in the thesis we have focused on the first aspect. We investigated the performance of several selection criteria when a filter approach is used: after quantifying the individual contribution of each input variable to the selection criterion under consideration, we simply selected the fixed number of variables corresponding to the optimum value of the selection criterion. In this approach the relative importance of each input variable is measured without any consideration of the other input variables. Generally this is a serious drawback. Depending on which of the other variables have already been selected or eliminated from the model, relationships among input variables may influence the relevance of variables. In this chapter we therefore evaluate the performance of selection criteria when they are used in combination with a selection strategy. The strategy investigated in this chapter is backward elimination.

In the literature backward elimination of input variables in the context of kernel techniques is known as *recursive feature elimination* (*RFE*). There are several reasons why this strategy is frequently preferred to forward selection. If p is large a forward strategy may terminate before the model containing all the variables, or models containing a large number of variables, is considered. This is generally undesirable, especially in situations where it is reasonable to expect a large percentage of input variables to be selected. The disadvantages of a forward selection strategy are also pointed out in a review paper on

variable selection by Guyon and Elisseeff (2003). They argue that '*weaker subsets are found by forward selection because the importance of variables is not assessed in the context of variables not included yet*' and illustrate this by means of an example. Finally, we will present empirical evidence in Chapter 6 supporting the preference for backward elimination to forward selection.

An outline of the remainder of the chapter is as follows. In Section 5.2 we present a review of the literature on RFE. Section 5.3 introduces the use of several selection criteria in backward elimination, which is then evaluated in a fairly extensive Monte Carlo simulation study described in Section 5.4. The results of this simulation study are discussed in Section 5.5, followed by a summary of the chapter in Section 5.6.

5.2 *LITERATURE REVIEW*

In the literature on kernel variable selection, RFE was initially studied as an input variable selection strategy for SVMs in the context of binary classification problems in micro-array analyses (Guyon *et al*., 2002). RFE is an iterative backward elimination strategy: starting with the classifier using the comprehensive set of p input variables, each step in the RFE-algorithm involves elimination of one (or more) input variables. Elimination of variables continues until only m variables remain. Importantly, note that the original RFE-algorithm assumes $m < p$ to be known.

The paper by Guyon *et al.* (2002) played an important role in introducing RFE and we therefore discuss this paper in some detail. Application of RFE requires specification of an appropriate criterion for identifying the variable(s) to be discarded at a given step of the algorithm. Guyon *et al*. (2002) initially propose use of the squared coefficients of the weight vector obtained after training a *linear* support vector classifier for this purpose. To explain their idea, let the observations on the input variables remaining after step $d-1$ be denoted by $\breve{\boldsymbol{x}}_{d-1,i}$, $i = 1, 2, \ldots, n$. Then step d in the algorithm requires one to train a linear SVM, yielding a discriminant function

$$f_d(\boldsymbol{x}) = b + \langle \boldsymbol{w}_d, \boldsymbol{x} \rangle, \tag{5.1}$$

where $\boldsymbol{w}_d = \sum_{i=1}^{n} \alpha_i y_i \breve{\boldsymbol{x}}_{d-1,i}$ and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the usual SVM outputs. Because a linear kernel function is used, $\boldsymbol{w}_d$ is a vector in $\Re^p$, and its j^{th} coefficient can therefore be interpreted as a measure of the importance of the j^{th} input variable. Hence Guyon *et al*. (2002) propose that in each step of the RFE-algorithm, variable with the smallest w_d^2-value should be discarded. This strategy has been shown to work well if the final SVM only makes use of a linear kernel function and is therefore restricted to operate in input space. However, in Chapter 3 it was found that, with exception of *NL* data sets, selection in input space performed rather poorly. Since the strength of SVMs mostly lies in their use of non-linear kernels and a classification function defined in feature space, we expect the performance of SVM-RFE based on $\left\{ w_{dj}^2, j = 1,2,\ldots,p \right\}$ in input space to be suboptimal in most cases.

When a non-linear kernel function is used, the SVM classifier is non-linear in input space, and then SVM-RFE should arguably also be based on non-linear relations in $\aleph$. In their follow-up proposal regarding RFE selection criteria, Guyon *et al*. (2002) therefore make use of the weight vector coefficients of a non-linear support vector classifier. That is, elimination of input variables at the d^{th} step in SVM-RFE is based on $\boldsymbol{w}_d$ in the non-linear SVM discriminant function

$$f_d(\boldsymbol{x}) = b + \langle \boldsymbol{w}_d, \Phi(\boldsymbol{x}) \rangle, \tag{5.2}$$

where $\boldsymbol{w}_d = \sum_{i=1}^{n} \alpha_i y_i \Phi\left(\breve{\boldsymbol{x}}_{d-1,i}\right)$ is now an N-dimensional vector in $\Im$. Importantly, note that the individual elements in $\boldsymbol{w}_d$ are associated with the N individual *features* in $\Im$. If a non-linear kernel function is used, the features are usually intricate functions of the original input variables. In cases where the kernel function does not imply an infinite value

of N, the elements in $\boldsymbol{w}_d$ can then be used for *feature selection*. However, since no single element in $\boldsymbol{w}_d$ can be used to quantify the importance of the role played by any individual input variable in the kernel classifier, the vector $\boldsymbol{w}_d$ cannot be used directly as a criterion for *input variable selection*.

It is however not unreasonable to assume that the SVM weight vector can be used for variable selection in RFE even when a non-linear kernel function is employed. Consider therefore once again step *d* and suppose $\left\|\boldsymbol{w}_d^{-v}\right\|^2$ denotes the squared norm of $\boldsymbol{w}_d$ if variable X_v is omitted at this step. Guyon *et al*. (2002) suggest using

$$\Delta W_v^2 = \left| \|\boldsymbol{w}_d\|^2 - \left\|\boldsymbol{w}_d^{-v}\right\|^2 \right| \tag{5.3}$$

as a criterion for identifying the variable which should be omitted. This is simply the absolute change in the value of $\|\boldsymbol{w}_d\|^2$ upon omission of variable X_v, and the proposal is to remove the variable minimising (5.3). The variable identified for omission at step *d* is therefore the variable whose exclusion has the smallest effect on $\|\boldsymbol{w}_d\|^2$. In this sense, ΔW_v^2 may be regarded as a so-called *sensitivity selection criterion* (*cf*. Rakotomamonjy, 2003).

It is necessary to refer to an approximation frequently employed in RFE to reduce the amount of computations in cases where p is large. Consider step *d* of the algorithm and assume that only one variable is discarded at each step. Then at the start of step *d* there are $p-d+1$ variables remaining in the model. Application of (5.3) as selection criterion would therefore entail fitting $p-d+1$ different SVMs, one corresponding to each of the remaining variables being omitted. In order to reduce the amount of computations required for implementation of this strategy, Guyon *et al*. (2002) propose the following approximation: assume that the SVM coefficients based on the reduced model obtained by

omitting a variable are identical to those based on the current full model. This assumption obviates re-calculation of $\alpha_1, \alpha_2, \ldots, \alpha_n$ after omission of each of the $p - d + 1$ candidate variables; we simply use $\alpha_1, \alpha_2, \ldots, \alpha_n$ computed after completion of step $d - 1$ and only re-calculate the entries in the kernel matrix after omitting each of the candidate variables in turn.

Guyon *et al*. (2002) evaluated SVM-RFE using gene expression data sets (the colon cancer data set, *cf*. Alon *et al*., 1999, and the leukaemia data set, *cf*. Golub *et al*., 1999). In both data examples pre-processing of the data features prominently. Standardisation of each input variable (*i.e*. subtracting the mean and dividing by the standard deviation of the variable) is consistently carried out. Guyon *et al*. (2002) also recommend standardisation of sample cases across variables. This of course only makes sense when the input variables are measured on the same scale, as is the case in micro-array data. The authors also use logarithmic transformations and apply a squashing function to reduce the effect of possible outliers.

Given that RFE is a numerically expensive procedure, a question may be raised as to whether it is worthwhile compared to simpler and computationally cheaper alternatives. Guyon *et al*. (2002) address this issue and point out that a full implementation of RFE consistently outperforms more naïve and cheaper alternatives, such as for example input variable ranking based on only a first RFE iteration. This is a consequence of the fact that RFE investigates the relevance of subsets of input variables simultaneously, thereby taking their joint effect into account. A procedure based on identifying a subset of input variables from their individual rankings may easily end up recommending highly correlated variables, many of which may in fact be superfluous. As a result, other useful variables may in the process be discarded. Moreover, there are further ways to speedup the RFE procedure. A very simple strategy, implemented by Guyon *et al*. (2002), is to remove half of the remaining input variables at each step during the first stages of the algorithm. As soon as approximately 100 variables remain, a single variable is then eliminated at each RFE step until the desired m variables is reached. Alternatively, one can make use of the

strategy presented in Furlanello *et al*. (2003), *viz*. entropy-based RFE, a faster version of the original RFE algorithm.

Prior to the introduction of SVM-RFE, gene selection in cancer classification problems was typically carried out using techniques related to calculating ordinary Pearson correlations (*cf*. for example Golub *et al*., 1999). In Guyon *et al*. (2002) SVM-RFE is compared with such correlation methods as well as with a naïve ranking method. The authors only make use of $\left\{ w_{dj}^2, j = 1,2,\ldots, p \right\}$ as selection criteria (since they argue this to be a sensible choice considering the data sets at hand), and find that SVM-RFE generally outperforms selection based on the correlation- and naïve ranking techniques.

We now discuss contributions in the literature that are connected to the Guyon *et al*. (2002) paper, firstly considering contributions which propose and evaluate relatively small modifications to the original SVM-RFE algorithm, specifically the use of other selection criteria. More sophisticated criteria may yield post-selection SVMs with better properties, and more general criteria can widen the scope of RFE. We also discuss papers which make more comprehensive and fundamental changes, regarding for example the selection search strategy used, or core assumptions underlying the basic RFE algorithm.

An important contribution regarding the use of alternative selection criteria in SVM-RFE is the paper by Rakotomamonjy (2003). In this paper various selection criteria for RFE are proposed, such as $\|\boldsymbol{w}\|^2$, and criteria based on generalisation error bounds (for example the radius/margin bound and the span estimate). For each of the criteria two methods of determining the variable which should be eliminated at each step are investigated: a zero-order method which eliminates the variable whose omission optimises the criterion, as well as a first-order method, which optimises the first derivative of the criterion. Rakotomamonjy (2003) points out that the zero-order method based on optimising $\|\boldsymbol{w}\|^2$ is equivalent to the RFE method suggested for non-linear SVMs by Guyon *et al*. (2002). He investigates RFE using the various criteria on two simulated data sets, as well as on four real-world data sets. An SVM based on variables selected by means of correlation methods

is also evaluated in the study. In general, the error rates of the SVM classifiers based on the selected subsets are lower (in some cases markedly so) than the error rates of the SVM classifier using all the input variables. Although there is no single criterion which consistently yields a classifier with the lowest error rate, the zero and first order criteria based on $\|\boldsymbol{w}\|^2$ generally perform well.

It should be noted that prior to the Rakotomamonjy (2003) paper, work on selection criteria derived from (SVM) generalisation error bounds had already been reported. A first example is the paper by Weston *et al*. (2001), where the authors make use of scaling factors and the radius-margin bound on the SVM generalisation error in combination with a gradient descent algorithm for variable selection. A second example is found in Weston *et al*. (2003). In this paper the number of non-zero elements in the SVM weight vector is used as selection criterion. Denote this so-called *zero-norm* selection criterion by $\|\boldsymbol{w}\|_0 = card\{w_i | w_i \neq 0\}$. The zero-norm criterion is minimised over input variable subsets, and the subset for which $\|\boldsymbol{w}\|_0$ is a minimum is selected. This is not an easy minimisation problem to solve. Bradley *et al*. (1998) and Bradley and Mangasarian (1998) developed a method for this purpose. Weston *et al*. (2003) also developed a minimisation procedure (l_2-*AROM*) which they then compared with the method of Bradley *et al*. (1998) and Bradley and Mangasarian (1998). The method by Bradley and Mangasarian (1998) solves an approximation to the above minimisation problem, and is referred to as *FSV* selection. In addition, l_2-*AROM* and *FSV* are compared to SVM-RFE (Guyon *et al*., 2002), the gradient descent method minimising the radius-margin bound (Weston *et al*., 2001), and naïve ranking using correlation coefficients. Summarising their findings, Weston *et al*. (2003) conclude that only SVM-RFE and the gradient descent method minimising the radius-margin bound are acceptable for variable selection when groups are not necessarily linearly separable.

In Louw and Steel (2006) RFE is extended to KFDA (KFDA-RFE). The authors consider the use of the alignment and of the Rayleigh coefficient as selection criteria in an RFE scheme, and then evaluate KFDA-RFE using simulated data and two benchmark data sets

(the heart disease and breast cancer data sets in Rätsch *et al*., 2001). In their conclusion it is stated that '*using either the Rayleigh coefficient or the alignment as selection criterion in KFDA-RFE, in the vast majority of cases leads to more parsimonious models with higher classification accuracy than using the model based on all the input variables*'.

Regarding changes related to the RFE search strategy, one may choose to allow re-entrance of input variables eliminated during previous steps of the algorithm. One can for example adjust the original SVM-RFE algorithm to no longer implement a backward elimination strategy, but rather forward or stepwise selection. Fujarewicz and Wiench (2003), for example, compared the performance of a recursive forward selection or replacement strategy (RFR) with ordinary RFE in real life classification contexts. A choice between RFE and RFR was however not that apparent and seemed to depend on the sizes of the gene subsets under consideration. RFE appeared to outperform RFR when gene subsets were large.

Recall that the originally proposed SVM-RFE assumes a known value for the optimal number of input variables to retain. Furlanello *et al*. (2003) treat the more realistic scenario of an unknown optimal dimension for the final classifier in an RFE context.

Finally in this section, note that there are several papers in the literature illustrating the successful application of RFE. See for example Aliferis *et al*. (2003), Brown *et al*. (2000), Furey *et al*. (2000), Komura *et al*. (2004), Notterman *et al*. (2001), and Schummer *et al*. (1999). Many researchers also comment on the good performance of RFE in many contexts (*cf*. Krishnapuram *et al*., 2004, Ambroise *et al*., 2002, and Li *et al*., 2002) and subsequently use RFE to benchmark new proposals. Comprehensive discussions of SVM-RFE may also be found in several textbooks, for example McLachlan *et al*. (2004) and Chen *et al*. (2004). An interesting application of RFE in penalised logistic regression is given in Zhu and Hastie (2004).

5.3 *SELECTION CRITERIA IN BACKWARD ELIMINATION*

In this section we discuss several selection criteria that can be used in a backward elimination strategy. During each step we seek to quantify sensitivity of a criterion to elimination of the individual input variables in the model at that stage. For this purpose we follow Rakotomamonjy (2003) and optimise values of a criterion in one of two ways: in a so-called *zero-order* approach we optimise values of the criterion itself, while in a *first-order* approach we optimise values of the derivative of the criterion with respect to input variables in the model.

Section 5.3.1 is devoted to a discussion of $\|\boldsymbol{w}\|^2$ as a zero-order selection criterion, while in Section 5.3.2 we discuss this quantity as a first-order criterion. During the discussion in Section 5.3.2 we introduce results that are used later on to derive first-order versions of other criteria. In Section 5.3.3 we introduce several further selection criteria for RFE: the Rayleigh coefficient which is applicable in KFDA, and algorithm-independent criteria, *viz*. the alignment (*A*), the sum of the similarites between pairs of inputs belonging to opposite groups (*SS*), the variation ratio (*VA*), and differences between the two group means in feature space (*G*). These algorithm-independent criteria have the advantage that the post-selection classifier may be any kernel classifier. For all the criteria in Section 5.3.3 we discuss both the zero- and the first-order form.

5.3.1 *THE SQUARED NORM OF THE SVM WEIGHT VECTOR: ZERO-ORDER FORM*

Consider an SVM discriminant function, $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b$, where the weight vector $\boldsymbol{w}$ is given by $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\boldsymbol{x}_i)$, with $\alpha_1, \alpha_2, \ldots, \alpha_n$ output from the SVM algorithm. It follows that $\|\boldsymbol{w}\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with k the kernel function. How can we use $\|\boldsymbol{w}\|^2$ as RFE-criterion? We answer this question first from a zero-order perspective,

dealing with the first-order version in Section 5.3.2. Consider therefore an RFE strategy where at each step we optimise $\|\boldsymbol{w}\|^2$ with respect to omission of every individual variable which is still in the model. Recall that in Section 4.7.1, it was indicated that by 'optimise' in this context it is meant that one should proceed in an RFE strategy by sequentially omitting variables to *maximise* $\|\boldsymbol{w}\|^2$. Empirical evidence lead to the conclusion that RFE based on maximising $\|\boldsymbol{w}\|^2$ may indeed be a worthwhile proposal.

5.3.2 *THE SQUARED NORM OF THE SVM WEIGHT VECTOR: FIRST-ORDER FORM*

In this section we describe a way in which the first-order version of $\|\boldsymbol{w}\|^2$ can be used as selection criterion in an RFE scheme. Much of the discussion is based on contributions in the important papers by Rakotomamonjy (2002 and 2003) and Chapelle *et al*. (2004). The results introduced in this section are also used in Section 5.3.3 when we derive the first-order versions of several other selection criteria.

The basic idea underlying the use of a first-order selection criterion in RFE is to discard variables with respect to which the criterion is relatively insensitive. This broadly implies that we have to consider the derivative of the criterion with respect to each of the variables (still) in the model, and eliminate the variable corresponding to the smallest absolute derivative. If we wish to implement this idea, it is necessary to have computable expressions for the relevant derivatives. We proceed with a discussion of how expressions for these derivatives can be found in the case of $\|\boldsymbol{w}\|^2$.

Consider therefore $\|\boldsymbol{w}\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\widetilde{\alpha}_i\,\widetilde{\alpha}_j\,y_i\,y_j\,k\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)$ and suppose we wish to differentiate this quantity with respect to variable X_l. In this we have to keep in mind that both $\boldsymbol{x}_i$ and

$\boldsymbol{x}_j$ contain an observation on X_l, and that in finding $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_n$ we optimised over quantities depending on X_l. We require modification of our notation: thus far we have indicated an observation on the l^{th} input variable by x_l. In keeping with the notation used in the literature, in this section we will however associate a new interpretation with x_l, and therefore now use v_l to represent a measurement on variable X_l. According to Rakotomamonjy (2002) two approaches are possible:

i. we can differentiate directly with respect to X_l, using the results provided below, or

ii. we can introduce a so-called scale factor for each variable and then perform the differentiation with respect to the relevant scale factor.

We discuss both these approaches, starting with *i*.

The following theoretical result is required.

LEMMA 5.1: ***THE DERIVATIVE OF A QUADRATIC OPTIMISATION OBJECTIVE FUNCTION AT ITS OPTIMISING ARGUMENT***

Let $\boldsymbol{v}_\theta$ be an $n \times 1$ vector, and $\boldsymbol{P}_\theta$ an $n \times n$ matrix, and suppose $\boldsymbol{P}_\theta$ is smoothly dependent on θ. Consider the following optimisation problem:

$$\max_{\boldsymbol{\alpha} \in F} \left\{ \boldsymbol{\alpha}' \boldsymbol{v}_\theta - \tfrac{1}{2} \boldsymbol{\alpha}' \boldsymbol{P}_\theta \boldsymbol{\alpha} \right\} \tag{5.4}$$

conditional on $F = \left\{ \boldsymbol{\alpha} : \boldsymbol{\alpha}' \boldsymbol{b} = c ; \boldsymbol{\alpha} \geq \boldsymbol{0} \right\}$.

If $\tilde{\boldsymbol{\alpha}}$ is the vector where the maximum in (5.4) is attained, we may differentiate the objective function with respect to θ as if $\tilde{\boldsymbol{\alpha}}$ does not depend on θ. Therefore, we may write

$$\frac{\partial\left\{\boldsymbol{\alpha}'\boldsymbol{v}_\theta - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{P}_\theta\boldsymbol{\alpha}\right\}}{\partial\theta} = \widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{v}_\theta}{\partial\theta} - \frac{1}{2}\widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{P}_\theta}{\partial\theta}\widetilde{\boldsymbol{\alpha}}\,. \tag{5.5}$$

□

Clearly the SVM quadratic optimisation problem can be written as in (5.4): setting $\boldsymbol{v}_\theta = \boldsymbol{1}$ and $\boldsymbol{P}_\theta = \boldsymbol{K}_\theta^y$ with $\left\{\boldsymbol{K}_\theta^y\right\}_{ij} = y_i y_j k_{ij}$, $\boldsymbol{b} = \boldsymbol{y}$, $c = 0$ and $\theta = x_i$, we obtain

$$\max_{\boldsymbol{\alpha}\in F}\left\{\,\boldsymbol{\alpha}'\boldsymbol{1} - \tfrac{1}{2}\boldsymbol{\alpha}'\boldsymbol{K}_\theta^y\boldsymbol{\alpha}\,\right\}. \tag{5.6}$$

Note that (5.6) is identical to (4.13) in Problem 4.3, *i.e.* we get the dual optimisation problem for the SVM. According to Chapelle *et al*. (2004) it can be shown that $\frac{1}{2}\|\boldsymbol{w}\|^2 = \boldsymbol{\alpha}'\boldsymbol{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{K}_\theta^y\boldsymbol{\alpha}$ (*cf*. also Vapnik, 1998). We can therefore apply (5.5) to obtain

$$\begin{aligned}
\frac{\partial\|\boldsymbol{w}\|^2}{\partial x_l} &= \frac{2\partial\left[\widetilde{\boldsymbol{\alpha}}\boldsymbol{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{K}_\theta^y\widetilde{\boldsymbol{\alpha}}\right]}{\partial x_l} \\
&= 2\left[\widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{1}}{\partial x_l} - \frac{1}{2}\widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{K}_\theta^y}{\partial x_l}\widetilde{\boldsymbol{\alpha}}\right] \\
&= 2\widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{1}}{\partial x_l} - \widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{K}_\theta^y}{\partial x_l}\widetilde{\boldsymbol{\alpha}} = -\widetilde{\boldsymbol{\alpha}}'\frac{\partial\boldsymbol{K}_\theta^y}{\partial x_l}\widetilde{\boldsymbol{\alpha}} \\
&= -\sum_j\sum_k \widetilde{\alpha}_j\widetilde{\alpha}_k y_j y_k \frac{\partial k\left(\boldsymbol{x}_j, \boldsymbol{x}_k\right)}{\partial x_l},
\end{aligned} \tag{5.7}$$

as given in Rakotomamonjy (2003, *p*. 12) and Chapelle *et al*. (2004, *p*. 16). At this point the reader should note that it is also possible to arrive at the same result without the use of Lemma 5.1. Such an alternative approach may be found in Chapelle *et al*. (2004, *p*. 17).

The term $\partial k(\boldsymbol{x}_j, \boldsymbol{x}_k)/\partial x_l$ and the meaning of x_l is explained in the appendix of Rakotomamonjy (2002). We follow the author by re-defining the bivariate kernel function $k : \Re^p \times \Re^p \to \Re$ using $\hat{k} : \Re^{2p} \to \Re$. That is, we write

$$\begin{aligned} k(\boldsymbol{x}_j, \boldsymbol{x}_k) &= k\left(\left[x_{j1}, x_{j2}, \ldots, x_{jp}\right]', \left[x_{k1}, x_{k2}, \ldots, x_{kp}\right]' \right) \\ &= \hat{k}\left(x_1, x_2, \ldots, x_l, \ldots x_p, x_{p+1}, x_{p+2}, \ldots, x_{p+l}, \ldots, x_{2p}\right) \\ &= \hat{k}(\boldsymbol{x}). \end{aligned} \tag{5.8}$$

Importantly, note the distinction between $\boldsymbol{v} : p \times 1$ and $\boldsymbol{x} : 2p \times 1$ in this section. Here v_l refers to the l^{th} of the p original input variables which occur in both arguments (or input patterns) in the original kernel function (k). The symbols x_l and x_{l+p} however are two observations (belonging to two different input patterns) of the l^{th} of the p input variables. That is, x_l and x_{l+p} are two distinct observations of variable X_l.

The following definition is from the theory of differentiation of multivariate functions.

DEFINITION 5.1: ***DIFFERENTIABILITY OF A MULTIVARIATE FUNCTION***

A function $f : \Re^{2p} \to \Re$ is differentiable at a point $\boldsymbol{a} \in \Re^{2p}$ if

$$f(\boldsymbol{a} + \boldsymbol{h}) = f(\boldsymbol{a}) + \left(d_{\boldsymbol{a}} f\right)(\boldsymbol{h}) + o(\|\boldsymbol{h}\|) \tag{5.9}$$

where $o(\|\boldsymbol{h}\|) \to 0$ as $\boldsymbol{h} : 2p \times 1 \to \boldsymbol{0}$. In (5.9), $d_{\boldsymbol{a}} f : \Re^{2p} \to \Re$, *i.e.* $d_{\boldsymbol{a}} f$ is a function from $\Re^{2p}$ to $\Re$. □

We can now also define the gradient of a multivariate function $f : \Re^{2p} \to \Re$.

DEFINITION 5.2: ***THE GRADIENT OF A MULTIVARIATE FUNCTION***

The gradient of $f : \Re^{2p} \to \Re$ in a point $\boldsymbol{a} \in \Re^{2p}$ is the 2 p-vector $\nabla_{\boldsymbol{a}} f$ with i^{th} element the partial derivative of f with respect to x_i in the point $\boldsymbol{a}$, *i.e.* $(\nabla_{\boldsymbol{a}} f)_i = \dfrac{\partial}{\partial x_i} f(\boldsymbol{x})\Big|_{\boldsymbol{x} = \boldsymbol{a}}$, also denoted by $D_i f(\boldsymbol{a}) = \dfrac{\partial}{\partial x_i} f(\boldsymbol{x})\Big|_{\boldsymbol{x} = \boldsymbol{a}}$.

There is a relation between $\nabla_{\boldsymbol{a}} f$ and the function $d_{\boldsymbol{a}} f$ in (5.9), *viz.* $(d_{\boldsymbol{a}} f)(\boldsymbol{h}) = \langle \nabla_{\boldsymbol{a}} f, \boldsymbol{h} \rangle = \sum_{i=1}^{2p} h_i \dfrac{\partial}{\partial x_i} f(\boldsymbol{x})\Big|_{\boldsymbol{x} = \boldsymbol{a}}$. Note also that we can write $\nabla_{\boldsymbol{a}} f = \sum_{i=1}^{2p} D_i f(\boldsymbol{a})\, \boldsymbol{e}_i$, where $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{2p}\}$ is the well known canonical basis in $\Re^{2p}$.

We now consider our application where f is a bivariate function of two $p-$component vectors, *i.e.* $f : \Re^p \times \Re^p \to \Re$ with value $f(\boldsymbol{x}_j, \boldsymbol{x}_k)$ when applied to the pair $(\boldsymbol{x}_j, \boldsymbol{x}_k)$ of input patterns. As explained earlier we can also view such a function as a real-valued function defined on $\Re^{2p}$.

DEFINITION 5.3: ***THE GRADIENT OF A MULTIVARIATE FUNCTION WITH RESPECT TO AN INPUT VARIABLE***

The gradient of $f : \Re^{2p} \to \Re$ with respect to variable v_i at a point $\boldsymbol{a} \in \Re^{2p}$ is the $2p-$component vector

$$\nabla_{\boldsymbol{a}}^{v_i} f \;= \frac{\partial}{\partial x_i} f(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{a}}\, \boldsymbol{e}_i \;+\; \frac{\partial}{\partial x_{p+i}} f(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{a}}\, \boldsymbol{e}_{p+i}\,. \tag{5.10}$$

It is clear from (5.10) that

$$\left\| \nabla_{\boldsymbol{a}}^{v_i} f \right\|^2 = D_i^2 f(\boldsymbol{a}) + D_{p+i}^2 f(\boldsymbol{a}). \tag{5.11}$$

We would now like to apply these results to the case where the function f is $\|\boldsymbol{w}\|^2$, the squared norm of the SVM weight vector. This requires differentiation of the kernel function. Consider therefore the kernel function evaluated at the pair $(\boldsymbol{x}_j, \boldsymbol{x}_k)$, where $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ are two input patterns. The point $\boldsymbol{a}: 2p \times 1$ above is taken to be the vector $[\boldsymbol{x}_j' \;\; \boldsymbol{x}_k']'$. Now $\frac{\partial}{\partial x_l} k(\boldsymbol{x}_j, \boldsymbol{x}_k) = -2\gamma\,(x_{jl} - x_{kl})\, k(\boldsymbol{x}_j, \boldsymbol{x}_k)$, with a similar expression for $\frac{\partial}{\partial x_{p+l}} k(\boldsymbol{x}_j, \boldsymbol{x}_k)$. It follows from (5.7) and (5.11) that the first-order $\|\boldsymbol{w}\|^2$ sensitivity selection criterion based on direct differentiation with respect to variable v_l and using the Gaussian kernel to quantify similarities between input patterns, is

$$J(l) \;=\; \left\| \nabla_{\boldsymbol{a}}^{v_i} \|\boldsymbol{w}\|^2 \right\|^2 \;=\; \left(2\gamma \sum_j \sum_k \tilde{\alpha}_j \tilde{\alpha}_k y_j y_k (x_{jl} - x_{kl}) k(\boldsymbol{x}_j, \boldsymbol{x}_k) \right)^2 \tag{5.12}$$

(*cf.* also Rakotomamonjy, 2002, *p*.11).

Now consider the second approach for deriving a first-order version of $\|\boldsymbol{w}\|^2$ as selection criterion. This approach makes use of differentiation with respect to virtual scaling factors (*cf.* Rakotomamonjy, 2003). Let $\boldsymbol{s}: p \times 1 = [s_1, s_2, \ldots, s_p]'$ be a vector of virtual scaling factors, with one scaling factor for each of the variables. We consider the kernel function

in the form $k(\boldsymbol{x}_j, \boldsymbol{x}_k) = k(\boldsymbol{s} \cdot \boldsymbol{x}_j, \boldsymbol{s} \cdot \boldsymbol{x}_k)$, where $\boldsymbol{s} \cdot \boldsymbol{x}$ denotes the component-wise vector product, *i.e.* $\boldsymbol{s} \cdot \boldsymbol{x} = [s_1 x_1, s_2 x_2, \ldots, s_p x_p]'$. This view of the kernel function is valid since we later put $\boldsymbol{s} = \boldsymbol{1}$, the $p-$component vector with 1 in every position.

Consider once again $\|\boldsymbol{w}\|^2 = 2(\boldsymbol{\alpha}'\boldsymbol{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{K}_\theta^y \boldsymbol{\alpha})$, where the elements of the $n \times n$ matrix $\boldsymbol{K}_\theta^y$ are now viewed as $\{\boldsymbol{K}_\theta^y\}_{ij} = y_i y_j \, k(\boldsymbol{s} \cdot \boldsymbol{x}_i, \boldsymbol{s} \cdot \boldsymbol{x}_j)$. If we now differentiate $\|\boldsymbol{w}\|^2$ with respect to the l^{th} scaling factor s_l, we obtain

$$\frac{\partial \|\boldsymbol{w}\|^2}{\partial s_l} = -\tilde{\boldsymbol{\alpha}}' \frac{\partial \boldsymbol{K}_\theta^y}{\partial s_l} \tilde{\boldsymbol{\alpha}} = -\sum_j \sum_k \tilde{\alpha}_j \tilde{\alpha}_k y_j y_k \frac{\partial k(\boldsymbol{s} \cdot \boldsymbol{x}_j, \boldsymbol{s} \cdot \boldsymbol{x}_k)}{\partial s_l}. \tag{5.13}$$

Now,

$$\begin{aligned} \frac{\partial}{\partial s_l} k(\boldsymbol{s} \cdot \boldsymbol{x}_j, \boldsymbol{s} \cdot \boldsymbol{x}_k) &= \frac{\partial}{\partial s_l} exp\left(-\gamma \|\boldsymbol{s} \cdot \boldsymbol{x}_j - \boldsymbol{s} \cdot \boldsymbol{x}_k\|^2 \right) \\ &= -2\gamma s_l (x_{jl} - x_{kl})^2 \, k(\boldsymbol{s} \cdot \boldsymbol{x}_j, \boldsymbol{s} \cdot \boldsymbol{x}_k) \end{aligned}$$

and if we put $\boldsymbol{s} = \boldsymbol{1}$, this simplifies to

$$\frac{\partial}{\partial s_l} k(\boldsymbol{s} \cdot \boldsymbol{x}_j, \boldsymbol{s} \cdot \boldsymbol{x}_k)\Big|_{\boldsymbol{s} = \boldsymbol{1}} = -2\gamma s_l (x_{jl} - x_{kl})^2 \, k(\boldsymbol{s} \cdot \boldsymbol{x}_j, \boldsymbol{s} \cdot \boldsymbol{x}_k).$$

The scaling factor version of the first-order $\|\boldsymbol{w}\|^2$ criterion is now defined to be

$$J(l) = \left| \frac{\partial}{\partial s_l} \|\boldsymbol{w}\|^2 \Big|_{\boldsymbol{s} = \boldsymbol{1}} \right| = \left| -2\gamma \sum_j \sum_k \tilde{\alpha}_j \tilde{\alpha}_k y_j y_k (x_{jl} - x_{kl})^2 k(\boldsymbol{x}_j, \boldsymbol{x}_k) \right| \tag{5.14}$$

In Rakotomamonjy (2002) the sensitivity criterion in (5.14) mostly outperformed the criterion in (5.12). Hence the only selection criteria investigated in Rakotomamonjy (2003) are derivatives calculated with respect to scale parameters. In light of this we will also restrict attention to criteria found through differentiation with respect to scale factors.

5.3.3 *FURTHER SENSITIVITY SELECTION CRITERIA*

Following a backward elimination strategy, Rakotomamonjy (2003) investigates the use of quantities based on the radius-margin and span bounds on the SVM generalisation error as sensitivity selection criteria. These criteria can however only be used in the context of SVMs, and they were found by Rakotomamonjy (2003) to perform worse than $\|\boldsymbol{w}\|^2$. In this section we therefore discuss several other criteria which can be used for selection prior to the application of any type of kernel classifier.

Any of the algorithm-independent criteria in Chapter 4 (*viz*. *A*, *G*, *SS* or *VA*) can be used in an RFE scheme of selection. In most cases, finding derivatives of these criteria is much simpler than obtaining the gradient vector with respect to the squared norm of the SVM weight vector. Note that in this section we also discuss use of the Rayleigh quotient as sensitivity selection criterion within the context of KFDA. Importantly, note that for all the criteria we assume the use of a Gaussian kernel function. That is, we always assume

$$\left\{ k_{ij} = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = exp\left(-\gamma \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2 \right) \; i, j = 1,2,\ldots,n \right\}, \tag{5.15}$$

leading to

$$\left\{ \frac{\partial k_{ij}}{\partial v_k} = -2\gamma \left(x_{ij} - x_{jk} \right)^2 k_{ij} \;\; i, j = 1,2,\ldots,n \right\} \quad . \tag{5.16}$$

It is of course also possible to find the first order forms of selection criteria when other kernel functions are used.

Since *G* and *SS* are the simplest selection criteria, we start by finding their first order forms.

THE SUM OF SIMILARITIES

In Chapter 4 the sum of similarities criterion was defined as the sum of the entries in the $n_1 \times n_2$ sub-matrix of $\boldsymbol{K}$, *i.e.* $SS = \sum_{i \in I_1, j \in I_2} k_{ij}$. In order to measure the sensitivity of *SS* with respect to input variable X_k, we simply require

$$\frac{\partial SS}{\partial v_k} = -2\gamma \sum_{i \in I_1, j \in I_2} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \,. \tag{5.17}$$

THE DIFFERENCE BETWEEN GROUP MEANS

We showed in Chapter 4 that the squared difference between the two group means in feature space can easily be calculated using entries in the empirical kernel matrix. Let the difference in group means be denoted by G, *i.e.*

$$G = \sum_{i \in I_1, j \in I_1} k_{ij} \Big/ n_1^2 + \sum_{i \in I_2, j \in I_2} k_{ij} \Big/ n_2^2 \; - 2 \sum_{i \in I_1, j \in I_2} k_{ij} \Big/ n_1 n_2 \,. \tag{5.18}$$

The first order form of the selection criterion calculated with respect to variable X_k is

$$\begin{aligned} \frac{\partial G}{\partial v_k} &= \sum_{i \in I_1, j \in I_1} \frac{\partial k_{ij}}{\partial v_k} \Big/ n_1^2 + \sum_{i \in I_2, j \in I_2} \frac{\partial k_{ij}}{\partial v_k} \Big/ n_2^2 - 2 \sum_{i \in I_1, j \in I_2} \frac{\partial k_{ij}}{\partial v_k} \Big/ n_1 n_2 \\ &= -2\gamma \left\{ \sum_{i, j \in I_1} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_1^2 + \sum_{i, j \in I_2} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_2^2 \right. \end{aligned}$$

$$-2\sum_{i\in I_1, j\in I_2}\left(x_{ik}-x_{jk}\right)^2 k_{ij}\Big/ n_1 n_2\Bigg\}. \tag{5.19}$$

Note that both Expressions (5.18) and (5.20) will feature in the derivation of the first-order form of the alignment, of the Rayleigh quotient and also of the variation ratio.

THE ALIGNMENT

Consider the alignment selection criterion, *viz*.

$$A=\frac{\sum_{i,j\in I_1} k_{ij}+\sum_{i,j\in I_2} k_{ij}-2\sum_{i\in I_1, j\in I_2} k_{ij}}{n\sqrt{\sum_{i,j} k_{ij}^2}}. \tag{5.20}$$

Let the numerator and denominator of (5.21) be denoted by A_N and A_D respectively, *i.e.* let $A = A_N / A_D$. Hence we require

$$\frac{\partial A}{\partial v_k}=\left[A_D\frac{\partial A_N}{\partial v_k}-A_N\frac{\partial A_D}{\partial v_k}\right]\Big/ A_D^2, \tag{5.21}$$

where

$$\begin{aligned}\frac{\partial A_N}{\partial v_k}&=\sum_{i,j\in I_1}\frac{\partial k_{ij}}{\partial v_k}+\sum_{i,j\in I_2}\frac{\partial k_{ij}}{\partial v_k}-2\sum_{i\in I_1, j\in I_2}\frac{\partial k_{ij}}{\partial v_k}\\ &=-2\gamma\left\{\sum_{i,j\in I_1}\left(x_{ij}-x_{jk}\right)^2 k_{ij}+\sum_{i,j\in I_2}\left(x_{ij}-x_{jk}\right)^2 k_{ij}-2\sum_{i\in I_1, j\in I_2}\left(x_{ij}-x_{jk}\right)^2 k_{ij}\right\},\end{aligned} \tag{5.22}$$

and

$$\frac{\partial A_D}{\partial v_k}=n\left(\sum_{i,j=1}^{n} k_{ij}^2\right)^{-\frac{1}{2}}.\sum_{i,j=1}^{n} k_{ij}\left(\frac{\partial k_{ij}}{\partial v_k}\right)$$

$$= -2n\gamma\left(\sum_{i,j=1}^{n} k_{ij}^2\right)^{-\frac{1}{2}} \sum_{i,j=1}^{n} \left(x_{ik} - x_{jk}\right)^2 k_{ij}^2 \ . \tag{5.23}$$

Substituting the derivatives in (5.23) and (5.24) into (5.22), the first-order form of the alignment criterion can now easily be obtained. We write the criterion in its unsimplified form, *viz*.

$$\frac{\partial A}{\partial v_k} = \left\{ -2\gamma\left(\sum_{i,j=1}^{n} k_{ij}^2\right)^{\frac{1}{2}} \left(\sum_{i,j\in I_1} \left(x_{ij} - x_{jk}\right)^2 k_{ij} + \sum_{i,j\in I_2} \left(x_{ij} - x_{jk}\right)^2 k_{ij} - 2 \sum_{i\in I_1, j\in I_2} \left(x_{ij} - x_{jk}\right)^2 k_{ij} \right) \right.$$
$$\left. + \left(\sum_{i,j\in I_1} k_{ij} + \sum_{i,j\in I_2} k_{ij} - 2 \sum_{i\in I_1, j\in I_2} k_{ij} \right) \left(2\gamma\left(\sum_{i,j=1}^{n} k_{ij}^2\right)^{\frac{1}{2}} \sum_{i,j=1}^{n} \left(x_{ik} - x_{jk}\right)^2 k_{ij}^2 \right) \right\} \Bigg/ n \sum_{i,j=1}^{n} k_{ij}^2 \ . \tag{5.24}$$

THE RAYLEIGH QUOTIENT

For simpler derivation of a sensitivity selection criterion based on the Rayleigh quotient, consider $log\ R = log[\boldsymbol{\alpha}'\boldsymbol{M}\boldsymbol{\alpha}] - log[\boldsymbol{\alpha}'\boldsymbol{N}\boldsymbol{\alpha}]$, where as before,

$$M = \left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2\right)\left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2\right)' \tag{5.25}$$

and

$$\boldsymbol{N} = \boldsymbol{K}_1\boldsymbol{K}_1{}' - \boldsymbol{K}_1\boldsymbol{1}_{n_1}\boldsymbol{K}_1 + \boldsymbol{K}_2\boldsymbol{K}_2{}' - \boldsymbol{K}_2\boldsymbol{1}_{n_2}\boldsymbol{K}_2, \tag{5.26}$$

with $\boldsymbol{1}_{n_j}$ an $n_j \times n_j$ matrix with common element $1/n_j$, and $\boldsymbol{K}_j$ an $n \times n_j$ submatrix of $\boldsymbol{K}$; $j = 1,2$. That is, $\boldsymbol{K} = [\boldsymbol{K}_1 \mid \boldsymbol{K}_2]$. In order to measure the sensitivity of $log\ R$ with respect to the k^{th} input variable, we require

$$\frac{\partial logR}{\partial v_k} = \frac{1}{\boldsymbol{\alpha}'\boldsymbol{M}\boldsymbol{\alpha}}\left\{\frac{\partial[\boldsymbol{\alpha}'\boldsymbol{M}\boldsymbol{\alpha}]}{\partial v_k}\right\} - \frac{1}{\boldsymbol{\alpha}'\boldsymbol{M}\boldsymbol{\alpha}}\left\{\frac{\partial[\boldsymbol{\alpha}'\boldsymbol{N}\boldsymbol{\alpha}]}{\partial v_k}\right\}$$

$$= \frac{1}{\boldsymbol{\alpha}'\boldsymbol{M}\boldsymbol{\alpha}}\left\{\boldsymbol{\alpha}'\left[\frac{\partial \boldsymbol{M}}{\partial v_k}\right]\boldsymbol{\alpha}\right\} - \frac{1}{\boldsymbol{\alpha}'\boldsymbol{N}\boldsymbol{\alpha}}\left\{\boldsymbol{\alpha}'\left[\frac{\partial \boldsymbol{N}}{\partial v_k}\right]\boldsymbol{\alpha}\right\}. \qquad (5.27)$$

The l^{th} element in $\frac{\partial \boldsymbol{M}}{\partial v_k}$ is

$$\left[\frac{\partial}{\partial v_k}\left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2\right)\right]\left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2\right)' + \left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2\right)\left[\frac{\partial}{\partial v_k}\left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2\right)\right]'$$

$$= \left[\sum_{j\in I_1}\frac{\partial}{\partial v_k}\boldsymbol{K}_{lj}\Big/ n_1 - \sum_{j\in I_2}\frac{\partial}{\partial v_k}\boldsymbol{K}_{lj}\Big/ n_2\right]\left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_1\right)' + \left(\overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_1\right)\left[\sum_{j\in I_1}\frac{\partial}{\partial v_k}\boldsymbol{K}_{lj}\Big/ n_1 - \sum_{j\in I_2}\frac{\partial}{\partial v_k}\boldsymbol{K}_{lj}\Big/ n_2\right]'$$

Since the $(il)^{th}$ element in $\boldsymbol{K}_j \boldsymbol{1}_{n_j} \boldsymbol{K}_j$ is $\frac{1}{n_j}\left(\sum_{r\in I_1} k_{ir}\right)\left(\sum_{r\in I_j} k_{lr}\right)$, the $(il)^{th}$ element in $\frac{\partial \boldsymbol{N}}{\partial v_k}$ is

$$\sum_{j=1,2}\left\{\left[\frac{\partial \boldsymbol{K}_j}{\partial v_k}\boldsymbol{K}_j{}'\right]_{il} + \left[\boldsymbol{K}_j\frac{\partial \boldsymbol{K}_j{}'}{\partial v_k}\right]_{il} - \left(\sum_{r\in I_j}\frac{\partial k_{ir}}{\partial v_k}\right)\left(\sum_{r\in I_j} k_{lr}\right)\Big/ n_j - \left(\sum_{r\in I_j} k_{ir}\right)\left(\sum_{r\in I_j}\frac{\partial k_{lr}}{\partial v_k}\right)\Big/ n_j\right\}.$$

Substituting the above two expressions into (5.28) yields the first-order form of the selection criterion based on the KFDA Rayleigh quotient.

THE VARIATION RATIO

Very similar to the alignment, the first order form of the variation ratio selection criterion can be derived as follows. We denote the numerator of the variation ratio by R_N, and the denominator by R_D, *i.e.* $R_N = s_1^2 + s_2^2$ and $R_D = \left\| \overline{\boldsymbol{\Phi}}_1 - \overline{\boldsymbol{\Phi}}_2 \right\|^2$. Hence,

$$\frac{\partial R}{\partial v_k} = \left[R_D \frac{\partial R_N}{\partial v_k} - R_N \frac{\partial R_D}{\partial v_k} \right] \Big/ R_D^2 \,. \tag{5.28}$$

A formula for calculating the terms in R_N was given in Chapter 4. Using this we have

$$R_N = n - \sum_{i,j \in I_1} k_{ij} \Big/ n_1 - \sum_{i,j \in I_2} k_{ij} \Big/ n_2 \,. \tag{5.29}$$

Also in Chapter 4 we saw that R_D can easily be obtained using entries in the empirical kernel matrix, *viz*.

$$R_D = \sum_{i,j \in I_1} k_{ij} \Big/ n_1^2 + \sum_{i,j \in I_2} k_{ij} \Big/ n_2^2 - 2 \sum_{i \in I_1, j \in I_2} k_{ij} \Big/ n_1 n_2 \,. \tag{5.30}$$

The following derivatives are now easily obtained:

$$\frac{\partial R_N}{\partial v_k} = 2\gamma \left\{ \sum_{i,j \in I_1} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_1 + \sum_{i,j \in I_2} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_2 \right\} \tag{5.31}$$

$$\begin{aligned} \frac{\partial R_D}{\partial v_k} = -2\gamma \Bigg\{ & \sum_{i,j \in I_1} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_1^2 + \sum_{i,j \in I_2} \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_2^2 \\ & - 2 \sum_{i \in I_1, j \in I_2} 2 \left(x_{ik} - x_{jk} \right)^2 k_{ij} \Big/ n_1 n_2 \Bigg\} \end{aligned} \tag{5.32}$$

5.4 *MONTE CARLO SIMULATION STUDY*

When applied in an RFE selection strategy, the performances of selection criteria discussed in the previous section may differ from their one-step selection performances in Chapter 4. In the next section we therefore report on a fairly extensive Monte Carlo simulation study that was carried out in order to study the performance of the proposed variable selection criteria when they are used in combination with a backward elimination strategy.

We used the same parameter configurations and data generating strategy as discussed in Section 4.6.2. These differ from the setup in Section 3.4.1, in that we omitted all *NL* data sets, and always generated the relevant and irrelevant subsets of input variables to be uncorrelated (*i.e.* we se $\rho_{S\bar{S}} = 0$ throughout). The various levels for the correlation (amongst the relevant set of input variables), the number of relevant variables, and the training and test sample sizes amounted to 16 configurations to consider per data scenario (*viz*. the *LL*, *LS* and *NS* setups), and thus 48 configurations in total – each requiring its own simulation program. The output of each program were the average post-selection test errors (with their standard errors), and per input variable and selection criterion, the percentage of times that the particular variable was selected. The averages and percentages were calculated across 500 Monte Carlo simulation repetitions.

To obtain the output described in the previous paragraph, we performed the following actions in each simulation repetition. We started by generating a training and test data set using the comprehensive set of available input variables in $\mathcal{V}$. Now consider a given selection criterion. We applied this criterion to the training data, and identified a variable to discard. This process was continued until the predetermined number m of input variables remained. For each of these variables a selection frequency counter was increased. Finally, the relevant kernel classifier (KFDA and SVMs in the case of algorithm-independent criteria, and only the appropriate one of the two for the algorithm-dependent criteria) based on the identified variables was trained and used to classify the test data cases, thereby yielding a test error value. Averaging these results over the 500

repetitions yielded estimates of the probability of selecting the different variables, and the error rate associated with each criterion.

5.5 *RESULTS AND CONCLUSIONS*

The average test errors obtained in the *LL*, *LS* and *NS* data scenarios are given in Tables A.5 to A.28 in Appendix A.2.2. Standard errors ranged between 0.000 and 0.008, and were omitted from the tables. Four tables are reported for each data scenario: one corresponding to each of the sample sizes considered. Furthermore, each table is divided into four segments, thus providing for the different correlation structures and number of relevant input variables considered. The four sets of average test errors in each table correspond to the following configurations:

Table 5.1: Correlation structures and number of relevant variables considered

SMALL, MIXED** and **LARGE SAMPLES	***WIDE SAMPLES***
i. $\rho_S = 0,\ m = 1$ (out of $p = 10$)	*i.* $\rho_S = 0,\ m = 6$ (out of $p = 60$)
ii. $\rho_S = 0,\ m = 4$ (out of $p = 10$)	*ii.* $\rho_S = 0,\ m = 24$ (out of $p = 60$)
iii. $\rho_S = 0.7,\ m = 1$ (out of $p = 10$)	*iii.* $\rho_S = 0.7,\ m = 6$ (out of $p = 60$)
iv. $\rho_S = 0.7,\ m = 4$ (out of $p = 10$)	*iv.* $\rho_S = 0.7,\ m = 24$ (out of $p = 60$)

In turn, each of the four table segments consists of three rows. In the first row of each segment the average test errors pertaining to the classifier using the full set of available input variables appear in the *FULL* column, the test errors based on the classifier using only the subset of relevant variables appear in the *oracle* (*ORA*) column, and the average test errors that were obtained for the respective post-RFE selection classifiers are given in the remaining columns. These columns are indicated as follows: The algorithm-independent RFE selection criteria (*viz.* the alignment, the sum of dissimilarities, the difference in group means and the variation ratio) are respectively denoted by *A*, *G*, *SS* or *VA*. The notation is additionally augmented with a zero or one, depending on whether the

zero- or first order form of the criterion was used. For example, $A0$ refers to the zero-order alignment, while the first-order alignment is denoted by $A1$. Note also that in the case of the alignment criterion further distinction has to be made between the ordinary zero- and first order forms ($A0$ and $A1$) and their respective translated versions ($A0T$ and $A1T$). The two sets of algorithm-dependent RFE selection criteria (*viz*. the norm of the SVM weight vector and the KFDA Rayleigh coefficient) are denoted by N and R respectively. Once again a zero or one is added to the notation in order to differentiate between zero- and first order criteria. Values in the second row of each table segment depict the size of the average post-selection errors relative to the average error achieved by the oracle. For example, in the first segment of Table A.5 we see that the zero-order Rayleigh criterion ($R0$) yields a post-selection average KFDA test error of 0.145, compared to an oracle average test error of 0.134. Relative to the oracle, the $R0$ error is therefore $0.145/0.134 = 1.082$, indicating that selection based on the $R0$ criterion yields an average error which is 8.2% higher than the best one can possibly do. Rankings of the performances of the post-selection classifiers are based on the above relative errors, and are given in the third row (with a 1 indicating the next best performance after that of the oracle). For example, in the first segment of Table A.5, the ranking of $R0$ indicates RFE selection using the $R0$ criterion to be the best performer, followed by selection based on $A1T$, and then by selection using the $V0$ criterion.

In Appendix A, we first report the results obtained for the *LL* data sets (in Tables A.5-A.12), followed by the output obtained in the *LS* (in Tables A.13-A.20) and the *NS* case (in Tables A.21-A.28). Tables A.5-A.8, A.13-A.16 and A.21-A.24 report KFDA errors, whereas Tables A.9-A12, A.17-A20 and A.25-A.28 contain the average test errors when the post-selection classifiers evaluated were SVMs.

With exception of sets of results pertaining to large sample sizes, we see that the relative performance of the RFE selection criteria tend to vary much across the various correlation structures and number of relevant input variables considered. We therefore further summarise the small, mixed and wide sample results in Tables A.5-A.28 by averaging rows 1 and 2 over the four segments in each table (*i.e*. over the four levels of correlation

structures and the number of relevant variables). In each table we also average the relative errors in the second row of each segment, and based on the obtained average relative risks, obtain new relative rankings for the selection criteria. For each data scenario and sample size, the summary tables are provided below. The first and second rows in each summary table contains average errors and average relative errors across correlation structures and values of m. Newly determined relative ranks based on the average relative errors (across correlation structures and m-value)s are given in the third row. Note that our conclusions regarding the performances of RFE selection criteria will largely be based upon these relative rankings.

We first provide the set of summary tables pertaining to the *NS* data scenario (refer to Tables 5.2-5.7). These are given in pairs: each first table reports KFDA test errors, and each second table contains SVM test errors. The first pair of tables corresponds to the small sample case, whereas the second and third pairs of tables contain test errors obtained in the case of mixed and wide sample sizes, respectively. The same structure is followed in reporting the sets of summary tables corresponding to the *LL* and *LS* data scenarios (in Tables 5.8-5.13 and Tables 5.14-5.19 respectively).

Considering the results reported in Tables 5.2-5.19, we first ask whether it is possible to infer whether algorithm-independent or algorithm-dependent selection criteria are preferred. A clear pronouncement is possible for *NS* data, where the *R* criterion does exceptionally well for KFDA, although the *N* criterion for SVMs generally performs worse than most of the algorithm-independent criteria. For *LL* and *LS* data the algorithm-dependent criteria are seldom best, finishing mostly in the middle of the rankings. Note that one cannot conclude that the *R* criterion generally outperforms the *N* criterion. In fact, for lognormal data the *N* criterion is better.

Table 5.2: Average small sample *NS* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.185	1.181	1.426	1.290	1.365	1.266	1.301	1.9	1.317	2.698	2.110	1.421
2	1	9	4	7	3	5	10	6	12	11	8

Table 5.3: Average small sample *NS* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.316	1.361	1.356	1.234	1.296	1.207	1.263	1.707	1.305	2.353	1.873	1.302
7	9	8	2	4	1	3	10	6	12	11	5

Table 5.4: Average mixed sample *NS* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.111	1.006	1.739	1.046	1.848	1.282	1.011	1.739	1.025	1.739	1.739	1.505
5	1	8	4	12	6	2	9	3	10	11	7

Table 5.5: Average mixed sample *NS* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.106	1.187	2.265	1.043	2.273	1.276	1.021	2.161	1.031	2.277	2.258	1.524
4	5	10	3	11	6	1	8	2	12	9	7

Table 5.6: Average wide sample *NS* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.944	2.079	2.292	2.171	2.260	2.147	2.128	2.714	2.221	3.889	3.031	2.805
1	2	8	5	7	4	3	9	6	12	11	10

Table 5.7: Average wide sample *NS* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
2.150	3.672	2.901	2.746	2.381	2.235	2.373	5.410	2.579	2.64	3.95	2.190
1	9	8	7	5	3	4	10	6	12	11	2

Table 5.8: Average small sample *LL* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.224	2.253	1.207	1.168	1.205	1.168	1.155	1.839	1.171	2.328	1.532	2.147
6	10	5	2	4	2	1	8	3	11	7	9

Table 5.9: Average small sample *LL* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.197	1.221	1.208	1.165	1.204	1.144	1.162	1.930	1.183	2.433	1.559	2.257
5	8	7	3	6	1	2	10	4	12	9	11

Table 5.10: Average mixed sample *LL* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.110	1.724	1.322	1.299	1.151	1.285	1.012	1.696	1.010	1.239	1.519	1.255
3	12	9	8	4	7	2	11	1	5	10	6

Table 5.11: Average mixed sample *LL* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.030	1.046	1.402	1.347	1.189	1.295	1.012	1.897	1.011	1.294	1.653	1.325
3	4	10	9	5	7	2	12	1	6	11	8

Table 5.12: Average wide sample *LL* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
2.068	3.416	1.633	1.550	1.788	1.542	1.4	3.126	1.534	3.371	1.882	3.092
8	12	5	4	6	3	1	10	2	11	7	9

Table 5.13: Average wide sample *LL* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.591	1.749	1.505	1.439	1.650	1.549	1.461	2.888	1.528	3.307	1.692	3.428
6	9	3	1	7	5	2	10	4	11	8	12

Table 5.14: Average small sample *LS* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.500	1.502	1.583	1.384	1.485	1.300	1.484	1.915	1.532	2.295	1.922	1.808
5	6	8	2	4	1	3	10	7	12	11	9

Table 5.15: Average small sample *LS* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.351	1.263	1.532	1.262	1.386	1.210	1.416	1.746	1.346	2.098	1.696	1.601
5	3	8	2	6	1	7	11	4	12	10	9

Table 5:16: Average mixed sample *LS* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.393	1.066	1.632	1.359	1.632	1.465	1.056	1.619	1.111	1.632	1.632	1.632
5	2	8	4	9	6	1	7	3	10	11	12

Table 5:17: Average mixed sample *LS* KFDA relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.199	1.334	2.380	1.491	2.288	1.699	1.108	2.015	1.181	2.442	2.369	1.945
3	4	11	5	9	6	1	8	2	12	10	7

Table 5.18: Average wide sample *LS* KFDA relative- test errors and ranks

R0	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
3.347	2.344	2.915	2.761	4.648	2.772	2.523	3.604	2.760	5.040	4.013	3.634
7	1	6	4	11	5	2	8	3	12	10	9

Table 5.19: Average wide sample *LS* SVM relative- test errors and ranks

N0	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
1.944	2.079	2.292	2.171	2.260	2.147	2.128	2.714	2.221	3.889	3.031	2.805
1	2	8	5	7	4	3	9	6	12	11	10

We now consider the relative performance of the criteria in the different sample size scenarios. In the small sample cases the $A1T$ and $V0$ criteria perform very well: in all of these cases, except the *NS* KFDA results, one of these two criteria does best. Overall, the best performing criteria in mixed samples are $V0$ and $G0$: with exception of a single case, one of these two criteria always performs best. The relative performances are slightly more erratic in wide samples. Now the algorithm-dependent criteria perform very well in the *NS* and *LS* scenarios, while the $V0$ and $A1$ criteria also do well, especially in the *LL* data setup.

Regarding a choice between use of the zero- or first-order forms of the selection criteria, we consider only those that perform well. For the *G*, *N* and *V* criteria the zero-order forms are better, but for the *A* criterion, the first-order version performs best. A general conclusion cannot be made for the *R* criterion, except for *LL* data, where $R0$ dominates. It is interesting to observe that the greater complexity of the first-order versions does not generally yield superior results, and we are led to recommend use of the zero-order criteria.

5.6 *SUMMARY*

In this chapter we once again considered selection in feature space, but now combined with a specific selection strategy, *viz*. backward elimination. We started with a review of recursive feature elimination, as it is generally known in the literature. This was followed by a description of selection criteria for RFE. Following the precedent set in a standard paper in this area, we defined zero-order and first-order versions of each criterion. Several conclusions followed from the empirical study. Regarding a comparison of the algorithm-independent and -dependent criteria, we found that in *NS* data the *R* criterion was clearly best in KFDA, but not so the *N* criterion in SVMs. Interestingly, in *LL* and *LS* data, the algorithm-dependent criteria were outperformed by the algorithm-independent ones. Training sample size had a significant influence on the relative performances of the different criteria. Overall the alignment and variation ratio seem recommendable. The zero- and first-order versions of the different criteria were also compared. Generally, the benefit from the greater complexity of the first-order criteria was doubtful.

CHAPTER 6

VARIABLE SELECTION FOR SUPPORT VECTOR MACHINES: A TWO-STAGE APPROACH

6.1 *INTRODUCTION*

Up to this point in the thesis we have emphasised the importance of variable selection when kernel methods are used, and assuming the number of variables to include to be known, investigated the use of several selection criteria in combination with a backward elimination strategy. In this chapter we address the important problem of using the data to decide on the number of variables to include in the final model. As before, the number of relevant variables in a given scenario will be denoted by m, *i.e.* in simulation studies we generate data in such a way that the two groups differ from one another with respect to m of the p input variables. We can view the decision on the number of variables to use as one of estimating the value of m. Attention is restricted to input variable selection for SVMs in binary classification. Combining our proposal for estimating m with a strategy that can be used to search through the available models, we arrive at a complete proposal for input variable selection.

The chapter is structured as follows. In Section 6.2 we review the literature related to the work presented in this chapter. Section 6.3 contains an illustration of the importance of accurate estimation of m, as well as a discussion of our proposal in this regard. In Section 6.4 we start with an evaluation of our proposal: we report the results of a simulation investigation where we studied the behaviour of the criterion which we propose for deciding on a value of m as a function of the number of variables included in a model. At each model dimension we purposely consider the model containing the best (relevant)

variables, possibly combined with one or more irrelevant variables. The idea behind this is to see whether the proposed criterion reaches an 'optimal' value at the correct model dimension. We empirically investigate various aspects of the performance of the proposed criterion when it is combined with other previously discussed criteria, as well as with a strategy for moving through the space of different variable subsets. The chapter concludes with a summary in Section 6.7.

6.2 *RELATED LITERATURE*

Data-dependent specification of the number of variables to include is an important aspect of kernel variable selection which has thus far not received much attention in the literature: presupposing a fixed value for this number has somehow become common practice in most contributions to kernel variable selection. Generally in applications the value of m is of course unknown and has to be determined from the data. In this section we first provide pointers to some papers on data-dependent specification of the model dimension in classical statistical procedures, and then we refer to a current approach which entails automatic determination of a value for m when kernel techniques are used.

Important contributions to selecting the model dimension in the context of traditional statistical procedures (based on maximum likelihood principles) are Akaike (1970) and Schwarz (1978). In a more general framework Breiman (1992) and Rao (1999) propose use of the bootstrap (*cf*. Efron, 1979 and 1982). The number of input variables to include in CART is determined through *pruning* which is typically based on cross-validation (*cf*. Geisser, 1975, and Stone, 1977). Bunke and Droge (1984) compare the bootstrap with cross-validation for deciding on a model dimension in linear regression problems. Other references regarding a choice between statistical models of varying complexity may be found in Rao (1999).

Cross-validation is currently the predominant way of determining the dimension of post-selection kernel models. The computational expense of cross-validation however often

renders it an unattractive option, especially in applications where p is large. Speedup cross-validation algorithms for least squares SVMs and kernel ridge regression are discussed in Senjian *et al*. (2007).

In some approaches to the variable selection problem, the value of m is automatically estimated from the data. If selection is performed through the use of regularised optimisation (briefly described in Chapter 2), also called *soft selection* (*cf*. for example Chapelle *et al*., 2004), a coefficient (or scaling factor) is associated with each input variable. The selection algorithm returns the estimated coefficient associated with each variable, where some of the estimated values may be zero. The number of non-zero scaling factors is taken as the number of variables to select. For more details regarding soft selection, the reader may consult references provided in Chapter 2 and also the paper by Chapelle *et al*. (2004).

6.3 *A PROPOSAL FOR DECIDING ON THE MODEL DIMENSION*

This section consists of two parts: in the first part we provide an illustration of the importance of determining the number of input variables to be used in an SVM. We then proceed with a discussion of a method which can be used for this purpose. The following notation will be used: as indicated earlier, m will denote the number of relevant variables; an estimate of m will be denoted by $\hat{m}$, *i.e*. this will represent the number of variables which we propose to include in the model; and finally, k will be used as an index for the different possible model dimensions.

Figure 6.1 displays boxplots of the test errors obtained after training SVMs on subsets of variables of different sizes. The sizes of the subsets are indicated on the x-axis. We considered a binary classification setup, and all training cases (which consisted of measurements on $p = 5$ input variables) were generated so that differences between the two groups were caused only by a subset of $m = 3$ relevant input variables. In the figure the boxplot at $k = 1$ depicts the test error values obtained from SVMs based on only one of

the three relevant variables, the boxplot at $k = 2$ corresponds to SVMs based on two relevant input variables, and so on, up to $k = 5$, where of course two irrelevant variables are also included. We used *NS* and *LS* data with $\rho_S = 0$ and 0.7, and $\rho_{S\bar{S}} = 0$ or 0.9 in the normal case, and $\rho_S = 0$ and 0.7, and $\rho_{S\bar{S}} = 0$ or 0.2 in the lognormal case. Regarding the cost parameter, we used a series of values, *viz.* $C = 0.001$, 0.01, 0.1, 1, 10, 100 and 1000, and we report the best results obtained. Figure 6.1 summarises the results for *NS* data with $\rho_S = 0$ and $\rho_{S\bar{S}} = 0$. Figures for the other cases which were investigated convey the same message and are therefore not shown. The full set of average test errors for the different parameter configurations are summarised per data configuration in Tables 6.1 and 6.2.

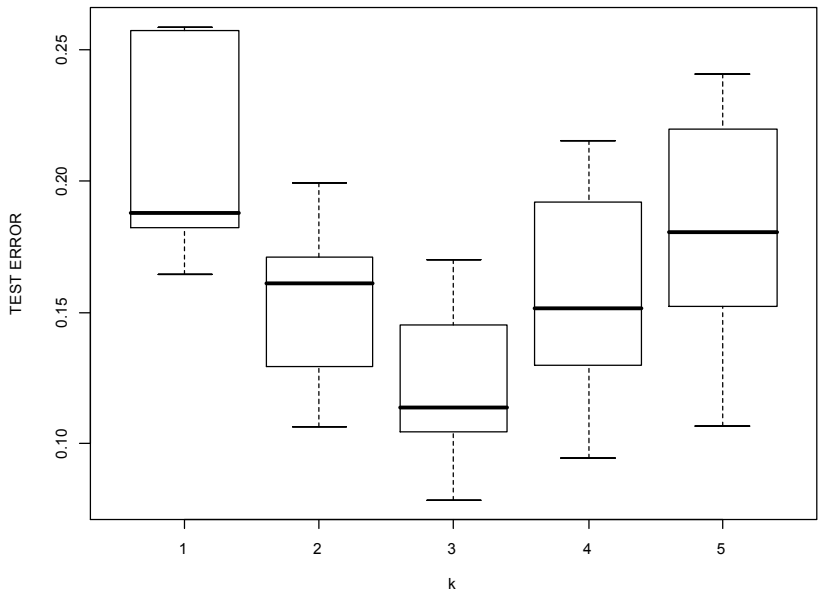


Figure 6.1: *Boxplots of test errors at different model dimensions*

Clearly in Figure 6.1, if we assume that it is possible to consistently identify the best subset per dimension correctly, then the SVM test errors based on the correct model dimension ($m = 3$ in the above scenario) is significantly smaller and have smaller variation than the errors at other values of k.

Table 6.1: Average SVM test errors at different model dimensions in the *NS* case

		k				
ρ_S	$\rho_{S\bar{S}}$	1	2	3	4	5
0.0	0.0	.258	.166	.120	.151	.175
0.0	0.9	.258	.166	.120	.149	.171
0.7	0	.258	.182	.145	.181	.210
0.7	0.9	.253	.182	.145	.178	.206

Table 6.2: Average SVM test errors at different model dimensions in the *LS* case

		k				
ρ_S	$\rho_{S\bar{S}}$	1	2	3	4	5
0.0	0.0	.184	.118	.093	.127	.157
0.0	0.2	.182	.115	.092	.120	.143
0.7	0	.184	.145	.129	.163	.193
0.7	0.2	.184	.145	.130	.160	.186

From Figure 6.1 and the average test errors reported at the different model dimensions in Tables 6.1 and 6.2, the importance of correctly determining the SVM model dimension (if one is given the best variable subset of each dimension), is evident. In the above scenarios the SVM misclassification rate can be significantly improved if instead of $k = 2$ or 4 one uses the correct number of input variables, *viz.* $k = m = 3$.

We now proceed with a discussion of our proposal for a data-dependent decision regarding the value of m. In the end this forms part of a two-stage approach for selecting a subset of $\{X_1, X_2, \ldots, X_p\}$ to construct an SVM classifier. Initially, p subsets are identified as candidates for final selection: a subset V_1 consisting of the single variable deemed best in some sense, a subset V_2 consisting of the best two variables, and so on up to $V_p = \{X_1, X_2, \ldots, X_p\}$. As seen in previous chapters, two aspects arise here: firstly, a criterion has to be specified to decide on an optimal subset of a given size, and secondly, one has to decide on the method which will be used to search through the different subsets of variables. We call the criterion in terms of which the optimal subsets of different dimensions are defined the *inner* (*selection*) *criterion*, and we discuss two possibilities in

this regard further on in this section. For a given inner criterion, forward selection, backward elimination, or an all possible subsets approach may be considered for scanning the subsets of variables. We know that the all possible subsets approach is only feasible for moderately large values of p, say $p < 40$. For larger values of p, the number of subsets of a given dimension which have to be examined becomes prohibitively large, and a choice has to be made between a forward selection and backward elimination approach. Several authors prefer the backward elimination approach, arguing that an unwise decision during the early stages of a forward selection approach may lead to good models of higher dimensions not being considered at all. In the empirical evaluation later on we will see that also in our proposed selection approach, backward elimination is generally to be preferred to forward selection. In any case, an important requirement for the inner criterion is that we should be able to calculate this criterion quickly.

During the second stage of variable selection a so-called *outer* (*selection*) *criterion* has to be used to choose one of the previously identified subsets $V_1, V_2, \ldots, V_p$. At this stage fast computation of the criterion becomes less important, since only p candidate subsets remain. However, an important requirement for the outer criterion is that it should not be a monotone function of the number of variables in the subsets under consideration. This requirement is essential to prevent the criterion from simply selecting the best 1-variable subset, or the set containing all the input variables.

What recommendations can be made regarding the inner and outer criterion for use in SVMs? As seen in Chapter 5, different (inner) selection criteria have been proposed in the literature, including $\|\boldsymbol{w}\|^2$ and criteria based on generalisation error bounds (for example, the radius-margin bound and the span estimate). Since training an SVM is computationally fairly expensive, the inner criterion should preferably not be based on quantities which depend on a trained SVM. To be more specific in this regard, we recommend that the inner criterion should only be a function of the kernel matrix. Of the algorithm-independent (and thus more quickly computable) selection criteria discussed in Chapter 4, the zero-order forms of the alignment (see Cristianini *et al.*, 2002) and also of the variation ratio (see

Wang *et al*., 2004) performed well, and we will evaluate our proposal using these two inner criteria.

Regarding the outer criterion, we stated in Section 6.2 that one possibility would be to use cross-validation estimates of the error rates of the SVM classifiers constructed from $V_1, V_2, \ldots, V_p$ to select one of these subsets. Since cross-validation is infeasible for data sets with a large number of input variables, we consider using an alternative outer criterion. A well known upper bound on the (leave-one-out) generalisation performance of SVMs is based on the number of support vectors (see Schölkopf and Smola, 2002, *p*. 198, as well as Vapnik and Chapelle, 2000, and Chapelle *et al*., 2004, for more detail on various bounds on the generalisation errors of SVMs). The following theorem summarises this bound.

THEOREM 6.1

The expectation of the number of support vectors obtained during training on a training data set of size n, divided by n, is an upper bound on the expected probability of test error of the SVM trained on training data sets of size $n-1$.

□

Based on the above theorem, we propose using the number of support vectors (NSV) as an outer selection criterion. We therefore propose estimation of m by $\hat{m}$, where $\hat{m}$ minimises the NSV criterion, *i.e*.

$$\hat{m} = \underset{k=1,2,\ldots,p}{arg\ min} \{NSV(V_k)\}, \tag{6.1}$$

where $NSV(V_k)$ is the number of support vectors (the number of positive SVM α-coefficients) of the SVM based on the variables in V_k. Note that this proposal is based on the fact that the bound in Theorem 6.1 is a monotone non-decreasing function of the number of support vectors.

The only aspect remaining is to specify a selection strategy. As pointed out before, in the literature on variable selection for SVMs a backward elimination approach is preferred to forward selection and an all possible subsets approach. In the simulation experiments described in later sections we will compare these different strategies and see that backward elimination is indeed to be preferred in our proposed approach as well.

6.4 *PRELIMINARY EVALUATION*

The aim of this section is to report a limited initial evaluation of the use of the NSV outer criterion introduced in Section 6.3, independently of the use of an inner criterion. Hence in this section we assume that the NSV criterion is given the nested sequence of best variable subsets corresponding to all possible model dimensions, *viz.* $V_1 \subset V_2 \subset \ldots \subset V_p$. If the NSV criterion does not perform well in this simplified scenario, evaluation of its performance in combination with an inner criterion will of course be pointless.

In our preliminary study we restricted attention to large (i.e. $n_1 = n_2 = 100$) *NS* and *LS* data sets. We simulated 500 binary classification training and test data sets, and in each generated differences between the two groups to be caused by $m = 3$ out of a total of $p = 5$ input variables. We considered all four correlation configurations described in Chapter 3 and used $s_2^2 = 10$ throughout. In total we therefore had eight simulation setups to evaluate. During each simulation repetition we trained a support vector classifier using the best variable subset per possible model dimension ($k = 1,2,\ldots,5$), and calculated the corresponding NSV criterion values and test errors for each of the five trained SVMs. In training the SVMs for each model dimension we used $\gamma = 1/k$. These steps were repeated for each simulated data set, at cost parameter values $C = 0.001, 0.01, 0.1, 1, 10, 100$ and 1000. For each model dimension and cost parameter value we then calculated an average NSV value, and an average test error. This yielded $8 \times 7 = 56$ sets of $p = 5$ average NSV and test error figures.

Since the results for different data configurations were very similar, we summarise only a sample of average *NSV* and test error values: the figures obtained for each model dimension and each value of C, with $\rho_S = 0$ and $\rho_{S\bar{S}} = 0$ in *NS* and *LS* data sets are given in Tables 6.3 and 6.4 respectively. The *NSV* values appear in the first row of each cell, and the test errors follow in the second row. We indicate minimum values of the *NSV* criterion and of the test errors in bold face.

Table 6.3: Average *NSV* values and test errors for $\rho_S = 0$ and $\rho_{S\bar{S}} = 0$ in the *NS* case

	k				
C	1	2	3	4	5
.001	200.00 .257	200.00 .161	200.00 **.107**	200.00 .152	200.00 .182
.01	**199.99** .258	200.00 .161	200.00 **.106**	200.00 .152	200.00 .182
.1	153.34 .258	141.25 .161	**134.10** **.106**	161.83 .134	179.08 .167
1	119.32 .257	87.27 .163	**71.70** **.110**	87.12 .118	102.26 .130
10	110.13 .258	75.22 .167	**55.76** **.118**	62.89 .131	71.86 .149
100	106.78 .259	71.33 .171	**52.37** **.134**	58.01 .164	64.78 .193
1000	106.87 .259	68.36 .178	**50.22** **.158**	56.78 .207	62.33 .221

Table 6.4: Average NSV values and test errors for $\rho_S = 0$ and $\rho_{S\bar{S}} = 0$ in the *LS* case

	k				
C	1	2	3	4	5
.001	**199.85** .183	200.00 .129	200.00 **.104**	200.00 .154	200.00 .192
.01	**199.92** .182	200.00 .129	200.00 **.104**	200.00 .156	200.00 .195
.1	163.23 .180	**161.76** .123	162.32 **.101**	179.86 .146	189.79 .187
1	113.45 .181	91.33 .114	**83.83** **.091**	102.29 .110	117.63 .130
10	97.39 .186	64.174 .111	**53.48** **.081**	64.82 .098	75.44 .115
100	90.54 .188	51.60 .107	**38.90** **.079**	48.02 .103	56.94 .128
1000	86.86 .191	44.08 .108	**32.40** **.088**	41.08 .120	49.49 .149

The results of this initial evaluation of the NSV criterion were encouraging: in only a small percentage of all 56 cases the minimum average NSV value occurred at an 'incorrect' value of k (*i.e.* in this setup at model dimension $k \neq m = 3$). Therefore, assuming that the inner criterion always correctly identifies the best subset per dimension, using the NSV criterion to determine the number of separating input variables would have yielded the incorrect number relatively rarely. It should also be noted though that in 12 out of 56 instances an inappropriate value of C (mostly $C = 0.001$ or 0.01) caused the number of support vectors to be large and equal at all values of k, rendering a decision based on the NSV criterion regarding the number of variables to use impracticable. Still, in the evaluated setups it seems as if *NSV*-based specification of the number of variables to use is not overly sensitive to potential misspecification of C. In Tables 6.3 and 6.4 for example, the NSV criterion correctly identifies m at $C = 1, 10, 100$ and 1000.

The average *NSV* values and SVM test errors (as percentages) for $\rho_S = 0.7$ and $\rho_{S\bar{S}} = 0$ in the *NS* and *LS* scenarios are displayed in Figures 6.2 and 6.3 respectively, plotted

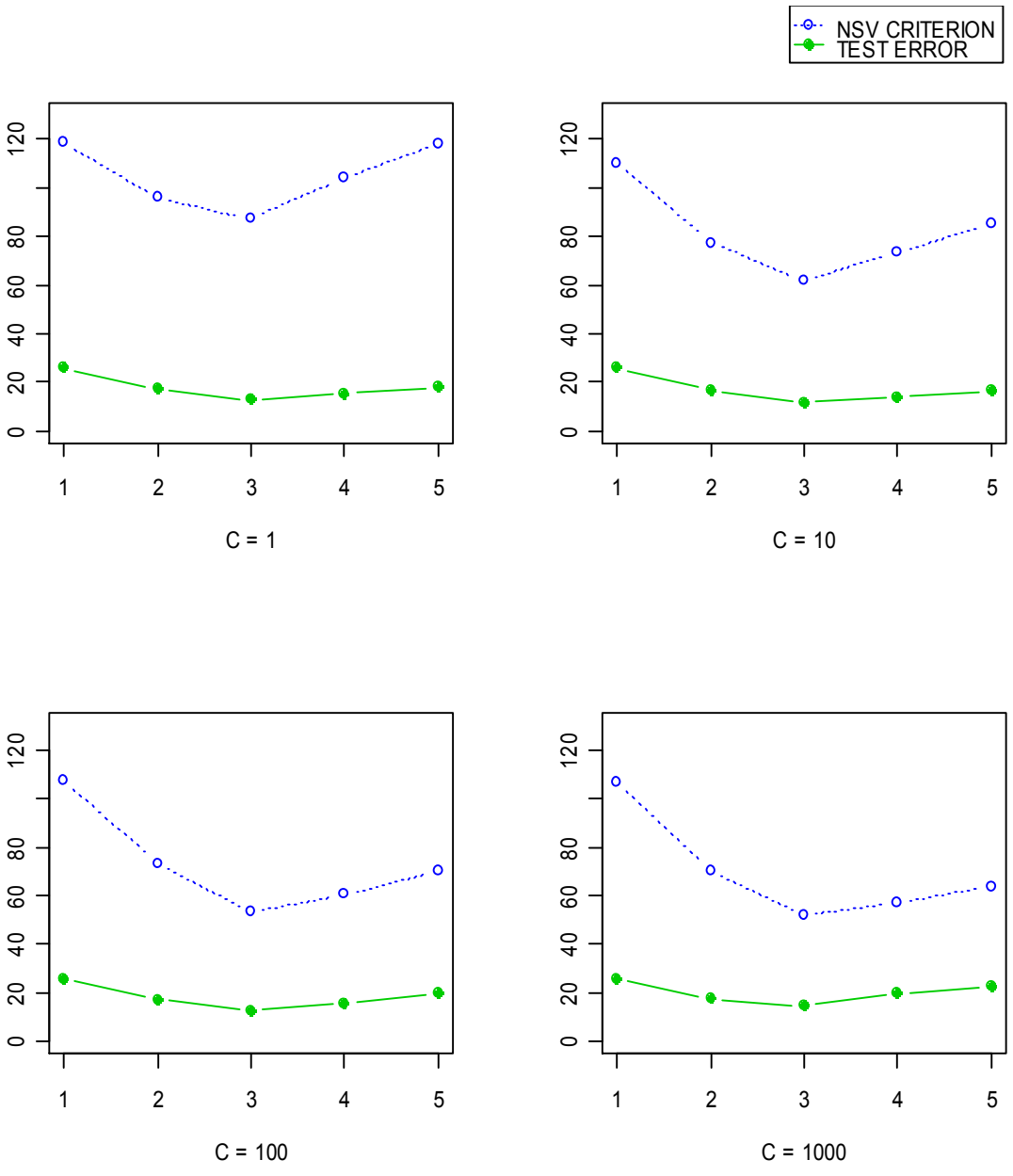


Figure 6.2: *The average number of support vectors, and the average test error (as a percentage), when* $\rho_S = 0.7$ *and* $\rho_{S\bar{S}} = 0$ *in the NS case*

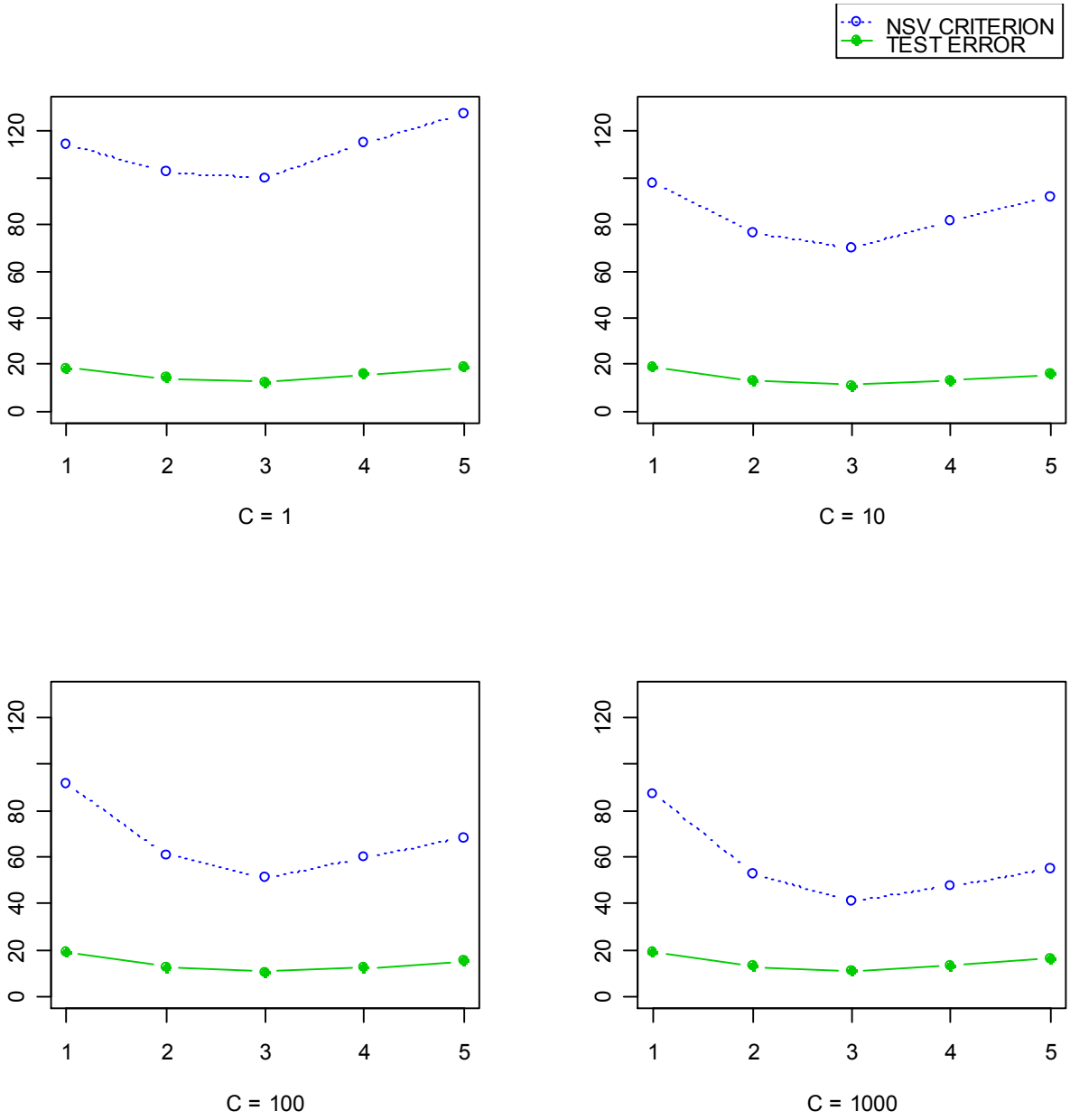


Figure 6.3: *The average number of support vectors, and the average test error (as a percentage) when* $\rho_S = 0.7$ *and* $\rho_{S\bar{S}} = 0$ *in the LS case*

against the different values of k. Results corresponding to other correlation structures were similar.

In both figures it is clear that the average test errors decrease from $k = 1$, reach a minimum at $k = 3$, and steadily increase at $k = 4$ and 5. The same pattern also holds for the *NSV* criterion, and we see that in these scenarios, a decision on the number of input variables to include, if it is based on minimising the number of support vectors, would yield the correct number ($m = 3$ in this setup) in all cases.

6.5 *MONTE CARLO SIMULATION STUDY*

The promising results in Section 6.4 encouraged us to evaluate the performance of the selection strategy proposed in Section 6.3 via a more extensive Monte Carlo simulation study. In essence we made use of the experimental design described in Chapter 3, but in some cases we considered different factor levels. We now discuss these modifications. Firstly, since we saw in previous chapters that little benefit is gained by using SVMs in *NL* cases, we only evaluated our dimension selection procedure using *NS*, *LL* and *LS* data sets. Also, the total number of available variables (in $\mathcal{V}$) was taken as either $p = 5$ or $p = 50$. In data sets where p equalled 5, we generated observations from the two groups to differ with respect to $m = 3$ input variables. When $p = 50$, we used $m = 10$. Depending on the value of p the proportion of relevant input variables was therefore either $\pi = 0.2$ (when $p = 50$) or $\pi = 0.6$ (when $p = 5$). A second modification was with respect to the sample sizes considered. In parts of the study where we compared the various inner criteria and selection strategies, we restricted attention to small (i.e. $n_1 = n_2 = 15$) and large (i.e. $n_1 = n_2 = 100$) data sets, whereas in cases where we considered the use of the *NSV* criterion versus cross-validation as outer criterion, we also included mixed samples (where $n_1 = 50$ and $n_2 = 150$). Thirdly, in *NS* and *LS* configurations we restricted attention to only a single value for the variance of the second group, viz. $s_2^2 = 10$ (as opposed to

$s_2^2 = 10$ and $s_2^2 = 100$ in Chapter 3). Regarding hyperparameter specifications, we followed a recommendation by Schölkopf and Smola (2002), and consistently used a cost parameter value of $C = n/10$. Note that more details with regard to specification of γ will be given as we proceed with the discussion. We used 1000 Monte Carlo simulation repetitions throughout, and for each model dimension we report the average test error (based on test data sets consisting of 2000 cases) and the fraction of times that each of the input variables was selected.

Recall that the first step during each simulation repetition involves computation of the zero-order versions of the alignment and the variation ratio as inner criteria on the training data, in order to identify $V_1 \subset V_2 \subset \ldots \subset V_p$. We used $\gamma = 1/p$ to calculate these inner criteria. Let the nested subsets of input variables obtained after using the alignment and the variation ratio be denoted by $V(A)_1 \subset V(A)_2 \subset \ldots \subset V(A)_p$ and $V(R)_1 \subset V(R)_2 \subset \ldots \subset V(R)_p$ respectively. The second step in our approach requires application of an outer selection criterion. After calculating the outer criterion for the best variable subsets of sizes $k = 1,2,\ldots,p$, we then have to select the number of variables corresponding to the optimal outer criterion value. In our proposal we use the NSV as outer selection criterion. Therefore, during the second step in each simulation repetition we (potentially) needed to train $2p$ SVMs, based on two (potentially) different variable subsets of sizes $k = 1,2,\ldots,p$ as identified by the alignment and the variation ratio as inner criteria. Here we specified $\gamma = 1/k$. We then calculated the NSV values corresponding to these $2p$ SVMs. Let the NSV values for selection using the alignment and the variation ratio be denoted by $NSV(A)_1, NSV(A)_2, \ldots, NSV(A)_p$ and $NSV(R)_1, NSV(R)_2, \ldots, NSV(R)_p$ respectively. A value for m using the alignment as inner selection criterion was then obtained as

$$\hat{m}_A = \underset{k=1,2,\ldots,p}{arg\,min}\{NSV(A)_k\}. \qquad (6.2)$$

Similarly, using the variation ratio as inner criterion we identified the number of variables to include in the final model as

$$\hat{m}_R = \underset{k=1,2,\ldots,p}{arg\ min}\{NSV(R)_k\}. \tag{6.3}$$

Finally then in this section, note that $\hat{m}_A$ and $\hat{m}_R$ respectively indicate the number of input variables to select as identified by the NSV outer criterion, after using the alignment and the variation ratio as inner criteria. Since prior to determining $\hat{m}_A$ and $\hat{m}_R$ we reduced the sets of variables to consider to only the best variable subsets of each dimension, note that $\hat{m}_A$ and $\hat{m}_R$ also index the variable subset selected using the alignment and the NSV outer criterion, and the variation ratio and the NSV criterion, in that order.

The final step in each simulation repetition involved separate calculation of several quantities when using both the alignment and the variation ratio for selection. We report averages of these quantities over the 1000 simulation repetitions. The quantities are:

i. Test errors of the post-selection SVM, which we will refer to as the *alignment NSV* and *variation ratio NSV* test errors.

ii. SVM test errors based on subsets containing only the correct number of input variables, which we will call *correct dimension* test errors (denoted by *CDIM* in the tables).

iii. The number of times that each of $X_1, X_2, \ldots, X_p$ was selected.

iv. The number of times that each of the possible model dimensions, *i.e.* $k = 1,2,\ldots,p$, was selected.

v. SVM test errors based on the subset of relevant input variables. We will refer to these errors as *oracle* test errors (denoted by *ORA* in the tables), since they represent a gold standard against which the performance of proposed procedures can be measured.

vi. SVM test errors based on all available input variables, *i.e.* the so-called *no selection* test errors (denoted by *NO* in the tables).

6.6 *RESULTS AND CONCLUSIONS*

The results obtained from the Monte Carlo simulation study described in Section 6.5 are summarised in this section.

Consider first the relative performance of the two selection strategies, *i.e.* backward elimination and forward selection. For this purpose we compare corresponding entries in Tables 6.5 and 6.6 below. Note that the first entry in each cell of the *CDIM* and *NSV* columns in these (and later similar) tables corresponds to the alignment as inner criterion, while the second entry corresponds to the variation ratio as inner criterion. We generally observe smaller average test errors in backward elimination. This is particularly evident in small sample cases, although we also observe relatively large improvements in classification accuracy if backward elimination is used in large *LS* samples. The more pronounced differences between the performances of backward and forward selection are summarised in Tables 6.7 and 6.8.

Table 6.5: Average test errors in forward selection

			SMALL SAMPLES				*LARGE SAMPLES*			
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***NS*1**	5	0.0	.039	.045 .042	.066 .064	.122	.016	.016 .016	.016 .016	.047
***NS*2**	5	0.7	.041	.048 .043	.065 .064	.125	.015	.015 .015	.016 .016	.047
***NS*3**	50	0.0	.008	.168 .143	.252 .231	.259	.002	.003 .003	.012 .011	.158
***NS*4**	50	0.7	.009	.152 .127	.238 .213	.259	.003	.004 .004	.013 .013	.158

			SMALL SAMPLES				*LARGE SAMPLES*			
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***LL*1**	5	0.0	.088	.095 .096	.121 .122	.133	.041	.041 .041	.046 .046	.066
***LL*2**	5	0.7	.133	.140 .134	.139 .137	.187	.087	.087 .087	.090 .090	.121
***LL*3**	50	0.0	.067	.159 .156	.115 .115	.178	.028	.028 .029	.047 .046	.072
***LL*4**	50	0.7	.130	.184 .147	.143 .141	.273	.084	.084 .084	.086 .086	.204

			SMALL SAMPLES				*LARGE SAMPLES*			
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***LS*1**	5	0.0	.133	.158 .152	.179 .178	.210	.079	.080 .079	.082 .082	.117
***LS*2**	5	0.7	.161	.197 .177	.203 .200	.255	.105	.106 .105	.109 .108	.151
***LS*3**	50	0.0	.089	.311 .301	.262 .261	.384	.054	.077 .073	.078 .077	.295
***LS*4**	50	0.7	.133	.367 .342	.362 .346	.405	.066	.165 .137	.148 .141	.340

Table 6.6: Average test errors in backward elimination

			SMALL SAMPLES				*LARGE SAMPLES*			
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***NS*1**	5	0.0	.039	.043 .039	.062 .060	.121	.015	.015 .015	.016 .016	.048
***NS*2**	5	0.7	.042	.046 .042	.066 .063	.122	.015	.015 .015	.016 .016	.047
***NS*3**	50	0.0	.008	.040 .021	.128 .088	.259	.002	.002 .002	.011 .010	.158
***NS*4**	50	0.7	.009	.041 .023	.128 .091	.259	.004	.004 .004	.012 .012	.158

			SMALL SAMPLES				*LARGE SAMPLES*			
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***LL*1**	5	0.0	.088	.095 .097	.118 .119	.134	.041	.041 .041	.045 .045	.066
***LL*2**	5	0.7	.133	.143 .136	.138 .137	.188	.087	.087 .087	.090 .090	.121
***LL*3**	50	0.0	.067	.100 .098	.115 .115	.178	.029	.029 .029	.046 .045	.073
***LL*4**	50	0.7	.129	.184 .146	.142 .139	.277	.084	.086 .085	.085 .084	

			SMALL SAMPLES				*LARGE SAMPLES*			
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***LS*1**	5	0.0	.132	.156 .153	.177 .177	.207	.079	.079 .079	.082 .082	.116
***LS*2**	5	0.7	.162	.203 .175	.205 .195	.257	.106	.106 .106	.110 .109	.151
***LS*3**	50	0.0	.087	.239 .227	.234 .225	.381	.054	.067 .063	.066 .065	.294
***LS*4**	50	0.7	.132	.314 .260	.304 .263	.404	.066	.119 .071	.093 .076	.339

Table 6.7: Average test errors for backward and forward selection in small samples when $p = 50$

		BACKWARD		***FORWARD***	
	ρ_S	***CDIM***	***NSV***	***CDIM***	***NSV***
***NS*3**	0.0	.040 .021	.128 .088	.168 .143	.252 .231
***NS*4**	0.7	.041 .023	.128 .091	.152 .127	.238 .213
***LS*3**	0.0	.239 .227	.234 .225	.311 .301	.262 .261
***LS*4**	0.7	.314 .260	.304 .263	.367 .342	.362 .346

Table 6.8: Average test errors for backward and forward selection in large samples when $p = 50$

		BACKWARD		***FORWARD***	
	ρ_S	***CDIM***	***NSV***	***CDIM***	***NSV***
***LS*3**	0.0	.067 .063	.066 .065	.077 .073	.078 .077
***LS*4**	0.7	.119 .071	.093 .076	.165 .137	.148 .141

It is clear that backward elimination is preferable to forward selection. The question may now be asked whether one should consider using an all possible subsets approach. We compared backward elimination and an all possible subsets approach in a limited simulation study ($p = 5$), and found the two strategies to perform largely the same (see Table 6.9 for the test errors using an all possible subsets approach, and Table 6.10 for easier comparison of the three selection strategies). Since an all possible subsets approach is impracticable for large values of p, we recommend the use of a backward elimination strategy. Most of the results reported and discussed further on in this section were obtained using backward elimination.

Table 6.9: Average test errors in all possible subsets selection when $p = 5$

		SMALL SAMPLES				*LARGE SAMPLES*			
	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***NS*1**	0.0	.034	.043 .039	.064 .063	.121	.016	.016 .016	.016 .016	.047
***NS*2**	0.7	.041	.044 .041	.064 .062	.123	.015	.015 .015	.016 .016	.047

		SMALL SAMPLES				*LARGE SAMPLES*			
	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***LL*1**	0.0	.090	.094 .097	.122 .123	.132	.041	.041 .041	.045 .045	.065
***LL*2**	0.7	.134	.141 .135	.140 .138	.187	.087	.087 .087	.089 .089	.120

		SMALL SAMPLES				*LARGE SAMPLES*			
	ρ_S	***ORA***	***CDIM***	***NSV***	***NO***	***ORA***	***CDIM***	***NSV***	***NO***
***LS*1**	0.0	.131	.157 .151	.176 .177	.205	.079	.080 .079	.082 .082	.119
***LS*2**	0.7	.161	.197 .172	.202 .194	.251	.106	.107 .106	.110 .109	.152

Consider next the quality of the proposed inner criteria. Given the superiority of backward elimination, we focus on the results in Table 6.6, and compare corresponding entries in the *ORA* and *CDIM* columns. These are summarised in Table 6.11. Recall that the figures in the *CDIM* columns are average test errors of models of the correct dimension identified by the alignment and the variation ratio inner criterion. The extent to which these test errors exceed the corresponding oracle test errors reflects failure of the inner criteria to identify the variables separating the two groups correctly.

Table 6.10: Average test errors in forward selection, backward elimination and an all possible subsets approach

			FORWARD				***BACKWARD***				***ALL POSSIBLE***			
			SMALL		***LARGE***		***SMALL***		***LARGE***		***SMALL***		***LARGE***	
	p	ρ_S	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***
***NS*1**	5	0.0	.045	.066	.016	.016	.043	.062	.015	.016	.043	.064	.016	.016
			.042	.064	.016	.016	.039	.060	.015	.016	.039	.063	.016	.016
***NS*2**	5	0.7	.048	.065	.015	.016	.046	.066	.015	.016	.044	.064	.015	.016
			.043	.064	.015	.016	.042	.063	.015	.016	.041	.062	.015	.016

			FORWARD				***BACKWARD***				***ALL POSSIBLE***			
			SMALL		***LARGE***		***SMALL***		***LARGE***		***SMALL***		***LARGE***	
	p	ρ_S	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***
***LL*1**	5	0.0	.095	.121	.041	.046	.095	.118	.041	.045	.094	.122	.041	.045
			.096	.122	.041	.046	.097	.119	.041	.045	.097	.123	.041	.045
***LL*2**	5	0.7	.140	.139	.087	.090	.143	.138	.087	.090	.141	.140	.087	.089
			.134	.137	.087	.090	.136	.137	.087	.090	.135	.138	.087	.089

			FORWARD				***BACKWARD***				***ALL POSSIBLE***			
			SMALL		***LARGE***		***SMALL***		***LARGE***		***SMALL***		***LARGE***	
	p	ρ_S	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***	***CDIM***	***NSV***
***LS*1**	5	0.0	.158	.179	.080	.082	.156	.177	.079	.082	.157	.176	.080	.082
			.152	.178	.079	.082	.153	.177	.079	.082	.151	.177	.079	.082
***LS*2**	5	0.7	.197	.203	.106	.109	.203	.205	.106	.110	.197	.202	.107	.110
			.177	.200	.105	.108	.175	.195	.106	.109	.172	.194	.106	.109

Table 6.11: Average test errors in backward elimination

			SMALL SAMPLES		*LARGE SAMPLES*	
	p	ρ_S	***ORA***	***CDIM***	***ORA***	***CDIM***
***NS*1**	5	0.0	.039	.043 .039	.015	.015 .015
***NS*2**	5	0.7	.042	.046 .042	.015	.015 .015
***NS*3**	50	0.0	.008	.040 .021	.002	.002 .002
***NS*4**	50	0.7	.009	.041 .023	.004	.004 .004

			SMALL SAMPLES		*LARGE SAMPLES*	
	p	ρ_S	***ORA***	***CDIM***	***ORA***	***CDIM***
***LL*1**	5	0.0	.088	.095 .097	.041	.041 .041
***LL*2**	5	0.7	.133	.143 .136	.087	.087 .087
***LL*3**	50	0.0	.067	.100 .098	.029	.029 .029
***LL*4**	50	0.7	.129	.184 .146	.084	.086 .085

			SMALL SAMPLES		*LARGE SAMPLES*	
	p	ρ_S	***ORA***	***CDIM***	***ORA***	***CDIM***
***LS*1**	5	0.0	.132	.156 .153	.079	.079 .079
***LS*2**	5	0.7	.162	.203 .175	.106	.106 .106
***LS*3**	50	0.0	.087	.239 .227	.054	.067 .063
***LS*4**	50	0.7	.132	.314 .260	.066	.119 .071

In Table 6.11 we see that for a small number of input variables our inner criteria perform well, especially so in large sample cases. As expected the performance deteriorates with an increase in the number of variables, especially in small samples. The worst cases in this regard are *NS3* and *NS*4, as well as *LS*3 and *LS*4. Note that the average test errors when the

alignment was used as inner criterion are given in the first row of each cell, and test errors pertaining to selection based on variation ratios follow in the second row. Comparing the performance of the two proposed inner criteria, we see that the variation ratio mostly outperforms the alignment, especially in small sample cases. In light of the above results our conclusion at this stage is that the proposed variable selection strategy performs well when it is based on backward elimination using the variation ratio as inner selection criterion.

We now move on to a discussion of the performance of the outer criteria. Results appear in Table 6.12. Note that only results based on the variation ratio as inner criterion combined with backward elimination are reported. Table 6.12 also contains a further column (denoted by *CV*) and representing test errors if cross-validation is used to estimate the value of m. Consider the correct dimension test errors and the test errors in the *NSV* and *CV* columns. Bear in mind that when we apply an outer criterion such as *NSV* or *CV*, we use the data to select a model dimension. This introduces an extra element of uncertainty and one would thus expect the outer criterion error rates to exceed corresponding correct dimension values. If an outer criterion selected the correct model dimension with probability 1, the resulting average test errors would be identical to those in the *CDIM* column. Therefore any difference between entries in the *CDIM* and an outer criterion column reflects failure of the outer criterion to determine the model dimension correctly in all cases. Interestingly, this is not always undesirable. In some cases we actually found that the *NSV* test error is smaller than the corresponding *CDIM* test error. This can only be the result of the *NSV* criterion fortuitously identifying a lower model dimension in cases where a sizable proportion of the variables selected by the inner criterion is useless. In general the outer criteria test errors summarised in Table 6.12 are not far above the *CDIM* values, except in the *NS*3 and *NS*4 cases.

Table 6.12: Average test errors for outer criteria in backward elimination

			SMALL SAMPLES					*MIXED SAMPLES*					*LARGE SAMPLES*				
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***CV***	***NO***	***ORA***	***CDIM***	***NSV***	***CV***	***NO***	***ORA***	***CDIM***	***NSV***	***CV***	***NO***
***NS*1**	5	0.0	.039	.039	.060	.065	.121	.017	.017	.017	.019	.052	.017	.017	.017	.019	.052
***NS*2**	5	0.7	.042	.042	.063	.067	.122	.017	.017	.018	.021	.052	.017	.017	.018	.021	.052
***NS*3**	50	0.0	.008	.021	.088	.057	.259	.004	.004	.009	.009	.198	.004	.004	.009	.009	.198
***NS*4**	50	0.7	.009	.023	.091	.063	.259	.005	.005	.011	.011	.197	.005	.005	.011	.011	.197

			SMALL SAMPLES					*MIXED SAMPLES*					*LARGE SAMPLES*				
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***CV***	***NO***	***ORA***	***CDIM***	***NSV***	***CV***	***NO***	***ORA***	***CDIM***	***NSV***	***CV***	***NO***
***LL*1**	5	0.0	.088	.097	.119	.111	.134	.034	.034	.038	.037	.055	.034	.034	.038	.037	.055
***LL*2**	5	0.7	.133	.136	.137	.137	.188	.071	.071	.065	.064	.100	.071	.071	.065	.064	.100
***LL*3**	50	0.0	.067	.098	.115	.104	.178	.0002	.024	.039	.028	.062	.0002	.024	.039	.028	.062
***LL*4**	50	0.7	.129	.146	.139	.167	.277	.0004	.079	.068	.067	.115	.0004	.079	.068	.067	.115

			SMALL SAMPLES					*MIXED SAMPLES*					*LARGE SAMPLES*				
	p	ρ_S	***ORA***	***CDIM***	***NSV***	***CV***	***NO***	***ORA***	***CDIM***	***NSV***	***CV***	***NO***	***ORA***	***CDIM***	***NSV***	***CV***	***NO***
***LS*1**	5	0.0	.132	.153	.177	.167	.207	.081	.082	.083	.087	.116	.081	.082	.083	.087	.116
***LS*2**	5	0.7	.162	.175	.195	.187	.257	.102	.102	.106	.107	.166	.102	.102	.106	.107	.166
***LS*3**	50	0.0	.087	.227	.225	.235	.381	.042	.064	.064	.066	.261	.042	.064	.064	.066	.261
***LS*4**	50	0.7	.132	.260	.263	.275	.404	.055	.087	.084	.084	.290	.055	.087	.084	.084	.290

We see that it is impossible to express a general preference for either of the two outer criteria based only on test errors: in some cases the *NSV* criterion performs better, but in other cases *CV* is preferred. It does however seem that the difference between the two criteria is larger in cases where the *NSV* criterion performs better.

In addition to the post-selection error rate we also estimated the probabilities with which each of the two outer criteria identified different final model dimensions. A selection of the results is presented in Figures 6.4 and 6.5. The small sample results in the first row of these figures are for the uncorrelated *NS*, *LL* and *LS* cases respectively, while the second and third rows contain the corresponding mixed and large sample results. Note that similar results were obtained for the correlated case (*i.e.* when $\rho_S = 0.7$). Scenarios where we had $p = 5$ and $m = 3$ relevant variables are displayed in Figure 6.4, and cases where $p = 50$ and $m = 10$ are given in Figure 6.5.

The ideal outer criterion would identify a model dimension of three with probability 1. If an outer criterion decides on a lower final model dimension, this amounts to underfitting, whilst overfitting occurs when the identified model dimension exceeds the correct model dimension.

In order to avoid too many categories in Figure 6.5 we grouped together the selection percentages for model dimensions $k = 1, 2, 3,$ and 4 into a so-called *severe underfitting* (*SU*) category, selection percentages for $k = 5, 6, \ldots, 9$ into an *underfitting* (*U*) category, for $k = 11, \ldots, 15$ into an *overfitting* (*O*) category, and for $k = 16, 17, \ldots, 50$ into a severe overfitting (*SO*) category. The percentage of times that the correct number of input variables was selected is labelled *C*.

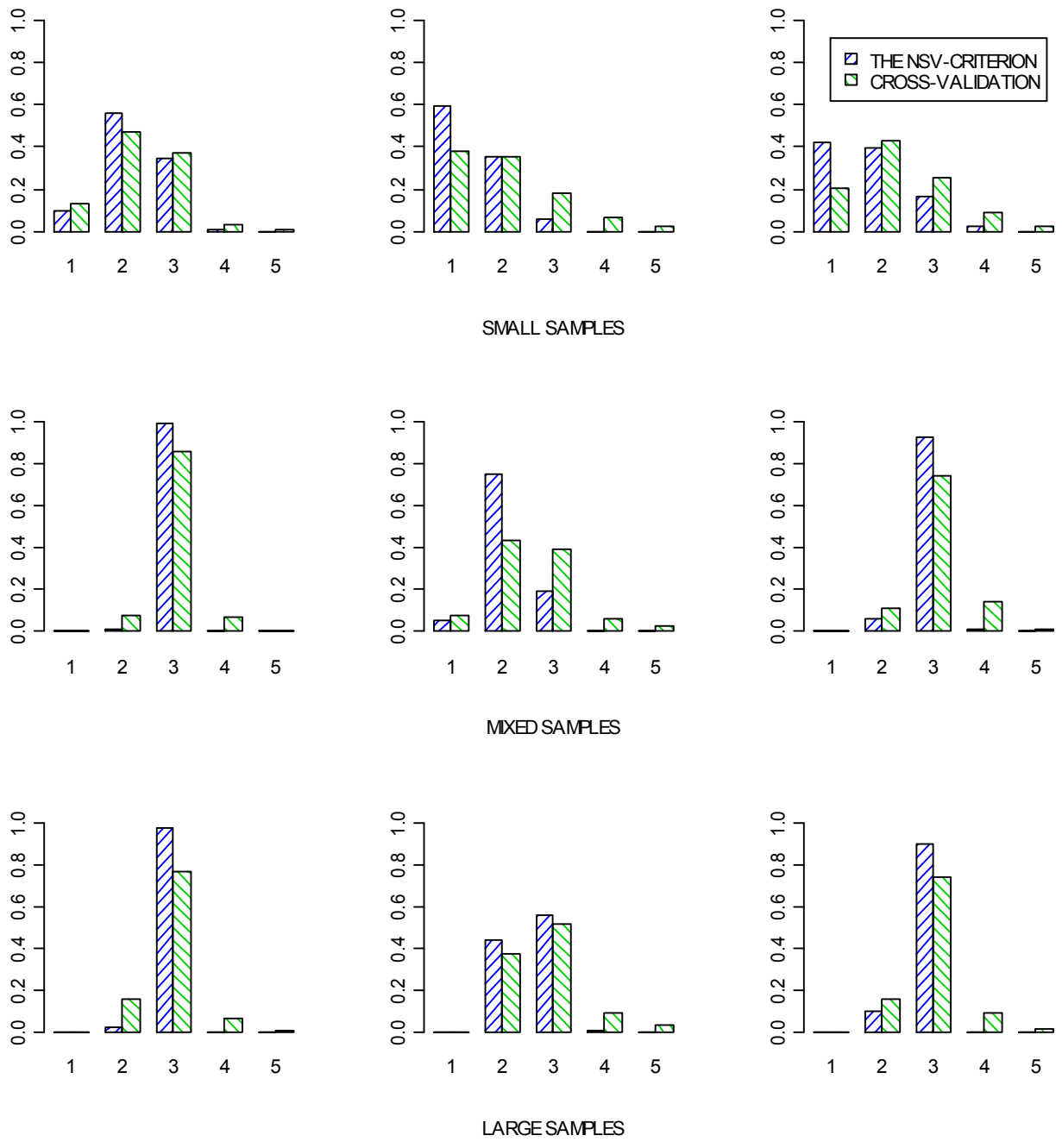


Figure 6.4: *Selection percentages for the NSV outer criterion compared to those obtained for cross-validation when* $p = 5$

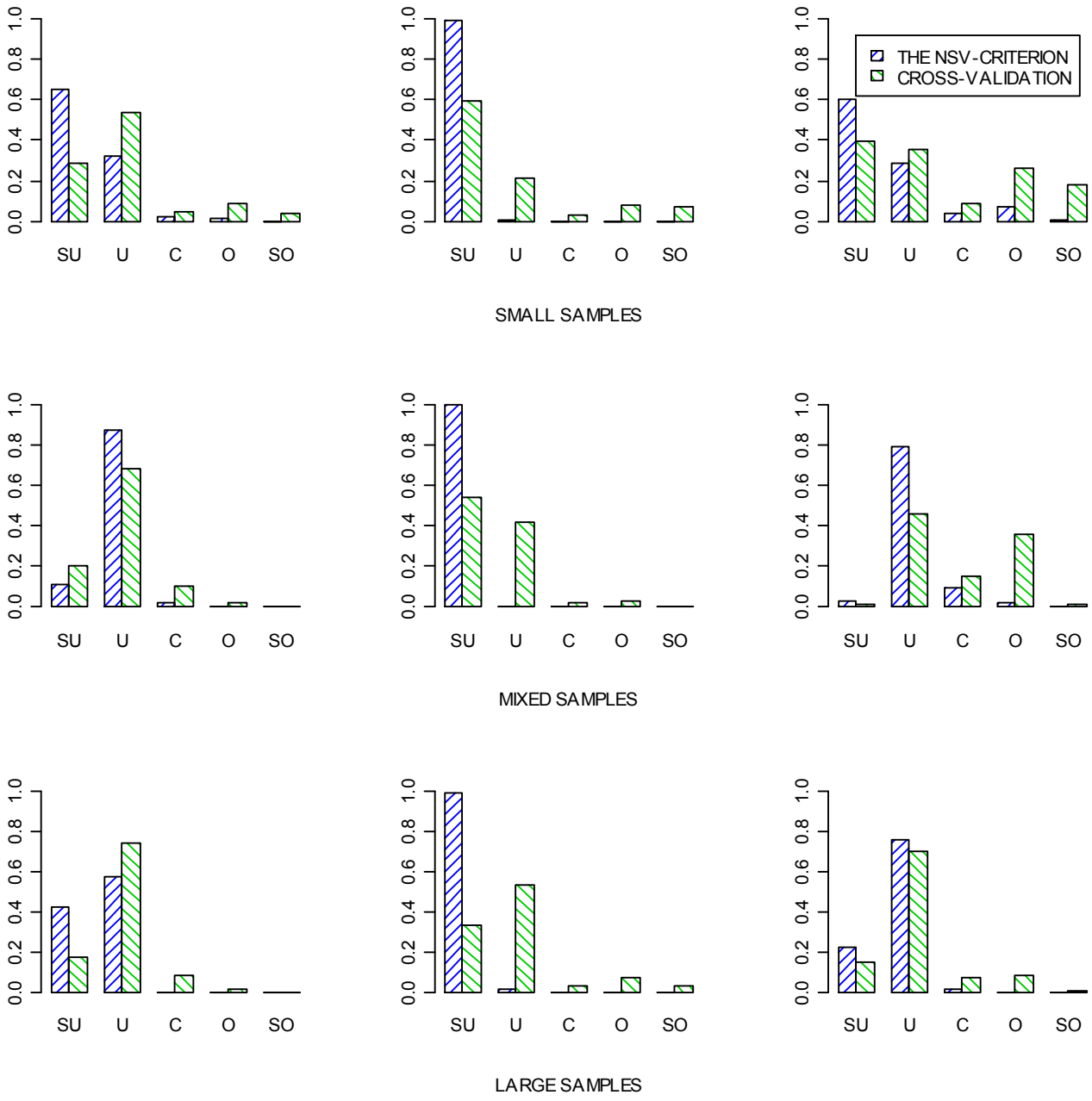


Figure 6.5: *Selection percentages for the NSV outer criterion compared to those obtained for cross-validation when* $p = 50$

Some general trends are evident from Figure 6.4. For small samples *CV* appears to identify the correct model dimension with somewhat higher probability than the *NSV* criterion. The reverse is true for large samples. Also, for small samples the *NSV* criterion tends to underfit, whilst *CV* shows some overfitting and therefore definitely includes irrelevant variables in some cases. Both criteria perform very well in the large sample, scale difference situations, especially so the *NSV* criterion.

For both outer criteria the scenarios reported in Figure 6.5, with their large number of input variables, proved to be much more difficult to handle. Even for large samples the selection percentages at the correct model dimension are relatively small. We again see that the *NSV* criterion is prone to underfitting, whereas cross-validation tends to overfit. Whether underfitting is more acceptable than overfitting, or vice versa, depends on the main variable selection objective: improving the accuracy of the model, or easier interpretation. In the latter case a simpler model is probably to be preferred, in which case we would prefer the *NSV* criterion. However, including seemingly irrelevant variables may sometimes improve the accuracy of a classifier, and if this is of primary interest one would prefer *CV* to *NSV*, given the tendency of the former criterion to overfit rather than underfit. Although it is difficult to make a firm recommendation regarding the outer criterion, we have a slight preference for *NSV*. This is based on the better performance of *NSV* in small p scenarios and the fact that cross-validation is computationally expensive when p is large.

6.7 *SUMMARY*

In this chapter we addressed an important aspect of variable selection for kernel classifiers, *viz*. how to make a data-dependent decision regarding the number of input variables to include. In our opinion, this is a problem to which a satisfactory solution has not yet been proposed in the literature. We restricted attention to support vector classifiers, and based on an upper bound on the expected probability of test error which can be given in terms of the expected number of support vectors, proposed using the number of support vectors to decide on the post-selection dimension.

The *NSV* criterion performed very well when we assumed the best variable subsets of each dimension to be known. We therefore proceeded with a numerical evaluation of the performance of the *NSV* criterion when the optimal variable subsets of each size had to be determined in a data-dependent manner. In this regard, we made use of the selection criteria which performed well in the previous chapters, *viz*. the zero-order versions of the alignment and the variation ratio criteria. With regard to ways of searching through the variable subsets, we compared a backward elimination strategy with a forward selection approach.

Overall it also seemed that the complete variable selection approach which we proposed (*i.e*., a backward elimination strategy with variation ratio as inner criterion and the number of support vectors or cross-validation as outer criterion) performed quite well in terms of generalisation error and the probability of identifying the correct model dimension. Based on the simulation study results there can be little doubt that backward elimination is superior to forward selection, and that the variation ratio is superior to the alignment criterion. No firm recommendation can however be made regarding an outer criterion, although the number of support vectors criterion is simpler and maybe slightly better than cross-validation.

CHAPTER 7

SUMMARY AND DIRECTIONS FOR FURTHER RESEARCH

In this thesis the problem of (input) variable selection for kernel classification methods was investigated, with particular emphasis being placed on support vector machines and, to a lesser extent, kernel Fisher discriminant analysis. This chapter is devoted to a summary of the main findings emanating from the research and an indication of directions for further research.

It is clear from the empirical results reported in the thesis that it is worthwhile to perform variable selection when a kernel classifier is applied. Two purposes are thereby served: the kernel classifier based on the selected variables frequently classifies new cases more accurately than the classifier based on all the variables, and we obtain an indication of the input variables which are important in describing the response. The relevance of the variable selection problem for kernel classifiers is enhanced by the fact that kernel classification is often performed in cases where the number of variables exceeds the number of data points (so-called *wide* data sets). An important example is provided by data sets encountered in micro-array analyses. The work reported in this thesis is therefore a contribution to solving a highly relevant problem.

A point which has also been emphasised repeatedly in the thesis is that there are several factors which complicate variable selection for kernel classifiers. The first and probably most important complicating factor is the (implicit) feature mapping which underlies kernel methods. A direct result of this transformation is that a kernel discriminant function is typically of the form $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b$, *i.e.* we obtain an expression in which a weight α_i is naturally associated with the i^{th} data point. This contrasts with, for example,

a procedure such as ordinary discriminant analysis where an expression which is a linear combination of the variables is obtained. It is clearly much more straightforward to propose variable selection for procedures falling into the latter category than for those where the discriminant function is a sum of terms associated with the individual data points.

In an attempt to deal with this complicating factor we investigated in Chapter 3 the merit of selection in different spaces: selection performed in input space (thereby effectively ignoring the feature transformation), in feature space (thereby explicitly taking the transformation to feature space into account), and so-called feature-to-input space selection (an attempt to combine information generated in feature space with the easier interpretation afforded by working in input space). In this last context we introduced and applied the idea of using the pre-images in input space of suitable quantities in feature space for variable selection. Although we found that input space selection performed well in certain data configurations (notably the case where the data distribution in the two groups was normal with a separation between the groups in terms of location), it turned out that selection making use of feature space information generally performed better.

Going hand in hand with the concept of transforming data from input to feature space is specification of a kernel function to compute inner products in feature space. This also complicates the selection process: we have to decide on the form of the kernel function and we have to specify, or determine from the data, values for its hyperparameters. In our investigation we restricted the discussion to the frequently used Gaussian kernel function, and we implemented a simple recommendation for obtaining a value for its hyperparameter, *viz*. $\gamma =$ the reciprocal of the number of variables being considered. This brings us to a direction for further research: how do different kernel functions compare in terms of the ease and the success with which we can perform variable selection? Is it possible to construct a kernel function specifically with variable selection in mind? And finally, what options, apart from the obvious but numerically intensive possibility of cross-validation, are available for determining hyperparameter values while performing variable selection?

Focusing on feature space selection in Chapter 4, we introduced a distinction between algorithm-independent and algorithm-dependent selection criteria. The former are criteria depending only on the empirical kernel matrix and can therefore be applied in any kernel procedure based on this matrix. Algorithm-dependent criteria use information which is derived from a specific kernel classification algorithm and as such these criteria can only be applied when this specific algorithm is employed. The results of an empirical study are presented and discussed in Chapter 4, and we see that two of the algorithm-independent criteria are very competitive compared to the more sophisticated algorithm-dependent criteria. Further research may therefore also include consideration of alternative algorithm-independent selection criteria.

Another aspect of variable selection which was investigated concerns the strategy used to search through the available models. Since an all possible subsets approach is infeasible when the number of input variables p becomes large, different stepwise procedures were investigated. Empirical evidence and results reported in the literature suggest that in this context backward elimination is superior to forward selection. We therefore presented a fairly extensive discussion of recursive feature elimination in Chapter 5, and found this to be an acceptable selection approach, especially in large p scenarios. Zero- and first-order forms of the new selection criteria proposed earlier in the thesis were presented for use in recursive feature eliminationand their properties investigated in a numerical study. It was found that some of the simpler zero-order criteria performed better than the more complicated first-order ones, making further investigation regarding the use of alternative zero-order criteria seem promising.

In Chapter 6 the focus fell on using the data to motivate a decision regarding the number of variables to select. We restricted attention to support vector machines, and proposed a new criterion, namely the number of support vectors, for this purpose. A complete strategy for solving the variable selection problem was obtained by combining this criterion with a backward elimination approach for searching through the space of variable subsets, and with the alignment and variation ratio for identifying the best subset of a given dimension. The alignment and variation ratio were used because of their good performance in earlier

chapters. Overall it seemed that the resulting complete variable selection approach (*i.e.*, a backward elimination strategy with variation ratio as inner criterion and the number of support vectors or cross-validation as outer criterion) performed quite well. Given that no complete solution to the variable selection problem for SVMs is available, we feel that this is an important contribution. It was shown in Chapter 6 that use of the number of support vectors is motivated by it being an important quantity in a well known upper bound on the generalisation performance of SVMs. Many alternative forms of such upper bounds can be found in the literature, and in these upper bounds, other quantities play an important role. Therefore it may prove worthwhile to investigate the use of such alternative quantities for a decision regarding the number of variables to select.

Other directions for further research are finding efficient ways to integrate variable selection and the specification of hyperparameter values, and also, to estimate the generalisation ability of a post-selection kernel classifier. Solutions to both of the above problems will undoubtedly prove very useful, but lie outside the scope of this thesis.

REFERENCES

Aizerman, M.A., Bravermann, E.M. and Rozonoér, L.I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, **25**, 821-837.

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, **2**, 203-217.

Aliferis, C.F., Tsamardinos, I., Massion, P., Statnikov, A., Fananapazir, N., and Hardin, D. (2003). Machine Learning models for classification of lung cancer and selection of genomic markers using array gene expression data. *Proceedings of FLAIRS*, *special track AI in Medicine*.

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D. and Levine, A.J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumour and normal colon tissues probed by oligonucleotide arrays. *Cell Biology*, **96**, 6745-6750.

Amari, S. and Wu, S. (1999). Improving support vector machines by modifying kernel functions. *Neural Networks*, **12**, 783-789.

Ambroise, C. and McLachlan, G.J. (2002). Selection bias in gene extraction on the basis of microarray gene expression data. *Proceedings of the National Academy of Science USA*, **99**, 6562-6566.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, **9**, 1545-1588.

Anlauf, J.K. and Biehl, M. (1989). The adatron: an adaptive perceptron algorithm. *Europhis. Letters*, **10**, 687-692.

Aronszajn, N. (1950). Theory of reproducing kernels. *Proceedings of the Cambridge Philosophical Society*, **40**, 337-401.

Barla, A., Odone, F. and Verri, A. (2003). Histogram intersection kernel for image classification. *Proceedings of the International Conference on Image Processing*, **3**, 513-516.

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms. *Machine Learning*, **36**(1,2), 105-139.

Bertero, M. (1986). Regularization methods for linear inverse problems. *Inverse problems*. Springer-Verlag, Berlin.

Bertero, M., Poggio, T. and Torre, V. (1988). Ill-posed problems in early vision. *Proceedings of the IEEE*, **76**, 869-889.

Boser, B.E., Guyon, I.M, and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th annual ACM Workshop on Computational Learning Theory*, 144-152.

Bousquet, O. and Elisseeff, A. (2002). Stability and generalisation. *Journal of Machine Learning Research*, **2**, 499-526.

Blum, A. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, **97**(1-2), 245-271.

Bradley, P.S., and Mangasarian, O.L. (1998). Feature selection via concave minimization and support vector machines. *Proceedings of the Fifteenth International Conference on Machine Learning*, 82-90.

Bradley, P.S., Mangasarian, O.L. and Street, W.N. (1998). Feature selection via mathematical programming. *INFORMS Journal on Computing*, **10**, 209-217.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and regression trees*. Chapman and Hall, New York.

Breiman, L. (1992). Little bootstrap and other methods for dimensionality selection in regression – X fixed prediction error. *Journal of the American Statistical Association*, **87**, 738-754.

Breiman, L. (1995). Better subset selection using the nonnegative garotte. *Technometrics*, **37**, 373-384.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **26**(2), 123-140.

Breiman, L. (1996). Heuristics of instability and stabilization in model selection. *Annals of Statistics*, **24**, 2350-2383.

Breiman, L. (2001). Random Forests. *Machine Learning*, **45**(1), 5-32.

Breiman, L. and Cutler, A. (2002). Manual On Setting Up, Using, And Understanding Random Forests V3.1, *http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1*.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares, M. Jr. and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Science USA*, **97**, 262-267.

Bunke, O. and Droge, B. (1984). Bootstrap and cross-validation estimates of the prediction error of linear regression models. *Annals of Statistics*, **12**, 1400-1424.

Burnham, K.P. and Anderson, D.R. (2002). *Model selection and multimodel inference*: *a practical information-theoretic approach*. Springer-Verlag, New York.

Cancedda, N., Gaussier, E., Goutte, C., Renders, J.-M., Kandola, J., Hofmann, T., Poggio, T. and Shawe-Taylor, J. (2003). Word-sequence kernels. *Journal of Machine Learning Research*, **3**(6), 1059-1082.

Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S. (2004). Choosing multiple parameters for support vector machines. *Machine Learning*, **46**(1-3), 131-159.

Chen, N., Lu, W.,Yang, J. and Li, G. (2004). *Support vector machines in chemistry*. World Scientific Publishing, Singapore.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, **20**(**3**), 273-297.

Cover, T.M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans*. *Electron*. *Comput*., **14**, 326–334.

Cox, D. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society*, *Series B*, **34**, 187-220.

Cristianini, N., Campbell, C. and Shawe-Taylor, J. (1998). Dynamically adapting kernels in support vector machines. In: *Proceedings of Neural Information Processing Workshop*, 204-210.

Cristianini, N., Lodhi, H. and Shawe-Taylor, J. (2000). Latent semantic kernels for feature selection. *NeuroCOLT*2 *Technical Report Series* NC-TR-2000-080, Royal Holloway College, University of London.

Cristianini, N. and Shawe-Taylor, J. (2000). *Support vector machines and other kernel-based learning algorithms*. Cambridge University Press, United Kingdom.

Cristianini, N., Shawe-Taylor, J. and Williamson, R.C. (2001). Introduction to the special issue on kernel methods. *Journal of Machine Learning Research*, **2**, 95-96.

Cristianini, N., Shawe-Taylor, J., Ellisseef, A. and Kandola, J. (2002). On kernel target alignment. *Advances in Neural Information Processing Systems*, MIT Press, Cambridge.

Dash, M. and Liu, H. (1997). Feature selection for classification. *International Journal of Intelligent Data Analysis* I, 131-156.

De'ath, G. and Fabricius, K.E. (2000). Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, **81**(11), 3178-3192.

Dietterich, T.G. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomisation. *Machine Learning*, **40**(2), 139-157.

Duan, K., Keerthi, S.S. and Poo, A.N. (2001). An empirical evaluation of simple performance measures for tuning SVM hyperparameters. *Unpublished manuscript*.

Duda, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. Wiley.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics*, **7**, 1-26.

Efron, B. (1982). *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, Philadelphia.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression (with discussion). *Annals of Statistics*, **32**, 407-499.

Evgeniou, T., Pontil, M. and Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, **13**(1), 1-50.

Friedman, J.H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, **84**, 165-175.

Friedman, J.H. (1991). Multivariate Adaptive Regression Splines (with discussion). *Annals of Statistics*, **19**, 1-141.

Friedman, J., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics*, **28**, 337-407.

Friedman, J., Hastie, T., Höfling, H. and Tibshirani, R. (2007). Pathwise coordinate optimization. To appear in *Annals of Applied Statistics*.

Fujarewicz and Wiench (2003). Selecting differentially expressed genes for colon tumor classification. *International Journal of Applied Mathematics and Computer Science*, **13**(3), 101-110.

Furey, T., Cristianini, N., Duffy, N., Bednarski, D., Schummer, M. and Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**, 906-914.

Furlanello, C., Serafini, M., Merler, S. and Jurman, G. (2003). Gene selection and classification by entropy-based recursive feature elimination. *Proceedings of the Joint International Conference on Neural Networks*, **4**, 3077-3082.

Geisser, S. (1975). The predictive sample re-use methods with application. *Journal of the American Statistical Association*, **70**, 320-328.

Genton, M.G. (2001). Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research*, **2**, 299-312.

George, E.I. (2000). The variable selection problem. *Journal of the American Statistical Association*, **95**(452), 1304-1308.

Ghosh, D. and Chinnaiyan, A. (2004). Classification and selection of biomarkers in genomic data using lasso. The University of Michigan, Department of Biostatistics *working paper series*, **42**. Berkeley Electronic Press.

Girosi, F., Jones, M. and Poggio, T. (1995). Regularization theory and neural network architectures. *Neural Computation*, **7**, 219-269.

Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., and Lander, E.S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression array monitoring. *Science*, **286**, 531-537.

Grandvalet, Y. and Canu, S. (2002). Adaptive scaling for feature selection in SVMs. In *Neural Information Processing Systems*, Paper AA09.

Guyon, I.M., Weston, J., Barnhill, S. and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, **46**, 389-422.

Guyon, I.M. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**, 1157-1182.

Habbema, J.D.F. and Hermans, J. (1977). Selection of variables in discriminant analysis by *F*-statistic and error rate. *Technometrics*, **19**, 487-493.

Hall, M.A. (1998). *Correlation-based feature selection for machine learning*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning*: *data mining*, *inference*, *and prediction*. Springer-Verlag, New York.

Haykin, S. (1999). *Neural networks*: *A comprehensive foundation*. Prentice Hall, New Jersey.

Herbrich, R. (2002). *Learning kernel classifiers*. MIT Press, Cambridge.

Ho, T.K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(8), 832-844.

Hoerl, A.E. and Kennard, R. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, **12**, 55-67.

Ishwaran, H. (2004). Discussion of "Least angle regression" by B. Efron, T. Hastie, I. Johnston and R. Tibshirani. *Annals of Statistics*, **32**, 452-457.

Ishwaran, H., Rao, J.S. (2005). Spike and slab variable selection: Frequentist and Bayesian strategies. *Annals of Statistics*, **33**(2), 730-773.

John, G. H., Kohavi, R. and Pfleger, K. (1994). Irrelevant features and the subset selection problem, *Machine Learning*: *Proceedings of the Eleventh International Conference*, 121-129.

Johnson, M.E. (1987). *Multivariate statistical simulation*. New York: Wiley.

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.

Keerthi, S.S. and Lin, C.J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, **15**, 1667-1689.

Keerthi, S.S. (2005). Generalized LARS as an effective feature selection tool for text classification with SVMs. *ACM International Conference Proceeding Series*, **119**, 417-424.

Kimeldorf, G.S. and Wahba, G. (1971). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, **41**, 495-502.

Kira, K. and Rendell, L. (1992). A practical approach to feature selection. *Proceedings of the* 9^{th} *International Conference on Machine Learning*, 249-256.

Kohavi, R. and John, G. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273-324.

Komura, D., Nakamura, H., Tsutsumi, S., Aburatani, H. and Ihara, S. (2004). Multidimensional support vector machines for visualization of gene expression data. *Bioinformatics*, **21**(4), 439-444.

Krishnapuram, B., Carin, L. and Hartemink, A. (2004). Gene expression array analysis: joint feature selection and classifier design. *Kernel Methods in Computational Biology*, MIT Press.

Kroon, R.S. (2003). Support Vector Machines, generalization bounds, and transduction. *MComm thesis*, Department of Computer Science, University of Stellenbosch.

Leray, P. and Gallinari, P. (1999). Feature selection with neural networks. *Behaviormetrika*, **26**(1),145-166.

Linhart, H. and Zucchini, W. (1986). *Model selection*. Wiley, New York.

Liu, H., Li, J. and Wong, L. (2002). A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, **13**, 51-60.

Li, Y., Campbell, C. and Tipping, M. (2002). Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, **18**, 1332-1339.

Lockhorst, J. (1999). The lasso and generalised linear models, Technical Report, University of Adelaide.

Lodhi, H., Saunders, C. Shawe-Taylor, J., Cristianini, N. Watkins, C. and Schölkopf, B. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, **2**:3, 419-444.

Louw, N. (1997). *Aspects of the pre- and post-selection classification performance of discriminant analysis and logistic regression*. *PhD Thesis*, The Department of Statistics and Actuarial Science, University of Stellenbosch, South Africa.

Louw, N. and Steel, S.J. (2005). A review of kernel Fisher discriminant analysis for statistical classification, *South African Statistical Journal*, **39**, 1-21.

Louw, N. and Steel, S.J. (2006). Variable selection in kernel Fisher discriminant analysis by means of recursive feature elimination. *Computational Statistics and Data Analysis*, **51**(3), 2043-2055.

Mardia, K.V., Kent, J.T. and Bibby, J.M. (1988). *Multivariate Analysis*. Academic Press, London.

Masotti, M. (2005). Exploring ranklets performances in mammographic mass classification using recursive feature elimination. *Unpublished Manuscript*, University of Bologna, Department of Physics, Bologna, Italy.

McKay, R.J. and Campbell, N.A. (1982 *a*). Variable selection techniques in discriminant analysis I. Description. *British Journal of Mathematical and Statistical Psychology*, **35**, 1-29.

McKay, R.J. and Campbell, N.A. (1982 *b*). Variable selection techniques in discriminant analysis II. Allocation. *British Journal of Mathematical and Statistical Psychology*, **35**, 30-41.

McLachlan, G.J. (1992). *Discriminant analysis and statistical pattern recognition*. Wiley, New York.

McLachlan, G.J., Do, K.A. and Ambroise, C. (2004). *Analysing microarray gene expression data*. Wiley, New York.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., Smola, A.J. and Müller, K.-R. (1999). Fisher discriminant analysis with kernels. Proceedings of *Neural Networks for Signal Processing*, **9**, 41-48.

Mika, S., Rätsch, G., and Müller, K.-R. (2001). A mathematical programming approach to the kernel Fisher algorithm. *Advances in Neural Information Processing Systems*, **13**, 591-597.

Mika, S. (2002). *Kernel Fisher Discriminants*. PhD thesis, Techniche Universität Berlin.

Miller, A. (2002). *Subset selection in regression*. Chapman and Hall, London.

Morrison, D.F. (1976). *Multivariate statistical models*. McGraw-Hill Series in Probability and Statistics, New York.

Mukherjee, S., Osuna, E. and Girosi, F. (1997). Non-linear prediction of chaotic time series using a support vector machine, *Neural Networks for Signal Processing*: *Proceedings of the* 1997 *IEEE Workshop*, **7**, 511-520.

Müller, K-R., Mika, S., Rätsch, G., Tsuda, K. and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. In: *IEEE Neural Networks*, **12**(2), 181-201.

Niijima, S., and Kuhara, S. (2006). Gene subset selection in kernel-induced feature space. *Pattern Recognition Letters*, **27**(16), 1884-1892.

Notterman, D., Alon, U., Sierk, A. and Levine, A. (2001). Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays. *Cancer Res*., **61**, 3124-3130.

Osborne, M., Presnell, B. and Turlach, B. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, **20**, 389-404.

Park, M.Y. and Hastie, T. (2006). L_1 regularization path algorithm for generalized linear models. *The Journal of the Royal Statistical Society B*, **58**, 267-288.

Pozdnoukhov, A. and Bengio, S. (2004). Tangent vector kernels for invariant image classification with SVMs. In: *Proceedings of the International Conference on Pattern Recognition*, 486-489.

Rakotomamonjy, A. (2002). Variable selection using SVM-based criteria. *Perception Systeme Information*, Insa de Rouen , Technical Report PSI 2002-04.

Rakotomamonjy, A. (2003). Variable selection using SVM-based criteria. *Journal of Machine Learning Research*, **3**, 1357-1370.

Rätsch, G., Onoda, T. and Muller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning*, **42**, 287-320.

Rao, J.S. (1999). Bootstrap choice of cost complexity for better subset selection. *Statistica Sinica*, **9**, 273-287.

Rockafellar, R.T. (1970). *Convex analysis*. Princeton University Press.

Rosenblatt, F. (1959). *Principles of Neurodynamics*. Spartan Books, New York.

Schölkopf, B., Burges, C.J.C. and Vapnik, V. (1996). Incorporating invariances in support vector learning machines. *Artificial Neural Networks*, *Springer Lecture Notes in Computer Science*, **112**, 47-52.

Schölkopf, B., Kah-Kay, S., Burges, C.J.C., Girosi, F., Niyogi, P., Poggio, T. and Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radialbasis function classifiers. In: *IEEE Signal Processing*, **45**(11), 2758-2765.

Schölkopf, B., Burges, C.J.C. and Smola, A.J. (1999). *Advances in kernel methods*: *support vector learning*. MIT Press, Cambridge.

Schölkopf, B. and Smola, A.J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, London.

Schummer, M., Ng, W., Bumgarner, R., Nelson, P., Schummer, B., Bednarski, D., Hassell, L., Baldwin, R., Karlan, B., and Hood, L. (1999). Comparative hybridization of an array of 21 500 ovarian cDNAs for the discovery of genes overexpressed in ovarian carcinomas. *Genes*, **238**, 375-385.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 461-464.

Shao, J. and Rao, J.S. (2000). The GIC for model selection: A hypothesis testing approach. Linear models. *Journal of Statistical Plann. Inference*, **88**, 215-231.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press, Cambridge.

Shi, T. and Horvath, S. (2006). *Unsupervised learning with Random Forest predictors*. Journal of Computational and Graphical Statistics, **15**(1), 118-138.

Shi, T., Seligson, D., Belldegrun, A.S., Palotie, A. and Horvath, S. (2005). *Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma*. Modern Pathology, **18**(4), 547-57.

Steel, S.J. and Louw, N. (2004). Simple and fast model selection for the Gaussian kernel. *Unpublished manuscript*.

Stitson, M.O., Gammerman, A., Vapnik,V., Vovk, V., Watkins, C. and Weston, J. (1997). Support vector ANOVA decomposition. *Technical report*, *Royal Holloway College*, *Report number* CSD-TR-97-23.

Stone, M. (1977). An asymptotic equivalence of choice of model by cross-validation and Akaike's Criterion. *Journal of the Royal Statistical Society B*, **38**, 44-47.

Stoppiglia, H., and Dreyfus, G. (2003). Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, **3**, 1399-1414.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, **58**, 267–288.

Tikhonov, A.N. and Arsenin, V.Y. (1977). *Solutions of ill-posed problems*. W.H. Winston, Washington D.C.

Van Gestel, T., Suykens, J.A.K., Lanckriet, G., Lambrechts, A., de Moor, B. and Vandewalle, J. (2002). Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel Fisher discriminant analysis. *Neural Computation*, **14**, 1115-1147.

Vapnik, V.N. and Lerner, A. (1963). Pattern recognition using a generalized portrait method. *Automation and Remote Control*, **24**, 774-780.

Vapnik, V.N. (1998). *Statistical Learning Theory*, Wiley, New York.

Vapnik, V.N. and Chapelle, O. (2000). Bounds on error expectation for support vector machines. *Neural Computation*, **12**, 2013-2036.

Wahba, G. (1990). *Spline models for observational data*. Series in Applied Mathematics, **59**, SIAM, Philadelphia.

Wahba, G. (1998). Support vector machines, reproducing kernel Hilbert spaces and the GACV. *Technical Report 984rr*, Department of Statistics, University of Wisconsin, Madison WI.

Wang, W., Xu, Z., Lu, W., Zhang, X. (2004). Determination of the spread parameter in the Gaussian kernel for classification and regression. *Neurocomputing*, **58-60**, 655-662.

Weston, J., Gammerman, A., Stitson, O., Vapnik, V., Vovk, V. and Watkins, C. (1997). Density estimation using support vector machines. *Technical Report*, Royal Holloway College, Report number CSD-TR-97-23.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. and Vapnik, V. (2001). Feature selection for SVMs. *Neural Information Processing Systems*, Cambridge MA, MIT Press.

Weston, J., Elisseeff, A., Schölkopf, B. and Tipping, M. (2003). Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, **3**, 1439-1461.

Zhang, T. (2004). Statistical behavior and consistency of classification methods based on convex risk minimisation. *Annals of Statistics*, **32**, 56-85.

Zhu, J. and Hastie, T. (2004). Classification of gene microarrays by penalized logistic regression. *Biostatistics*, **5**, 427-443.

Zhu, J. and Hastie, T. (2005). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, **14**, 185-205.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, **67**(2), 301-320.

APPENDIX A

SIMULATION STUDY DISTRIBUTIONS AND COMPREHENSIVE RESULTS

There are two sections in this appendix. In Section A.1 we describe the procedure which was used in the empirical work to generate correlated random variables with lognormal marginal distributions. Section A.2 contains tables which provide a comprehensive summary of the empirical study results discussed in Chapters 4 and 5.

A.1 *GENERATING MULTIVARIATE DATA WITH LOGNORMAL MARGINAL DISTRIBUTIONS*

In simulation experiments conducted in the course of the thesis, we used different correlation structures between pairs of input variables in the normal and lognormal data scenarios. In this section some points to consider when generating lognormal data are discussed, and a motivation is given for using different sets of correlation values for the normal and lognormal data setups in our simulation design. In Section A.1.1 we start with a series of results from matrix algebra required to understand the technical difficulties one might run into when generating multivariate lognormal data. Since we need to specify correlations between input variables which yield a positive-definite covariance matrix, results required to ensure positive-definiteness of a (covariance) matrix are given in Section A.1.2. Using the results in Sections A.1.1 and A.1.2, a motivation for the correlation structure used in generating lognormal data, is given in Section A.1.3.

A.1.1 *RESULTS FROM MATRIX ALGEBRA*

The following results are required in Section A.1.2.

RESULT A.1.1: ***THE DETERMINANT OF AN EQUI-CORRELATION MATRIX***

(See Mardia and Kent, 1988, *p*. 461)

Consider the equi-correlation matrix

$$\underset{p\times p}{\boldsymbol{A}} = (1-\rho)\boldsymbol{I}_p + \rho\ \underset{p\times p}{\boldsymbol{E}}\ . \tag{A.1}$$

Then

$$\underset{p\times p}{\boldsymbol{A}}^{-1} = \frac{1}{1-\rho}\left(\boldsymbol{I}_p - \frac{\rho}{1+(p-1)\rho}\ \underset{p\times p}{\boldsymbol{E}}\right), \tag{A.2}$$

and

$$|\boldsymbol{A}| = (1-\rho)^{p-1}\left(1+(p-1)\rho\right)\ . \tag{A.3}$$

RESULT A.1.2: ***THE INVERSE OF A MATRIX UNDER SPECIAL CONDITIONS*** (See Morrison, 1976, *pp*. 69-70)

Consider the $p\times p$ matrix

$$\underset{p\times p}{\boldsymbol{K}} = \begin{bmatrix} a & c & \cdots & c \\ c & a & \ddots & c \\ \vdots & \ddots & \ddots & \vdots \\ c & c & \cdots & a \end{bmatrix}. \tag{A.4}$$

The inverse of this matrix has common diagonal element

$$\frac{a+(p-2)c}{(a-c)(a+(p-1)c)}, \tag{A.5}$$

and common off-diagonal element

$$-\frac{c}{(a-c)(a+(p-1)c)}. \tag{A.6}$$

RESULT A.1.3: ***THE DETERMINANT OF A MATRIX UNDER SPECIAL CONDITIONS***

Consider again the matrix $\boldsymbol{K}$ in Result A.1.2. Since

$$\underset{p\times p}{\boldsymbol{K}} = a \begin{bmatrix} 1 & \frac{c}{a} & \cdots & \frac{c}{a} \\ \frac{c}{a} & 1 & \cdots & \frac{c}{a} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{c}{a} & \frac{c}{a} & \cdots & 1 \end{bmatrix}, \tag{A.7}$$

it follows from Result A.1.1 that

$$\left|\boldsymbol{K}\right| = a^{p}\left(1-\frac{c}{a}\right)^{p-1}\left(1+(p-1)\frac{c}{a}\right). \tag{A.8}$$

A.1.2 *NON-SINGULARITY (POSITIVE DEFINITENESS) OF A GIVEN MATRIX*

We consider a $p \times p$ matrix

$$\underset{p\times p}{\boldsymbol{\Sigma}} = \begin{bmatrix} \underset{p_1\times p_1}{\boldsymbol{\Sigma}_{11}} & \underset{p_1\times p_2}{\boldsymbol{\Sigma}_{12}} \\ \underset{p_2\times p_1}{\boldsymbol{\Sigma}_{21}} & \underset{p_2\times p_2}{\boldsymbol{\Sigma}_{22}} \end{bmatrix}. \tag{A.9}$$

We assume that $\boldsymbol{\Sigma}$ is symmetric and we would like to determine conditions which guarantee that $\boldsymbol{\Sigma}$ is positive definite, *i.e.* conditions under which $\left|\boldsymbol{\Sigma}\right| > 0$.

The following notation is used regarding the submatrices of $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma}_{11} : p_1 \times p_1 = \begin{bmatrix} 1 & a & \dots & a \\ a & 1 & \ddots & a \\ \vdots & \ddots & \ddots & \vdots \\ a & a & \dots & 1 \end{bmatrix}, \tag{A.10}$$

$$\boldsymbol{\Sigma}_{22} : p_2 \times p_2 = \begin{bmatrix} \sigma^2 & b & \cdots & b \\ b & \sigma^2 & \ddots & b \\ \vdots & \ddots & \ddots & \vdots \\ b & b & \dots & \sigma^2 \end{bmatrix}, \tag{A.11}$$

$$\text{and } \boldsymbol{\Sigma}_{12} : p_1 \times p_2 = \begin{bmatrix} c & c & \cdots & c \\ c & c & \ddots & c \\ \vdots & \ddots & \ddots & \vdots \\ c & c & \cdots & c \end{bmatrix} = \boldsymbol{\Sigma}_{21}' . \tag{A.12}$$

It is known that

$$\left|\boldsymbol{\Sigma}\right| = \left|\boldsymbol{\Sigma}_{11}\right| . \left|\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right| , \tag{A.13}$$

and from Result A.1.1. we may write

$$\left|\boldsymbol{\Sigma}_{11}\right| = \left(1-a\right)^{p_1-1}\left(1+\left(p_1-1\right)a\right) . \tag{A.14}$$

We now require $\left|\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right|$. From Result A.1.2 we know that $\boldsymbol{\Sigma}_{11}^{-1}$ is a matrix with common diagonal element

$$r = \frac{1+\left(p_1-2\right)a}{\left(1-a\right)\left(1+\left(p_1-1\right)a\right)} , \tag{A.15}$$

and common off-diagonal element

$$s = -\frac{a}{\left(1-a\right)\left(1+\left(p_1-1\right)a\right)} . \tag{A.16}$$

We therefore have

$$\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12} = c \underset{p_2\times p_1}{\boldsymbol{E}} \boldsymbol{\Sigma}_{11}^{-1} c \underset{p_1\times p_2}{\boldsymbol{E}} = c^2\left(r+\left(p_1-1\right)s\right) \underset{p_2\times p_1}{\boldsymbol{E}} \underset{p_1\times p_2}{\boldsymbol{E}} ,$$

which yields

$$\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12} = c^2\left(r+\left(p_1-1\right)s\right)p_1 \underset{p_2\times p_1}{\boldsymbol{E}} = p_1c^2\left(r+\left(p_1-1\right)s\right)\underset{p_2\times p_1}{\boldsymbol{E}} .$$

Hence $\boldsymbol{\Sigma}_{22}-\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}$ will be a matrix with common diagonal element $t=\sigma^2-p_1c^2\left(r+\left(p_1-1\right)s\right)$ and common off-diagonal element $u=b-p_1c^2\left(r+\left(p_1-1\right)s\right)$. We are now in a position to use Result A.1.3 to write down $\left|\boldsymbol{\Sigma}_{22}-\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right|$. It follows that

$$\left|\boldsymbol{\Sigma}_{22}-\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right| = t^{p_2}\left(1-\tfrac{u}{t}\right)^{p_2-1}\left(1+\left(p_2-1\right)\tfrac{u}{t}\right)$$

Combining all the above results we find

$$\begin{aligned}\left|\boldsymbol{\Sigma}\right| &= \left|\boldsymbol{\Sigma}_{11}\right|.\left|\boldsymbol{\Sigma}_{22}-\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right| \\ &= \left(1-a\right)^{p_1-1}\left(1+\left(p_1-1\right)a\right)t^{p_2}\left(1-\tfrac{u}{t}\right)^{p_2-1}\left(1+\left(p_2-1\right)\tfrac{u}{t}\right) .\end{aligned}$$

Our problem now is to determine the conditions under which the above expression is positive. We first of all require $\boldsymbol{\Sigma}_{11}$ to be positive-definite, since $\boldsymbol{\Sigma}_{11}$ is the covariance matrix of the first p_1 random variables. Hence we require

$$\begin{aligned}\left(1-a\right)^{p_1-1}\left(1+\left(p_1-1\right)a\right)>0 \quad &\Leftrightarrow \quad 1-a>0 \text{ and } 1+\left(p_1-1\right)a>0 \text{, or} \\ &\qquad\quad 1-a<0 \text{ and } 1+\left(p_1-1\right)a<0 \\ &\Leftrightarrow \quad -\frac{1}{p_1-1}<a<1 \\ &\qquad\quad \text{or } a>1 \text{ and } a<-\frac{1}{p_1-1} \text{ (impossible)} .\end{aligned}$$

Our first requirement is therefore that

$$-\frac{1}{p_1-1} < a < 1, \tag{A.17}$$

and this guarantees that $\left|\boldsymbol{\Sigma}_{11}\right| > 0$. We now also require $\left|\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right| > 0$. Hence consider

$$\begin{aligned}
\left|\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right| &= t^{p_2}\left(1-\tfrac{u}{t}\right)^{p_2-1}\left(1+\left(p_2-1\right)\tfrac{u}{t}\right) \\
&= t^{p_2}\left(1-\tfrac{u}{t}\right)^{p_2-1}\left(\left(1-\tfrac{u}{t}\right)+p_2\,\tfrac{u}{t}\right) \\
&= t^{p_2}\left(1-\tfrac{u}{t}\right)^{p_2} + t^{p_2}\left(1-\tfrac{u}{t}\right)^{p_2-1} p_2\,\tfrac{u}{t} \\
&= \left(t-u\right)^{p_2} + p_2 u t^{p_2-1}\left(1-\tfrac{u}{t}\right)^{p_2-1} \\
&= \left(t-u\right)^{p_2-1}\left(t-u+p_2 u\right) \\
&= \left(t-u\right)^{p_2-1}\left(t+\left(p_2-1\right)u\right) .
\end{aligned}$$

Therefore $\left|\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right| > 0 \quad \Leftrightarrow \quad t > u$ and $t+\left(p_2-1\right)u > 0$, or

$t < u$ and $t+\left(p_2-1\right)u < 0$.

From the earlier expressions for r and s it follows that

$$r+\left(p_1-1\right)s = \frac{1}{1+\left(p_1-1\right)a} , \tag{A.18}$$

and this is a positive quantity since we require $a > -\dfrac{1}{p_1-1}$ for $\left|\boldsymbol{\Sigma}_{11}\right|$ to be positive.

Also, since $t = \sigma^2 - p_1 c^2 \left(r + \left(p_1 - 1 \right) s \right)$ and $u = b - p_1 c^2 \left(r + \left(p_1 - 1 \right) s \right)$, it now follows that

$$t = \sigma^2 - \frac{p_1 c^2}{1 + \left(p_1 - 1 \right) a} = \frac{\sigma^2 + \left(p_1 - 1 \right) a \sigma^2 - p_1 c^2}{1 + \left(p_1 - 1 \right) a}$$

and $u = b - \dfrac{p_1 c^2}{1 + \left(p_1 - 1 \right) a} = \dfrac{b + \left(p_1 - 1 \right) ab - p_1 c^2}{1 + \left(p_1 - 1 \right) a}$, so that it can easily be seen that

$$t < u \quad \Leftrightarrow \quad \sigma^2 < b$$
$$\text{or} \quad t > u \quad \Leftrightarrow \quad \sigma^2 > b \, .$$

Also,

$$t + \left(p_2 - 1 \right) u < 0 \qquad \text{or} \qquad t + \left(p_2 - 1 \right) u > 0$$

$$\Leftrightarrow \quad \sigma^2 - \frac{p_1 c^2}{1 + \left(p_1 - 1 \right) a} + \left(p_2 - 1 \right) \left(b - \frac{p_1 c^2}{1 + \left(p_1 - 1 \right) a} \right) < 0$$

$$\text{or} \quad \sigma^2 - \frac{p_1 c^2}{1 + \left(p_1 - 1 \right) a} + \left(p_2 - 1 \right) \left(b - \frac{p_1 c^2}{1 + \left(p_1 - 1 \right) a} \right) > 0$$

$$\Leftrightarrow \quad \left(\sigma^2 - b \right) + p_2 b - \frac{p_1 p_2 c^2}{1 + \left(p_1 - 1 \right) a} < 0$$

$$\text{or} \quad \left(\sigma^2 - b \right) + p_2 b - \frac{p_1 p_2 c^2}{1 + \left(p_1 - 1 \right) a} > 0 \, .$$

Hence, our second requirement, which guarantees that $\left| \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12} \right| > 0$, is

$$\sigma^2 > b \text{ and } \left(\sigma^2 - b \right) + p_2 b - \frac{p_1 p_2 c^2}{1 + \left(p_1 - 1 \right) a} > 0 \, , \qquad \text{(A.19)}$$

$$\text{or } \sigma^2 < b \text{ and } \left(\sigma^2 - b\right) + p_2 b - \frac{p_1 p_2 c^2}{1+\left(p_1 - 1\right)a} < 0 \,. \tag{A.20}$$

Therefore, if one specifies a (covariance) matrix (A.9) to have a structure as given in (A.10) to (A.12), positive definiteness of the matrix is guaranteed if (A.17) and (A.19) or (A.20) hold.

A.1.3 *OUR APPLICATION*

The Johnson translation system for generating multivariate data with lognormal marginal distributions (Johnson, 1987) starts with normally distributed random variables $Z_1, Z_2, \ldots, Z_p$, *i.e.*

$$Z_j \sim N\left(0, \sigma_j^2\right),\ j = 1, 2, \ldots, p \text{, where } covar\left(Z_j, Z_k\right) = \sigma_{jk},\ j \neq k \,. \tag{A.21}$$

The standard normal random variables $Z_1, Z_2, \ldots, Z_p$ are then transformed to lognormally distributed random variables $X_1, X_2, \ldots, X_p$, via

$$X_j = \lambda_j e^{Z_j} + \xi_j \quad ,\ j = 1, 2, \ldots, p \ . \tag{A.22}$$

This implies

i. $E\left(X_j\right) = \lambda_j E\left(e^{Z_j}\right) + \xi_j = \lambda_j e^{\sigma_j^2/2} + \xi_j$

ii. $E\left(X_j^2\right) = \lambda_j^2 e^{2\sigma_j^2} + 2\lambda_j \xi_j e^{\sigma_j^2/2} + \xi_j^2$, and

iii. $E\left(X_j X_k\right) = \lambda_j \lambda_k e^{\left(\sigma_j^2 + \sigma_k^2 + 2\sigma_{jk}\right)/2} + \lambda_j \xi_k e^{\sigma_j^2/2} + \lambda_k \xi_j e^{\sigma_k^2/2} + \xi_j \xi_k \ .$

Hence also

$$iv.\quad var\left(X_j\right)=\lambda_j^2 e^{2\sigma_j^2}-\lambda_j^2 e^{\sigma_j^2}=\lambda_j^2 e^{\sigma_j^2}\left(e^{\sigma_j^2}-1\right), \text{ and}$$

$$v.\quad covar\left(X_j,X_k\right)=\lambda_j\lambda_k e^{\left(\sigma_j^2+\sigma_k^2+2\sigma_{jk}\right)/2}-\lambda_j\lambda_k e^{\left(\sigma_j^2+\sigma_k^2\right)/2}\ .$$

We now require λ_j, ξ_j and σ_{jk}, $j,k=1,2,\ldots,p;\ \ j\neq k$, such that the following requirements are satisfied:

$$E\left(X_j\right)=\mu_j;\ \ var\left(X_j\right)=s_j^2;\ \text{and}\ \ corr\left(X_j,X_k\right)=\rho_{jk}\ . \tag{A.23}$$

Now,

$$var\left(X_j\right)=s_j^2\ \Rightarrow\ \ \lambda_j^2 e^{\sigma_j^2}\left(e^{\sigma_j^2}-1\right)=s_j^2$$

$$\Rightarrow\ \ \lambda_j=\frac{s_j}{\sqrt{e^{\sigma_j^2}\left(e^{\sigma_j^2}-1\right)}},\ j=1,2,\ldots,p\ . \tag{A.24}$$

Also,

$$E\left(X_j\right)=\mu_j\ \Rightarrow\ \ \lambda_j e^{\sigma_j^2/2}+\xi_j=\mu_j$$

$$\Rightarrow\ \ \xi_j=\mu_j-\frac{s_j\, e^{\sigma_j^2/2}}{\sqrt{e^{\sigma_j^2}\left(e^{\sigma_j^2}-1\right)}},\ j=1,2,\ldots,p\ . \tag{A.25}$$

Finally,

$$corr\left(X_j, X_k\right) = \rho_{jk} \;\Rightarrow\; \frac{\lambda_j \lambda_k e^{\left(\sigma_j^2+\sigma_k^2\right)/2}\left(e^{\sigma_{jk}}-1\right)}{\lambda_j \lambda_k e^{\left(\sigma_j^2+\sigma_k^2\right)/2}\sqrt{\left(e^{\sigma_j^2}-1\right)\left(e^{\sigma_k^2}-1\right)}} = \rho_{jk}$$

$$\Rightarrow\; \sigma_{jk} = \log\left(1+\rho_{jk}\sqrt{\left(e^{\sigma_j^2}-1\right)\left(e^{\sigma_k^2}-1\right)}\right)$$

$$j,k = 1,2,\ldots,p;\;\; j \neq k\,. \qquad \text{(A.26)}$$

Use of the Johnson translation system for generating correlated lognormal data first requires specification of $\sigma_j^2, j = 1,2,\ldots,p$, and ρ_{jk}, $j,k = 1,2,\ldots,p;\;\; j \neq k$ (in A.26). These specifications then imply the values for ξ_j, λ_j and σ_{jk}, $j,k = 1,2,\ldots,p;\;\; j \neq k$, such that (A.23) will hold. In this section it was shown that care must however be taken when specifying $\sigma_j^2, j = 1,2,\ldots,p$, and ρ_{jk}, $j,k = 1,2,\ldots,p;\;\; j \neq k$, so that use of the Johnson translation system does not imply σ_{jk} values rendering the covariance matrix for the initial normal input variables a singular matrix. Via Johnson translations, generating lognormal data with a particular structure first requires normal data with a specific structure. In Section 3.4.1, since lognormal data with $\rho_S = 0$ or 0.7, and $\rho_{S\bar{S}} = 0.9$ implied the use of normal data with a singular covariance matrix, we were forced to use $\rho_{S\bar{S}} = 0.2$ or $\rho_{S\bar{S}} = 0.25$ instead.

A.2 *COMPREHENSIVE SIMULATION STUDY RESULTS*

A.2.1 *KERNEL VARIABLE SELECTION IN FEATURE SPACE*

In this section we present the remaining set of results of the Monte Carlo simulation study described in Chapter 4, where we studied the relative performance of selection criteria defined in feature space.

Table A.1: Average test errors in the *LL* case

		KFDA						
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(VA)$	$Err(M)$	$Err(SS)$	$Err(A)$	$Err(AT)$
LL1	***SMALL***	.337	.135	.155	.159	.219	.139	.139
	MIXED	.165	.079	.079	.080	.127	.080	.080
	LARGE	.215	.122	.122	.122	.183	.122	.122
	WIDE	.301	.058	.088	.089	.095	.066	.064
LL2	***SMALL***	.166	.078	.083	.085	.103	.079	.079
	MIXED	.078	.042	.042	.042	.056	.042	.042
	LARGE	.072	.032	.032	.041	.035	.032	.032
	WIDE	.122	.039	.045	.045	.050	.043	.042
LL3	***SMALL***	.330	.137	.154	.157	.214	.140	.139
	MIXED	.160	.079	.079	.079	.127	.079	.080
	LARGE	.213	.121	.121	.121	.179	.121	.121
	WIDE	.352	.117	.164	.164	.169	.127	.124
LL4	***SMALL***	.219	.124	.128	.129	.151	.125	.125
	MIXED	.100	.079	.079	.079	.092	.079	.079
	LARGE	.151	.084	.084	.084	.092	.084	.084
	WIDE	.228	.099	.132	.132	.143	.108	.106

Table A.2: Average test errors in the *LL* case (continued)

		SVM						
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(VA)$	$Err(M)$	$Err(SS)$	$Err(A)$	$Err(AT)$
LL1	***SMALL***	.331	.122	.139	.145	.208	.126	.126
	MIXED	.171	.064	.064	.065	.116	.064	.065
	LARGE	.232	.115	.115	.115	.176	.115	.115
	WIDE	.265	.068	.091	.092	.099	.066	.065
LL2	***SMALL***	.157	.084	.089	.090	.108	.086	.085
	MIXED	.071	.045	.042	.043	.057	.042	.042
	LARGE	.077	.033	.033	.033	.043	.033	.033
	WIDE	.089	.020	.030	.030	.032	.023	.022
LL3	***SMALL***	.334	.121	.141	.144	.203	.126	.125
	MIXED	.171	.064	.065	.065	.114	.064	.067
	LARGE	.229	.113	.113	.113	.174	.113	.113
	WIDE	.324	.122	.162	.164	.170	.131	.130
LL4	***SMALL***	.216	.127	.134	.135	.156	.130	.129
	MIXED	.100	.068	.068	.069	.078	.069	.069
	LARGE	.143	.084	.084	.084	.098	.084	.084
	WIDE	.218	.128	.144	.145	.150	.132	.132

Table A.3: Average test errors in the *LS* case

		KFDA						
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(VA)$	$Err(M)$	$Err(SS)$	$Err(A)$	$Err(AT)$
***LS*1**	***SMALL***	.438	.225	.367	.369	.407	.275	.269
	MIXED	.250	.250	.250	.250	.250	.250	.250
	LARGE	.360	.201	.207	.219	.451	.202	.201
	WIDE	.499	.119	.385	.384	.377	.185	.179
***LS*2**	***SMALL***	.262	.126	.181	.182	.239	.151	.149
	MIXED	.193	.096	.139	.145	.250	.250	.250
	LARGE	.154	.085	.086	.089	.214	.085	.085
	WIDE	.279	.054	.215	.215	.233	.099	.093
***LS*3**	***SMALL***	.442	.223	.375	.378	.421	.278	.270
	MIXED	.250	.250	.250	.250	.250	.250	.250
	LARGE	.360	.201	.210	.223	.448	.201	.201
	WIDE	.457	.141	.421	.420	.413	.243	.236
***LS*4**	***SMALL***	.323	.149	.205	.209	.276	.175	.171
	MIXED	.243	.129	.177	.182	.250	.250	.250
	LARGE	.206	.126	.127	.130	.250	.126	.126
	WIDE	.342	.082	.313	.313	.317	.179	.172

Table A.4: Average test errors in the *LS* case (continued)

		SVM						
		$Err(\mathcal{V})$	$Err(V_S)$	$Err(VA)$	$Err(M)$	$Err(SS)$	$Err(A)$	$Err(AT)$
***LS*1**	***SMALL***	.441	.213	.368	.375	.414	.153	.150
	MIXED	.335	.227	.242	.250	.250	.250	.250
	LARGE	.361	.180	.187	.203	.444	.181	.181
	WIDE	.449	.136	.391	.388	.385	.194	.188
***LS*2**	***SMALL***	.269	.125	.185	.186	.252	.153	.150
	MIXED	.195	.086	.125	.130	.330	.295	.194
	LARGE	.163	.076	.078	.081	.199	.076	.076
	WIDE	.260	.062	.237	.236	.236	.101	.097
***LS*3**	***SMALL***	.442	.208	.364	.375	.418	.267	.259
	MIXED	.333	.224	.249	.248	.265	.268	.266
	LARGE	.361	.179	.187	.201	.448	.181	.181
	WIDE	.449	.174	.423	.423	.420	.248	.241
***LS*4**	***SMALL***	.315	.166	.214	.222	.294	.193	.189
	MIXED	.251	.095	.143	.147	.335	.309	.308
	LARGE	.221	.092	.093	.096	.239	.092	.092
	WIDE	.327	.168	.316	.316	.317	.209	.205

A.2.2 *BACKWARD ELIMINATION FOR KERNEL CLASSIFIERS*

In this section we present the full set of results that was obtained in the Monte Carlo simulation study in Chapter 5, where the performances of RFE selection criteria were evaluated. The average KFDA test errors for the *NS* case and small, mixed, large and wide sample sizes are given first, followed by the average SVM test errors corresponding to the same cases. Results pertaining to the *LL* and *LS* data scenarios are given in similar order. Conclusions drawn from Tables A.5-A.28 (given below) are discussed in Chapter 5.

Table A.5: Average KFDA test errors for the *NS* case and small samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.455	.285	.343 1.204 3	.348 1.221 4	.414 1.453 9	.355 1.246 5	.377 1.323 6	.340 1.193 1	.405 1.421 7	.487 1.709 11	.411 1.442 8	.498 1.747 12	.461 1.618 10	.341 1.196 2
.255	.104	.110 1.058 1	.117 1.125 3	.141 1.356 7	.132 1.269 5	.135 1.298 6	.132 1.269 5	.122 1.173 4	.208 2.000 9	.128 1.085 2	.450 4.327 11	.281 2.702 10	.171 1.644 8
.453	.281	.343 1.221 3	.332 1.181 1	.416 1.480 9	.364 1.295 5	.384 1.367 6	.345 1.228 4	.408 1.452 7	.489 1.740 11	.412 1.466 8	.499 1.776 12	.461 1.641 10	.341 1.214 2
.306	.152	.191 1.257 4	.182 1.197 2	.215 1.414 8	.205 1.349 6	.224 1.474 9	.209 1.375 7	.176 1.158 1	.327 2.151 3	.194 1.276 5	.447 2.941 12	.377 2.480 11	.248 1.632 10

Table A.6: Average KFDA test errors for the *NS* case and mixed samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.250	.250	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1
.201	.086	.089 1.035 3	.086 1.000 1	.250 2.907 6	.087 1.012 2	.250 2.907 6	.140 1.628 4	.087 1.012 2	.250 2.907 6	.089 1.035 3	.250 2.907 6	.250 2.907 6	.180 2.093 5
.250	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1
.237	.122	.172 1.410 5	.125 1.025 1	.250 2.049 8	.143 1.172 4	.303 2.484 9	.183 1.500 6	.126 1.033 2	.250 2.049 8	.130 1.066 3	.250 2.049 8	.250 2.049 8	.235 1.926 7

Table A.7: Average KFDA test errors for the *NS* case and large samples

FULL	*ORA*	*R0*	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
.379	.257	.259 1.008 3	.272 1.058 4	.258 1.004 2	.257 1.000 1	.257 1.000 1	.257 1.000 1	.257 1.000 1	.498 1.938 6	.257 1.000 1	.257 1.000 1	.442 1.720 5	.257 1.000 1
.136	.084	.084 1.000 1	.109 1.298 3	.084 1.000 1	.084 1.000 1	.084 1.000 1	.084 1.000 1	.084 1.000 1	.109 1.298 3	.084 1.000 1	.084 2.429 4	.204 1.012 2	.085 1.012 2
.379	.258	.259 1.004 2	.273 1.058 3	.259 1.004 2	.258 1.000 1	.258 1.000 1	.258 1.000 1	.258 1.000 1	.499 1.934 4	.259 1.004 2	.258 1.000 1	.435 1.686 4	.259 1.004 2
.205	.111	.133 1.198 4	.123 1.108 2	.111 1.000 1	.111 1.000 1	.111 1.000 1	.111 1.000 1	.111 1.000 1	.291 2.622 5	.111 1.000 1	.111 1.000 1	.327 2.946 6	.131 1.180 3

Table A.8: Average KFDA test errors for the *NS* case and wide samples

FULL	*ORA*	*R0*	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
.410	.087	.106 1.218 1	.199 2.287 3	.438 5.034 9	.414 4.759 6	.326 3.747 5	.268 3.080 4	.421 4.839 7	.485 5.575 11	.432 4.966 8	.496 5.701 12	.467 5.368 10	.158 1.816 2
.130	.003	.004 1.333 1	.004 1.333 1	.007 2.333 4	.007 2.333 4	.006 2.000 3	.006 2.000 3	.005 1.667 2	.022 7.333 6	.006 2.000 3	.452 150.7 8	.135 45.00 7	.009 3.000 5
.427	.142	.222 1.563 1	.226 1.592 2	.448 3.155 9	.427 3.007 7	.375 2.641 5	.342 2.408 4	.402 2.831 6	.492 3.465 11	.433 3.049 8	.498 3.507 12	.482 3.394 10	.268 1.887 3
.222	.019	.136 7.158 4	.038 2.000 1	.146 7.684 6	.139 7.316 5	.164 8.632 8	.160 8.421 7	.054 2.842 2	.308 16.21 10	.098 5.158 3	.475 25.00 12	.397 20.89 11	.159 8.368 9

Table A.9: Average SVM test errors for the *NS* case and small samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.453	.294	.381 1.296 5	.365 1.241 3	.416 1.415 8	.358 1.218 2	.379 1.289 4	.343 1.167 1	.407 1.384 6	.488 1.660 10	.413 1.405 7	.499 1.697 11	.462 1.571 9	.343 1.167 1
.260	.128	.162 1.266 5	.187 1.461 7	.162 1.266 5	.155 1.211 3	.157 1.227 4	.153 1.195 2	.147 1.148 1	.215 1.680 8	.153 1.195 2	.454 3.547 10	.287 2.242 9	.184 1.438 6
.451	.293	.383 1.307 5	.365 1.246 3	.418 1.427 9	.368 1.256 4	.388 1.324 6	.349 1.191 2	.411 1.403 7	.489 1.669 11	.415 1.416 8	.498 1.700 12	.461 1.573 10	.345 1.177 1
.320	.183	.255 1.393 6	.274 1.497 9	.241 1.317 5	.229 1.251 3	.246 1.344 7	.233 1.273 4	.204 1.115 1	.333 1.820 10	.220 1.202 2	.452 2.470 12	.385 2.104 11	.261 1.426 8

Table A.10: Average SVM test errors for the *NS* case and mixed samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.332	.253	.261 1.032 5	.259 1.024 4	.268 1.059 8	.253 1.000 1	.268 1.059 8	.253 1.000 1	.258 1.020 3	.266 1.051 7	.259 1.024 4	.265 1.047 6	.268 1.059 9	.256 1.012 2
.209	.096	.100 1.042 4	.103 1.073 5	.367 3.823 10	.097 1.010 1	.369 3.844 11	.152 1.583 6	.098 1.021 2	.332 3.458 8	.099 1.031 3	.371 3.865 12	.366 3.813 9	.190 1.979 7
.332	.253	.262 1.036 3	.260 1.028 2	.268 1.059 6	.253 1.000 1	.268 1.059 6	.253 1.000 1	.260 1.028 2	.265 1.047 5	.260 1.028 2	.264 1.043 4	.268 1.059 6	.256 1.012 1
.264	.117	.154 1.316 4	.190 1.624 6	.365 3.12 10	.136 1.162 3	.366 3.128 11	.178 1.521 5	.119 1.017 1	.361 3.085 8	.122 1.043 2	.369 3.154 12	.363 3.103 9	.245 2.094 7

Table A.11: Average SVM test errors for the *NS* case and large samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.397	.257	.262 1.019 2	.257 1.000 1	.257 1.000 1	.257 1.000 1	.257 1.000 1	.257 1.000 1	.257 1.000 1	.499 1.942 4	.257 1.000 1	.257 1.000 1	.442 1.720 3	.257 1.000 1
.163	.076	.076 1.000 1	.082 1.079 3	.076 1.000 1	.076 1.000 1	.076 1.000 1	.076 1.000 1	.076 1.000 1	.102 1.342 4	.076 1.000 1	.076 1.000 1	.195 2.566 5	.077 1.013 2
.398	.258	.261 1.012 2	.258 1.000 1	.258 1.000 1	.258 1.000 1	.258 1.000 1	.258 1.000 1	.258 1.000 1	.499 1.934 4	.258 1.000 1	.258 1.000 1	.434 1.682 3	.258 1.000 1
.227	.091	.091 1.000 1	.093 1.022 2	.091 1.000 1	.091 1.000 1	.091 1.000 1	.091 1.000 1	.091 1.000 1	.283 3.11 4	.091 1.000 1	.091 1.000 1	.323 3.545 5	.116 1.275 3

Table A.12: Average SVM test errors for the *NS* case and wide samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.433	.129	.267 2.070 3	.265 2.054 2	.451 3.496 9	.431 3.341 6	.369 2.860 5	.319 2.473 4	.440 3.411 7	.485 3.76 11	.447 3.465 8	.497 3.853 12	.470 3.643 10	.206 1.597 1
.085	.006	.019 3.167 3	.054 9 6	.025 4.167 5	.023 3.833 4	.018 3.000 2	.018 3.000 2	.016 2.667 1	.077 12.83 7	.019 3.167 3	.485 80.83 9	.280 46.67 8	.025 1.013 2
.430	.179	.363 2.028 3	.356 1.989 2	.454 2.536 9	.435 2.430 7	.396 2.212 5	.364 2.034 4	.415 2.318 6	.492 2.749 11	.440 2.458 8	.499 2.788 12	.483 2.698 10	.288 1.609 1
.252	.155	.207 1.335 3	.255 1.645 9	.218 1.406 6	.214 1.381 4	.225 1.452 8	.222 1.432 7	.170 1.097 1	.356 2.297 10	.190 1.226 2	.480 3.097 12	.432 2.787 11	.215 1.387 5

Table A.13: Average KFDA test errors for the *LL* case and small samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.332	.134	.145 1.082 1	.396 2.955 12	.168 1.254 7	.166 1.239 6	.164 1.224 4	.151 1.127 2	.163 1.216 3	.322 2.403 9	.165 1.231 5	.376 2.806 11	.223 1.664 8	.359 2.679 10
.165	.076	.105 1.382 8	.138 1.816 10	.088 1.158 4	.085 1.118 1	.090 1.184 5	.097 1.276 6	.087 1.145 3	.097 1.276 6	.086 1.132 2	.153 2.013 11	.103 1.335 7	.128 1.684 9
.331	.134	.150 1.119 2	.387 2.888 12	.163 1.216 6	.158 1.179 5	.162 1.209 4	.146 1.09 1	.161 1.201 3	.309 2.306 9	.166 1.239 7	.375 2.799 11	.207 1.545 8	.348 2.60 10
.215	.124	.163 1.315 6	.168 1.355 7	.149 1.202 5	.141 1.137 3	.149 1.202 5	.146 1.177 4	.131 1.056 1	.170 1.371 8	.134 1.081 2	.210 1.694 11	.194 1.565 9	.202 1.629 10

Table A.14: Average KFDA test errors for the *LL* case and mixed samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.164	.078	.078 1 1	.176 2.256 10	.117 1.5 6	.105 1.346 4	.092 1.179 8	.094 1.205 3	.078 1 1	.174 2.231 9	.079 1.013 2	.105 1.346 4	.134 1.718 7	.108 1.385 5
.079	.042	.059 1.405 9	.053 1.262 6	.050 1.19 5	.058 1.381 7	.047 1.119 4	.068 1.619 8	.044 1.048 2	.050 1.19 5	.042 1 1	.050 1.19 5	.058 1.381 6	.049 1.167 3
.162	.078	.078 1 1	.174 2.231 10	.117 1.5 7	.105 1.346 5	.097 1.244 4	.089 1.141 3	.078 1 1	.170 2.179 9	.080 1.026 2	.105 1.346 5	.132 1.692 8	.107 1.372 6
.101	.081	.084 .084 1.037	.093 1.148 7	.089 1.099 5	.091 1.123 6	.086 1.062 4	.095 1.173 8	.081 1 1	.096 1.185 9	.081 1 1	.087 1.074 3	.104 1.284 10	.089 1.099 5

Table A.15: Average KFDA test errors for the *LL* case and large samples

FULL	*ORA*	*R0*	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
.215	.121	.121 1.000 1	.367 3.033 5	.121 1.000 1	.121 1.000 1	.121 1.000 1	.121 1.000 1	.121 1.000 1	.317 2.620 4	.121 1.000 1	.499 4.124 6	.186 1.537 2	.213 1.760 3
.075	.033	.040 1.212 3	.057 1.727 5	.033 1.000 1	.033 1.000 1	.033 1.000 1	.033 1.000 1	.033 1.000 1	.039 1.182 2	.033 1.000 1	.033 1.000 1	.043 1.303 4	.043 1.303 4
.212	.121	.121 1 1	.369 3.050 5	.121 1.000 1	.121 1.000 1	.121 1.000 1	.121 1.000 1	.121 1.000 1	.309 2.554 4	.121 1.000 1	.500 4.132 6	.193 1.595 2	.225 1.860 3
.152	.084	.121 1.440 5	.085 1.012 2	.084 1.000 1	.084 1.000 1	.084 1.000 1	.084 1.000 1	.084 1.000 1	.105 1.250 3	.084 1.000 1	.084 1.000 1	.121 1.441 5	.108 1.286 4

Table A.16: Average KFDA test errors for the *LL* case and wide samples

FULL	*ORA*	*R0*	*R1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
.300	.059	.139 2.356 6	.292 4.949 8	.091 1.542 2	.091 1.542 2	.097 1.644 5	.092 1.559 3	.090 1.525 1	.285 4.831 5	.092 1.559 3	.312 5.288 9	.095 1.610 4	.229 3.881 7
.126	.038	.057 1.500 3	.157 4.132 11	.064 1.684 6	.056 1.474 2	.072 1.895 8	.049 1.289 1	.062 1.632 5	.067 1.763 7	.064 1.684 6	.097 2.553 10	.061 1.605 4	.095 2.500 9
.349	.115	.253 2.200 8	.321 2.791 9	.175 1.522 4	.168 1.461 2	.195 1.696 6	.179 1.557 5	.147 1.278 1	.425 3.696 12	.169 1.470 3	.361 3.139 11	.216 1.878 7	.334 2.904 10
.229	.097	.215 2.216 8	.174 1.794 6	.173 1.784 5	.167 1.722 3	.186 1.918 7	.171 1.763 4	.113 1.165 2	.215 2.216 8	.138 1.423 1	.243 2.505 10	.236 2.433 9	.299 3.082 11

Table A.17: Average SVM test errors for the *LL* case and small samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.335	.121	.150 1.24 2	.150 1.24 2	.157 1.298 6	.154 1.273 5	.152 1.256 3	.138 1.140 1	.152 1.256 3	.317 2.620 4	.154 1.273 5	.373 3.083 9	.215 1.777 7	.357 2.950 8
.157	.081	.096 1.185 5	.105 1.296 8	.092 1.136 3	.089 1.099 1	.094 1.160 4	.102 1.259 7	.092 1.136 3	.101 1.247 6	.090 1.111 2	.159 1.963 11	.106 1.309 9	.133 1.642 10
.337	.121	.155 1.281 7	.144 1.19 2	.152 1.256 6	.146 1.207 3	.151 1.248 5	.133 1.099 1	.150 1.240 4	.305 2.521 9	.155 1.281 7	.372 3.074 11	.199 1.645 8	.346 2.86 10
.217	.132	.143 1.083 4	.153 1.159 7	.151 1.144 5	.143 1.083 4	.152 1.152 6	.142 1.076 3	.134 1.015 1	.176 1.333 8	.141 1.068 2	.213 1.614 11	.199 1.508 9	.208 1.576 10

Table A.18: Average SVM test errors for the *LL* case and mixed samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.174	.064	.064 1.000 1	.064 1.000 1	.105 1.641 6	.093 1.453 4	.078 1.219 2	.081 1.266 3	.064 1.000 1	.170 2.656 8	.064 1.000 1	.093 1.453 4	.123 1.922 7	.097 1.516 5
.072	.042	.047 1.190 4	.046 1.095 3	.050 1.190 6	.057 1.357 3	.048 1.143 5	.066 1.571 9	.044 1.048 2	.050 1.190 6	.042 1.000 1	.050 1.190 6	.059 1.405 8	.050 1.190 6
.175	.065	.065 1.000 1	.065 1.000 1	.107 1.646 8	.095 1.462 6	.085 1.308 4	.078 1.200 3	.065 1.000 1	0.166 2.554 10	.067 1.031 2	.094 1.446 5	.124 1.908 9	.098 0.508 7
.100	.069	.069 1.000 1	.075 1.087 3	.078 1.130 4	.077 1.116 6	.075 1.087 3	.079 1.145 5	.069 1.00 1	.082 1.188 7	.070 1.014 2	.075 1.087 3	.095 1.377 8	.075 1.087 3

Table A.19: Average SVM test errors for the *LL* case and large samples

FULL	*ORA*	*N0*	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
.233	.114	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.314 2.754 4	.114 1.000 1	.499 4.377 5	.181 1.588 2	.208 1.825 3
.079	.033	.034 1.030 2	.034 1.030 2	.034 1.030 2	.033 1.000 1	.033 1.000 1	.033 1.000 1	.033 1.000 1	.039 1.182 3	.033 1.000 1	.033 1.000 1	.043 1.303 4	.043 1.303 4
.229	.114	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.114 1.000 1	.306 2.684 4	.114 1.000 1	.500 4.386 5	.187 1.640 2	.220 1.930 3
.145	.084	.090 1.071 2	.091 1.083 3	.084 1.000 1	.084 1.000 1	.084 1.000 1	.084 1.000 1	.084 1.000 1	.109 1.298 4	.084 1.000 1	.084 1.000 1	.125 1.488 6	.113 1.345 5

Table A.20: Average SVM test errors for the *LL* case and wide samples

FULL	*ORA*	*N0*	*N1*	*A0*	*A1*	*A0T*	*A1T*	*VA0*	*VA1*	*G0*	*G1*	*SS0*	*SS1*
.269	.060	.111 1.850 9	.104 1.733 8	.094 1.567 2	.093 1.550 1	.102 1.700 7	.095 1.583 3	.097 1.617 5	.292 4.867 11	.096 1.600 4	.298 4.967 12	.099 1.650 6	.241 4.017 10
.092	.021	.042 2.000 6	.057 2.714 7	.037 1.762 2	.033 1.571 1	.042 2.000 6	.040 1.905 4	.041 1.952 5	.037 1.762 2	.042 2.000 6	.079 3.762 8	.038 1.810 3	.105 5.000 9
.348	.123	.169 1.374 3	.166 1.350 2	.174 1.415 5	.170 1.382 4	.192 1.561 7	.177 1.439 6	.150 1.220 1	.424 3.447 11	.169 1.374 3	.348 2.829 10	.212 1.724 8	.344 2.797 9
.219	.130	.148	.156 1.200 3	.166 1.277 6	.163 1.254 4	.174 1.338 7	.165 1.269 5	.137 1.054 1	.192 1.477 8	.148 1.138 2	.217 1.669 10	.206 1.585 9	.247 1.900 11

Table A.21: Average KFDA test errors for the *LS* case and small samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.438	.221	.303 1.371 2	.360 1.629 5	.380 1.719 8	.321 1.452 3	.340 1.538 4	.291 1.317 1	.374 1.692 6	.446 2.018 11	.377 .706 7	.479 2.167 12	.411 1.860 9	.412 1.864 10
.266	.129	.211 1.636 7	.182 1.411 4	.185 1.434 6	.174 1.349 2	.183 1.419 5	.168 1.302 1	.182 1.411 4	.232 1.798 8	.181 1.403 3	.339 2.628 11	.247 1.915 10	.234 1.814 9
.437	.223	.298 1.336 2	.371 1.664 5	.378 1.695 7	.316 1.417 3	.344 1.543 4	.294 1.318 1	.376 1.686 6	.435 1.951 10	.376 1.686 6	.481 2.157 11	.410 1.839 9	.402 1.803 8
.308	.157	.260 1.656 8	.205 1.306 3	.233 1.484 7	.207 1.318 4	.226 1.439 6	.198 1.261 2	.180 1.146 1	.297 1.892 10	.209 1.331 5	.350 2.229 12	.326 2.076 111	.275 1.752 9

Table A.22: Average KFDA test errors for the *LS* case and mixed samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.250	.250	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1
.194	.096	.165 1.719 3	.114 1.188 2	.250 2.604 8	.152 1.583 5	.250 2.604 8	.186 1.938 6	.111 1.156 1	.245 2.552 7	.115 1.198 4	.250 2.604 8	.250 2.604 8	.250 2.604 8
.250	.250	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1	.250 1.000 1
.242	.130	.241 1.854 4	.140 1.077 2	.250 1.923 5	.241 1.854 4	.250 1.923 5	.250 1.923 5	.139 1.069 1	.250 1.923 5	.162 1.246 3	.250 1.923 5	.250 1.923 5	.250 1.923 5

Table A.23: Average KFDA test errors for the *LS* case and large samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.358	.200	.203 1.015 2	.301 1.505 6	.222 1.110 5	.203 1.015 2	.213 1.065 3	.202 1.010 1	.215 1.075 4	.474 2.370 9	.220 1.100 5	.499 2.495 10	.453 2.265 8	.378 1.890 7
.155	.085	.123 1.447 5	.100 1.176 4	.088 1.035 3	.086 1.012 2	.086 1.012 2	.086 1.012 2	.085 1.000 1	.201 2.365 7	.086 1.012 2	.085 1.000 1	.213 2.506 8	.146 1.718 6
.361	.202	.205 1.015 4	.300 1.485 7	.237 1.173 8	.208 .990 1	.218 1.079 5	.204 1.010 2	.219 1.084 3	.476 2.356 11	.234 1.158 6	.500 2.475 12	.453 2.243 10	.376 1.861 9
.208	.127	.180 1.417 4	.127 1.000 1	.131 1.031 3	.128 1.008 2	.128 1.008 2	.127 1.000 1	.127 1.000 1	.257 2.024 6	.128 1.008 2	.127 1.000 1	.274 2.157 7	.207 1.630 5

Table A.24: Average KFDA test errors for the *LS* case and wide samples

FULL	***ORA***	***R0***	***R1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.449	.120	.371 1.015 2	.304 1.505 6	.358 1.110 5	.328 1.015 2	.334 1.065 3	.291 1.010 1	.354 1.075	.421 2.370 9	.356 1.100 5	.471 2.495 10	.382 2.265 8	.384 1.890 7
.282	.054	.194 3.593 8	.146 2.704 6	.135 2.500 4	.130 2.407 3	.141 2.611 5	.146 2.704 6	.126 2.333 1	.164 3.037 7	.129 2.389 2	.402 7.444 11	.248 4.593 10	.224 4.148 9
.455	.143	.419 2.930 8	.324 2.266 1	.398 2.783 6	.370 2.578 3	.385 2.692 7	.347 2.427 2	.371 2.594 4	.456 3.189 11	.395 2.762 5	.475 3.322 12	.436 3.049 10	.423 2.958 9
.342	.079	.298 3.772 7	.148 1.873 1	.268 3.392 5	.262 3.316 4	.283 10.51 12	.279 3.532 6	.175 2.215 2	.370 4.684 9	.231 2.924 3	.432 5.468 11	.413 5.228 10	.334 4.228 8

Table A.25: Average SVM test errors for the *LS* case and small samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.438	.212	.277 1.307 2	.266 1.255 1	.377 1.778 8	.317 1.495 4	.337 1.590 5	.288 1.358 3	.371 1.750 6	.446 2.104 11	.375 1.769 7	.478 2.255 12	.412 1.943 10	.408 1.925 9
.277	.129	.173 1.341 2	.175 1.357 3	.187 1.450 7	.177 1.372 4	.184 1.426 6	.170 1.318 1	.184 1.426 6	.241 1.868 9	.183 1.419 5	.346 2.682 11	.254 1.969 10	.233 1.806 8
.438	.212	.287 1.354 2	.271 1.278 1	.375 1.769 8	.313 1.476 4	.340 1.604 5	.290 1.368 3	.372 1.755 6	.435 2.052 11	.373 1.759 7	.480 2.264 12	.408 1.925 10	.399 1.882 9
.315	.164	.230 1.402 5	.239 1.457 7	.245 1.494 8	.223 1.360 3	.238 1.451 6	.213 1.299 2	.196 1.195 1	.305 1.860 10	.224 1.366 4	.356 2.171 12	.333 2.030 11	.283 1.726 9

Table A.26: Average SVM test errors for the *LS* case and mixed samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.336	.223	.243 1.090 4	.241 1.081 3	.268 1.202 10	.241 1.081 2	.268 1.202 10	.246 1.103 1	.246 1.103 1	.261 1.170 7	.248 1.112 5	.262 1.175 8	.266 1.193 9	.256 1.148 6
.197	.090	.107 1.189 3	.126 1.400 4	.337 3.744 10	.140 1.556 5	.337 3.744 10	.175 1.944 6	.101 1.122 1	.246 2.733 8	.105 1.167 2	.348 3.867 11	.325 3.611 9	.233 2.589 7
.335	.225	.246 1.093 2	.248 1.102 4	.268 1.191 10	.241 1.071 1	.269 1.196 11	.247 1.098 3	.251 1.116 5	.260 1.156 7	.251 1.116 5	.263 1.169 8	.267 1.187 9	.256 1.138 6
.252	.097	.138 1.423 3	.170 1.753 4	.328 3.381 10	.219 2.258 5	.292 3.010 9	.257 2.649 6	.106 1.093 1	.291 3.000 8	.129 1.330 2	.345 3.557 12	.338 3.485 11	.282 2.907 7

Table A.27: Average SVM test errors for the *LS* case and large samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.362	.182	.184 1.011 1	.185 1.016 2	.206 1.132 6	.185 1.016 2	.196 1.077 3	.184 1.011 1	.198 1.088 4	.473 2.599 9	.204 1.121 5	.499 2.742 10	.450 2.473 8	.371 2.038 7
.164	.076	.076 1.000 1	.076 1.000 1	.078 1.026 3	.077 1.013 2	.077 1.013 2	.077 1.013 2	.076 1.000 1	.190 2.500 5	.077 1.013 2	.076 1.000 1	.204 2.684 6	.136 1.789 4
.362	.181	.185 1.022 5	.184 1.017 4	.219 1.210 8	.188 1.039 6	.198 1.094 2	.183 1.011 1	.199 1.099 3	.475 2.624 11	.216 1.193 7	.499 2.757 12	.450 2.486 10	.368 2.033 9
.226	.093	.100 1.075 5	.111 1.194 6	.098 1.054 4	.094 1.011 2	.095 1.022 3	.093 1.000 1	.093 1.000 1	.252 2.710 8	.094 1.011 2	.093 1.000 1	.284 3.054 9	.196 2.108 7

Table A.28: Average SVM test errors for the *LS* case and wide samples

FULL	***ORA***	***N0***	***N1***	***A0***	***A1***	***A0T***	***A1T***	***VA0***	***VA1***	***G0***	***G1***	***SS0***	***SS1***
.447	.136	.265 1.949 2	.263 1.934 1	.365 2.684 8	.338 2.485 4	.339 2.493 5	.300 2.206 3	.363 2.669 9	.425 3.125 11	.361 2.654 6	.472 3.471 12	.396 2.912 10	.364 2.676 7
.261	.063	.133 2.111 1	.165 2.619 8	.153 2.429 5	.146 2.317 2	.158 2.508 7	.162 2.571 6	.147 2.333 3	.172 2.730 9	.149 2.365 4	.416 6.603 11	.256 4.063 10	.256 4.063 10
.450	.174	.333 1.914 2	.328 1.885 1	.393 2.259 8	.368 2.115 5	.378 2.172 6	.345 1.983 3	.367 2.109 4	.459 2.638 11	.391 2.247 7	.476 2.736 12	.438 2.517 10	.405 2.328 9
.334	.157	.283 1.803 5	.295 1.879 8	.282 1.796 4	.277 1.764 3	.293 1.866 7	.287 1.828 6	.220 1.401 1	.371 2.363 10	.254 1.618 2	.431 2.745 12	.413 2.631 11	.338 2.153 9

APPENDIX B

SOME MATHEMATICAL RESULTS

In this appendix, definitions and results required in the introduction to SVMs in Section 5.2 are given. We start with the definition of a hyperplane in $\Re^p$, followed by a brief description of the way in which a vector lying in the hyperplane may be obtained. A result for the distance from an arbitrary point to a hyperplane, and thus also for its functional and geometric margins, is given. We then provide the definition of a hyperplane in canonical form.

Consider the following definition of a hyperplane in $\Re^p$.

DEFINITION B.1: ***A HYPERPLANE IN*** $\Re^p$

A hyperplane in $\Re^p$ is defined as the affine set $L(\boldsymbol{w}, b)$ of all p-vectors $\boldsymbol{x}$ satisfying

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0, \tag{B.1}$$

where $\boldsymbol{w} \in \Re^p$ is a given vector and b is a given scalar. Hence

$$L(\boldsymbol{w}, b) = \{\boldsymbol{x} \in \Re^p : \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\}. \tag{B.2}$$

Note that for the case $b = 0$ in equations (B.1) and (B.2), $L(\boldsymbol{w}, 0) = \{\boldsymbol{x} \in \Re^p : \langle \boldsymbol{w}, \boldsymbol{x} \rangle = 0\}$ consists of all vectors in $\Re^p$ which are orthogonal to $\boldsymbol{w}$. This will be a $(p-1)$-

dimensional subspace in $\Re^p$. Let this subspace be denoted by W. Starting with $\boldsymbol{w}$, and performing Gram-Schmidt orthogonalisation, an orthogonal basis (say $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{p-1}$) can be found for W.

Now consider the case where $b \neq 0$. Let $\boldsymbol{x}$ be a typical vector in W, *i.e.* $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = 0$. Also, let $\boldsymbol{x}_b$ be a vector such that $\langle \boldsymbol{w}, \boldsymbol{x}_b \rangle = -b$. Put $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{x}_b$, then $\langle \boldsymbol{w}, \boldsymbol{z} \rangle = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \langle \boldsymbol{w}, \boldsymbol{x}_b \rangle = -b$, *i.e.* $\langle \boldsymbol{w}, \boldsymbol{z} \rangle + b = 0$, and we can conclude that any vector $\boldsymbol{z}$ of the form $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{x}_b$, $\boldsymbol{x} \in W$, $\langle \boldsymbol{w}, \boldsymbol{x}_b \rangle = -b$, will belong to the hyperplane $L(\boldsymbol{w}, b)$. Moreover, we can assume without loss of generality that $\boldsymbol{x}_b \perp W$, since $\langle \boldsymbol{w}, \boldsymbol{x}_b \rangle = \left\langle \boldsymbol{w}, \left[P_W \boldsymbol{x}_b + P_{W^\perp} \boldsymbol{x}_b \right] \right\rangle = \left\langle \boldsymbol{w}, P_{W^\perp} \boldsymbol{x}_b \right\rangle$.

In general therefore consider a vector $\boldsymbol{z} \in L(\boldsymbol{w}, b)$, *i.e.* $\langle \boldsymbol{w}, \boldsymbol{z} \rangle + b = 0$. Any such vector can be obtained in two steps: First, find a vector $\boldsymbol{x} \in W$, where W is the subspace with $\boldsymbol{w} \perp W$. Second, find a vector $\boldsymbol{x}_b \in W^\perp$ such that $\langle \boldsymbol{w}, \boldsymbol{x}_b \rangle = -b$. Now put $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{x}_b$. Note that we can put $\boldsymbol{x} = P_W \boldsymbol{z}$.

DEFINITION B.2: *PROJECTION OF AN ARBITRARY POINT ONTO A HYPERPLANE IN* $\Re^p$

The projection of an arbitrary point $\boldsymbol{q} \in \Re^p$ onto a hyperplane L is defined as

$$P_L \boldsymbol{q} = P_W \boldsymbol{q} + \boldsymbol{x}_b . \tag{B.3}$$

Note that $P_L \boldsymbol{q} \in L(\boldsymbol{w}, b)$, since $\langle \boldsymbol{w}, P_L \boldsymbol{q} \rangle = -b$. Also, $\min_{z \in L}(\|\boldsymbol{q} - \boldsymbol{z}\|) = \|\boldsymbol{q} - P_L \boldsymbol{q}\|$. This follows since $\|\boldsymbol{q} - \boldsymbol{z}\| = \|\boldsymbol{q} - \boldsymbol{x} - \boldsymbol{x}_b\| = \|(\boldsymbol{q} - \boldsymbol{x}_b) - \boldsymbol{x}\|$ is a minimum over $\boldsymbol{x} \in W$ if $\boldsymbol{x} = P_W(\boldsymbol{q} - \boldsymbol{x}_b) = P_W \boldsymbol{x}$. The $\boldsymbol{z} \in L(\boldsymbol{w}, b)$ minimising $\|\boldsymbol{q} - \boldsymbol{z}\|$ is therefore given by $\boldsymbol{z} = P_W \boldsymbol{x} + \boldsymbol{x}_b = P_L \boldsymbol{x}$. Note therefore that $\min_{\boldsymbol{q} \in L}(\|\boldsymbol{q} - \boldsymbol{z}\|) = \|\boldsymbol{x} - P_L \boldsymbol{x}\| = \left\| P_{W^\perp} \boldsymbol{x} - \boldsymbol{x}_b \right\|$ (since $\boldsymbol{x}_b \in W^\perp$).

We are now in a position to define the distance between an arbitrary vector $\boldsymbol{q} \in \Re^p$ and the hyperplane $L(\boldsymbol{w}, b)$ (*cf.* also Hastie *et al.*, 2001, *p.* 106).

DEFINITION B.3: ***DISTANCE FROM AN ARBITRARY POINT TO A HYPERPLANE IN*** $\Re^p$

The distance between an arbitrary point $\boldsymbol{q} \in \Re^p$ and a hyperplane L is

$$\min_{\boldsymbol{q} \in L}(\|\boldsymbol{q} - \boldsymbol{z}\|) = \left\| P_{W^\perp} \boldsymbol{x} - \boldsymbol{x}_b \right\|. \tag{B.4}$$

Note that $W^\perp$ is a one-dimensional space, spanned for example by $\boldsymbol{w}$. Hence $P_{W^\perp} \boldsymbol{x} = \left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle / \|\boldsymbol{w}\|^2 \right) \boldsymbol{w} = a\boldsymbol{w}$, where of course $a = \langle \boldsymbol{w}, \boldsymbol{x} \rangle / \|\boldsymbol{w}\|^2$.

We now find that

$$\begin{aligned} \left\| P_{W^\perp} \boldsymbol{q} - \boldsymbol{x}_b \right\|^2 &= \langle a\boldsymbol{w} - \boldsymbol{x}_b, a\boldsymbol{w} - \boldsymbol{x}_b \rangle \\ &= a^2 \|\boldsymbol{w}\|^2 - 2a\boldsymbol{w}'\boldsymbol{x}_b + \|\boldsymbol{x}_b\|^2 \end{aligned}$$

$$= \frac{\langle \boldsymbol{q}, \boldsymbol{w} \rangle^2}{\|\boldsymbol{w}\|^2} + \frac{2\langle \boldsymbol{q}, \boldsymbol{w} \rangle b}{\|\boldsymbol{w}\|^2} + \|\boldsymbol{x}_b\|^2 . \tag{B.5}$$

Since $\boldsymbol{x}_b \in W^{\perp}$, we also have

$$\boldsymbol{x}_b = P_{W^{\perp}} \boldsymbol{x}_b = \frac{\langle x_b, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|^2} \boldsymbol{w} = \frac{-b}{\|\boldsymbol{w}\|^2} \boldsymbol{w} \tag{B.6}$$

and

$$\|\boldsymbol{x}_b\|^2 = \frac{b^2}{\|\boldsymbol{w}\|^4} \|\boldsymbol{w}\|^2 = \frac{b^2}{\|\boldsymbol{w}\|^2} . \tag{B.7}$$

Therefore $\left\| P_{W^{\perp}} \boldsymbol{q} - \boldsymbol{x}_b \right\|^2 = \frac{1}{\|\boldsymbol{w}\|^2} \left(\langle \boldsymbol{q}, \boldsymbol{w} \rangle + b \right)^2$, and the distance from $\boldsymbol{q}$ to L simplifies to

$$\delta(\boldsymbol{q}, L) = \left\| P_{W^{\perp}} \boldsymbol{q} - \boldsymbol{x}_b \right\| = \frac{1}{\|\boldsymbol{w}\|} \left(\langle \boldsymbol{q}, \boldsymbol{w} \rangle + b \right) \tag{B.8}$$

Note that in the literature $\delta(\boldsymbol{x}_i, L_{\aleph})$ is frequently referred to as the *geometric margin of the* i^{th} *input pattern with respect to* $L_{\aleph}$. Note also that the minimum value of (B.8) over all training patterns $\boldsymbol{x}_i$, $i = 1,2,\ldots,n$, is referred to as the geometric margin of the hyperplane.

DEFINITION B.4: ***THE GEOMETRIC MARGIN OF A HYPERPLANE IN*** $\Re^p$ ***WITH RESPECT TO AN ARBITRARY POINT***

The geometric margin of a data case $(\boldsymbol{x}_i, y_i)$ with respect to a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is

$$\delta_i = \frac{1}{\|\boldsymbol{w}\|} \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right). \tag{B.9}$$

DEFINITION B.5: ***THE GEOMETRIC MARGIN OF A HYPERPLANE IN $\Re^p$ WITH RESPECT TO A TRAINING SET***

The geometric margin of a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ with respect to a training set is

$$\delta = min\left\{\frac{1}{\|\boldsymbol{w}\|}\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b\right), i = 1,2,\ldots,n\right\}. \tag{B.10}$$

DEFINITION B.6: ***THE FUNCTIONAL MARGIN OF A HYPERPLANE IN $\Re^p$ WITH RESPECT TO AN ARBITRARY POINT***

The (functional) margin of a a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ with respect to a data case $(\boldsymbol{x}_i, y_i)$ is defined to be

$$\eta_i = y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b\right). \tag{B.11}$$

In a two-group classification setup, note that $\eta_i > 0$ in (B.11) implies correct classification of the case $(\boldsymbol{x}_i, y_i)$. From (B.9), the relation between the (functional) margin of an arbitrary data case $(\boldsymbol{x}_i, y_i)$ with respect to $L_{\aleph}$ and the distance from input $\boldsymbol{x}_i$ to $L_{\aleph}$ is clear:

$$\delta(\boldsymbol{x}_i, L_{\aleph}) = \frac{1}{\|\boldsymbol{w}\|}\left(\frac{\eta_i}{y_i}\right). \tag{B.12}$$

Of course in a binary classification setup, assuming correct classification of the case $(\boldsymbol{x}_i, y_i)$, (B.12) simplifies to

$$\delta(\boldsymbol{x}_i, L_{\aleph}) = \frac{\eta_i}{\|\boldsymbol{w}\|}. \tag{B.13}$$

DEFINITION B.7: ***THE FUNCTIONAL MARGIN OF A HYPERPLANE IN*** $\Re^p$

The functional margin of a hyperplane $L_{\aleph}(\boldsymbol{w}, b)$ is defined as the minimum of its functional margin with respect to each point in the training data set, *viz*.

$$\eta = min\{y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b), i = 1,2,\ldots,n\} \tag{B.14}$$

DEFINITION B.8: ***A SEPARATING HYPERPLANE IN*** $\Re^p$

In a two-group classification problem, a separating hyperplane in $\Re^p$ is the hyperplane in input space such that all input patterns belonging to group 1 lie on the opposite side of the hyperplane to those belonging to group 2. Therefore, a separating hyperplane in $\Re^p$ is such that

$$\eta_i > 0\ ,\ i = 1,2,\ldots,n \tag{B.15}$$

Note that we may additionally require the set of separating hyperplanes in $\Re^p$ to have $\eta_i \geq 1,\ i = 1,2,\ldots,n$, with equality occurring at at least one $\boldsymbol{x}_i$. The set of hyperplanes satisfying this additional criterion is the same as the set of hyperplanes which satisfy

(B.15), but the additional requirement normalises the latter set of hyperplanes to each have only a single representation. The single representation induced by the requirement $\eta_i \geq 1$, $i = 1,2,\ldots,n$, is called the *canonical form of a separating hyperplane in* $\Re^p$.

DEFINITION B.9: ***THE CANONICAL FORM OF A SEPARATING HYPERPLANE IN*** $\Re^p$

The canonical form of a separating hyperplane in $\Re^p$ is defined as the affine set $L(\boldsymbol{w}, b)$ of all p-vectors $\boldsymbol{x}$ safisfying

$$yf(\boldsymbol{x}) = y\{\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b\} \geq 1. \tag{B.16}$$

APPENDIX C

EXAMPLES OF SIMULTATION PROGRAMS

In this appendix, an example of each of the programs used in the Monte Carlo simulation studies in Chapters 3 to 6 of the thesis, is given. All simulation programs were written in Fortran. Sections 1 and 2 contain the code for the simulation program used to generate the results reported in Chapter 3. An example of the programs used in Chapter 4 is given in Section 3, and Sections 4 and 5 pertain to the Monte Carlo simulation studies reported in Chapters 5 and 6 respectively. Subroutines and functions required in the main simulation programs appear in Section 6, in the order in which they occur in the programs in Sections 1 to 5, alphabetical per main simulation program. Note that the required Fortran subroutines and functions (in the Fortran IMSL library) are only listed in the programs where they are used. More details regarding these subroutines and functions can be found in the IMSL reference.

C.1 *NAÏVE SELECTION IN INPUT AND FEATURE SPACE*

An example of the Fortran code used to obtain the results in Section 2 of Chapter 3 (in Example 3.1) is given below. Note that Subroutines 1 to 7 will also be used in most of the simulation programs to follow.

```
C     IN THIS PROGRAM WE CALCULATE AVERAGE TEST ERRORS (AND STANDARD ERRORS)
C     PERTAINING TO THE FULL SVM, KFDA, AND LDA MODELS, AND POST-SELECTION
C     AVERAGE TEST ERRORS (AND STANDARD ERRORS) FOR SVM, KFDA, AND LDA MODELS
      AFTER USING
C        C.1.1  CORRELATIONS AND
C        C.1.2  ALIGNMENTS
C     AS SELECTION CRITERIA
C     THE DATA ARE GENERATED FROM A MULTIVARIATE NORMAL DISTRIBUTION AND
      THE TWO GROUPS DIFFER WITH RESPECT TO LOCATION
```

```
C    SETS OF RELEVANT AND IRRELEVANT INPUT VARIABLES ARE UNCORRELATED
C    THE FOLLOWING (OWN) FUNCTION IS REQUIRED:
C        ALIGNMENT

C    THE FOLLOWING (OWN) SUBROUTINES ARE REQUIRED:
C        1. BERFOUTKFDA
C        2. BERFOUTLDA
C        3. BERFOUTSVM
C        4. DOENKFDA
C        5. DOENSVM
C        6. GEMVARV
C        7. GRAMMAT
C        8. GRAMNUUT

C    THE FOLLOWING IMSL FUNCTIONS ARE REQUIRED:
C        1. DABS
C        2. DMACH
C        3. DSQRT

C    THE FOLLOWING IMSL SUBROUTINES ARE REQUIRE :
C        1. DCHFAC
C        2. DLINDS
C        3. DLSASF
C        4. DRNMVN
C        5. DSVRGP
C        6. DQPROG

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=1,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NMC=1000,CORR=0.0D0,SIGNOISE=20.0D0)
      PARAMETER (GAMPAR=1.0D0/IP)
      PARAMETER (NT=1000,MT=1000,NMT=NT+MT)

C    POPULATION PARAMETERS
      DIMENSION AMU1(IP),AMU2(IP)
      DIMENSION SIGMAM(IP,IP),RSIG(IP,IP)
```

```
C   X MATRICES AND Y VECTORS
    DIMENSION XM(NNPMM,IP),XMRF(IP,NNPMM)
    DIMENSION XT(NMT,IP)
    DIMENSION GEM(IP),VAR(IP)
    DIMENSION YV(NNPMM),YVT(NMT)

C   KFDA RELATED QUANTITIES
    DIMENSION EENM(NNPMM),EENP(NNPMM)

C   SVM RELATED QUANTITIES
    DIMENSION GRMATVOL(NNPMM,NNPMM),GRNUUTVOL(NMT,NNPMM)
    DIMENSION GRMATREG(NNPMM,NNPMM),GRNUUTREG(NMT,NNPMM)
    DIMENSION GRMATCOR(NNPMM,NNPMM),GRNUUTCOR(NMT,NNPMM)
    DIMENSION GRMATAL(NNPMM,NNPMM),GRNUUTAL(NMT,NNPMM)
    DIMENSION ALPHA(NNPMM),ALPHAW(NNPMM),AL(NNPMM)

C   VARIOUS OTHER QUANTITIES
    DIMENSION AKOR(IP),ALIGN(IP)
    DIMENSION IPERM(IP)
    DIMENSION INDVEK(IP),INDVEKVOL(IP),JIND(IP)
    DIMENSION VEKKOR(IP),VEKAL(IP)
    DIMENSION KIESKOR(IP),KIESAL(IP),KIESREG(IP)
    DIMENSION FOUTMAT(NMC,4,3),FOUTGEMMAT(4,3),FOUTSTDMAT(4,3)

C   OUTPUT FILES
    CHARACTER*70 FILEOUT1,FILEOUT2
    FILEOUT1='foutNL9.d'
    FILEOUT2='kiesNL9.d'

C   WRITE FILE HEADERS
    OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
      WRITE(1,*) 'NORMAL DISTRIBUTION GROUP DIFFERENCES IN LOCATION'
      WRITE(1,*) 'RELEVANT AND IRRELEVANT SUBSETS OF VARIABLES ARE CORRELATED'
      WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
      WRITE(1,*) 'KERNEL HYPERPARAMETER=',GAMPAR
      WRITE(1,*) 'N1=',NN,'N2=',MM,'NO. OF MONTE CARLO REPETITIONS=',NMC
      WRITE(1,*) 'NT1=',NT,'NT2=',MT
```

```
      WRITE(1,*) 'SIGNOISE=',SIGNOISE,'SIG12=0.9'
      WRITE(1,600)
    CLOSE(1)

    OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
      WRITE(1,*) 'NORMAL DISTRIBUTION DIFF IN LOCATION'
      WRITE(1,*) 'CORRELATION BETWEEN REL AND IRREL'
      WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
      WRITE(1,*) 'KERNEL HYPERPARAMETER=',GAMPAR
      WRITE(1,*) 'N1=',NN,'N2=',MM,'NO. OF MONTE CARLO REPETITIONS=',NMC
      WRITE(1,*) 'NT1=',NT,'NT2=',MT
      WRITE(1,*) 'SIGNOISE=',SIGNOISE,'SIG12=0.9'
      WRITE(1,600)
    CLOSE(1)

C   SET UP THE INDICES TO THE SET OF THE SEPARATING INPUT VARIABLES
    DO J=1,IP
      KIESREG(J)=J
    END DO

C   ASSIGN VALUES TO THE POPULATION PARAMETERS AND OBTAIN THE COVARIANCE
C   MATRICES REQUIRED TO GENERATE THE TRAINING AND TEST DATA SETS
    DO 3 I=1,IPTEL
      AMU1(I)=0.0D0
      AMU2(I)=1.0D0
      DO 2 J=1,IPTEL
          SIGMAM(I,J)=CORR
2     CONTINUE
      SIGMAM(I,I)=1.0D0
      DO J=IPTEL+1,IP
          SIGMAM(I,J)=0.9D0
      END DO
3   CONTINUE
    DO 5 I=IPTEL+1,IP
      AMU1(I)=0.0D0
      AMU2(I)=0.0D0
      DO J=1,IPTEL
```

```
                SIGMAM(I,J)=SIGMAM(J,I)
            END DO
            DO 4 J=IPTEL+1,IP
                SIGMAM(I,J)=0.0D0
4           CONTINUE
            SIGMAM(I,I)=SIGNOISE
5       CONTINUE

        TOL=1.0D2*DMACH(4)
        CALL DCHFAC(IP,SIGMAM,IP,TOL,IRANK,RSIG,IP)

C       INITIALISE VARIOUS QUANTITIES
        DO 9 I=1,NN
            YV(I)=-1.0D0
9       CONTINUE
        DO 10 I=NN+1,NNPMM
            YV(I)=1.0D0
10      CONTINUE
        SOMY=1.0D0*(MM-NN)
        SOMY2=NNPMM
        DO 11 I=1,NT
            YVT(I)=-1.0D0
11      CONTINUE
        DO 12 I=NT+1,NMT
            YVT(I)=1.0D0
12      CONTINUE

C       SET UP THE KFDA VECTORS
        DO 13 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
13      CONTINUE
```

```
C   THE FOLLOWING STEPS ARE ITERATED OVER FIVE SPECIFICATIONS FOR THE
C   COST PARAMETER
      DO 1010 ICP=1,5
        IF (ICP.EQ.1) CPAR=0.1D0
        IF (ICP.EQ.2) CPAR=1.0D0
        IF (ICP.EQ.3) CPAR=10.0D0
        IF (ICP.EQ.4) CPAR=100.0D0
        IF (ICP.EQ.5) CPAR=1000.0D0

C   INITIALISE THE TEST ERROR AND VARIABLE INDEX VECTORS
        DO 15 I=1,4
            DO 14 J=1,3
                FOUTGEMMAT(I,J)=0.0D0
                FOUTSTDMAT(I,J)=0.0D0
14          CONTINUE
15      CONTINUE
        DO 16 J=1,IP
            INDVEKVOL(J)=J
            VEKKOR(J)=0.0D0
            VEKAL(J)=0.0D0
16      CONTINUE

C   THE START OF THE MONTE CARLO SIMULATION LOOP
        DO 1004 MC=1,NMC

C   GENERATE THE TRAINING DATA
            CALL DRNMVN(NNPMM,IP,RSIG,IP,XM,NNPMM)
            DO 18 I=1,NN
                DO 17 J=1,IPTEL
                    XM(I,J)=XM(I,J)+AMU1(J)
17              CONTINUE
18          CONTINUE
            DO 20 I=1,NN
                DO 19 J=1,NNOISE
                    XM(I,J+IPTEL)=XM(I,J+IPTEL)+AMU1(J+IPTEL)
19              CONTINUE
20          CONTINUE
```

```
      DO 22 I=1,MM
            DO 21 J=1,IPTEL
                  XM(NN+I,J)=XM(NN+I,J)+AMU2(J)
21          CONTINUE
22    CONTINUE
      DO 24 I=1,MM
            DO 23 J=1,NNOISE
                  XM(NN+I,J+IPTEL)=XM(NN+I,J+IPTEL)+AMU2(J+IPTEL)
23          CONTINUE
24    CONTINUE

C     STANDARDISE THE TRAINING DATA
      DO 28 J=1,IP
            S=0.0D0
            S2=0.0D0
            DO 26 I=1,NNPMM
                  S=S+XM(I,J)
                  S2=S2+XM(I,J)**2.0D0
26          CONTINUE
            GEM(J)=S/NNPMM
            VAR(J)=(S2-(S**2.0D0)/NNPMM)/NNPMM
            DO 27 I=1,NNPMM
                  XM(I,J)=(XM(I,J)-GEM(J))/DSQRT(VAR(J))
27          CONTINUE
28    CONTINUE

C     GENERATE THE TEST DATA
      CALL DRNMVN(NMT,IP,RSIG,IP,XT,NMT)
      DO 36 I=1,NT
            DO 35 J=1,IPTEL
                  XT(I,J)=XT(I,J)+AMU1(J)
35          CONTINUE
36    CONTINUE
      DO 38 I=1,NT
            DO 37 J=1,NNOISE
                  XT(I,J+IPTEL)=XT(I,J+IPTEL)+AMU1(J+IPTEL)
37          CONTINUE
```

```
38          CONTINUE
            DO 40 I=1,MT
                  DO 39 J=1,IPTEL
                        XT(NT+I,J)=XT(NT+I,J)+AMU2(J)
39                CONTINUE
40          CONTINUE
            DO 42 I=1,MT
                  DO 41 J=1,NNOISE
                        XT(NT+I,J+IPTEL)=XT(NT+I,J+IPTEL)+AMU2(J+IPTEL)
41                CONTINUE
42          CONTINUE

C     STANDARDISE THE TEST DATA
            DO 45 J=1,IP
                  DO 44 I=1,NMT
                        XT(I,J)=(XT(I,J)-GEM(J))/(DSQRT(VAR(J)))
44                CONTINUE
45          CONTINUE

C     CALCULATE (AND SORT) ONE VARIABLE AT-A-TIME CORRELATIONS
            NVER=IPTEL
            DO 240 J=1,IP
                  S=0.0D0
                  DO 239 I=1,NNPMM
                        S=S+XM(I,J)*YV(I)
239               CONTINUE
                  AKOR(J)=S/(DSQRT(1.0D0*NNPMM*(SOMY2-SOMY*SOMY/NNPMM)))
                  AKOR(J)=DABS(AKOR(J))
240         CONTINUE
            DO 241 J=1,IP
                  IPERM(J)=J
241         CONTINUE
            CALL DSVRGP(IP,AKOR,AKOR,IPERM)
            DO 242 J=1,NVER
                  KIESKOR(J)=IPERM(IP-NVER+J)
242         CONTINUE
```

```
C     CALCULATE (AND SORT) ONE VARIABLE AT-A-TIME ALIGNMENTS
                DO 243 J=1,IP
                      ALIGN(J)=ALIGNMENT(J,XM)
243             CONTINUE
                DO 244 J=1,IP
                      IPERM(J)=J
244             CONTINUE
                CALL DSVRGP(IP,ALIGN,ALIGN,IPERM)
                DO 245 J=1,NVER
                      KIESAL(J)=IPERM(IP-NVER+J)
245             CONTINUE

C     UPDATE SELECTION PROPORTIONS
                DO 206 J=1,IPTEL
                      VEKKOR(KIESKOR(J))=VEKKOR(KIESKOR(J))+1.0D0
                      VEKAL(KIESAL(J))=VEKAL(KIESAL(J))+1.0D0
206             CONTINUE

C     CALCULATE THE TEST ERROR PERTAINING TO THE SVM BASED ON ALL AVAILABLE
C     INPUT VARIABLES
                NVER=IP
                CALL GRAMMAT(GAMPAR,XM,NVER,INDVEKVOL,GRMATVOL)
                CALL DOENSVM(YV,XM,GRMATVOL,CPAR,GAMPAR,NVER,INDVEKVOL,AL,BOPT)
                CALL GRAMNUUT(GAMPAR,XM,XT,NVER,INDVEKVOL,GRNUUTVOL)
                CALL BERFOUTSVM(YV,GRNUUTVOL,YVT,AL,BOPT,FOUT)
                FOUTMAT(MC,1,1)=FOUT

C     CALCULATE THE TEST ERROR PERTAINING TO THE SVM BASED ONLY ON THE
C     SEPARATING INPUT VARIABLES
                NVER=IPTEL
                CALL GRAMMAT(GAMPAR,XM,NVER,KIESREG,GRMATREG)
                CALL DOENSVM(YV,XM,GRMATREG,CPAR,GAMPAR,NVER,KIESREG,AL,BOPT)
                CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESREG,GRNUUTREG)
                CALL BERFOUTSVM(YV,GRNUUTREG,YVT,AL,BOPT,FOUT)
                FOUTMAT(MC,2,1)=FOUT
```

```
C   CALCULATE THE POST-SELECTION SVM TEST ERROR WHEN THE SELECTION CRITE-
C   RION IS PEARSON'S CORRELATION COEFFICIENT IN INPUT SPACE
        CALL GRAMMAT(GAMPAR,XM,NVER,KIESKOR,GRMATCOR)
        CALL DOENSVM(YV,XM,GRMATCOR,CPAR,GAMPAR,NVER,KIESKOR,AL,BOPT)
        CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESKOR,GRNUUTCOR)
        CALL BERFOUTSVM(YV,GRNUUTCOR,YVT,AL,BOPT,FOUT)
        FOUTMAT(MC,3,1)=FOUT

C   CALCULATE THE POST-SELECTION SVM TEST ERROR WHEN THE SELECTION CRITE-
C   RION IS THE ALIGNMENT IN FEATURE SPACE
        CALL GRAMMAT(GAMPAR,XM,NVER,KIESAL,GRMATAL)
        CALL DOENSVM(YV,XM,GRMATAL,CPAR,GAMPAR,NVER,KIESAL,AL,BOPT)
        CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESAL,GRNUUTAL)
        CALL BERFOUTSVM(YV,GRNUUTAL,YVT,AL,BOPT,FOUT)
        FOUTMAT(MC,4,1)=FOUT

C   CALCULATE THE TEST ERROR PERTAINING TO THE KFD BASED ON ALL AVAILABLE
C   INPUT VARIABLES
        CALL DOENKFDA(EENM,EENP,GRMATVOL,CPAR,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUTVOL,YVT,ALPHA,BOPT,FOUT)
        FOUTMAT(MC,1,2)=FOUT

C   CALCULATE THE TEST ERROR PERTAINING TO THE KFD BASED ONLY ON THE
C   SEPARATING INPUT VARIABLES
        CALL DOENKFDA(EENM,EENP,GRMATREG,CPAR,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUTREG,YVT,ALPHA,BOPT,FOUT)
        FOUTMAT(MC,2,2)=FOUT
C    CALCULATE THE POST-SELECTION KFD TEST ERROR WHEN THE SELECTION CRITE-
C    RION IS PEARSON'S CORRELATION COEFFICIENT IN INPUT SPACE
        CALL DOENKFDA(EENM,EENP,GRMATCOR,CPAR,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUTCOR,YVT,ALPHA,BOPT,FOUT)
        FOUTMAT(MC,3,2)=FOUT

C   CALCULATE THE POST-SELECTION SVM TEST ERROR WHEN THE SELECTION CRITE-
C   RION IS THE ALIGNMENT IN FEATURE SPACE
        CALL DOENKFDA(EENM,EENP,GRMATAL,CPAR,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUTAL,YVT,ALPHA,BOPT,FOUT)
```

```
            FOUTMAT(MC,4,2)=FOUT

C   CALCULATE THE TEST ERROR PERTAINING TO THE LD BASED ON ALL AVAILABLE
C   INPUT VARIABLES
            NVER=IP
            CALL BERFOUTLDA(XM,XT,YVT,NVER,INDVEKVOL,FOUT)
            FOUTMAT(MC,1,3)=FOUT

C   CALCULATE THE TEST ERROR PERTAINING TO THE LD BASED ONLY ON THE
C   SEPARATING INPUT VARIABLES
            NVER=IPTEL
            CALL BERFOUTLDA(XM,XT,YVT,NVER,KIESREG,FOUT)
            FOUTMAT(MC,2,3)=FOUT

C   CALCULATE THE POST-SELECTION LD TEST ERROR WHEN THE SELECTION CRITE-
C   RION IS PEARSON'S CORRELATION COEFFICIENT IN INPUT SPACE
            CALL BERFOUTLDA(XM,XT,YVT,NVER,KIESKOR,FOUT)
            FOUTMAT(MC,3,3)=FOUT

C   CALCULATE THE POST-SELECTION LD TEST ERROR WHEN THE SELECTION CRITE-
C   RION IS THE ALIGNMENT IN FEATURE SPACE
            CALL BERFOUTLDA(XM,XT,YVT,NVER,KIESAL,FOUT)
            FOUTMAT(MC,4,3)=FOUT

        1004 CONTINUE
C    THE END OF THE MONTE CARLO SIMULATION LOOP

C   AVERAGE THE OBTAINED TEST ERRORS AND SELECTION FREQUENCIES AND
C   CALCULATE STANDARD ERRORS
        DO 346 K=1,3
            DO 345 J=1,4
                S1=0.0D0
                S2=0.0D0
                DO 344 I=1,NMC
                    S1=S1+FOUTMAT(I,J,K)
                    S2=S2+FOUTMAT(I,J,K)**2.0D0
344             CONTINUE
```

```
                    FOUTGEMMAT(J,K)=S1/NMC
                    FOUTSTDMAT(J,K)=DSQRT((S2-(S1*S1)/NMC)/(NMC*(NMC-1.0D0)))
345             CONTINUE
346     CONTINUE
        DO 349 J=1,IP
                VEKKOR(J)=VEKKOR(J)/NMC
                VEKAL(J)=VEKAL(J)/NMC
349     CONTINUE

C   WRITE THE AVERAGE TEST ERRORS AND STANDARD ERRORS TO A FILE
        OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
                WRITE(1,*) 'SVM C: ',CPAR,'G: ',GAMPAR
                WRITE(1,599) 'FUL','ORA','KOR','AL'
                WRITE(1,600) (FOUTGEMMAT(J,1),J=1,4)
                WRITE(1,600) (FOUTSTDMAT(J,1),J=1,4)
                WRITE(1,*) 'KFDA '
                WRITE(1,600) (FOUTGEMMAT(J,2),J=1,4)
                WRITE(1,600) (FOUTSTDMAT(J,2),J=1,4)
                WRITE(1,*) 'LDA '
                WRITE(1,600) (FOUTGEMMAT(J,3),J=1,4)
                WRITE(1,600) (FOUTSTDMAT(J,3),J=1,4)
                WRITE(1,600)
        CLOSE(1)

C   WRITE THE SELECTION PERCENTAGES TO A FILE
        OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
                WRITE(1,*) 'C: ',CPAR,'G: ',GAMPAR
                WRITE(1,*) 'CORRELATIONS'
                WRITE(1,602) (VEKKOR(J),J=1,IP)
                WRITE(1,*) 'ALIGNMENTS'
                WRITE(1,602) (VEKAL(J),J=1,IP)
        CLOSE(1)
    1010 CONTINUE
C   THE COST PARAMETER LOOP ENDS HERE

C   FILE FORMATS
599 FORMAT(1X,A3,2(4X,A3),5X,A2,3(4X,A3),4X,A4,3(4X,A3))
```

```
600  FORMAT(12(F5.3,2X))
601  FORMAT(20I3)
602  FORMAT(12(F5.2,1X))
605  FORMAT(14X,9(F5.1,2X))

7000 STOP
END
C    THE SIMULATION PROGRAM ENDS HERE
```

C.2 *FEATURE-TO-INPUT SPACE SELECTION*

The following simulation program is an example of the Fortran code used to obtain the results in Section 4.6 of Chapter 3.

```
C    IN THIS PROGRAM WE CALCULATE AVERAGE TEST ERRORS (AND STANDARD ERRORS)
C    PERTAINING TO THE FULL SVM AND KFD MODELS, AND POST-SELECTION AVERAGE
C    TEST ERRORS (AND STANDARD ERRORS) FOR SVM AND KFD MODELS AFTER USING
C        C.2.1  ALIGNMENTS IN FEATURE SPACE
C        C.2.2  CORRELATIONS IN INPUT SPACE
C        C.2.3  LEAST SQUARES APPROXIMATION OF DISCRIMINANT FUNCTION VALUES
C        C.2.4  REGRESSION RANDOM FOREST APPROXIMATION OF DISCRIMINANT
                FUNCTION VALUES
C        C.2.5  THE FOLLOWING PRE-IMAGE APPROXIMATIONS:
                MEANS IN FEATURE SPACE
                VARIATION RATIOS

C    NOTE THAT THE DISCRIMINANT FUNCTION VALUES  ARE OBTAINED AFTER
C    APPLICATION OF AN SVM ON THE FULL SET OF AVAILABLE INPUT VARIABLES
C    THE DATA ARE GENERATED FROM A MULTIVARIATE NORMAL DISTRIBUTION AND
C    THE TWO GROUPS DIFFER WITH RESPECT TO THEIR VARIANCE-COVARIANCE
C     STRUCTURE
C    SETS OF RELEVANT AND IRRELEVANT INPUT VARIABLES ARE CORRELATED

C    THE FOLLOWING (OWN) FUNCTION IS REQUIRED:
C        ALIGNMENT
```

```
C     THE FOLLOWING (OWN) SUBROUTINES ARE REQUIRED:
C          1. BERFOUTSVM
C          2. DOENSVM
C          3. FCN
C          4. GRAMMAT
C          5. GRAMNUUT
C          6. PREIMAGE
C          7. RANFOR
C          8. BUILDTREE
C          9. FINDBESTSPLIT
C          10. QUICKSORT

C     THE FOLLOWING IMSL SUBROUTINES ARE REQUIRED:
C          1. DABS
C          2. DMACH
C          3. DSQRT
C          4. CPSEC

C     THE FOLLOWING IMSL SUBROUTINES ARE REQUIRED:
C          1. DCHFAC
C          2. DRNMVN
C          3. DSVRGP
C          4. DRLSE

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=4,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NMC=1000,CORR=0.7D0,SIGFAKTOR=10.0D0,SIGNOISE=20.0D0)
      PARAMETER (GAMPAR=1.0D0/IP)
      PARAMETER (NT=1000,MT=1000,NMT=NT+MT)

C     POPULATION PARAMETERS
      DIMENSION SIGMAM1(IP,IP),RSIG1(IP,IP)
      DIMENSION SIGMAM2(IP,IP),RSIG2(IP,IP)

C     X MATRICES AND Y VECTORS
      DIMENSION XM(NNPMM,IP),XMRF(IP,NNPMM)
      DIMENSION XM1(NN,IP),XM2(MM,IP)
```

```
      DIMENSION XGEM1(IP),XGEM2(IP),GEM(IP),VAR(IP)
      DIMENSION XT(NMT,IP),XT1(NT,IP),XT2(MT,IP)
      DIMENSION YV(NNPMM),YVT(NMT),IYV(NNPMM)

C     SVM RELATED QUANTITIES
      DIMENSION GRMAT(NNPMM,NNPMM),GRNUUT(NMT,NNPMM)
      DIMENSION ALPHA(NNPMM),ALPHAW(NNPMM),AL(NNPMM)

C     VARIOUS QUANTITIES
      DIMENSION AKOR(IP),ALIGN(IP)
      DIMENSION XVERSKIL(IP),KIESX(IP)
      DIMENSION XPRE1(IP),XPRE2(IP),XPREVERSKIL(IP),KIESF(IP)
      DIMENSION WPRE1(IP),WPRE2(IP),WPREVERSKIL(IP),KIESWD(IP)
      DIMENSION WPRE(IP),KIESW(IP)
      DIMENSION FW(NNPMM),BV(IP),INDVEK(IP)
      DIMENSION Z(IP)
      DIMENSION KIESL(IP),KIESRF(IP),KIESKOR(IP),KIESAL(IP)
      DIMENSION IPERM(IP),INDVEKVOL(IP),JIND(IP),KIESREG(IPTEL)
      DIMENSION FOUTMAT(NMC,11),FOUTGEM(11),FOUTSTD(11)
      DIMENSION VEKX(IP),VEKF(IP),VEKWD(IP),VEKW(IP),VEKL(IP),VEKRF(IP)
      DIMENSION VEKKOR(IP),VEKAL(IP)
      DIMENSION TYD(11)

C     OUTPUT FILES
      CHARACTER*70 FILEOUT1,FILEOUT2
      FILEOUT1='foutNS.d'
      FILEOUT2='kiesNS.d'

C     WRITE FILE HEADERS
      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
        WRITE(1,*) 'NORMAL DISTRIBUTION DIFFERENCES IN SPREAD'
        WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
        WRITE(1,*) 'GAMPAR=',GAMPAR,'CPARS=',CPARS
        WRITE(1,*) 'NN=',NN,'MM=',MM,'NMC=',NMC
        WRITE(1,*) 'NT=',NT,'MT=',MT
        WRITE(1,*) 'SIGFAKTOR=',SIGFAKTOR,'SIGNOISE=',SIGNOISE
        WRITE(1,*) 'SIG12=0.9'
```

```
      WRITE(1,600)
    CLOSE(1)
    OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
      WRITE(1,*) 'NORMAL DISTRIBUTION DIFFERENCES IN SPREAD'
      WRITE(1,*) 'REGRESSION RF AND TREE: USING F-VALUES'
      WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
      WRITE(1,*) 'GAMPAR=',GAMPAR,'CPARS=',CPARS
      WRITE(1,*) 'NN=',NN,'MM=',MM,'NMC=',NMC
      WRITE(1,*) 'NT=',NT,'MT=',MT
      WRITE(1,*) 'SIGFAKTOR=',SIGFAKTOR,'SIGNOISE=',SIGNOISE
      WRITE(1,*) 'SIG12=0.9'
      WRITE(1,600)
    CLOSE(1)

C   ASSIGN VALUES TO THE POPULATION PARAMETERS AND OBTAIN THE COVARIANCE
C   MATRICES REQUIRED TO GENERATE TRAINING AND TEST DATA SETS
    DO 3 I=1,IPTEL
      KIESREG(I)=I
      DO 2 J=1,IPTEL
            SIGMAM1(I,J)=CORR
            SIGMAM2(I,J)=SIGFAKTOR*CORR
2     CONTINUE
      SIGMAM1(I,I)=1.0D0
      SIGMAM2(I,I)=SIGFAKTOR*1.0D0
3   CONTINUE
    DO 5 I=1,IPTEL
      DO 4 J=IPTEL+1,IP
            SIGMAM1(I,J)=0.9D0
            SIGMAM2(I,J)=0.9D0
            SIGMAM1(J,I)=SIGMAM1(I,J)
            SIGMAM2(J,I)=SIGMAM2(I,J)
4     CONTINUE
5   CONTINUE
    DO 7 I=IPTEL+1,IP
      DO 6 J=IPTEL+1,IP
            SIGMAM1(I,J)=0.0D0
            SIGMAM2(I,J)=0.0D0
```

```
6       CONTINUE
        SIGMAM1(I,I)=SIGNOISE*1.0D0
        SIGMAM2(I,I)=SIGNOISE*1.0D0
7   CONTINUE

    TOL=1.0D2*DMACH(4)
    CALL DCHFAC(IP,SIGMAM1,IP,TOL,IRANK,RSIG1,IP)
    CALL DCHFAC(IP,SIGMAM2,IP,TOL,IRANK,RSIG2,IP)

C   INITIALISE VARIOUS QUANTITIES
    DO 9 I=1,NN
        YV(I)=-1.0D0
9   CONTINUE
    DO 10 I=NN+1,NNPMM
        YV(I)=1.0D0
10  CONTINUE
    SOMY=1.0D0*(MM-NN)
    SOMY2=NNPMM
    DO 11 I=1,NT
        YVT(I)=-1.0D0
11  CONTINUE
    DO 12 I=NT+1,NMT
        YVT(I)=1.0D0
12  CONTINUE

C   THE FOLLOWING STEPS ARE ITERATED OVER FIVE SPECIFICATIONS FOR THE COST
    PARAMETER
    DO 1010 ICP=1,5
        IF (ICP.EQ.1) CPARS=0.1D0
        IF (ICP.EQ.2) CPARS=1.0D0
        IF (ICP.EQ.3) CPARS=10.0D0
        IF (ICP.EQ.4) CPARS=100.0D0
        IF (ICP.EQ.5) CPARS=1000.0D0

C   INITIALISE THE TEST ERROR AND VARIABLE INDEX VECTORS
        DO 15 J=1,11
                FOUTGEM(J)=0.0D0
```

```
                FOUTSTD(J)=0.0D0
                TYD(J)=0.0D0
15          CONTINUE
            DO 16 J=1,IP
                INDVEKVOL(J)=J
                VEKKOR(J)=0.0D0
                VEKAL(J)=0.0D0
                VEKX(J)=0.0D0
                VEKF(J)=0.0D0
                VEKWD(J)=0.0D0
                VEKW(J)=0.0D0
                VEKL(J)=0.0D0
                VEKRF(J)=0.0D0
16          CONTINUE

C       THE  START OF THE MONTE CARLO SIMULATION LOOP
            DO 1004 MC=1,NMC
C               WRITE(6,*) MC

C       GENERATE THE TRAINING DATA SETS
                CALL DRNMVN(NN,IP,RSIG1,IP,XM1,NN)
                DO 18 I=1,NN
                    DO 17 J=1,IP
                        XM(I,J)=XM1(I,J)
17                  CONTINUE
18              CONTINUE
                CALL DRNMVN(MM,IP,RSIG2,IP,XM2,MM)
                DO 22 I=1,MM
                    DO 21 J=1,IP
                        XM(NN+I,J)=XM2(I,J)
21                  CONTINUE
22              CONTINUE

C       STANDARDISE THE TRAINING DATA
                DO 28 J=1,IP
                    S=0.0D0
                    S2=0.0D0
```

```
                DO 26 I=1,NNPMM
                    S=S+XM(I,J)
                    S2=S2+XM(I,J)**2.0D0
26              CONTINUE
                GEM(J)=S/NNPMM
                VAR(J)=(S2-(S**2.0D0)/NNPMM)/NNPMM
                DO 27 I=1,NNPMM
                    XM(I,J)=(XM(I,J)-GEM(J))/DSQRT(VAR(J))
27              CONTINUE
28          CONTINUE

C   GENERATE TEST DATA SETS
            CALL DRNMVN(NT,IP,RSIG1,IP,XT1,NT)
            CALL DRNMVN(MT,IP,RSIG2,IP,XT2,MT)
            DO 36 I=1,NT
                DO 35 J=1,IP
                    XT(I,J)=XT1(I,J)
35              CONTINUE
36          CONTINUE
            DO 40 I=1,MT
                DO 39 J=1,IP
                    XT(NT+I,J)=XT2(I,J)
39              CONTINUE
40          CONTINUE

C   STANDARDISE THE TEST DATA
            DO 45 J=1,IP
                DO 44 I=1,NMT
                    XT(I,J)=(XT(I,J)-GEM(J))/(DSQRT(VAR(J)))
44              CONTINUE
45          CONTINUE

C   CALCULATE (AND SORT) ONE-VARIABLE-AT-A-TIME CORRELATIONS
            TYD1=CPSEC()
            NVER=IPTEL
            DO 240 J=1,IP
                S=0.0D0
```

```
                DO 239 I=1,NNPMM
                    S=S+XM(I,J)*YV(I)
239             CONTINUE
            AKOR(J)=S/(DSQRT(1.0D0*NNPMM*(SOMY2-SOMY*SOMY/NNPMM)))
            AKOR(J)=DABS(AKOR(J))
240         CONTINUE
            DO 241 J=1,IP
                IPERM(J)=J
241         CONTINUE
            CALL DSVRGP(IP,AKOR,AKOR,IPERM)
            DO 242 J=1,NVER
                KIESKOR(J)=IPERM(IP-NVER+J)
242         CONTINUE
            TYD2=CPSEC()
            TYD(1)=TYD(1)+TYD2-TYD1

C   CALCULATE (AND SORT) ONE-VARIABLE-AT-A-TIME ALIGNMENTS
            DO 243 J=1,IP
                ALIGN(J)=ALIGNMENT(J,XM)
243         CONTINUE
            DO 244 J=1,IP
                IPERM(J)=J
244         CONTINUE
            CALL DSVRGP(IP,ALIGN,ALIGN,IPERM)
            DO 245 J=1,NVER
                KIESAL(J)=IPERM(IP-NVER+J)
245         CONTINUE
            TYD3=CPSEC()
            TYD(2)=TYD(2)+TYD3-TYD2

C   DETERMINE THE VARIABLES PROVIDING BEST POSSIBLE SEPARATION BETWEEN
C   MEAN VECTORS IN INPUT SPACE
            DO 64 J=1,IP
                S=0.0D0
                DO 62 I=1,NN
                    S=S+XM(I,J)
62              CONTINUE
```

```
                XGEM1(J)=S/NN
                S=0.0D0
                DO 63 I=NN+1,NNPMM
                    S=S+XM(I,J)
63              CONTINUE
                XGEM2(J)=S/MM
                XVERSKIL(J)=DABS(XGEM1(J)-XGEM2(J))
64          CONTINUE
            DO 65 J=1,IP
                IPERM(J)=J
65          CONTINUE
            CALL DSVRGP(IP,XVERSKIL,XVERSKIL,IPERM)
            DO 68 J=1,IP
                KIESX(J)=IPERM(IP-J+1)
68          CONTINUE
            TYD4=CPSEC()
            TYD(3)=TYD(3)+TYD4-TYD3

C     OBTAIN PRE-IMAGES OF TWO MEAN VECTORS IN FEATURE SPACE
            DO 50 I=1,NN
                ALPHA(I)=1.0D0/NN
50          CONTINUE
            DO 51 I=NN+1,NNPMM
                ALPHA(I)=0.0D0/NN
51          CONTINUE
            CALL PREIMAGE(XM,ALPHA,XPRE1)
            DO 52 I=1,NN
            ALPHA(I)=0.0D0
52          CONTINUE
            DO 53 I=NN+1,NNPMM
                ALPHA(I)=1.0D0/MM
53          CONTINUE
            CALL PREIMAGE(XM,ALPHA,XPRE2)
```

```
C   DETERMINE VARIABLES PROVIDING BEST POSSIBLE SEPARATION BETWEEN MEAN
C   VECTORS IN FEATURE SPACE
                DO 55 J=1,IP
                        IPERM(J)=J
                        XPREVERSKIL(J)=DABS(XPRE1(J)-XPRE2(J))
55              CONTINUE
                CALL DSVRGP(IP,XPREVERSKIL,XPREVERSKIL,IPERM)
                DO 58 J=1,IP
                        KIESF(J)=IPERM(IP-J+1)
58              CONTINUE
                TYD5=CPSEC()
                TYD(4)=TYD(4)+TYD5-TYD4

C   APPLY AN SVM USING THE FULL SET OF AVAILABLE INPUT VARIABLES
                TYDSVM1=CPSEC()
                CALL GRAMMAT(GAMPAR,XM,IP,INDVEKVOL,GRMAT)
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,IP,INDVEKVOL,
               &ALPHAW,BOPT)
                TYDSVM2=CPSEC()
                TYDSVM=TYDSVM2-TYDSVM1

C   DETERMINE THE PRE-IMAGE OF THE SVM WEIGHT VECTOR
                DO 81 I=1,NNPMM
                        ALPHA(I)=ALPHAW(I)
81              CONTINUE
                CALL PREIMAGE(XM,ALPHA,WPRE)

C   DETERMINE THE INPUT VARIABLES CORRESPONDING TO LARGEST ABSOLUTE SVM
C   WEIGHT COEFFICIENTS
                DO 85 J=1,IP
                        IPERM(J)=J
                        WPRE(J)=DABS(WPRE(J))
85              CONTINUE
                CALL DSVRGP(IP,WPRE,WPRE,IPERM)
                DO 88 J=1,IP
                        KIESW(J)=IPERM(IP-J+1)
88              CONTINUE
```

```
      TYD6=CPSEC()
      TYD(5)=TYD(5)+TYD6-TYDSVM2+TYDSVM

C     DETERMINE PRE-IMAGES OF TWO LINEAR COMBINATIONS FOR THE TWO GROUPS
C     WHERE COEFFICIENTS IN THE LINEAR COMBINATIONS ARE OBTAINED FROM THE
C     RESPECTIVE SVM WEIGHT VECTORS
      DO 71 I=1,NN
          ALPHA(I)=ALPHAW(I)
71    CONTINUE
      DO 72 I=NN+1,NNPMM
          ALPHA(I)=0.0D0
72    CONTINUE
      CALL PREIMAGE(XM,ALPHA,WPRE1)
      DO 73 I=1,NN
          ALPHA(I)=0.0D0
73    CONTINUE
      DO 74 I=NN+1,NNPMM
          ALPHA(I)=ALPHAW(I)
74    CONTINUE
      CALL PREIMAGE(XM,ALPHA,WPRE2)

C     DETERMINE THE INPUT VARIABLES MAXIMALLY SEPARATING THE ABOVE TWO PRE-
C     IMAGES
      DO 75 J=1,IP
          IPERM(J)=J
          WPREVERSKIL(J)=DABS(WPRE1(J)-WPRE2(J))
75    CONTINUE
      CALL DSVRGP(IP,WPREVERSKIL,WPREVERSKIL,IPERM)
      DO 78 J=1,IP
          KIESWD(J)=IPERM(IP-J+1)
78    CONTINUE
      TYD7=CPSEC()
      TYD(6)=TYD(6)+TYD7-TYD6+TYDSVM
```

```
C   CALCULATE THE SVM DISCRIMINANT FUNCTION VALUE FOR ALL THE TRAINING
C   PATTERNS
            TYDFW1=CPSEC()
            DO 98 I=1,NNPMM
                ALPHAW(I)=ALPHAW(I)*YV(I)
98          CONTINUE
            DO 105 J=1,NNPMM
                S=0.0D0
                DO 104 I=1,NNPMM
                    S=S+ALPHAW(I)*GRMAT(I,J)
104             CONTINUE
                FW(J)=S
105         CONTINUE
            TYDFW2=CPSEC()
            TYDFW=TYDFW2-TYDFW1

C   PERFORM LEAST SQUARES REGRESSION TO APPROXIMATE SVM DISCRIMINANT
C   FUNCTION VALUES
            INTCEP=0
            CALL DRLSE(NNPMM,FW,IP,XM,NNPMM,INTCEP,BV,SST,SSE)
            DO 185 J=1,IP
                IPERM(J)=J
                BV(J)=DABS(BV(J))
185         CONTINUE
            CALL DSVRGP(IP,BV,BV,IPERM)
            DO 188 J=1,IP
                KIESL(J)=IPERM(IP-J+1)
188         CONTINUE
            TYD8=CPSEC()
            TYD(7)=TYD(7)+TYD8-TYDFW2+TYDSVM+TYDFW
            DO 195 I=1,NNPMM
                DO 194 J=1,IP
                    XMRF(J,I)=XM(I,J)
194             CONTINUE
195         CONTINUE
```

```
C   USE A REGRESSION RANDOM FOREST TO APPROXIMATE SVM DISCRIMINANT
C   FUNCTION VALUES AND RETURN VARIABLE RANKINGS ACCORDING TO THE APPLIED
C   RANDOM FOREST
            CALL RANFOR(XMRF,FW,Z)
            DO J=1,IP
                  IPERM(J)=J
                  Z(J)=DABS(Z(J))
            END DO
            CALL DSVRGP(IP,Z,Z,IPERM)
            DO 198 J=1,IP
                  KIESRF(J)=IPERM(IP-J+1)
198         CONTINUE
            TYD9=CPSEC()
            TYD(8)=TYD(8)+TYD9-TYD8+TYDSVM+TYDFW

C   USE A REGRESSION TREE TO APPROXIMATE SVM DISCRIMINANT
C   FUNCTION VALUES AND RETURN VARIABLE RANKINGS ACCORDING TO THE APPLIED
C   REGRESSION TREE
C           CALL REGTREE(XMRF,FW,KIESTREE)
C           TYD10=CPSEC()
C           TYD(9)=TYD(9)+TYD10-TYD9+TYDSVM+TYDFW

C   UPDATE SELECTION FREQUENCIES
            DO 206 J=1,IPTEL
                  VEKKOR(KIESKOR(J))=VEKKOR(KIESKOR(J))+1.0D0
                  VEKAL(KIESAL(J))=VEKAL(KIESAL(J))+1.0D0
                  VEKX(KIESX(J))=VEKX(KIESX(J))+1.0D0
                  VEKF(KIESF(J))=VEKF(KIESF(J))+1.0D0
                  VEKW(KIESW(J))=VEKW(KIESW(J))+1.0D0
                  VEKWD(KIESWD(J))=VEKWD(KIESWD(J))+1.0D0
                  VEKL(KIESL(J))=VEKL(KIESL(J))+1.0D0
                  VEKRF(KIESRF(J))=VEKRF(KIESRF(J))+1.0D0
206         CONTINUE
```

```
C   CALCULATE THE SVM TEST ERROR BASED ON THE FULL SET OF AVAILABLE INPUT
C   VARIABLES
          NVER=IP
          CALL GRAMMAT(GAMPAR,XM,NVER,INDVEKVOL,GRMAT)
          CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,INDVEKVOL,AL,BOPT)
          CALL GRAMNUUT(GAMPAR,XM,XT,NVER,INDVEKVOL,GRNUUT)
          CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
          FOUTMAT(MC,1)=FOUT

C   CALCULATE THE SVM TEST ERROR BASED ONLY ON THE SET OF TRULY SEPARATING
C   INPUT VARIABLES
          NVER=IPTEL
          CALL GRAMMAT(GAMPAR,XM,NVER,KIESREG,GRMAT)
          CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESREG,AL,BOPT)
          CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESREG,GRNUUT)
          CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
          FOUTMAT(MC,2)=FOUT

C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA THE USE OF CORRELATIONS IN INPUT SPACE
          CALL GRAMMAT(GAMPAR,XM,NVER,KIESKOR,GRMAT)
          CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESKOR,AL,BOPT)
          CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESKOR,GRNUUT)
          CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
          FOUTMAT(MC,3)=FOUT

C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA THE USE OF ALIGNMENTS IN FEATURE SPACE
          CALL GRAMMAT(GAMPAR,XM,NVER,KIESAL,GRMAT)
          CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESAL,AL,BOPT)
          CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESAL,GRNUUT)
          CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
          FOUTMAT(MC,4)=FOUT
```

```
C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA DIFFERENCES IN GROUP MEANS IN INPUT SPACE
        CALL GRAMMAT(GAMPAR,XM,NVER,KIESX,GRMAT)
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESX,AL,BOPT)
        CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESX,GRNUUT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMAT(MC,5)=FOUT

C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA DIFFERENCES IN PRE-IMAGES BASED ON  MEANS IN INPUT SPACE
        CALL GRAMMAT(GAMPAR,XM,NVER,KIESF,GRMAT)
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESF,AL,BOPT)
        CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESF,GRNUUT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMAT(MC,6)=FOUT

C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA DIFFERENCES IN PRE-IMAGES BASED LINEAR COMBINATIONS USING
C   THE SVM WEIGHT VECTOR
        CALL GRAMMAT(GAMPAR,XM,NVER,KIESW,GRMAT)
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESW,AL,BOPT)
        CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESW,GRNUUT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMAT(MC,7)=FOUT

C   CALCULATE THE TEST ERROR USING DIFFERENCES IN PRE-IMAGES (OF THE WEIGHT
C   VECTOR IN FEATURE SPACE)
        CALL GRAMMAT(GAMPAR,XM,NVER,KIESWD,GRMAT)
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESWD,AL,BOPT)
        CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESWD,GRNUUT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMAT(MC,8)=FOUT
```

```
C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA LEAST SQUARES APPROXIMATION OF THE SVM DISCRIMINANT
C   FUNCTION VALUES
            CALL GRAMMAT(GAMPAR,XM,NVER,KIESL,GRMAT)
            CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESL,AL,BOPT)
            CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESL,GRNUUT)
            CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
            FOUTMAT(MC,9)=FOUT

C   CALCULATE THE SVM TEST ERROR BASED ON THE SUBSET OF INPUT VARIABLES
C   SELECTED VIA RANDOM FOREST APPROXIMATION OF THE SVM DISCRIMINANT
C   FUNCTION VALUES
            CALL GRAMMAT(GAMPAR,XM,NVER,KIESRF,GRMAT)
            CALL DOENSVM(YV,XM,GRMAT,CPARS,GAMPAR,NVER,KIESRF,AL,BOPT)
            CALL GRAMNUUT(GAMPAR,XM,XT,NVER,KIESRF,GRNUUT)
            CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
            FOUTMAT(MC,10)=FOUT

1004    CONTINUE
C   THE MONTE CARLO SIMULATION LOOP ENDS HERE

C   AVERAGE THE OBTAINED TEST ERRORS AND SELECTION FREQUENCIES AND
C   CALCULATE STANDARD ERRORS
        DO 345 J=1,11
            S1=0.0D0
            S2=0.0D0
            DO 344 I=1,NMC
                S1=S1+FOUTMAT(I,J)
                S2=S2+FOUTMAT(I,J)**2.0D0
344         CONTINUE
            FOUTGEM(J)=S1/NMC
            FOUTSTD(J)=DSQRT((S2-(S1*S1)/NMC)/(NMC*(NMC-1.0D0)))
345     CONTINUE
```

```
C    CALCULATE THE AVERAGE TIME TO COMPLETE EACH SELECTION PROCEDURE
      DO 348 J=1,9
            TYD(J)=1.0D3*TYD(J)/NMC
348   CONTINUE

C    CALCULATE SELECTION PROPORTIONS
      DO 349 J=1,IP
            VEKKOR(J)=VEKKOR(J)/NMC
            VEKAL(J)=VEKAL(J)/NMC
            VEKX(J)=VEKX(J)/NMC
            VEKF(J)=VEKF(J)/NMC
            VEKW(J)=VEKW(J)/NMC
            VEKWD(J)=VEKWD(J)/NMC
            VEKL(J)=VEKL(J)/NMC
            VEKRF(J)=VEKRF(J)/NMC
349   CONTINUE

C    WRITE AVERAGE TEST ERRORS, STANDARD ERRORS, AND AVERAGE TIMES TO
C    COMPLETE EACH SELECTION PROCEDURE TO A FILE
      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
            WRITE(1,*) 'C: ',CPARS,'G: ',GAMPAR
            WRITE(1,599) 'FUL','ORA','KOR','AL','XME','FME','WGH','WGHD',
            & 'LS','RFR','RTRE'
            WRITE(1,600) (FOUTGEM(J),J=1,11)
            WRITE(1,600)
            WRITE(1,600) (FOUTSTD(J),J=1,11)
            WRITE(1,600)
            WRITE(1,605) (TYD(J),J=1,9)
            WRITE(1,600)
            CLOSE(1)

C    WRITE SELECTION PROPORTIONS TO A FILE
      OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
            WRITE(1,*) 'C: ',CPARS,'G: ',GAMPAR
            WRITE(1,*) 'CORRELATIONS'
            WRITE(1,602) (VEKKOR(J),J=1,IP)
            WRITE(1,*) 'ALIGNMENTS'
```

```
                WRITE(1,602) (VEKAL(J),J=1,IP)
                WRITE(1,*) 'DIF IN MEANS: INPUT SPACE'
                WRITE(1,602) (VEKX(J),J=1,IP)
                WRITE(1,*) 'DIF IN MEANS: FEATURE SPACE'
                WRITE(1,602) (VEKF(J),J=1,IP)
                WRITE(1,*) 'WEIGHT VECTOR'
                WRITE(1,602) (VEKW(J),J=1,IP)
                WRITE(1,*) 'WEIGHT VECTOR DIFFERENCES'
                WRITE(1,602) (VEKWD(J),J=1,IP)
                WRITE(1,*) 'LEAST SQUARES'
                WRITE(1,602) (VEKL(J),J=1,IP)
                WRITE(1,*) 'REGRESSION RANDOM FOREST'
                WRITE(1,602) (VEKRF(J),J=1,IP)
                WRITE(1,*) 'REGRESSION TREE'
                WRITE(1,602) (VEKTREE(J),J=1,IP)
        CLOSE(1)
1010CONTINUE
C     END OF THE COST PARAMETER LOOP

C     FILE FORMATS
599  FORMAT(1X,A3,2(4X,A3),5X,A2,3(4X,A3),4X,A4,3(4X,A3))
600  FORMAT(12(F5.3,2X))
601  FORMAT(20I3)
602  FORMAT(12(F5.2,1X))
605  FORMAT(14X,9(F5.1,2X))

7000STOP
END
C     END OF THE MAIN SIMULATION PROGRAM
```

C.3 *ALGORITHM-INDEPENDENT AND ALGORITHM-DEPENDENT SELECTION IN FEATURE SPACE*

The following Fortran code is an example of the programs used in the numerical evaluations reported in Section 6.2 and 7.2 of Chapter 4.

```
C   IN THIS PROGRAM WE CALCULATE AVERAGE TEST ERRORS (AND STANDARD ERRORS)
C   PERTAINING TO THE FULL SVM AND KFD (BINARY) CLASSIFIERS, AND POST-
C   SELECTION AVERAGE TEST ERRORS (AND STANDARD ERRORS) FOR SVM AND KFD
    CLASSIFIERS WHEN ALGORITHM- INDEPENDENT SELECTION CRITERIA
C       C.3.1  CORRELATIONS
C       C.3.2  ALIGNMENTS
C       C.3.3  BETWEEN GROUP DISSIMILARITIES IN INPUT SPACE
C       C.3.4  DIFFERENCES BETWEEN GROUP MEANS IN INPUT SPACE
C       C.3.5  VARIATION RATIOS IN INPUT SPACE
C   AND ALGORITHM-DEPENDENT SELECTION CRITERIA, VIZ.
C       C.3.6  THE NORM OF THE SVM WEIGHT VECTOR
C       C.3.7  THE RAYLEIGH QUOTIENT
C   ARE USED.
C   THE DATA ARE GENERATED FROM A MULTIVARIATE LOGNORMAL DISTRIBUTION AND
    THE TWO GROUPS DIFFER WITH RESPECT TO LOCATION
C   SETS OF RELEVANT AND IRRELEVANT INPUT VARIABLES ARE UNCORRELATED

C   THE FOLLOWING (OWN) FUNCTIONS ARE REQUIRED:
C       1.  ALIGNMENT
C       2.  ALIGNMENTTRANS

C   THE FOLLOWING (OWN) SUBROUTINES ARE REQUIRED:
C       1.  BERCRIT12
C       2.  BERCRITCC
C       3.  BERCRITR
C       4.  BERCRITW
C       5.  BERFOUTKFDA
C       6.  BERFOUTSVM
C       7.  DOENKFDA
C       8.  DOENSVM
```

```
C       9.  GRAMMAT
C       10. GRAMNUUT

C   THE FOLLOWING IMSL FUNCTIONS ARE REQUIRED:
C       1.  CPSEC
C       2.  DEXP
C       3.  DSQRT

C   THE FOLLOWING IMSL SUBROUTINES ARE REQUIRED:
C       1.  DCHFAC
C       2.  DSVRGP

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=4,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NMC=500,CORR=0.0D0,SIGNOISE=20.0D0)
      PARAMETER (GAMPAR=1.0D0/IP)
      PARAMETER (NT=1000,MT=1000,NMT=NT+MT)

C   POPULATION PARAMETERS
      DIMENSION AMU1(IP),AMU2(IP)
      DIMENSION SIGMAM(IPTEL,IPTEL),RSIG(IPTEL,IPTEL)
      DIMENSION SIGMAMNOISE(NNOISE,NNOISE),RSIGNOISE(NNOISE,NNOISE)

C   X MATRICES AND Y VECTORS
      DIMENSION XM(NNPMM,IP),XMRF(IP,NNPMM)
      DIMENSION XM11(NN,IPTEL),XM21(MM,IPTEL)
      DIMENSION XM12(NN,NNOISE),XM22(MM,NNOISE)
      DIMENSION XGEM1(IP),XGEM2(IP),GEM(IP),VAR(IP)
      DIMENSION XT(NMT,IP),XT11(NT,IPTEL),XT21(MT,IPTEL)
      DIMENSION XT12(NT,NNOISE),XT22(MT,NNOISE)
      DIMENSION YV(NNPMM),YVT(NMT)

C   SVM AND KFDA RELATED QUANTITIES
      DIMENSION GRMAT(NNPMM,NNPMM),GRNUUT(NMT,NNPMM),GRMATV(IP,NNPMM,NNPMM)
      DIMENSION ALPHAV(IP,NNPMM),ALPHA(NNPMM),ALPHAW(NNPMM),AL(NNPMM)
      DIMENSION EENP(NNPMM),EENM(NNPMM)
      DIMENSION B(NNPMM),H(NNPMM,NNPMM)
```

```
C   VARIOUS QUANTITIES
      DIMENSION TYD(9)
      DIMENSION FOUTMATK(NMC,10),FOUTVERK(NMC,10)
      DIMENSION FOUTMATS(NMC,10),FOUTVERS(NMC,10)
      DIMENSION FOUTGEMK(10),FOUTSTDK(10)
      DIMENSION FOUTGEMS(10),FOUTSTDS(10)
      DIMENSION FREKWKIESAL(IP),FREKWKIESALT(IP),FREKWKIESCC(IP)
      DIMENSION FREKWKIESET(IP),FREKWKIESG(IP),FREKWKIESSV(IP)
      DIMENSION FREKWKIESN(IP),FREKWKIESW(IP),FREKWKIESR(IP)
      DIMENSION INDVEKVOL(IP),INDVEKREG(IPTEL),INDVEKEEN(IP)
      DIMENSION IPERMU(IP),IPERM(IP)
      DIMENSION IPERMCC(IP),IPERMET(IP),IPERMSV(IP),IPERMAL(IP),IPERMALT(IP)
      DIMENSION IPERMG(IP),IPERMWW(IP),IPERMRR(IP),IPERMN(IP)
      DIMENSION INDVEKCC(IP),INDVEKET(IP),INDVEKSV(IP),INDVEKAL(IP)
      DIMENSION INDVEKG(IP),INDVEKW(IP),INDVEKR(IP),INDVEKN(IP),INDVEKALT(IP)
      DIMENSION CRITVEKCC(IP),CRITVEKET(IP),CRITVEKSV(IP),CRITVEKAL(IP)
      DIMENSION CRITVEKG(IP),CRITVEKW(IP),CRITVEKR(IP),CRITVEKN(IP),CRITVEKALT(IP)

      CHARACTER*70 FILEOUT1,FILEOUT2
      FILEOUT1='foutLL.d'
      FILEOUT2='kiesLL.d'

C   WRITE FILE HEADERS
      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
          WRITE(1,*) 'LOGNORMAL DISTRIBUTION DIFF IN LOCATION'
          WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
          WRITE(1,*) 'GAMPAR=',GAMPAR
          WRITE(1,*) 'NN=',NN,'MM=',MM,'NMC=',NMC
          WRITE(1,*) 'NT=',NT,'MT=',MT
          WRITE(1,*) 'SIGNOISE=',SIGNOISE
          WRITE(1,600)
      CLOSE(1)

      OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
          WRITE(1,*) 'LOGNORMAL DISTRIBUTION DIFF IN LOCATION'
          WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
          WRITE(1,*) 'GAMPAR=',GAMPAR
```

```
      WRITE(1,*) 'NN=',NN,'MM=',MM,'NMC=',NMC
      WRITE(1,*) 'NT=',NT,'MT=',MT
      WRITE(1,*) 'SIGNOISE=',SIGNOISE
      WRITE(1,600)
    CLOSE(1)

C   ASSIGN VALUES TO THE POPULATION PARAMETERS
    E=DEXP(1.0D0)
    BLAM=DSQRT(1.0D0/(E*(E-1.0D0)))
    EP=-1.0D0*DSQRT(1.0D0/(E-1.0D0))

    DO 3 I=1,IPTEL
      INDVEKREG(I)=I
      AMU1(I)=0.0D0
      AMU2(I)=1.0D0
      DO 2 J=1,IPTEL
            SIGMAM(I,J)=DLOG(1.0D0+CORR*(E-1.0D0))
2     CONTINUE
      SIGMAM(I,I)=1.0D0
3   CONTINUE

    DO 5 I=1,NNOISE
      AMU1(IPTEL+I)=0.0D0
      AMU2(IPTEL+I)=0.0D0
      DO 4 J=1,NNOISE
            SIGMAMNOISE(I,J)=0.0D0
4     CONTINUE
      SIGMAMNOISE(I,I)=1.0D0
5   CONTINUE
    TOL=1.0D2*DMACH(4)
    CALL DCHFAC(IPTEL,SIGMAM,IPTEL,TOL,IRANK,RSIG,IPTEL)
    CALL DCHFAC(NNOISE,SIGMAMNOISE,NNOISE,TOL,IRANK,RSIGNOISE,NNOISE)

C   INITIALISE VARIOUS QUANTITIES
    DO 9 I=1,NN
      YV(I)=-1.0D0
9   CONTINUE
```

```
      DO 10 I=NN+1,NNPMM
         YV(I)=1.0D0
10    CONTINUE
      SOMY=1.0D0*(MM-NN)
      SOMY2=NNPMM

      DO 11 I=1,NT
         YVT(I)=-1.0D0
11    CONTINUE
      DO 12 I=NT+1,NMT
         YVT(I)=1.0D0
12    CONTINUE

C     SET UP THE KFDA VECTORS
      DO 15 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
15    CONTINUE

C     THE NEXT LOOP SPECIFIES DIFFERENT VALUES FOR THE COST PARAMETER
      DO 1010 ICP=1,6
         IF (ICP.EQ.1) CPAR=0.00001D0
         IF (ICP.EQ.2) CPAR=0.001D0
         IF (ICP.EQ.3) CPAR=0.1D0
         IF (ICP.EQ.4) CPAR=10.0D0
         IF (ICP.EQ.5) CPAR=1000.0D0
         IF (ICP.EQ.6) CPAR=1000.0D0

C     INITIALISE THE TEST ERROR AND STANDARD ERROR VECTORS
         DO 16 J=1,10
               FOUTGEMK(J)=0.0D0
               FOUTSTDK(J)=0.0D0
               FOUTGEMS(J)=0.0D0
               FOUTSTDS(J)=0.0D0
16       CONTINUE
```

```
C    INITIALISE THE VECTORS MEASURING THE IMPLEMENTATION TIME PERTAINING TO
C    EACH SELECTION CRITERION
     DO J=1,9
        TYD(J)=0.0D0
     ENDDO

C    INITIALISE SELECTION FREQUENCY VECTORS
     DO J=1,IP
        FREKWKIESAL(J)=0.0D0
        FREKWKIESCC(J)=0.0D0
        FREKWKIESET(J)=0.0D0
        FREKWKIESG(J)=0.0D0
        FREKWKIESN(J)=0.0D0
        FREKWKIESW(J)=0.0D0
        FREKWKIESR(J)=0.0D0
        FREKWKIESALT(J)=0.0D0
     ENDDO

C    THE MONTE CARLO SIMULATION LOOP STARTS HERE
     DO 1004 MC=1,NMC
C       WRITE(6,*) MC

C    GENERATE THE TRAINING DATA
        CALL DRNMVN(NN,IPTEL,RSIG,IPTEL,XM11,NN)
        DO 18 I=1,NN
              DO 17 J=1,IPTEL
                    XM(I,J)=(BLAM*DEXP(XM11(I,J)))+EP+AMU1(J)
17            CONTINUE
18      CONTINUE
        CALL DRNMVN(NN,NNOISE,RSIGNOISE,NNOISE,XM12,NN)
        DO 20 I=1,NN
              DO 19 J=1,NNOISE
                    XM(I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XM12(I,J)))+
                   &EP+AMU1(J+IPTEL)/DSQRT(SIGNOISE))
19            CONTINUE
20      CONTINUE
        CALL DRNMVN(MM,IPTEL,RSIG,IPTEL,XM21,MM)
```

```
      DO 22 I=1,MM
            DO 21 J=1,IPTEL
                  XM(NN+I,J)=(BLAM*DEXP(XM21(I,J)))+EP+AMU2(J)
21          CONTINUE
22    CONTINUE
      CALL DRNMVN(MM,NNOISE,RSIGNOISE,NNOISE,XM22,MM)
      DO 24 I=1,MM
            DO 23 J=1,NNOISE
                  XM(NN+I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XM22(I,J)))+
                 &EP+AMU2(J+IPTEL)/DSQRT(SIGNOISE))
23          CONTINUE
24    CONTINUE

C     STANDARDISE THE TRAINING DATA
      DO 28 J=1,IP
            S=0.0D0
            S2=0.0D0
            DO 26 I=1,NNPMM
                  S=S+XM(I,J)
                  S2=S2+XM(I,J)**2.0D0
26          CONTINUE
            GEM(J)=S/NNPMM
            VAR(J)=(S2-(S**2.0D0)/NNPMM)/NNPMM
            DO 27 I=1,NNPMM
                  XM(I,J)=(XM(I,J)-GEM(J))/DSQRT(VAR(J))
27          CONTINUE
28    CONTINUE

C     GENERATE THE TEST DATA
      CALL DRNMVN(NT,IPTEL,RSIG,IPTEL,XT11,NT)
      CALL DRNMVN(NT,NNOISE,RSIGNOISE,NNOISE,XT12,NT)
      CALL DRNMVN(MT,IPTEL,RSIG,IPTEL,XT21,MT)
      CALL DRNMVN(MT,NNOISE,RSIGNOISE,NNOISE,XT22,MT)

      DO 36 I=1,NT
            DO 35 J=1,IPTEL
                  XT(I,J)=(BLAM*DEXP(XT11(I,J)))+EP+AMU1(J)
```

```
35          CONTINUE
36      CONTINUE
        DO 38 I=1,NT
            DO 37 J=1,NNOISE
                XT(I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XT12(I,J)))
               &+EP+AMU1(J+IPTEL)/ DSQRT(SIGNOISE))
37          CONTINUE
38      CONTINUE
        DO 40 I=1,MT
            DO 39 J=1,IPTEL
                XT(NT+I,J)=(BLAM*DEXP(XT21(I,J)))+EP+AMU2(J)
39          CONTINUE
40      CONTINUE
        DO 42 I=1,MT
            DO 41 J=1,NNOISE
                XT(NT+I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XT22(I,J)))+EP
               &+AMU2(J+IPTEL)/DSQRT(SIGNOISE))
41          CONTINUE
42      CONTINUE

C   STANDARDISE THE TEST DATA
        DO 45 J=1,IP
            DO 44 I=1,NMT
                XT(I,J)=(XT(I,J)-GEM(J))/(DSQRT(VAR(J)))
44          CONTINUE
45      CONTINUE

C   INITIALISE THE VARIABLE INDEX VECTORS
        DO 50 J=1,IP
            INDVEKVOL(J)=J
50      CONTINUE
        DO 51 J=1,IP
            IPERMCC(J)=J
            IPERMET(J)=J
            IPERMG(J)=J
            IPERMWW(J)=J
            IPERMRR(J)=J
```

```
            IPERMN(J)=J
            IPERMSV(J)=J
            IPERMAL(J)=J
            IPERMALT(J)=J
51      CONTINUE

C   CALCULATE THE TEST ERRORS PERTAINING TO THE FULL (SVM AND KFD) MODELS
        CALL GRAMMAT(GAMPAR,XM,IP,INDVEKVOL,GRMAT)
        CALL GRAMNUUT(GAMPAR,XM,XT,IP,INDVEKVOL,GRNUUT)
        CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
        FOUTMATK(MC,1)=FOUT
        CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IP,INDVEKVOL,AL,BOPT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMATS(MC,1)=FOUT

C   CALCULATE THE TEST ERRORS PERTAINING TO THE CORRECT MODEL
        CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKREG,GRMAT)
        CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKREG,GRNUUT)
        CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
        FOUTMATK(MC,2)=FOUT
        CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKREG,AL,BOPT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMATS(MC,2)=FOUT
        DO K=1,IP
            INDVEKEEN(1)=K
            TYD1=CPSEC()
            CALL GRAMMAT(GAMPAR,XM,1,INDVEKEEN,GRMAT)
            TYD2=CPSEC()
            TYD0=TYD2-TYD1
            TYD1=CPSEC()
            CALL BERCRITW(GRMAT,CRIT1,CRIT2)
            TYD2=CPSEC()
            TYD(3)=TYD(3)+TYD0+TYD2-TYD1
            TYD(4)=TYD(4)+TYD0+TYD2-TYD1
            CRITVEKW(K)=CRIT1
```

```
CRITVEKG(K)=CRIT2
TYD1=CPSEC()
CALL BERCRIT12(GRMAT,CRIT)
TYD2=CPSEC()
TYD(5)=TYD(5)+TYD0+TYD2-TYD1
CRITVEKET(K)=CRIT
TYD1=CPSEC()
CALL BERCRITCC(GRMAT,CRIT)
TYD2=CPSEC()
TYD(6)=TYD(6)+TYD0+TYD2-TYD1
CRITVEKCC(K)=CRIT
TYD1=CPSEC()
CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
TYD2=CPSEC()
CALL BERCRITR(GRMAT,EENP,EENM,ALPHA,CPAR,CRIT)
CRITVEKR(K)=CRIT
TYD(2)=TYD(2)+TYD0+TYD2-TYD1

C   CALCULATE THE SQUARED NORM OF THE SVM WEIGHT VECTOR
TYD1=CPSEC()
CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,1,INDVEKEEN,AL,BOPT)
S=0.0D0
DO I=1,NNPMM
        DO J=1,NNPMM
                S=S+AL(I)*AL(J)*YV(I)*YV(J)*GRMAT(I,J)
        END DO
END DO
TYD2=CPSEC()
TYD(1)=TYD(1)+TYD0+TYD2-TYD1
CRITVEKN(K)=S

C   CALCULATE THE NUMBER OF SUPPORT VECTORS
TYD3=CPSEC()
TEL=0.0D0
DO I=1,NNPMM
        IF (AL(I).GT.0.00000001D0) TEL=TEL+1.0D0
ENDDO
```

```
            TYD4=CPSEC()
            TYD(7)=TYD(7)+TYD0+TYD2-TYD1+TYD4-TYD3
            CRITVEKSV(K)=TEL

C   CALCULATE THE ALIGNMENT CRITERION
            TYD1=CPSEC()
            CRITVEKAL(K)=ALIGNMENT(INDVEKEEN,XM)
            TYD2=CPSEC()
            TYD(8)=TYD(8)+TYD2-TYD1
            TYD1=CPSEC()
            CRITVEKALT(K)=ALIGNMENTTRANS(INDVEKEEN,XM)
            TYD2=CPSEC()
            TYD(9)=TYD(9)+TYD2-TYD1
        ENDDO

C   ORDER THE SELECTION CRITERIA VALUES PERTAINING TO EACH SINGLE VARIABLE
C   MODEL
        CALL DSVRGP(IP,CRITVEKW,CRITVEKW,IPERMWW)
        CALL DSVRGP(IP,CRITVEKG,CRITVEKG,IPERMG)
        CALL DSVRGP(IP,CRITVEKET,CRITVEKET,IPERMET)
        CALL DSVRGP(IP,CRITVEKCC,CRITVEKCC,IPERMCC)
        CALL DSVRGP(IP,CRITVEKR,CRITVEKR,IPERMRR)
        CALL DSVRGP(IP,CRITVEKN,CRITVEKN,IPERMN)
        CALL DSVRGP(IP,CRITVEKSV,CRITVEKSV,IPERMSV)
        CALL DSVRGP(IP,CRITVEKAL,CRITVEKAL,IPERMAL)
        CALL DSVRGP(IP,CRITVEKALT,CRITVEKALT,IPERMALT)

        DO 244 J=1,IP
            IPERM(J)=J
244     CONTINUE

C   UPDATE THE VARIABLE INDEX VECTORS
        DO 53 J=1,IP
            INDVEKW(J)=IPERMWW(IP-J+1)
            INDVEKG(J)=IPERMG(IP-J+1)
            INDVEKET(J)=IPERMET(J)
            INDVEKCC(J)=IPERMCC(J)
```

```
            INDVEKR(J)=IPERMRR(IP-J+1)
            INDVEKN(J)=IPERMN(IP-J+1)
            INDVEKSV(J)=IPERMSV(J)
            INDVEKAL(J)=IPERMAL(IP-J+1)
            INDVEKALT(J)=IPERMALT(IP-J+1)
53      CONTINUE

C   UPDATE THE SELECTION PROPORTIONS
        DO 54 J=1,IPTEL
            FREKWKIESET(INDVEKET(J))=FREKWKIESET(INDVEKET(J))+1.0D0
            FREKWKIESCC(INDVEKCC(J))=FREKWKIESCC(INDVEKCC(J))+1.0D0
            FREKWKIESR(INDVEKR(J))=FREKWKIESR(INDVEKR(J))+1.0D0
            FREKWKIESG(INDVEKG(J))=FREKWKIESG(INDVEKG(J))+1.0D0
            FREKWKIESW(INDVEKW(J))=FREKWKIESW(INDVEKW(J))+1.0D0
            FREKWKIESSV(INDVEKSV(J))=FREKWKIESSV(INDVEKSV(J))+1.0D0
            FREKWKIESN(INDVEKN(J))=FREKWKIESN(INDVEKN(J))+1.0D0
            FREKWKIESAL(INDVEKAL(J))=FREKWKIESAL(INDVEKAL(J))+1.0D0
            FREKWKIESALT(INDVEKALT(J))=FREKWKIESALT(INDVEKALT(J))+1.0D0
54      CONTINUE

C   CALCULATE SELECTION TEST ERRORS PERTAINING TO SELECTION USING
C   THE RAYLEIGH QUOTIENT
            CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKR,GRMAT)
            CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKR,GRNUUT)
            CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
            CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
            FOUTMATK(MC,3)=FOUT

C   THE SQUARED NORM OF THE SVM WEIGHT VECTOR
            CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKN,AL,BOPT)
            CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
            FOUTMATS(MC,3)=FOUT

C   THE VARIATION RATIO
            CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKW,GRMAT)
            CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKW,GRNUUT)
            CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
```

```
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,4)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKW,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,4)=FOUT

C     DIFFERENCES IN GROUP MEANS
      CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKG,GRMAT)
      CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKG,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,5)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKG,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,5)=FOUT

C     THE SUM OF DISSIMILARITIES BETWEEN THE TWO GROUPS
      CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKET,GRMAT)
      CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKET,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,6)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKET,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,6)=FOUT

C     THE CUT COST
      CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKCC,GRMAT)
      CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKCC,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,7)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKCC,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,7)=FOUT
```

```
C     THE NUMBER OF SUPPORT VECTORS
                CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKSV,GRMAT)
                CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKSV,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,8)=FOUT
                CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKSV,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,8)=FOUT

C     THE ALIGNMENT CRITERION
                CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKAL,GRMAT)
                CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKAL,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,9)=FOUT
                CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKAL,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,9)=FOUT

C     THE TRANSFORMED ALIGNMENT CRITERION
                CALL GRAMMAT(GAMPAR,XM,IPTEL,INDVEKALT,GRMAT)
                CALL GRAMNUUT(GAMPAR,XM,XT,IPTEL,INDVEKALT,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,10)=FOUT
                CALL DOENSVM(YV,XM,GRMAT,CPAR,GAMPAR,IPTEL,INDVEKALT,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,10)=FOUT

1004    CONTINUE
C     THE MONTE CARLO SIMULATION LOOP ENDS HERE
```

```
C    CALCULATE AVERAGES OF THE OBTAINED TEST ERRORS AND OBTAIN STANDARD
C    ERRORS
      DO 345 J=1,10
            S1=0.0D0
            S2=0.0D0
            S3=0.0D0
            S4=0.0D0
            DO 344 I=1,NMC
                  S1=S1+FOUTMATK(I,J)
                  S2=S2+FOUTMATK(I,J)**2.0D0
                  S3=S3+FOUTMATS(I,J)
                  S4=S4+FOUTMATS(I,J)**2.0D0
344         CONTINUE
            FOUTGEMK(J)=S1/NMC
            FOUTSTDK(J)=DSQRT((S2-(S1*S1)/NMC)/(NMC*(NMC-1.0D0)))
            FOUTGEMS(J)=S3/NMC
            FOUTSTDS(J)=DSQRT((S4-(S3*S3)/NMC)/(NMC*(NMC-1.0D0)))
345   CONTINUE

C    CALCULATE SELECTION PERCENTAGES
      DO 349 J=1,IP
         FREKWKIESET(J)=FREKWKIESET(J)/NMC
         FREKWKIESCC(J)=FREKWKIESCC(J)/NMC
         FREKWKIESR(J)=FREKWKIESR(J)/NMC
         FREKWKIESSV(J)=FREKWKIESSV(J)/NMC
         FREKWKIESN(J)=FREKWKIESN(J)/NMC
         FREKWKIESW(J)=FREKWKIESW(J)/NMC
         FREKWKIESAL(J)=FREKWKIESAL(J)/NMC
         FREKWKIESALT(J)=FREKWKIESALT(J)/NMC
         FREKWKIESG(J)=FREKWKIESG(J)/NMC
349   CONTINUE

C    WRITE THE AVERAGE TIMES, TEST ERRORS AND STANDARD ERRORS TO A FILE
      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
         WRITE(1,*) 'C: ',CPAR,'G: ',GAMPAR
         WRITE(1,*) 'KFDA'
         WRITE(1,599) 'FULL','ORACLE','NO/RAY','WANG','FMEANS','BL12','CC','SVs','AL','ALT'
```

```
      WRITE(1,600) (FOUTGEMK(J),J=1,10)
      WRITE(1,600) (FOUTSTDK(J),J=1,10)
      WRITE(1,601) TYD(2),(TYD(J),J=3,9)
      WRITE(1,*) 'SVM'
      WRITE(1,600) (FOUTGEMS(J),J=1,10)
      WRITE(1,600) (FOUTSTDS(J),J=1,10)
      WRITE(1,601) TYD(1),(TYD(J),J=3,9)
      WRITE(1,600)
    CLOSE(1)

C   WRITE THE SELECTION PERCENTAGES TO A FILE
    OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
      WRITE(1,*) 'C: ',CPAR,'G: ',GAMPAR
      WRITE(1,*) 'WEIGHT VECTOR NORM'
      WRITE(1,602) (FREKWKIESN(J),J=1,IP)
      WRITE(1,*) 'RAYLEIGH'
      WRITE(1,602) (FREKWKIESR(J),J=1,IP)
      WRITE(1,*) 'DIF IN MEANS: FEATURE SPACE'
      WRITE(1,602) (FREKWKIESG(J),J=1,IP)
      WRITE(1,*) 'WANG'
      WRITE(1,602) (FREKWKIESW(J),J=1,IP)
      WRITE(1,*) 'BLOCK 12'
      WRITE(1,602) (FREKWKIESET(J),J=1,IP)
      WRITE(1,*) 'CUT COST'
      WRITE(1,602) (FREKWKIESCC(J),J=1,IP)
      WRITE(1,*) 'ALIGNMENTS'
      WRITE(1,602) (FREKWKIESAL(J),J=1,IP)
      WRITE(1,*) 'ALIGNMENTS-TRANS'
      WRITE(1,602) (FREKWKIESALT(J),J=1,IP)
      WRITE(1,*)
    CLOSE(1)
1010  CONTINUE
C   THE COST PARAMETER LOOP ENDS HERE
```

```
C     FILE FORMATS
599   FORMAT(3X,A4,3X,A6,2X,A6,3X,A4,2X,A6,3X,A4,5X,A2,7X,A3,3X,A2,6X,A3)
500   FORMAT(20I3)
600   FORMAT(10(F7.3,1X))
601   FORMAT(16X,9(F7.3,1X))
602   FORMAT(12(F5.2,1X))
605   FORMAT(I6,2X,4(F12.5,1X))
700   FORMAT(10(F20.10))

7000  STOP
      END
C     END OF THE SIMULATION PROGRAM
```

C.4 *BACKWARD ELIMINATION FOR KERNEL CLASSIFIERS*

An example of the simulation program used to investigate use of recursive elimination in variable selection is given below. Results of the study are given in Section 5 of Chapter 5.

```
C   IN THIS PROGRAM WE INVESTIGATE SELECTION USING A BACKWARD STRATEGY
C   WE CALCULATE AVERAGE TEST ERRORS (AND STANDARD ERRORS)
C   PERTAINING TO THE FULL SVM AND KFD MODELS, AND POST-SELECTION AVERAGE
C   TEST ERRORS (AND STANDARD ERRORS) FOR SVM AND KFD MODELS AFTER USING
C       C.3.1  ZERO ORDER ALIGNMENTS IN FEATURE SPACE
C       C.3.2  FIRST ORDER ALIGNMENTS IN FEATURE SPACE
C       C.3.3  TRANSFORMED ZERO ORDER ALIGNMENTS IN FEATURE SPACE
C       C.3.4  TRANSFORMED FIRST ORDER ALIGNMENTS IN FEATURE SPACE
C       C.3.5  ZERO ORDER VARIATION RATIOS
C       C.3.6  FIRST ORDER VARIATION RATIOS
C       C.3.7  ZERO ORDER DIFFERENCES IN MEANS IN INPUT SPACE
C       C.3.8  FIRST ORDER DIFFERENCES IN MEANS IN INPUT SPACE
C       C.3.9  THE ZERO ORDER DISSIMILARITY CRITERION
C       C.3.10  THE FIRST ORDER DISSIMILARITY CRITERION
C   FOR SVM MODELS AFTER USING
C       C.3.11  THE ZERO ORDER NORM OF THE SVM WEIGHT VECTOR
C       C.3.12  THE FIRST ORDER NORM OF THE SVM WEIGHT VECTOR
C   AND FOR KFDA MODELS AFTER USING
C       C.3.13  THE ZERO ORDER RAYLEIGH QUOTIENT
C       C.3.14  THE FIRST ORDER RAYLEIGH QUOTIENT
C   THE DATA ARE GENERATED FROM A MULTIVARIATE LOGNORMAL DISTRIBUTION AND
C   THE TWO GROUPS DIFFER WITH RESPECT TO THEIR VARIANCE-COVARIANCE
C   STRUCTURE
C   SETS OF RELEVANT AND IRRELEVANT INPUT VARIABLES ARE UNCORRELATED

C   THE FOLLOWING (OWN) SUBROUTINES ARE REQUIRED:
C       1.  BERCRIT12
C       2.  BERCRITA0
C       3.  BERCRITA1
C       4.  BERCRITET1
C       5.  BERCRITG1
C       6.  BERCRITW
C       7.  BERCRTW1
```

```
C       8. BERCRITR
C       9. BERCRITR1
C       10. BERCRITN1
C       11. BERFOUTKFDA
C       12. BERFOUTSVM
C       13. DOENKFDA
C       14. DOENSVM
C       15. GRAMMAT
C       16. GRAMMATL
C       17. GRAMNUUT
C       18. GRAMNUUTL

C   THE FOLLOWING IMSL FUNCTIONS ARE REQUIRED:
C       1. DEXP
C       2. DLOG
C       3. DMACH
C       4. DSQRT

C   THE FOLLOWING IMSL SUBROUTINES ARE REQUIRED:
C       1. DCHFAC
C       2. DRNMVN

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=4,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NMC=500,CORR=0.0D0,SIGNOISE=20.0D0,SIGFAKTOR=10.0D0)
      PARAMETER (GAM=1.0D0/IP,GAMPAR=GAM)
      PARAMETER (NT=1000,MT=1000,NMT=NT+MT)

C   POPULATION PARAMETERS
      DIMENSION AMU1(IP),AMU2(IP)
      DIMENSION SIGMAM(IPTEL,IPTEL),RSIG(IPTEL,IPTEL)
      DIMENSION SIGMAMNOISE(NNOISE,NNOISE),RSIGNOISE(NNOISE,NNOISE)

C   X MATRICES AND Y VECTORS
      DIMENSION XM(NNPMM,IP),XMRF(IP,NNPMM)
      DIMENSION XM11(NN,IPTEL),XM21(MM,IPTEL)
      DIMENSION XM12(NN,NNOISE),XM22(MM,NNOISE)
```

```
      DIMENSION XGEM1(IP),XGEM2(IP),GEM(IP),VAR(IP)
      DIMENSION XT(NMT,IP),XT11(NT,IPTEL),XT21(MT,IPTEL)
      DIMENSION XT12(NT,NNOISE),XT22(MT,NNOISE)
      DIMENSION YV(NNPMM),YVT(NMT),IYV(NNPMM)

C     SVM RELATED QUANTITIES
      DIMENSION GRMAT(NNPMM,NNPMM),GRNUUT(NMT,NNPMM),GRMATN(NMT,NNPMM)
      DIMENSION ALPHA(NNPMM),ALPHAW(NNPMM),AL(NNPMM)

C     KFDA QUANTITIES
      DIMENSION EENM(NNPMM),EENP(NNPMM)

C     VARIOUS OTHER QUANTITIES
      DIMENSION FOUTMATK(NMC,14,5),FOUTVERK(NMC,14,5)
      DIMENSION FOUTMATS(NMC,14,5),FOUTVERS(NMC,14,5)
      DIMENSION FOUTGEM(28,5),FOUTSTD(28,5),R(IP)
      DIMENSION FREKWKIES(4,IP,5),FREKWKIESINDEP(10,IP)
      DIMENSION INDVEKVOL(IP),INDVEKP(IP),IPERMU(IP)
      DIMENSION INDVEKA0(IP),INDVEKA1(IP),INDVEKA2(IP),INDVEKA3(IP)
      DIMENSION INDVEKA0T(IP),INDVEKA1T(IP),INDVEKA2T(IP),INDVEKA3T(IP)
      DIMENSION INDVEKW0(IP),INDVEKW1(IP),INDVEKW2(IP),INDVEKW3(IP)
      DIMENSION INDVEKG0(IP),INDVEKG1(IP),INDVEKG2(IP),INDVEKG3(IP)
      DIMENSION INDVEKET0(IP),INDVEKET1(IP),INDVEKET2(IP),INDVEKET3(IP)
      DIMENSION INDVEKN0(IP),INDVEKN1(IP),INDVEKN2(IP),INDVEKN3(IP)
      DIMENSION INDVEKR0(IP),INDVEKR1(IP),INDVEKR2(IP),INDVEKR3(IP)
      DIMENSION CRITVEKA0(IP),CRITVEKA1(IP)
      DIMENSION CRITVEKA0T(IP),CRITVEKA1T(IP)
      DIMENSION CRITVEKW0(IP),CRITVEKW1(IP)
      DIMENSION CRITVEKG0(IP),CRITVEKG1(IP)
      DIMENSION CRITVEKET0(IP),CRITVEKET1(IP)
      DIMENSION CRITVEKN0(IP),CRITVEKN1(IP)
      DIMENSION CRITVEKR0(IP),CRITVEKR1(IP)

      CHARACTER*70 FILEOUT1,FILEOUT2
      FILEOUT1='foutLS.d'
      FILEOUT2='kiesLS.d'
```

```
C   WRITE FILE HEADERS
      NVERIN=IPTEL
      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
           WRITE(1,*) 'LOGNORMAL DISTRIBUTION DIFF IN SPREAD'
           WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
           WRITE(1,*) 'GAMPAR=',GAMPAR
           WRITE(1,*) 'NN=',NN,'MM=',MM,'NMC=',NMC
           WRITE(1,*) 'NT=',NT,'MT=',MT
           WRITE(1,*) 'SIGFAKTOR=',SIGFAKTOR,'SIGNOISE=',SIGNOISE
           WRITE(1,600)
      CLOSE(1)

      OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
           WRITE(1,*) 'LOGNORMAL DISTRIBUTION DIFF IN SPREAD'
           WRITE(1,*) 'IP=',IP,'IPTEL=',IPTEL,'CORR=',CORR
           WRITE(1,*) 'GAMPAR=',GAMPAR
           WRITE(1,*) 'NN=',NN,'MM=',MM,'NMC=',NMC
           WRITE(1,*) 'NT=',NT,'MT=',MT
           WRITE(1,*) 'SIGFAKTOR=',SIGFAKTOR,'SIGNOISE=',SIGNOISE
           WRITE(1,600)
      CLOSE(1)

C   ASSIGN VALUES TO THE POPULATION PARAMETERS AND OBTAIN THE COVARIANCE
C   MATRICES REQUIRED FOR GENERATING THE DATA
      E=DEXP(1.0D0)
      BLAM=DSQRT(1.0D0/(E*(E-1.0D0)))
      EP=-1.0D0*DSQRT(1.0D0/(E-1.0D0))
      DO 3 I=1,IPTEL
C          KIESREG(I)=I
           AMU1(I)=0.0D0
           AMU2(I)=0.0D0
           DO 2 J=1,IPTEL
                SIGMAM(I,J)=DLOG(1.0D0+CORR*(E-1.0D0))
2          CONTINUE
           SIGMAM(I,I)=1.0D0
3     CONTINUE
      DO 4 I=IPTEL+1,IP
```

```
        AMU1(I)=0.0D0
        AMU2(I)=0.0D0
4   CONTINUE
    DO 8 I=1,NNOISE
        DO 7 J=1,NNOISE
                SIGMAMNOISE(I,J)=0.0D0
7       CONTINUE
        SIGMAMNOISE(I,I)=1.0D0
8   CONTINUE
    TOL=1.0D2*DMACH(4)
    CALL DCHFAC(IPTEL,SIGMAM,IPTEL,TOL,IRANK,RSIG,IPTEL)
    CALL DCHFAC(NNOISE,SIGMAMNOISE,NNOISE,TOL,IRANK,RSIGNOISE,NNOISE)

C   INITIALISE VARIOUS QUANTITIES
    DO 9 I=1,NN
        YV(I)=-1.0D0
9   CONTINUE
    DO 10 I=NN+1,NNPMM
        YV(I)=1.0D0
10  CONTINUE
    SOMY=1.0D0*(MM-NN)
    SOMY2=NNPMM
    DO 11 I=1,NT
        YVT(I)=-1.0D0
11  CONTINUE
    DO 12 I=NT+1,NMT
        YVT(I)=1.0D0
12  CONTINUE
    DO 15 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
15  CONTINUE
```

```
C   INITIALISE THE TEST ERROR, STANDARD ERROR AND SELECTION FREQUENCY
C   VECTORS
    DO K=1,5
        DO I=1,28
                FOUTGEM(I,K)=0.0D0
                FOUTSTD(I,K)=0.0D0
        END DO
        DO I=1,4
                DO J=1,IP
                        FREKWKIES(I,J,K)=0.0D0
                END DO
        END DO
    END DO
    DO I=1,10
        DO J=1,IP
                FREKWKIESINDEP(I,J)=0.0D0
        END DO
    END DO

C   THE SIMULATION LOOP STARTS HERE
    DO 390 MC=1,NMC
C       WRITE(6,*) MC

C   GENERATE THE TRAINING DATA
    CALL DRNMVN(NN,IPTEL,RSIG,IPTEL,XM11,NN)
    DO 18 I=1,NN
        DO 17 J=1,IPTEL
                XM(I,J)=(BLAM*DEXP(XM11(I,J)))+AMU1(J)+EP
17      CONTINUE
18  CONTINUE
    CALL DRNMVN(NN,NNOISE,RSIGNOISE,NNOISE,XM12,NN)
    DO 20 I=1,NN
        DO 19 J=1,NNOISE
                XM(I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XM12(I,J)))+AMU1(J+IPTEL)
               &/(DSQRT(SIGNOISE))+EP)
19       CONTINUE
20  CONTINUE
```

```
      CALL DRNMVN(MM,IPTEL,RSIG,IPTEL,XM21,MM)
      DO 22 I=1,MM
          DO 21 J=1,IPTEL
                XM(NN+I,J)=DSQRT(SIGFAKTOR)*((BLAM*DEXP(XM21(I,J)))+AMU2(J)+EP)
21        CONTINUE
22    CONTINUE
      CALL DRNMVN(MM,NNOISE,RSIGNOISE,NNOISE,XM22,MM)
      DO 24 I=1,MM
          DO 23 J=1,NNOISE
          XM(NN+I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XM22(I,J)))+AMU2(J+IPTEL)
         &/(DSQRT(SIGNOISE))+EP)
23        CONTINUE
24    CONTINUE

C     STANDARDISE THE TRAINING DATA
      DO 28 J=1,IP
          S=0.0D0
          S2=0.0D0
          DO 26 I=1,NNPMM
                S=S+XM(I,J)
                2=S2+XM(I,J)**2.0D0
26        CONTINUE
          GEM(J)=S/NNPMM
          VAR(J)=(S2-(S**2.0D0)/NNPMM)/NNPMM
          DO 27 I=1,NNPMM
                XM(I,J)=(XM(I,J)-GEM(J))/DSQRT(VAR(J))
27        CONTINUE
28        CONTINUE

C     GENERATE THE TEST DATA
      CALL DRNMVN(NT,IPTEL,RSIG,IPTEL,XT11,NT)
      CALL DRNMVN(MT,IPTEL,RSIG,IPTEL,XT21,MT)
      CALL DRNMVN(NT,NNOISE,RSIGNOISE,NNOISE,XT12,NT)
      CALL DRNMVN(MT,NNOISE,RSIGNOISE,NNOISE,XT22,MT)
      DO 36 I=1,NT
          DO 35 J=1,IPTEL
                XT(I,J)=(BLAM*DEXP(XT11(I,J)))+AMU1(J)+EP
```

```
35        CONTINUE
36   CONTINUE
     DO 38 I=1,NT
     DO 37 J=1,NNOISE
     XT(I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XT12(I,J)))+AMU1(J+IPTEL)/DSQRT(SIGNOIS
    &E)+EP)
37   CONTINUE
38   CONTINUE
     DO 40 I=1,MT
          DO 39 J=1,IPTEL
                XT(NT+I,J)=DSQRT(SIGFAKTOR)*((BLAM*DEXP(XT21(I,J)))+AMU2(J)+EP)
39        CONTINUE
40   CONTINUE

     DO 42 I=1,MT
     DO 41 J=1,NNOISE
     XT(NT+I,J+IPTEL)=DSQRT(SIGNOISE)*((BLAM*DEXP(XT22(I,J)))+AMU2(J+IPTEL)/DSQRT(SIG
    &NOISE)+EP )
41   CONTINUE
42   CONTINUE

C    STANDARDISE THE TEST DATA
     DO 45 J=1,IP
          DO 44 I=1,NMT
                XT(I,J)=(XT(I,J)-GEM(J))/(DSQRT(VAR(J)))
44        CONTINUE
45   CONTINUE
     DO 50 J=1,IP
          INDVEKVOL(J)=J
50   CONTINUE

C    INITIALISE THE VARIABLE INDEX VECTORS
     DO 52 J=1,IP
          INDVEKA0(J)=J
          INDVEKA1(J)=J
          INDVEKA0T(J)=J
          INDVEKA1T(J)=J
```

```
        INDVEKW0(J)=J
        INDVEKW1(J)=J
        INDVEKG0(J)=J
        INDVEKG1(J)=J
        INDVEKET0(J)=J
        INDVEKET1(J)=J
52  CONTINUE

C   OMIT A SINGLE VARIABLE-AT-A-TIME UNTIL ONLY IPTEL=NVERIN VARIABLES REMAIN
    DO 150 NVERUIT=1,NNOISE
        NV=IP-NVERUIT+1
        NVV=NV-1

C   FIRST IMPLEMENT TECHNIQUE DEPENDENT VARIABLE SELECTION
C   CALCULATE THE ZERO ORDER ALIGNMENT CRITERION
        AMAXA=-1.1D0
        DO 70 KK=1,NV
                ITEL=0
                DO 65 J=1,NV
                        IF (J.NE.KK) THEN
                                ITEL=ITEL+1
                                INDVEKA2(ITEL)=INDVEKA0(J)
                        ENDIF
65              CONTINUE
                CALL GRAMMAT(GAM,XM,NVV,INDVEKA2,GRMAT)
                CALL BERCRITA0(GRMAT,CRITA)
                CRITVEKA0(KK)=CRITA
                IF (CRITA.GT.AMAXA) THEN
                AMAXA=CRITA
                IVERUITA=INDVEKA0(KK)
                ENDIF
70      CONTINUE
        DO 72 J=1,NV
                INDVEKA3(J)=INDVEKA0(J)
72      CONTINUE
        ITEL=0
        DO 73 J=1,NV
```

```
      IF (INDVEKA3(J).NE.IVERUITA) THEN
            ITEL=ITEL+1
            INDVEKA0(ITEL)=INDVEKA3(J)
      ENDIF
      INDVEKA0(ITEL+1)=0
73    CONTINUE

C     CALCULATE THE FIRST ORDER ALIGNMENT CRITERION
      AMINA=1.0D100
      CALL GRAMMAT(GAM,XM,NV,INDVEKA1,GRMAT)
      DO 80 KK=1,NV
            ITEL=0
            IDIFA=INDVEKA1(KK)
            CALL BERCRITA1(IDIFA,XM,GRMAT,GAM,CRITA)
            CRITVEKA1(KK)=CRITA
            IF (CRITA.LT.AMINA) THEN
                  AMINA=CRITA
                  IVERUITA=INDVEKA1(KK)
      ENDIF
80    CONTINUE
      DO 82 J=1,NV
            INDVEKA3(J)=INDVEKA1(J)
82    CONTINUE
      ITEL=0
      DO 83 J=1,NV
            IF (INDVEKA3(J).NE.IVERUITA) THEN
                  ITEL=ITEL+1
                  INDVEKA1(ITEL)=INDVEKA3(J)
            ENDIF
      INDVEKA1(ITEL+1)=0
83    CONTINUE

C     CALCULATE THE ZERO ORDER TRANSFORMED ALIGNMENT CRITERION
      AMAXA=-1.1D0
      DO 170 KK=1,NV
      ITEL=0
      DO 165 J=1,NV
```

```
            IF (J.NE.KK) THEN
                  ITEL=ITEL+1
                  INDVEKA2T(ITEL)=INDVEKA0T(J)
            ENDIF
165   CONTINUE
      CALL GRAMMATL(GAM,XM,NVV,INDVEKA2T,GRMAT)
      CALL BERCRITA0(GRMAT,CRITA)
      CRITVEKA0T(KK)=CRITA
      IF (CRITA.GT.AMAXA) THEN
            AMAXA=CRITA
            IVERUITA=INDVEKA0T(KK)
      ENDIF
170   CONTINUE
      DO 172 J=1,NV
            INDVEKA3T(J)=INDVEKA0T(J)
172   CONTINUE
      ITEL=0
      DO 173 J=1,NV
            IF (INDVEKA3T(J).NE.IVERUITA) THEN
                  ITEL=ITEL+1
                  INDVEKA0T(ITEL)=INDVEKA3T(J)
            ENDIF
            INDVEKA0T(ITEL+1)=0
173   CONTINUE

C     CALCULATE THE FIRST ORDER TRANSFORMED ALIGNMENT CRITERION
      AMINA=1.0D100
      CALL GRAMMATL(GAM,XM,NV,INDVEKA1T,GRMAT)
      DO 180 KK=1,NV
      ITEL=0
      IDIFA=INDVEKA1T(KK)
      CALL BERCRITA1(IDIFA,XM,GRMAT,GAM,CRITA)
      CRITVEKA1T(KK)=CRITA
      IF (CRITA.LT.AMINA) THEN
            AMINA=CRITA
            IVERUITA=INDVEKA1T(KK)
      ENDIF
```

```
180     CONTINUE
        DO 182 J=1,NV
              INDVEKA3T(J)=INDVEKA1T(J)
182     CONTINUE
        ITEL=0
        DO 183 J=1,NV
              IF (INDVEKA3T(J).NE.IVERUITA) THEN
                    ITEL=ITEL+1
                    INDVEKA1T(ITEL)=INDVEKA3T(J)
              ENDIF
        INDVEKA1T(ITEL+1)=0
183     CONTINUE

C    CALCULATE THE ZERO ORDER VARIATION RATIO CRITERION
        AMAXW=-1.1D0
        DO 90 KK=1,NV
              ITEL=0
              DO 85 J=1,NV
                    IF (J.NE.KK) THEN
                          ITEL=ITEL+1
                          INDVEKW2(ITEL)=INDVEKW0(J)
                    ENDIF
85            CONTINUE
        CALL GRAMMAT(GAM,XM,NVV,INDVEKW2,GRMAT)
        CALL BERCRITW0(GRMAT,CRITW)
        CRITVEKW0(KK)=CRITW
        IF (CRITW.GT.AMAXW) THEN
              AMAXW=CRITW
              IVERUITW=INDVEKW0(KK)
        ENDIF
90      CONTINUE
        DO 91 J=1,NV
              INDVEKW3(J)=INDVEKW0(J)
91      CONTINUE
        ITEL=0
        DO 92 J=1,NV
              IF (INDVEKW3(J).NE.IVERUITW) THEN
```

```
                ITEL=ITEL+1
                 INDVEKW0(ITEL)=INDVEKW3(J)
            ENDIF
        INDVEKW0(ITEL+1)=0
92      CONTINUE

C   CALCULATE THE FIRST ORDER VARIATION RATIO CRITERION
        AMINW=1.0D100
        CALL GRAMMAT(GAM,XM,NV,INDVEKW1,GRMAT)
        DO 94 KK=1,NV
            ITEL=0
            IDIFW=INDVEKW1(KK)
            CALL BERCRITW1(IDIFW,XM,GRMAT,GAM,CRITW)
            CRITVEKW1(KK)=CRITW
            IF (CRITW.LT.AMINW) THEN
                AMINW=CRITW
                IVERUITW=INDVEKW1(KK)
            ENDIF
94      CONTINUE
        DO 95 J=1,NV
            INDVEKW3(J)=INDVEKW1(J)
95      CONTINUE
        ITEL=0
        DO 96 J=1,NV
            IF (INDVEKW3(J).NE.IVERUITW) THEN
                ITEL=ITEL+1
                INDVEKW1(ITEL)=INDVEKW3(J)
            ENDIF
            INDVEKW1(ITEL+1)=0
96      CONTINUE

C   CALCULATE THE ZERO ORDER DIFFERENCES IN MEANS
        AMAXG=-1.1D0
        DO 98 KK=1,NV
            ITEL=0
            DO 97 J=1,NV
                IF (J.NE.KK) THEN
```

```
                        ITEL=ITEL+1
                        INDVEKG2(ITEL)=INDVEKG0(J)
                    ENDIF
97              CONTINUE
                CALL GRAMMAT(GAM,XM,NVV,INDVEKG2,GRMAT)
                CALL BERCRITW(GRMAT,CRITW,CRITG)
                CRITVEKG0(KK)=CRITG
                IF (CRITG.GT.AMAXG) THEN
                    AMAXG=CRITG
                    IVERUITG0=INDVEKG0(KK)
                ENDIF
98          CONTINUE
            DO 99 J=1,NV
                INDVEKG3(J)=INDVEKG0(J)
99          CONTINUE
            ITEL=0
            DO 100 J=1,NV
                IF (INDVEKG3(J).NE.IVERUITG0) THEN
                    ITEL=ITEL+1
                    INDVEKG0(ITEL)=INDVEKG3(J)
                ENDIF
                INDVEKG0(ITEL+1)=0
100         CONTINUE

C   CALCULATE THE FIRST ORDER DIFFERENCES IN MEANS
            AMING=1.0D100
            CALL GRAMMAT(GAM,XM,NV,INDVEKG1,GRMAT)
            DO 101 KK=1,NV
                ITEL=0
                IDIFG=INDVEKG1(KK)
                CALL BERCRITG1(IDIFG,XM,GRMAT,GAM,CRITG)
                CRITVEKG1(KK)=CRITG
                IF (CRITG.LT.AMING) THEN
                    AMING=CRITG
                    IVERUITG1=INDVEKG1(KK)
                ENDIF
101         CONTINUE
```

```
      DO 102 J=1,NV
            INDVEKG3(J)=INDVEKG1(J)
102   CONTINUE
      ITEL=0
      DO 103 J=1,NV
            IF (INDVEKG3(J).NE.IVERUITG1) THEN
                  ITEL=ITEL+1
                  INDVEKG1(ITEL)=INDVEKG3(J)
            ENDIF
      INDVEKG1(ITEL+1)=0
103   CONTINUE

C     CALCULATE THE ZERO ORDER DISSIMILARITY CRITERION
      AMINET=1.1D50
      DO 105 KK=1,NV
            ITEL=0
            DO 104 J=1,NV
                  IF (J.NE.KK) THEN
                        ITEL=ITEL+1
                        INDVEKET2(ITEL)=INDVEKET0(J)
                  ENDIF
104         CONTINUE
            CALL GRAMMAT(GAM,XM,NVV,INDVEKET2,GRMAT)
            CALL BERCRIT12(GRMAT,CRITET)
            CRITVEKET0(KK)=CRITET
            IF (CRITET.LT.AMINET) THEN
                  AMINET=CRITET
                  IVERUITET0=INDVEKET0(KK)
            ENDIF
105         CONTINUE
      DO 106 J=1,NV
            INDVEKET3(J)=INDVEKET0(J)
106   CONTINUE
      ITEL=0
      DO 107 J=1,NV
            IF (INDVEKET3(J).NE.IVERUITET0) THEN
                  ITEL=ITEL+1
```

```
                INDVEKET0(ITEL)=INDVEKET3(J)
            ENDIF
            INDVEKET0(ITEL+1)=0
107     CONTINUE

C   CALCULATE THE FIRST ORDER DISSIMILARITY CRITERION
        AMINET1=1.0D100
        CALL GRAMMAT(GAM,XM,NV,INDVEKET1,GRMAT)
        DO 108 KK=1,NV
            ITEL=0
            IDIF=INDVEKET1(KK)
            CALL BERCRITET1(IDIF,XM,GRMAT,GAM,CRITET1)
            CRITVEKET1(KK)=CRITET1
            IF (CRITET1.LT.AMINET1) THEN
                AMINET1=CRITET1
                IVERUITET1=INDVEKET1(KK)
            ENDIF
108     CONTINUE
        DO 109 J=1,NV
            INDVEKET3(J)=INDVEKET1(J)
109     CONTINUE
        ITEL=0
        DO 110 J=1,NV
            IF (INDVEKET3(J).NE.IVERUITET1) THEN
                ITEL=ITEL+1
                INDVEKET1(ITEL)=INDVEKET3(J)
            ENDIF
            INDVEKET1(ITEL+1)=0
110     CONTINUE
150 CONTINUE

C   UPDATE THE (TECHNIQUE INDEPENDENT) SELECTION FREQUENCIES
    DO 853 I=1,NVERIN
        II=INDVEKA0(I)
        FREKWKIESINDEP(1,II)=FREKWKIESINDEP(1,II)+1.0D0
853 CONTINUE
    DO 854 I=1,NVERIN
```

```
        II=INDVEKA1(I)
        FREKWKIESINDEP(2,II)=FREKWKIESINDEP(2,II)+1.0D0
854 CONTINUE
    DO 855 I=1,NVERIN
        II=INDVEKA0T(I)
        FREKWKIESINDEP(3,II)=FREKWKIESINDEP(3,II)+1.0D0
855 CONTINUE
    DO 856 I=1,NVERIN
        II=INDVEKA1T(I)
        FREKWKIESINDEP(4,II)=FREKWKIESINDEP(4,II)+1.0D0
856 CONTINUE
    DO 857 I=1,NVERIN
        II=INDVEKW0(I)
        FREKWKIESINDEP(5,II)=FREKWKIESINDEP(5,II)+1.0D0
857 CONTINUE
    DO 858 I=1,NVERIN
        II=INDVEKW1(I)
        FREKWKIESINDEP(6,II)=FREKWKIESINDEP(6,II)+1.0D0
858 CONTINUE
    DO 859 I=1,NVERIN
        II=INDVEKG0(I)
        FREKWKIESINDEP(7,II)=FREKWKIESINDEP(7,II)+1.0D0
859 CONTINUE
    DO 860 I=1,NVERIN
        II=INDVEKG1(I)
        FREKWKIESINDEP(8,II)=FREKWKIESINDEP(8,II)+1.0D0
860 CONTINUE
    DO 161 I=1,NVERIN
        II=INDVEKET0(I)
        FREKWKIESINDEP(9,II)=FREKWKIESINDEP(9,II)+1.0D0
161 CONTINUE
    DO 162 I=1,NVERIN
    II=INDVEKET1(I)
        FREKWKIESINDEP(10,II)=FREKWKIESINDEP(10,II)+1.0D0
162 CONTINUE
```

```
C    NOW IMPLEMENT TECHNIQUE DEPENDENT VARIABLE SELECTION
C     THE NEXT LOOP CONSIDERS DIFFERENT VALUES OF THE COST PARAMETER
     DO 1090 ICP=1,5
          IF (ICP.EQ.1) CPARS=0.00001D0
          IF (ICP.EQ.2) CPARS=0.001D0
          IF (ICP.EQ.3) CPARS=0.1D0
          IF (ICP.EQ.4) CPARS=10.0D0
          IF (ICP.EQ.5) CPARS=1000.0D0
          IF (ICP.EQ.1) CPARK=0.00001D0
          IF (ICP.EQ.2) CPARK=0.001D0
          IF (ICP.EQ.3) CPARK=0.1D0
          IF (ICP.EQ.4) CPARK=10.0D0
          IF (ICP.EQ.5) CPARK=1000.0D0

C    INITIALISE THE VARIABLE INDEX VECTORS
          DO 752 J=1,IP
                    INDVEKN0(J)=J
                    INDVEKN1(J)=J
                    INDVEKR0(J)=J
                    INDVEKR1(J)=J
752       CONTINUE

          DO 750 NVERUIT=1,NNOISE
                    NV=IP-NVERUIT+1
                    NVV=NV-1

C    CALCULATE THE ZERO ORDER RAYLEIGH QUOTIENT
                    CALL GRAMMAT(GAM,XM,NV,INDVEKR0,GRMAT)
                    CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                    AMAXR=0.000000000001D0
                    DO 760 KK=1,NV
                              ITEL=0
                              DO 759 J=1,NV
                                        IF (J.NE.KK) THEN
                                                  ITEL=ITEL+1
                                                  INDVEKR2(ITEL)=INDVEKR0(J)
                                        ENDIF
```

```
759             CONTINUE
                CALL GRAMMAT(GAM,XM,NVV,INDVEKR2,GRMAT)
                CALL BERCRITR(GRMAT,EENP,EENM,ALPHA,CPARK,CRITR)
                IF (CRITR.GT.AMAXR) THEN
                    AMAXR=CRITR
                    IVERUITR0=INDVEKR0(KK)
                ENDIF
760             CONTINUE
                DO 762 J=1,NV
                    INDVEKR3(J)=INDVEKR0(J)
762             CONTINUE
                ITEL=0
                DO 763 J=1,NV
                    IF (INDVEKR3(J).NE.IVERUITR0) THEN
                        ITEL=ITEL+1
                        INDVEKR0(ITEL)=INDVEKR3(J)
                    ENDIF
763             CONTINUE
                INDVEKR0(ITEL+1)=0

C   CALCULATE THE FIRST ORDER RAYLEIGH QUOTIENT
                AMAXR1=-1.1D0
                CALL GRAMMAT(GAM,XM,NV,INDVEKR1,GRMAT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                DO 780 KK=1,NV
                    ITEL=0
                    IDIFR=INDVEKR1(KK)
                    CALL BERCRITR1(IDIFR,XM,GRMAT,GAM,ALPHA,CRITR1)
                    CRITVEKR1(KK)=CRITR1
                    IF (CRITR1.GT.AMAXR1) THEN
                        AMAXR1=CRITR1
                        IVERUITR1=INDVEKR1(KK)
                    ENDIF
780             CONTINUE
                DO 782 J=1,NV
                    INDVEKR3(J)=INDVEKR1(J)
782             CONTINUE
```

```
                ITEL=0
                DO 783 J=1,NV
                    IF (INDVEKR3(J).NE.IVERUITR1) THEN
                        ITEL=ITEL+1
                        INDVEKR1(ITEL)=INDVEKR3(J)
                    ENDIF
783             CONTINUE
                INDVEKR1(ITEL+1)=0

C   CALCULATE THE ZERO ORDER NORM OF THE SVM WEIGHT COEFFICIENT
                AMAXN=-1.0D100
                CALL GRAMMAT(GAM,XM,NV,INDVEKN0,GRMAT)
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NV,INDVEKN0,AL,BOPT)
                DO 716 KK=1,NV
                    ITEL=0
                    DO 711 J=1,NV
                        IF (J.NE.KK) THEN
                            ITEL=ITEL+1
                            INDVEKN2(ITEL)=INDVEKN0(J)
                        ENDIF
711                 CONTINUE
                    CALL GRAMMAT(GAM,XM,NVV,INDVEKN2,GRMAT)
                    S2=0.0D0
                    DO 713 I=1,NNPMM
                        DO 712 J=1,NNPMM
                            S2=S2+AL(I)*AL(J)*YV(I)*YV(J)*GRMAT(I,J)
712                     CONTINUE
713                 CONTINUE
                    CRITN=S2
                    CRITVEKN0(KK)=CRITN
                    IF (CRITN.GT.AMAXN) THEN
                        AMAXN=CRITN
                        IVERUITN0=INDVEKN0(KK)
                    ENDIF
716             CONTINUE
                DO 717 J=1,NV
                    INDVEKN3(J)=INDVEKN0(J)
```

```
717                 CONTINUE
                    ITEL=0
                    DO 718 J=1,NV
                        IF (INDVEKN3(J).NE.IVERUITN0) THEN
                            ITEL=ITEL+1
                            INDVEKN0(ITEL)=INDVEKN3(J)
                        ENDIF
718                 CONTINUE
                    INDVEKN0(ITEL+1)=0

C   CALCULATE THE FIRST ORDER NORM OF THE SVM WEIGHT COEFFICIENT
                    AMIN=1.0D100
                    DO 720 KK=1,NV
                        ITEL=0
                        IDIFN=INDVEKN1(KK)
                        CALL BERCRITN1(IDIFN,GAM,CPARS,YV,XM,NV,INDVEKN1,CRITN)
                        CRITVEKN1(KK)=CRITN
                        IF (CRITN.LT.AMIN) THEN
                            AMIN=CRITN
                            IVERUITN1=INDVEKN1(KK)
                        ENDIF
720                 CONTINUE
                    DO 721 J=1,NV
                        INDVEKN3(J)=INDVEKN1(J)
721                 CONTINUE
                    ITEL=0
                    DO 722 J=1,NV
                        IF (INDVEKN3(J).NE.IVERUITN1) THEN
                            ITEL=ITEL+1
                            INDVEKN1(ITEL)=INDVEKN3(J)
                        ENDIF
722                 CONTINUE
                    INDVEKN1(ITEL+1)=0
750             CONTINUE

C   UPDATE THE (TECHNIQUE DEPENDENT) SELECTION FREQUENCIES
                NVERIN=IPTEL
```

```
      DO 151 I=1,NVERIN
            II=INDVEKR0(I)
            FREKWKIES(1,II,ICP)=FREKWKIES(1,II,ICP)+1.0D0
151   CONTINUE
      DO 152 I=1,NVERIN
            II=INDVEKR1(I)
            FREKWKIES(2,II,ICP)=FREKWKIES(2,II,ICP)+1.0D0
152   CONTINUE
      DO 153 I=1,NVERIN
            II=INDVEKN0(I)
            FREKWKIES(3,II,ICP)=FREKWKIES(3,II,ICP)+1.0D0
153   CONTINUE
      DO 154 I=1,NVERIN
            II=INDVEKN1(I)
            FREKWKIES(4,II,ICP)=FREKWKIES(4,II,ICP)+1.0D0
154   CONTINUE

C     CALCULATE THE TEST ERROR PERTAINING TO THE FULL MODEL
      CALL GRAMMAT(GAM,XM,IP,INDVEKVOL,GRMAT)
      CALL GRAMNUUT(GAM,XM,XT,IP,INDVEKVOL,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,1,ICP)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,IP,INDVEKVOL,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,1,ICP)=FOUT

C     CALCULATE THE TEST ERROR PERTAINING TO THE 'ORACLE'
      CALL GRAMMAT(GAM,XM,NVERIN,INDVEKVOL,GRMAT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
      CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKVOL,GRNUUT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,2,ICP)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKVOL,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,2,ICP)=FOUT
```

```
C   CALCULATE THE FOLLOWING POST-SELECTION TEST ERRORS
C   AFTER USING THE ZERO-ORDER RAYLEIGH QUOTIENT
        CALL GRAMMAT(GAM,XM,NVERIN,INDVEKR0,GRMAT)
        CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKR0,GRNUUT)
        CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
        FOUTMATK(MC,3,ICP)=FOUT

C   AFTER USING THE ZERO-ORDER NORM OF THE SVM WEIGHT VECTOR
        CALL GRAMMAT(GAM,XM,NVERIN,INDVEKN0,GRMAT)
        CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKN0,GRNUUT)
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKN0,AL,BOPT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMATS(MC,3,ICP)=FOUT

C   AFTER USING THE FIRST-ORDER RAYLEIGH QUOTIENT
        CALL GRAMMAT(GAM,XM,NVERIN,INDVEKR1,GRMAT)
        CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKR1,GRNUUT)
        CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
        FOUTMATK(MC,4,ICP)=FOUT

C   AFTER USING THE FIRST-ORDER NORM OF THE SVM WEIGHT VECTOR
        CALL GRAMMAT(GAM,XM,NVERIN,INDVEKN1,GRMAT)
        CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKN1,GRNUUT)
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKN1,AL,BOPT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMATS(MC,4,ICP)=FOUT
C   AFTER USING THE ZERO-ORDER ALIGNMENT CRITERION
        CALL GRAMMAT(GAM,XM,NVERIN,INDVEKA0,GRMAT)
        CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKA0,GRNUUT)
        CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
        CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
        FOUTMATK(MC,5,ICP)=FOUT
        CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKA0,AL,BOPT)
        CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
        FOUTMATS(MC,5,ICP)=FOUT
```

```
C    AFTER USING THE FIRST-ORDER ALIGNMENT CRITERION
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKA1,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKA1,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,6,ICP)=FOUT
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKA1,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,6,ICP)=FOUT

C    AFTER USING THE ZERO-ORDER TRANSFORMED ALIGNMENT CRITERION
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKA0T,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKA0T,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,7,ICP)=FOUT
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKA0T,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKA0T,GRNUUT)
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKA0T,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,7,ICP)=FOUT

C    AFTER USING THE FIRST-ORDER TRANSFORMED ALIGNMENT CRITERION
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKA1T,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKA1T,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,8,ICP)=FOUT
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKA1T,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKA1T,GRNUUT)
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKA1T,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,8,ICP)=FOUT

C    AFTER USING THE ZERO-ORDER VARIATION RATIO CRITERION
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKW0,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKW0,GRNUUT)
```

```
      CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,9,ICP)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKW0,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,9,ICP)=FOUT

C     AFTER USING THE FIRST-ORDER VARIATION RATIO CRITERION
      CALL GRAMMAT(GAM,XM,NVERIN,INDVEKW1,GRMAT)
      CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKW1,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,10,ICP)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKW1,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,10,ICP)=FOUT

C     AFTER USING THE ZERO-ORDER DIFFERENCES IN MEANS
      CALL GRAMMAT(GAM,XM,NVERIN,INDVEKG0,GRMAT)
      CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKG0,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,11,ICP)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKG0,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,11,ICP)=FOUT

C     AFTER USING THE ZERO-ORDER DIFFERENCES IN MEANS
      CALL GRAMMAT(GAM,XM,NVERIN,INDVEKG1,GRMAT)
      CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKG1,GRNUUT)
      CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
      CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
      FOUTMATK(MC,12,ICP)=FOUT
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKG1,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      FOUTMATS(MC,12,ICP)=FOUT
```

```
C    AFTER USING ZERO-ORDER DISSIMILARITIES
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKET0,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKET0,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,13,ICP)=FOUT
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKET0,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,13,ICP)=FOUT

C    AFTER USING FIRST-ORDER DISSIMILARITIES
                CALL GRAMMAT(GAM,XM,NVERIN,INDVEKET1,GRMAT)
                CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDVEKET1,GRNUUT)
                CALL DOENKFDA(EENM,EENP,GRMAT,CPARK,ALPHA,BOPT)
                CALL BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
                FOUTMATK(MC,14,ICP)=FOUT
                CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NVERIN,INDVEKET1,AL,BOPT)
                CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
                FOUTMATS(MC,14,ICP)=FOUT

1090    CONTINUE
THE COST PARAMETER LOOP ENDS HERE

390 CONTINUE
C    THE MONTE CARLO SIMULATION LOOP ENDS HERE

C    CALCULATE AVERAGE TEST ERRORS AND STANDARD ERRORS
     DO 394 K=1,5
        DO 392 I=1,NMC
                DO 391 J=1,14
                        FOUTGEM(J,K)=FOUTGEM(J,K)+FOUTMATK(I,J,K)
                        FOUTSTD(J,K)=FOUTSTD(J,K)+FOUTMATK(I,J,K)**2.0D0
                        FOUTGEM(14+J,K)=FOUTGEM(14+J,K)+FOUTMATS(I,J,K)
                        FOUTSTD(14+J,K)=FOUTSTD(14+J,K)+FOUTMATS(I,J,K)**2.0D0
391             CONTINUE
392     CONTINUE
```

```
      DO 393 J=1,14
            FOUTGEM(J,K)=FOUTGEM(J,K)/NMC
            FOUTSTD(J,K)=DSQRT((FOUTSTD(J,K)-NMC*(FOUTGEM(J,K)**2.0D0))/
           & (NMC*(NMC-1)))
            FOUTGEM(14+J,K)=FOUTGEM(14+J,K)/NMC
            FOUTSTD(14+J,K)=DSQRT((FOUTSTD(14+J,K)-NMC*(FOUTGEM(14+J,K)**2.0D0))/
          & (NMC*(NMC-1)))
393     CONTINUE
394  CONTINUE

C    CALCULATE SELECTION PERCENTAGES
     DO 398 I=1,10
        DO 397 J=1,IP
            FREKWKIESINDEP(I,J)=FREKWKIESINDEP(I,J)/NMC
397     CONTINUE
398  CONTINUE
     DO 402 ICP=1,5
        DO 401 I=1,4
            DO 400 J=1,IP
                FREKWKIES(I,J,ICP)=FREKWKIES(I,J,ICP)/NMC
400         CONTINUE
401     CONTINUE
402  CONTINUE

C    WRITE THE OBTAINED SELECTION PERCENTAGES TO A FILE
     OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
        WRITE(1,*) 'INDVEKA0'
        WRITE(1,600) (FREKWKIESINDEP(1,J),J=1,IP)
        WRITE(1,*) 'INDVEKA1'
        WRITE(1,600) (FREKWKIESINDEP(2,J),J=1,IP)
        WRITE(1,*) 'INDVEKA0T'
        WRITE(1,600) (FREKWKIESINDEP(3,J),J=1,IP)
        WRITE(1,*) 'INDVEKA1T'
        WRITE(1,600) (FREKWKIESINDEP(4,J),J=1,IP)
        WRITE(1,*) 'INDVEKW0'
        WRITE(1,600) (FREKWKIESINDEP(5,J),J=1,IP)
        WRITE(1,*) 'INDVEKW1'
```

```
      WRITE(1,600) (FREKWKIESINDEP(6,J),J=1,IP)
      WRITE(1,*) 'INDVEKG0'
      WRITE(1,600) (FREKWKIESINDEP(7,J),J=1,IP)
      WRITE(1,*) 'INDVEKG1'
      WRITE(1,600) (FREKWKIESINDEP(8,J),J=1,IP)
      WRITE(1,*) 'INDVEKET0'
      WRITE(1,600) (FREKWKIESINDEP(9,J),J=1,IP)
      WRITE(1,*) 'INDVEKET1'
      WRITE(1,600) (FREKWKIESINDEP(10,J),J=1,IP)
      WRITE(1,600)
      DO ICP=1,5
            IF (ICP.EQ.1) CPARS=0.1D0
            IF (ICP.EQ.2) CPARS=1.0D0
            IF (ICP.EQ.3) CPARS=10.0D0
            IF (ICP.EQ.4) CPARS=100.0D0
            IF (ICP.EQ.5) CPARS=1000.0D0
            IF (ICP.EQ.1) CPARK=0.1D0
            IF (ICP.EQ.2) CPARK=1.0D0
            IF (ICP.EQ.3) CPARK=10.0D0
            IF (ICP.EQ.4) CPARK=100.0D0
            IF (ICP.EQ.5) CPARK=1000.0D0
            WRITE(1,*) 'CPARK=',CPARK
            WRITE(1,*) 'INDVEKRO'
            WRITE(1,600) (FREKWKIES(1,J,ICP),J=1,IP)
            WRITE(1,*) 'INDVEKR1'
            WRITE(1,600) (FREKWKIES(2,J,ICP),J=1,IP)
            WRITE(1,*) 'CPARS=',CPARS
            WRITE(1,*) 'INDVEKN0'
            WRITE(1,600) (FREKWKIES(3,J,ICP),J=1,IP)
            WRITE(1,*) 'INDVEKN1'
            WRITE(1,600) (FREKWKIES(4,J,ICP),J=1,IP)
            WRITE(1,600)
      END DO
CLOSE(1)
```

```
C    WRITE THE OBTAINED AVERAGE TEST ERRORS AND STANDARD ERRORS TO A FILE
     OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
          ICP=1,5
                    IF (ICP.EQ.1) CPARS=0.1D0
                    IF (ICP.EQ.2) CPARS=1.0D0
                    IF (ICP.EQ.3) CPARS=10.0D0
                    IF (ICP.EQ.4) CPARS=100.0D0
                    IF (ICP.EQ.5) CPARS=1000.0D0
                    IF (ICP.EQ.1) CPARK=0.1D0
                    IF (ICP.EQ.2) CPARK=1.0D0
                    IF (ICP.EQ.3) CPARK=10.0D0
                    IF (ICP.EQ.4) CPARK=100.0D0
                    IF (ICP.EQ.5) CPARK=1000.0D0
                    WRITE(1,*) CPARK
                    WRITE(1,605) 'FUL','ORA','R0','R1','A0','A1','A0T','A1T','W0','W1','G0','G1','ET0','ET1'
                    WRITE(1,601) (FOUTGEM(J,ICP),J=1,14)
                    WRITE(1,601) (FOUTSTD(J,ICP),J=1,14)
                    WRITE(1,*) CPARS
                    WRITE(1,605) 'FUL','ORA','N0','N1','A0','A1','A0T','A1T','W0','W1','G0','G1','ET0','ET1'
                    WRITE(1,601) (FOUTGEM(J,ICP),J=15,28)
                    WRITE(1,601) (FOUTSTD(J,ICP),J=15,28)
                    WRITE(1,600)
          END DO
     CLOSE(1)

C    FILE FORMATS
500  FORMAT(20I3)
600  FORMAT(14(F6.4,1X))
600  FORMAT(14(F5.3,1X))
605  FORMAT(1X,A3,3X,A3,3X,4(A2,4X),2(A3,3X),4(A2,4X),2(A3,3X))
700  FORMAT(10(F20.10))

7000STOP
     END
C    END OF THE SIMULATION PROGRAM
```

C.5 *VARIABLE SELECTION FOR SUPPORT VECTOR MACHINES: A TWO-STAGE APPROACH*

The following simulation program is an example of the Fortran code used in the Monte Carlo study described in Chapter 6 of the thesis. The aim of the study was to evaluate the integrated use of cross-validation and the *NSV* -criteria together with alignments and variation ratios in SVM selection.

```
C    IN THIS PROGRAM WE CALCULATE AVERAGE TEST ERRORS (AND STANDARD ERRORS)
C    PERTAINING TO
C       C.7.1  A FULL SUPPORT VECTOR CLASSIFIER
C       C.7.2  AN SVM BASED ONLY ON THE SUBSET OF TRULY SEPARATING VARIABLES
C       C.7.3  THE POST-SELECTION SVM AFTER USING
                  ALIGNMENTS AND THE NSV CRITERION
                  VARIATION RATIOS AND THE NSV CRITERION
                  VARIATION RATIOS AND CROSS-VALIDATION

C    NOTE THAT WE MAKE USE OF A BACKWARD SELECTION STRATEGY
C    THE DATA ARE GENERATED FROM A MULTIVARIATE NORMAL DISTRIBUTION AND
C    THE TWO GROUPS DIFFER WITH RESPECT TO THEIR VARIANCE-COVARIANCE
C    STRUCTURE
C    SETS OF RELEVANT AND IRRELEVANT INPUT VARIABLES ARE UNCORRELATED

C    THE FOLLOWING (OWN) SUBROUTINES ARE REQUIRED:
C       1.  BERCRITA
C       2.  BERCRITW
C       3.  BERFOUTSVM
C       4.  CROSSVAL
C       5.  DOENSVM
C       6.  GRAMMAT
C       7.  GRAMNUUT
C       8.

C    THE FOLLOWING IMSL FUNCTIONS ARE REQUIRED:
C       1.  DSQRT
C       2.  DMACH
```

```
C     THE FOLLOWING IMSL SUBROUTINES ARE REQUIRED:
C          1.  DCHFAC
C          2.  DRNMVN

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NT=1000,MT=1000,NMT=NT+MT)
      PARAMETER (IPTEL=10,NNOISE=IP-IPTEL)
      PARAMETER (NVAR=NNPMM,NCON=2*NVAR+1,NEQ=1)
      PARAMETER (LDA=2*NVAR+1,LDH=NVAR)
      PARAMETER (NMC=1000)
      PARAMETER (CORR=0.0D0)
      PARAMETER (CPAR=NNPMM/10.0D0)
      PARAMETER (SIGFAKTOR=100.0D0)

C     DISTRIBUTION PARAMETERS
      DIMENSION AMU1(IP),AMU2(IP)
      DIMENSION SIGMAM11(IPTEL,IPTEL),RSIG11(IPTEL,IPTEL)
      DIMENSION SIGMAM12(IPTEL,IPTEL),RSIG12(IPTEL,IPTEL)
      DIMENSION SIGMAM2(NNOISE,NNOISE),RSIG2(NNOISE,NNOISE)

C     X MATRICES AND Y VECTORS
      DIMENSION XM11(NN,IPTEL),XM21(MM,IPTEL)
      DIMENSION XM12(NN,NNOISE),XM22(MM,NNOISE)
      DIMENSION XM(NNPMM,IP),YV(NNPMM)
      DIMENSION XT11(NT,IPTEL),XT21(MT,IPTEL)
      DIMENSION XT12(NT,NNOISE),XT22(MT,NNOISE)
      DIMENSION XT(NMT,IP),YVT(NMT)
      DIMENSION GEM(IP),SA(IP)

C     SVM RELATED QUANTITIES
      DIMENSION GRMAT(NNPMM,NNPMM),GRNUUT(NMT,NNPMM)
      DIMENSION AL(NNPMM),NSV(IP,2)
      DIMENSION ALPHA(NVAR)

C     SELECTION QUANTITIES
      DIMENSION SOMVEK(IP+2,2),SOM2VEK(IP+2,2)
```

```
      DIMENSION INDVEKVOL(IP)
      DIMENSION INDVEKA0(IP),INDVEKA1(IP),INDVEKA2(IP),INDVEKA3(IP)
      DIMENSION INDVEKAMAT(IP,IP)
      DIMENSION INDVEKW0(IP),INDVEKW1(IP),INDVEKW2(IP),INDVEKW3(IP)
      DIMENSION INDVEKWMAT(IP,IP)
      DIMENSION INDREEKS(IP)
      DIMENSION CRITVEKA(IP),CRITVEKW(IP),CRITVEKG(IP)
      DIMENSION CRITVEKA0(IP),CRITVEKW0(IP)
      DIMENSION AKIES(IP,IP),WKIES(IP,IP),ADIM(IP),WDIM(IP),CVDIM(IP)
      DIMENSION FOUTAVEK(IP),FOUTWVEK(IP)

      CHARACTER*70 FILEOUT1,FILEOUT2
      FILEOUT1='kiesNS.d'
      FILEOUT2='foutNS.d'

C     WRITE FILE HEADERS
      OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
          WRITE(1,*) 'NORMAALVERDELING ',NMC,' MC HERHALINGS'
          WRITE(1,*) IP,' VERANDERLIKES',NN,' STEEKPROEFGR'
          WRITE(1,*) IPTEL,' RELEVANTE VERANDERLIKES'
          WRITE(1,*) 'KORRELASIE=',CORR,' SIGFAKTOR=',SIGFAKTOR
          WRITE(1,600)
      CLOSE(1)

      OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
          WRITE(1,*) 'NORMAALVERDELING ',NMC,' MC HERHALINGS'
          WRITE(1,*) IP,' VERANDERLIKES',NN,' STEEKPROEFGR'
          WRITE(1,*) IPTEL,' RELEVANTE VERANDERLIKES'
          WRITE(1,*) 'KOR=',CORR,' SIGFAKTOR=',SIGFAKTOR
          WRITE(1,600)
      CLOSE(1)

C     SETUP THE DISTRIBUTION PARAMETERS
      DO 3 I=1,IPTEL
          AMU1(I)=0.0D0
          AMU2(I)=0.0D0
          DO 2 J=1,IPTEL
```

```
                SIGMAM11(I,J)=CORR
                SIGMAM12(I,J)=CORR
2           CONTINUE
            SIGMAM11(I,I)=1.0D0
            SIGMAM12(I,I)=1.0D0*SIGFAKTOR
3       CONTINUE
        DO 5 I=1,NNOISE
            AMU1(IPTEL+I)=0.0D0
            AMU2(IPTEL+I)=0.0D0
            DO 4 J=1,NNOISE
                SIGMAM2(I,J)=0.0D0
4           CONTINUE
            SIGMAM2(I,I)=20.0D0
5       CONTINUE
        TOL=1.0D2*DMACH(4)
        CALL DCHFAC(IPTEL,SIGMAM11,IPTEL,TOL,IRANK,RSIG11,IPTEL)
        CALL DCHFAC(IPTEL,SIGMAM12,IPTEL,TOL,IRANK,RSIG12,IPTEL)
        CALL DCHFAC(NNOISE,SIGMAM2,NNOISE,TOL,IRANK,RSIG2,NNOISE)

C       SET UP THE RESPONSE VECTOR FOR THE TRAINING AND TEST DATA
        DO 8 I=1,NN
            YV(I)=-1.0D0
8       CONTINUE
        DO 9 I=NN+1,NNPMM
            YV(I)=1.0D0
9       CONTINUE
        DO 10 I=1,NT
            YVT(I)=-1.0D0
10      CONTINUE
        DO 11 I=NT+1,NMT
            YVT(I)=1.0D0
11      CONTINUE

C       INITIALISE THE TEST ERRORS
        FOUTA=0.0D0
        FOUTA2=0.0D0
        FOUTW=0.0D0
```

```
      FOUTW2=0.0D0
      FOUTCV=0.0D0
      FOUTCV2=0.0D0

C     INITIALISE THE TEST AND STANDARD ERRORS VECTORS
      DO J=1,IP+1
         SOMVEK(J,1)=0.0D0
         SOMVEK(J,2)=0.0D0
         SOM2VEK(J,1)=0.0D0
         SOM2VEK(J,2)=0.0D0
      END DO

C     INITIALISE THE SELECTED DIMENSION AND VARIABLE INDEX VECTORS
      DO I=1,IP
         ADIM(I)=0.0D0
         WDIM(I)=0.0D0
         CVDIM(I)=0.0D0
         DO J=1,IP
                  AKIES(I,J)=0.0D0
                  WKIES(I,J)=0.0D0
         END DO
      END DO

C     THE SIMULATION LOOP STARTS HERE
      DO 390 MC=1,NMC
C        WRITE(6,*) MC

C     GENERATE THE DATA FROM GROUP 1
         CALL DRNMVN(NN,IPTEL,RSIG11,IPTEL,XM11,NN)
         DO 18 I=1,NN
                  DO 17 J=1,IPTEL
                           XM(I,J)=XM11(I,J)+AMU1(J)
17                CONTINUE
18       CONTINUE
         CALL DRNMVN(NN,NNOISE,RSIG2,NNOISE,XM12,NN)
         DO 20 I=1,NN
                  DO 19 J=1,NNOISE
```

```
                XM(I,J+IPTEL)=XM12(I,J)+AMU1(J+IPTEL)
19          CONTINUE
20      CONTINUE

C   GENERATE THE DATA FROM GROUP 2
        CALL DRNMVN(MM,IPTEL,RSIG12,IPTEL,XM21,MM)
        DO 22 I=1,MM
            DO 21 J=1,IPTEL
                XM(NN+I,J)=XM21(I,J)+AMU2(J)
21          CONTINUE
22      CONTINUE
        CALL DRNMVN(MM,NNOISE,RSIG2,NNOISE,XM22,MM)
        DO 24 I=1,MM
            DO 23 J=1,NNOISE
                XM(NN+I,J+IPTEL)=XM22(I,J)+AMU2(J+IPTEL)
23          CONTINUE
24      CONTINUE

C   GENERATE THE TEST DATA
        CALL DRNMVN(NT,IPTEL,RSIG11,IPTEL,XT11,NT)
        CALL DRNMVN(NT,NNOISE,RSIG2,NNOISE,XT12,NT)
        CALL DRNMVN(MT,IPTEL,RSIG12,IPTEL,XT21,MT)
        CALL DRNMVN(MT,NNOISE,RSIG2,NNOISE,XT22,MT)
        DO 36 I=1,NT
            DO 35 J=1,IPTEL
                XT(I,J)=XT11(I,J)+AMU1(J)
35          CONTINUE
36      CONTINUE
        DO 38 I=1,NT
            DO 37 J=1,NNOISE
                XT(I,J+IPTEL)=XT12(I,J)+AMU1(J+IPTEL)
37          CONTINUE
38      CONTINUE
        DO 40 I=1,MT
            DO 39 J=1,IPTEL
                XT(NT+I,J)=XT21(I,J)+AMU2(J)
39          CONTINUE
```

```
40      CONTINUE
        DO 42 I=1,MT
              DO 41 J=1,NNOISE
                    XT(NT+I,J+IPTEL)=XT22(I,J)+AMU2(J+IPTEL)
41            CONTINUE
42      CONTINUE

C   STANDARDISE THE TRAINING DATA
        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 47 J=1,IP
              DO 46 I=1,NNPMM
                    XM(I,J)=(XM(I,J)-GEM(J))/SA(J)
46            CONTINUE
47      CONTINUE

C   STANDARDISE THE TEST DATA
        DO 49 J=1,IP
              DO 48 I=1,NMT
                    XT(I,J)=(XT(I,J)-GEM(J))/SA(J)
48            CONTINUE
49      CONTINUE
        DO 52 J=1,IP
              INDVEKVOL(J)=J
              INDVEKA0(J)=J
              INDVEKW0(J)=J
52      CONTINUE

        GAM=1.0D0/IP
```

```
C    OMIT A SINGLE VARIABLE-AT-A-TIME
      DO 150 NVERUIT=1,IP-1
            NV=IP-NVERUIT+1
            NVV=NV-1

            DO J=1,IP
                  INDVEKAMAT(IP+1-NVERUIT,J)=INDVEKA0(J)
                  INDVEKWMAT(IP+1-NVERUIT,J)=INDVEKW0(J)
            END DO

C    CALCULATE THE ALIGNMENT-0 CRITERION
            AMAXA=-1.1D0
            DO 70 KK=1,NV
                  ITEL=0
                  DO 65 J=1,NV
                        IF (J.NE.KK) THEN
                              ITEL=ITEL+1
                              INDVEKA2(ITEL)=INDVEKA0(J)
                        ENDIF
65                CONTINUE
            CALL GRAMMAT(GAM,XM,NVV,INDVEKA2,GRMAT)
            CALL BERCRITA(GRMAT,CRITA)
            CRITVEKA0(KK)=CRITA
            IF (CRITA.GT.AMAXA) THEN
                  AMAXA=CRITA
                  IVERUITA=INDVEKA0(KK)
            ENDIF
70          CONTINUE
            DO 72 J=1,NV
                  INDVEKA3(J)=INDVEKA0(J)
72          CONTINUE
            ITEL=0
            DO 73 J=1,NV
                  IF (INDVEKA3(J).NE.IVERUITA) THEN
                        ITEL=ITEL+1
                        INDVEKA0(ITEL)=INDVEKA3(J)
                  ENDIF
```

```
                INDVEKA0(ITEL+1)=0
73          CONTINUE

C   CALCULATE THE VARIATION RATIO CRITERION
            AMAXW=-1.1D0
            DO 90 KK=1,NV
                ITEL=0
                DO 85 J=1,NV
                    IF (J.NE.KK) THEN
                        ITEL=ITEL+1
                        INDVEKW2(ITEL)=INDVEKW0(J)
                    ENDIF
85              CONTINUE
                CALL GRAMMAT(GAM,XM,NVV,INDVEKW2,GRMAT)
                CALL BERCRITW(GRMAT,CRITW,CRITDUMMY)
                CRITVEKW0(KK)=CRITW
                IF (CRITW.GT.AMAXW) THEN
                    AMAXW=CRITW
                    IVERUITW=INDVEKW0(KK)
                ENDIF
90          CONTINUE
            DO 91 J=1,NV
                INDVEKW3(J)=INDVEKW0(J)
91          CONTINUE
            ITEL=0
            DO 92 J=1,NV
                IF (INDVEKW3(J).NE.IVERUITW) THEN
                    ITEL=ITEL+1
                    INDVEKW0(ITEL)=INDVEKW3(J)
                ENDIF
                INDVEKW0(ITEL+1)=0
92          CONTINUE
150     CONTINUE
C   END OF THE ONE VARIABLE AT-A-TIME LOOP
```

```
C   UPDATE THE SELECTED INPUT VARIABLE INDEX VECTORS
      DO J=1,IP
            INDVEKAMAT(1,J)=INDVEKA0(J)
            INDVEKWMAT(1,J)=INDVEKW0(J)
      END DO
      DO I=1,IP
            DO J=1,I
                  AKIES(I,INDVEKAMAT(I,J))=AKIES(I,INDVEKAMAT(I,J))+1.0D0
                  WKIES(I,INDVEKWMAT(I,J))=WKIES(I,INDVEKWMAT(I,J))+1.0D0
            END DO
      END DO

C   NOW CALCULATE THE ERROR RATES FOR THE SEQUENCES OF SELECTED INPUT
C   VARIABLES AND THE NUMBER OF SUPPORT VECTORS CORRESPONDING TO EACH
      DO J=1,IP
            NSV(J,1)=0
            NSV(J,2)=0
      END DO
      DO J=1,IP
            NVERIN=J
            GAM=1.0D0/J
            DO I=1,J
                  INDREEKS(I)=INDVEKAMAT(J,I)
            END DO
            CALL GRAMMAT(GAM,XM,NVERIN,INDREEKS,GRMAT)
            CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDREEKS,GRNUUT)
            CALL DOENSVM(YV,XM,GRMAT,CPAR,NVERIN,INDREEKS,AL,BOPT)
            DO I=1,NNPMM
                  IF(AL(I).GT.0.1D-20) NSV(J,1)=NSV(J,1)+1
            END DO
            CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
            FOUTAVEK(J)=FOUT
            SOMVEK(J,1)=SOMVEK(J,1)+FOUT
            SOM2VEK(J,1)=SOM2VEK(J,1)+FOUT**2.0D0
            DO I=1,J
                  INDREEKS(I)=INDVEKWMAT(J,I)
            END DO
```

```
            CALL GRAMMAT(GAM,XM,NVERIN,INDREEKS,GRMAT)
            CALL GRAMNUUT(GAM,XM,XT,NVERIN,INDREEKS,GRNUUT)
            CALL DOENSVM(YV,XM,GRMAT,CPAR,NVERIN,INDREEKS,AL,BOPT)
            DO I=1,NNPMM
                  IF(AL(I).GT.0.1D-20) NSV(J,2)=NSV(J,2)+1
            END DO
            CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
            FOUTWVEK(J)=FOUT
            SOMVEK(J,2)=SOMVEK(J,2)+FOUT
            SOM2VEK(J,2)=SOM2VEK(J,2)+FOUT**2.0D0
      END DO

C     USE CROSS-VALIDATION TO DETERMINE THE NUMBER OF INPUT VARIABLES
      CALL CROSSVAL(XM,YV,INDVEKWMAT,IBESTDIM)
      IAMIN=NNPMM+1
      IWMIN=NNPMM+1

C     USE THE NUMBER OF SUPPORT VECTORS (CALCULATED ABOVE) TO DETERMINE THE
C     NUMBER OF INPUT VARIABLES (BASED ON ALIGNMENTS AND VARIATION RATIOS)
      DO J=1,IP
            IF(NSV(J,1).LT.IAMIN) THEN
                  IAMIN=NSV(J,1)
                  IADIM=J
            END IF
            IF(NSV(J,2).LT.IWMIN) THEN
                  IWMIN=NSV(J,2)
                  IWDIM=J
            END IF
      END DO

C     UPDATE THE NUMBER OF INPUT VARIABLES SELECTED
      ADIM(IADIM)=ADIM(IADIM)+1.0D0
      WDIM(IWDIM)=WDIM(IWDIM)+1.0D0
      CVDIM(IBESTDIM)=CVDIM(IBESTDIM)+1.0D0

C     UPDATE THE TEST ERROR AND STANDARD ERROR VECTORS
      FOUTA=FOUTA+FOUTAVEK(IADIM)
```

```
      FOUTA2=FOUTA2+FOUTAVEK(IADIM)**2.0D0
      FOUTW=FOUTW+FOUTWVEK(IWDIM)
      FOUTW2=FOUTW2+FOUTWVEK(IWDIM)**2.0D0
      FOUTCV=FOUTCV+FOUTWVEK(IBESTDIM)
      FOUTCV2=FOUTCV2+FOUTWVEK(IBESTDIM)**2.0D0

C     CALCULATE THE TEST ERROR WHEN ONLY THE SUBSET OF SEPARATING
C     VARIABLES IS USED
      GAM=1.0D0/IPTEL
      CALL GRAMMAT(GAM,XM,IPTEL,INDVEKVOL,GRMAT)
      CALL GRAMNUUT(GAM,XM,XT,IPTEL,INDVEKVOL,GRNUUT)
      CALL DOENSVM(YV,XM,GRMAT,CPAR,IPTEL,INDVEKVOL,AL,BOPT)
      CALL BERFOUTSVM(YV,GRNUUT,YVT,AL,BOPT,FOUT)
      SOMVEK(IP+1,1)=SOMVEK(IP+1,1)+FOUT
      SOM2VEK(IP+1,1)=SOM2VEK(IP+1,1)+FOUT**2.0D0
390 CONTINUE
C     THE MONTE CARLO SIMULATION LOOP ENDS HERE

C     CALCULATE THE AVERAGE TEST ERRORS AND STANDARD ERRORS
      FOUTA=FOUTA/NMC
      FOUTW=FOUTW/NMC
      FOUTCV=FOUTCV/NMC
      FOUTA2=DSQRT((FOUTA2/NMC-FOUTA**2.0D0)/NMC)
      FOUTW2=DSQRT((FOUTW2/NMC-FOUTW**2.0D0)/NMC)
      FOUTCV2=DSQRT((FOUTCV2/NMC-FOUTCV**2.0D0)/NMC)
      DO J=1,IP+1
         SOMVEK(J,1)=SOMVEK(J,1)/NMC
         SOMVEK(J,2)=SOMVEK(J,2)/NMC
         SOM2VEK(J,1)=DSQRT((SOM2VEK(J,1)/NMC-SOMVEK(J,1)**2.0D0)/NMC)
         SOM2VEK(J,2)=DSQRT((SOM2VEK(J,2)/NMC-SOMVEK(J,2)**2.0D0)/NMC)
      END DO

C     CALCULATE THE AVERAGE NUMBER OF SELECTED INPUT VARIABLES AND SELECTION
C     PERCENTAGES
      DO I=1,IP
         ADIM(I)=ADIM(I)/NMC
         WDIM(I)=WDIM(I)/NMC
```

```
        CVDIM(I)=CVDIM(I)/NMC
        DO J=1,IP
                AKIES(I,J)=AKIES(I,J)/NMC
                WKIES(I,J)=WKIES(I,J)/NMC
        END DO
    END DO

C   WRITE THE OBTAINED AVERAGE TEST ERRORS AND STANDARD ERRORS TO A FILE
    OPEN(1,FILE=FILEOUT2,ACCESS='APPEND')
        WRITE(1,600) FOUTA,FOUTA2
        WRITE(1,600) (SOMVEK(J,1),J=1,IP+1)
        WRITE(1,600) (SOM2VEK(J,1),J=1,IP+1)
        WRITE(1,*)
        WRITE(1,600) FOUTW,FOUTW2
        WRITE(1,600) (SOMVEK(J,2),J=1,IP+1)
        WRITE(1,600) (SOM2VEK(J,2),J=1,IP+1)
        WRITE(1,*)
        WRITE(1,600) FOUTCV,FOUTCV2
        WRITE(1,600) (SOMVEK(J,2),J=1,IP+1)
        WRITE(1,600) (SOM2VEK(J,2),J=1,IP+1)
        WRITE(1,*)
    CLOSE(1)

C   WRITE THE AVERAGE NUMBER OF INPUT VARIABLES SELECTED AND THE OBTAINED
C    SELECTION PERCENTAGES TO A FILE
    OPEN(1,FILE=FILEOUT1,ACCESS='APPEND')
        DO I=1,IP
                WRITE(1,600) (AKIES(I,J),J=1,IP)
        END DO
        WRITE(1,*)
        WRITE(1,600) (ADIM(I),I=1,IP)
        WRITE(1,*)
        DO I=1,IP
                WRITE(1,600) (WKIES(I,J),J=1,IP)
        END DO
        WRITE(1,*)
        WRITE(1,600) (WDIM(I),I=1,IP)
```

```
        WRITE(1,*)
        DO I=1,IP
                WRITE(1,600) (WKIES(I,J),J=1,IP)
        END DO
        WRITE(1,*)
        WRITE(1,600) (CVDIM(I),I=1,IP)
        WRITE(1,*)
    CLOSE(1)

C   FILE FORMATS
500 FORMAT(20I3)
600 FORMAT(15(F10.5,1X))
605 FORMAT(I6,2X,4(F12.5,1X))
700 FORMAT(10(F20.10))

STOP
END
C   THE SIMULATION PROGRAM ENDS HERE
```

C.6 *FUNCTIONS AND SUBROUTINES*

FUNCTION ALIGNMENT
SUBROUTINE BERFOUTKFDA
SUBROUTINE BERFOUTLDA
SUBROUTINE BERFOUTSVM
SUBROUTINE DOENSVM
SUBROUTINE GEMVARV
SUBROUTINE GRAMMAT
SUBROUTINE GRAMNUUT
SUBROUTINE FCN
SUBROUTINE PREIMAGE
SUBROUTINE RANFOR
SUBROUTINE BUILDTREE
SUBROUTINE FINDBESTSPLIT
SUBROUTINE TESTREEBAG
SUBROUTINE PERMOBMR
SUBROUTINE ZERV
SUBROUTINE ZERVR
SUBROUTINE ZERM
SUBROUTINE PACKLB
SUBROUTINE UNPACKLB
SUBROUTINE QUICKSORT
SUBROUTINE PERM1
SUBROUTINE RAND1
SUBROUTINE LRND
SUBROUTINE LRND1
FUNCTION RNORM
SUBROUTINE BERCRITW
SUBROUTINE BERCRIT12
SUBROUTINE BERCRITCC
SUBROUTINE BERCRITR
FUNCTION ALIGNMENTTRANS
SUBROUTINE GRAMMATL
SUBROUTINE BERCRITA0
SUBROUTINE BERCRITA1
SUBROUTINE BERCRITW0

```
SUBROUTINE BERCRITW1
SUBROUTINE BERCRITW
SUBROUTINE BERCRITG1
SUBROUTINE BERCRIT12
SUBROUTINE BERCRITET1
SUBROUTINE BERCRITN1
SUBROUTINE BERCRITR
SUBROUTINE BERCRITR1
SUBROUTINE BERCRITA
SUBROUTINE BERCRITW
SUBROUTINE CROSSVAL
SUBROUTINE DEELGRAMNUUT
SUBROUTINE DOENSVMIN
SUBROUTINE BERFOUTSVMUIT

FUNCTION ALIGNMENT(J,XM)
C   CALCULATES THE ALIGNMENT SELECTION CRITERION TO EVALUATE THE
C   IMPORTANCE OF INPUT VARIABLE J
C   INPUT:   THE INDEX (J) OF THE INPUT VARIABLE TO EVALUATE
C                THE TRAINING INPUT PATTERNS
C   OUTPUT: THE ALIGNMENT VALUE
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,IPTEL=1,NN=100,MM=100,NNPMM=NN+MM,IPP1=IP+1)
    DIMENSION XM(NNPMM,IP)
    S1=0.0D0
    S2=0.0D0
    S3=0.0D0
    S4=0.0D0
    DO 5 I1=1,NN
        DO 4 I2=1,NN
                S=(XM(I1,J)-XM(I2,J))*(XM(I1,J)-XM(I2,J))
                S1=S1+DEXP(-S)
                S4=S4+DEXP(-2*S)
4       CONTINUE
5   CONTINUE
    DO 10 I1=NN+1,NNPMM
        DO 9 I2=NN+1,NNPMM
```

```
                S=(XM(I1,J)-XM(I2,J))*(XM(I1,J)-XM(I2,J))
                S2=S2+DEXP(-S)
                S4=S4+DEXP(-2*S)
9       CONTINUE
10  CONTINUE
    DO 15 I1=1,NN
        DO 14 I2=NN+1,NNPMM
                S=(XM(I1,J)-XM(I2,J))*(XM(I1,J)-XM(I2,J))
                S3=S3+DEXP(-S)
                S4=S4+2.0D0*DEXP(-2*S)
14      CONTINUE
15  CONTINUE
    S3=2.0D0*S3
    ALIGNMENT=(S1+S2-S3)/((1.0D0*NNPMM)*DSQRT(S4))
RETURN
END
C   END OF THE ALIGNMENT FUNCTION

SUBROUTINE BERFOUTKFDA(GRNUUT,YVT,ALPHA,BOPT,FOUT)
C   CALCULATES A KFDA TEST ERROR
C   INPUT:  KERNEL MATRIX ON THE TEST INPUT PATTERNS
            THE TEST LABELS
            THE FITTED KFD ALPHA VECTOR
C   OUTPUT: THE NUMBER OF INCORRECT CLASSIFICATIONS ON THE TEST DATA SET
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,IPTEL=1,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
    PARAMETER (NT=1000,MT=1000,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
C    END OF THE BERFOUTKFDA SUBROUTINE

SUBROUTINE BERFOUTLDA(XM,XT,YVT,NVER,JIND,FOUT)
C    CALCULATES AN LDA TEST ERROR
C    INPUT:    THE TEST AND TRAINING INPUT PATTERNS
C              THE TEST LABELS
C              THE NO. OF AND INDICES OF THE SUBSET OF INPUT VARIABLES
C    USES:     GEMVARV AND DLINDS
C    OUTPUT: THE NUMBER OF INCORRECT CLASSIFICATIONS ON THE TEST DATA SET
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=10,IPTEL=1,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
     PARAMETER (NT=1000,MT=1000,NMT=NT+MT)
     DIMENSION XM(NNPMM,IP),XT(NMT,IP),YVT(NMT)
     DIMENSION XGEM1(IP),XGEM2(IP)
     DIMENSION SINV(IP,IP),SMAT(IP,IP),SMAT1(IP,IP)
     DIMENSION XV(NVER),JIND(IP)

     CALL GEMVARV(XM,SMAT,XGEM1,XGEM2)

     DO I=1,NVER
        DO J=1,NVER
              SMAT1(I,J)=SMAT(JIND(I),JIND(J))
        END DO
     END DO

     CALL DLINDS(NVER,SMAT1,IP,SINV,IP)

     FOUT=0.0D0
     DO 1151 I=1,NMT
        DO 1148 J=1,NVER
              XV(J)=XT(I,JIND(J))
1148    CONTINUE
        SOM1=0.0D0
```

```
      SOM2=0.0D0
      DO 1150 I1=1,NVER
            DO 1149 I2=1,NVER
                  V1=XV(I1)-XGEM1(JIND(I1))
                  V2=XV(I2)-XGEM1(JIND(I2))
                  SOM1=SOM1+V1*SINV(I1,I2)*V2
                  V1=XV(I1)-XGEM2(JIND(I1))
                  V2=XV(I2)-XGEM2(JIND(I2))
                  SOM2=SOM2+V1*SINV(I1,I2)*V2
1149        CONTINUE
1150  CONTINUE
      DIST1=SOM1-DLOG((1.0D0*NN)/(1.0D0*MM))
      DIST2=SOM2
      IF (DIST1.LT.DIST2) GROEP=-1.0D0
      IF (DIST1.GE.DIST2) GROEP=1.0D0
      IF (DABS((YVT(I)-GROEP)).GT.0.1D0) FOUT=FOUT+1.0D0
1151CONTINUE
    FOUT=FOUT/NMT
RETURN
END
C   END OF THE BERFOUTLDA SUBROUTINE

SUBROUTINE BERFOUTSVM(YV,GRNUUT,YVT,ALPHA,BOPT,FOUT)
C   CALCULATES AN SVM TEST ERROR
C   INPUT:    KERNEL MATRIX ON THE TEST INPUT PATTERNS
C             THE TEST LABELS
C             THE FITTED SVM ALPHA VECTOR
C   OUTPUT: THE NUMBER OF INCORRECT CLASSIFICATIONS ON THE TEST DATA SET
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,IPTEL=1,NN=100,MM=100,NNPMM=NN+MM)
    PARAMETER (NT=1000,MT=1000,NMT=NT+MT)
    DIMENSION GRNUUT(NMT,NNPMM),ALPHA(NNPMM),YVT(NMT)
    DIMENSION YV(NNPMM)
    FOUT=0.0D0
    DO 10 I=1,NMT
      TOETS=1.0D0
      S=BOPT
```

```
        DO 5 J=1,NNPMM
                S=S+ALPHA(J)*YV(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
C   END OF THE BERFOUTSVM SUBROUTINE

SUBROUTINE DOENKFDA(EENM,EENP,GRMAT,CPAR,ALPHA,BOPT)
C   OBTAINS THE KFD ALPHA VECTOR  (USING ALL COMPONENTS, OR ONLY A SUBSET
C   OF COMPONENTS OF THE TRAINING INPUTS)
C   INPUT:      THE KERNEL MATRIX ON THE INPUT PATTERNS (ALL COMPONENTS OR
C               ONLY A SUBSET)
C   USES:       DLSASF
C   OUTPUT: THE KFD ALPHA VECTOR
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,IPTEL=1,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
    PARAMETER (LDH=NNPMM)
    DIMENSION GRMAT(NNPMM,NNPMM)
    DIMENSION EENP(NNPMM),EENM(NNPMM)
    DIMENSION B(NNPMM),H(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
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
C     END OF THE DOENKFDA SUBROUTINE

SUBROUTINE DOENSVM(YV,XM,GRMAT,CPAR,GAM,NVER,INDVEK,AL,BOPT)
C     OBTAINS THE SVM ALPHA VECTOR (USING ALL COMPONENTS, OR ONLY A SUBSET OF
C     COMPONENTS OF THE TRAINING INPUT PATTERNS)
C     INPUT:    A VALUE FOR THE KERNEL HYPERPARAMETER
C               THE KERNEL MATRIX
C               THE VECTOR OF INDICES TO THE COMPONENTS IN INPUT PATTERNS TO BE
C               USED
C     USES:     DQPROG, DSVRGP
C     OUTPUT: THE KERNEL MATRIX
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=1,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NVAR=NNPMM,NCON=2*NVAR+1,NEQ=1,LDA=NCON,LDH=NVAR)
      DIMENSION XM(NNPMM,IP),YV(NNPMM),GRMAT(NNPMM,NNPMM)
      DIMENSION XV(IP),XVV(IP)
      DIMENSION A(NCON,NVAR),B(NCON),G(NVAR),H(NVAR,NVAR)
      DIMENSION SOL(NVAR),ALAM(NVAR),AL(NNPMM)
      DIMENSION FW(NNPMM),FWR(NNPMM),YVR(NNPMM)
```

```
      DIMENSION IACT(NVAR),IPERM(NNPMM)
      DIMENSION INDVEK(IP)

      EP=1.0D-8
      DO 28 I=1,NVAR
         A(1,I)=YV(I)
28    CONTINUE
      DO 30 I=1,NVAR
         DO 29 J=1,NVAR
                  A(I+1,J)=0.0D0
                  A(NVAR+I+1,J)=0.0D0
29       CONTINUE
         A(I+1,I)=1.0D0
         A(NVAR+I+1,I)=-1.0D0
30    CONTINUE
      B(1)=0.0D0
      DO 35 I=1,NVAR
         B(I+1)=0.0D0
         B(NVAR+I+1)=-1.0D0*CPAR
35    CONTINUE
      DO 36 I=1,NVAR
         G(I)=-1.0D0
36    CONTINUE
      DO 40 I=1,NVAR
         DO 39 J=1,NVAR
                  H(I,J)=YV(I)*YV(J)*GRMAT(I,J)
39       CONTINUE
40    CONTINUE
      CALL DQPROG(NVAR,NCON,NEQ,A,LDA,B,G,H,LDH,DIAG,SOL,NACT,
     & IACT,ALAM)
      DO 45 I=1,NVAR
         IF (DABS(SOL(I)).LT.EP) SOL(I)=0.0D0
         IF (DABS(SOL(I)-CPAR).LT.EP) SOL(I)=CPAR
         AL(I)=SOL(I)
45    CONTINUE
```

```
C     DETERMINE THE Y INTERCEPT VALUE
      DO 200 J=1,NNPMM
          IPERM(J)=J
          S=0.0D0
          DO 199 I=1,NNPMM
                  S=S+AL(I)*YV(I)*GRMAT(I,J)
199       CONTINUE
          FW(J)=S
200   CONTINUE
      CALL DSVRGP(NNPMM,FW,FWR,IPERM)
      DO 205 I=1,NNPMM
          YVR(I)=YV(IPERM(I))
205   CONTINUE
      BPAR=-FWR(1)+1.0D0
      BOPT=BPAR
      NFOUTE=NN
      NFOUTEOPT=NFOUTE
      NTEL=0
210    NTEL=NTEL+1
      BPAR=-(FWR(NTEL)+FWR(NTEL+1))/2.0D0
      IF (YVR(NTEL).LE.0.0D0) NFOUTE=NFOUTE-1
      IF (YVR(NTEL).GT.0.0D0) NFOUTE=NFOUTE+1
      IF (NFOUTE.LT.NFOUTEOPT) THEN
          NFOUTEOPT=NFOUTE
          BOPT=BPAR
      ENDIF
      IF (NTEL.LE.NNPMM-2) GOTO 210
RETURN
END
END OF THE DOENSVM SUBROUTINE

SUBROUTINE GEMVARV(XX,SMAT,XGEM1,XGEM2)
C     REQUIRED IN BERFOUTLDA
C     OBTAINS
C     INPUT:
C     USES:      DCORVC,
C     OUTPUT:    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      PARAMETER (IP=10,IPTEL=1,NNOISE=IP-IPTEL,NN=100,MM=100,NNPMM=NN+MM)
      DIMENSION XX(NNPMM,IP),XX1(NN,IP),XX2(MM,IP)
      DIMENSION XGEM1(IP),XGEM2(IP)
      DIMENSION SMAT(IP,IP),S1(IP,IP),S2(IP,IP)
      EXTERNAL DCORVC,DLINDS
      DO 10 I=1,NN
          DO 5 J=1,IP
                XX1(I,J)=XX(I,J)
5         CONTINUE
10    CONTINUE
      DO 20 I=1,MM
          DO 15 J=1,IP
                XX2(I,J)=XX(NN+I,J)
15        CONTINUE
20    CONTINUE
      IDO=0
      NROW=NN
      NVAR=IP
      LDX=NN
      IFRQ=0
      IWT=0
      MOPT=0
      ICOPT=0
      LDCOV=IP
      LDINCD=1
      CALL DCORVC(IDO,NROW,NVAR,XX1,LDX,IFRQ,IWT,MOPT,
     &          ICOPT,XGEM1,S1,LDCOV,INCD,LDINCD,NOBS,
     &          NMISS,SUMWT)
      NROW=MM
      LDX=MM
      CALL DCORVC(IDO,NROW,NVAR,XX2,LDX,IFRQ,IWT,MOPT,
     &          ICOPT,XGEM2,S2,LDCOV,INCD,LDINCD,NOBS,
     &          NMISS,SUMWT)
      NOEM=NN+MM-2
      DO 30 I=1,IP
          DO 25 J=1,IP
                SMAT(I,J)=((NN-1)*S1(I,J)+(MM-1)*S2(I,J))/NOEM
```

```
25      CONTINUE
30   CONTINUE
RETURN
END
```
END OF THE GEMVARV SUBROUTINE

```
SUBROUTINE GRAMMAT(GAMPAR,XM,NV,INDVEK,GRMAT)
```
C CALCULATES THE KERNEL MATRIX ON THE TRAINING INPUT PATTERNS (USING ALL COMPONENTS, OR ONLY A SUBSET OF COMPONENTS)
C INPUT: A VALUE FOR THE KERNEL HYPERPARAMETER
C THE TRAINING INPUT PATTERNS
C THE VECTOR OF INDICES TO THE COMPONENTS IN INPUT PATTERNS TO BE
C USED
C OUTPUT: THE KERNEL MATRIX
```
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,IPTEL=1,NN=100,MM=100,NNPMM=NN+MM,IPP1=IP+1)
    DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
    DIMENSION INDVEK(IP)
    DO 10 I=1,NNPMM-1
        GRMAT(I,I)=1.0D0
        DO 5 J=I+1,NNPMM
                S=0.0D0
                DO 3 K=1,NV
                        KK=INDVEK(K)
                        S=S+(XM(I,KK)-XM(J,KK))*(XM(I,KK)-XM(J,KK))
3               CONTINUE
                GRMAT(I,J)=DEXP(-GAMPAR*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
```
C END OF THE GRAMMAT SUBROUTINE

```
SUBROUTINE GRAMNUUT(GAMPAR,XM,XT,NV,INDVEK,GRNUUT)
C   CALCULATES THE KERNEL MATRIX ENTRIES BETWEEN TRAINING AND TEST
C   PATTERNS (USING ALL COMPONENTS, OR ONLY A SUBSET OF COMPONENTS)
C   INPUT:   A VALUE FOR THE KERNEL HYPERPARAMETER
C            THE TRAINING AND TEST INPUT PATTERNS
C            THE VECTOR OF INDICES TO THE COMPONENTS IN INPUT PATTERNS TO BE
C            USED
C   OUTPUT: THE KERNEL MATRIX
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=1,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NT=1000,MT=1000,NMT=NT+MT)
      DIMENSION XM(NNPMM,IP),GRNUUT(NMT,NNPMM)
      DIMENSION XT(NMT,IP)
      DIMENSION INDVEK(IP)
      DO 10 I=1,NMT
         DO 5 J=1,NNPMM
                  S=0.0D0
                  DO 3 K=1,NV
                           KK=INDVEK(K)
                           S=S+(XT(I,KK)-XM(J,KK))*(XT(I,KK)-XM(J,KK))
3                 CONTINUE
                  GRNUUT(I,J)=DEXP(-GAMPAR*S)
5        CONTINUE
10    CONTINUE
RETURN
END
C   END OF THE GRAMNUUT SUBROUTINE

SUBROUTINE FCN(NDIM,X,F)
C   REQUIRED IN THE PREIMAGE SUBROUTINE
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=4,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (GAMPAR=1.0D0/IP)
      DIMENSION X(IP),X1(IP),X2(IP)
      COMMON /PREIM/ ALPHAC(NNPMM),XMC(NNPMM,IP),GAMPARC
      S=0.0D0
      DO 20 I=1,NNPMM
```

```
      DO 5 J=1,NDIM
            X1(J)=XMC(I,J)
            X2(J)=X(J)
5     CONTINUE
      S1=0.0D0
      DO 10 K=1,NDIM
            S1=S1+(X1(K)-X2(K))*(X1(K)-X2(K))
10    CONTINUE
      TERM=ALPHAC(I)*DEXP(-GAMPARC*S1)
      S=S+TERM
20  CONTINUE
    F=-1.0D0*S*S
RETURN
END
C   END OF THE FCN SUBROUTINE

SUBROUTINE PREIMAGE(XM,ALPHA,XPRE)
C   FINDS AN APPROXIMATE PRE-IMAGE FOR A GIVEN LINEAR COMBINATION OF THE
C   DATA POINTS IN FEATURE SPACE
C   THE LINEAR COMBINATION IS SPECIFIED BY SPECIFYING ITS COEFFICIENTS IN THE
C   VECTOR ALPHA.
C   THE IMSL ROUTINE DUMINF IS USED TO PERFORM THE REQUIRED OPTIMISATION.
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    EXTERNAL FCN
    PARAMETER (IP=10,IPTEL=4,NN=100,MM=100,NNPMM=NN+MM)
    PARAMETER (GAMPAR=1.0D0/IP)
    DIMENSION XM(NNPMM,IP),ALPHA(NNPMM)
    DIMENSION XGUESS(IP),XSCALE(IP),RPARAM(7),XPRE(IP)
    DIMENSION IPARAM(7)
    COMMON /PREIM/ ALPHAC(NNPMM),XMC(NNPMM,IP),GAMPARC

    DO 5 I=1,NNPMM
      ALPHAC(I)=ALPHA(I)
      DO 4 J=1,IP
            XMC(I,J)=XM(I,J)
4     CONTINUE
5   CONTINUE
```

```
      GAMPARC=GAMPAR
      NDIM=IP
      DO 8 J=1,IP
          XGUESS(J)=XM(1,J)
          XSCALE(J)=1.0D0
8     CONTINUE
      FSCALE=1.0D0
      IPARAM(1)=0
      CALL DUMINF(FCN,NDIM,XGUESS,XSCALE,FSCALE,IPARAM,RPARAM,
     & XPRE,FVALUE)
RETURN
END
```

END OF THE PREIMAGE SUBROUTINE

```
SUBROUTINE RANFOR(X,Y,ZSCORE)
C     USES:      DUMINF, FCN
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(MDIM=10,NSAMPLE=200,NTEST=1,NTHSIZE=5,
1     NRNODES=2*(NSAMPLE/NTHSIZE)+1,JPRINT=0,JBT=250,MTRY=MDIM/3,
1     IMP=1,NIMP=IMP*NSAMPLE,MIMP=IMP*MDIM,
1     TH=0.5D0,IMPPRINT=1)

      DIMENSION X(MDIM,NSAMPLE),Y(NSAMPLE),YB(NSAMPLE),
1     RSNODECOST(NRNODES),ZSCORE(MDIM),
1     RSTREECOST(0:NRNODES),BESTCRIT(NRNODES),SD(MDIM),WTS(NSAMPLE),
1     V(NSAMPLE),UT(NSAMPLE),XT(NSAMPLE),XB(MDIM,NSAMPLE),
1     ERRIMP(MIMP),  YTR(NSAMPLE),YPTR(NSAMPLE),YL(NSAMPLE),XA(3*MDIM),
1     AVNODE(NRNODES),UTR(NSAMPLE),PREDIMP(NIMP,MIMP),ZA(MDIM),
1     TGINI(MDIM),UPPER(NRNODES),
1     YPRED(NTEST),YTREE(NTEST),XTS(MDIM,NTEST),YTS(NTEST)

      DIMENSION JDEX(NSAMPLE),IITREEMAP(2,NRNODES),  NODESTATUS(NRNODES),
1     NODEPOP(NRNODES),NPERT(NSAMPLE),IP(MDIM),NTERM(0:NRNODES),
1     NPERM(NSAMPLE),IIPARENT(NRNODES),JJCAT(MDIM),NOUT(NSAMPLE),
1     JIN(NSAMPLE),ISORT(NSAMPLE),NODESTART(NRNODES),NCASE(NSAMPLE),
1     NBRTERM(NRNODES),JPERM(JBT),MBEST(NRNODES),INCL(MDIM)
```

```
DO N=1,717
    ZZ=RAND(1)
END DO

DO M=1,MDIM
    JJCAT(M)=1
END DO

QVERRTS=0
AVERRB=0

AVY=0
VARY=0
DO N=1,NSAMPLE
    NTRUE=N-1
    VARY=VARY+NTRUE*(Y(N)-AVY)**2/(NTRUE+1)
    AVY=(NTRUE*AVY+Y(N))/(NTRUE+1)
END DO
VARY=VARY/NSAMPLE

CALL ZERVR(YPTR,NSAMPLE)
CALL ZERV(NOUT,NSAMPLE)
ASTR=0
ASD=0

DO JB=1,JBT
    CALL ZERV(JIN,NSAMPLE)
    DO N=1,NSAMPLE
            K=INT(RAND1(1)*NSAMPLE)+1
            JIN(K)=1
            YB(N)=Y(K)
            DO M=1,MDIM
                XB(M,N)=X(M,K)
            END DO
    END DO

NLS=NSAMPLE
```

```
      CALL BUILDTREE(XB,YB,YL,MDIM,NLS,NSAMPLE,IITREEMAP,JDEX,
     1  UPPER,AVNODE,BESTCRIT, NODESTATUS,NODEPOP,NODESTART,
     1  NRNODES,NTHSIZE,RSNODECOST,NCASE,IIPARENT,UT,V,IPRINT,
     1  XT,MTRY,IP,NLIM,MBEST,JJCAT,TGINI)
      NDBIGTREE=NRNODES
      DO K=NRNODES,1,-1
         IF (NODESTATUS(K).EQ.0) NDBIGTREE=NDBIGTREE-1
         IF (NODESTATUS(K).EQ.2) NODESTATUS(K)=-1
      END DO
      CALL ZERVR(YTR,NSAMPLE)
      CALL TESTREEBAG(X,NSAMPLE,MDIM,IITREEMAP,NODESTATUS,
     1  NRNODES,NDBIGTREE,YTR,UPPER,AVNODE,MBEST,JJCAT)

      ERRB=0
      JOUT=0
      DO N=1,NSAMPLE
         IF(JIN(N).EQ.0) THEN
                  YPTR(N)=(NOUT(N)*YPTR(N)+YTR(N))/(NOUT(N)+1)
                  NOUT(N)=NOUT(N)+1
         END IF

         IF(NOUT(N).GT.0) JOUT=JOUT+1
                  ERRB=ERRB+(Y(N)-YPTR(N))**2
      END DO
      ERRB=ERRB/NSAMPLE

      IF(IMP.EQ.1) THEN
         DO MR=1,MDIM
                  CALL PERMOBMR(MR,X,UTR,XT,JIN,NSAMPLE,MDIM)
                  CALL TESTREEBAG(X,NSAMPLE,MDIM,IITREEMAP,NODESTATUS,
     1            NRNODES,NDBIGTREE,YTR,UPPER,AVNODE,MBEST,JJCAT)
                  DO N=1,NSAMPLE
                           X(MR,N)=XT(N)
                  END DO
                  EM=0
                  DO N=1,NSAMPLE
                           IF(JIN(N).EQ.0) THEN
```

```
                                PREDIMP(N,MR)=(NOUT(N)*PREDIMP(N,MR)+YTR(N))
1                               /(NOUT(N)+1)
                            END IF
                            EM=EM+(Y(N)-PREDIMP(N,MR))**2
                        END DO
                        ERRIMP(MR)=EM/NSAMPLE
                    END DO
                END IF
            END DO

            IF(IMP.EQ.1) THEN
                DO M=1,MDIM
                    ERRIMP(M)=100.0D0*((ERRIMP(M)/ERRB)-1)
                    IF(ERRIMP(M).LE.0.0D0) ERRIMP(M)=0.0D0
                END DO
            END IF

            DO M=1,MDIM
                ZSCORE(M)=ERRIMP(M)
            END DO

            END
C     END OF THE RANFOR SUBROUTINE

            SUBROUTINE BUILDTREE(X,Y,YL,MDIM,NLS,NSAMPLE,IITREEMAP,
           1    JDEX,UPPER,AVNODE,BESTCRIT, NODESTATUS,
           1    NODEPOP,NODESTART,NRNODES,NTHSIZE,RSNODECOST,
           1    NCASE,IIPARENT,UT,V,IPRINT,XT,MTRY,IP,NLIM,
           1    MBEST,JJCAT,TGINI)

                IMPLICIT DOUBLE PRECISION(A-H,O-Z)
                DIMENSION IITREEMAP(2,NRNODES),IIPARENT(NRNODES),
           1    NODESTATUS(NRNODES),IP(MDIM),NODEPOP(NRNODES),
           1    NODESTART(NRNODES),JDEX(NSAMPLE),NCASE(NSAMPLE),
           1    MBEST(NRNODES),JJCAT(MDIM)

                DIMENSION Y(NSAMPLE),BESTCRIT(NRNODES),X(MDIM,NSAMPLE),
```

```
1    AVNODE(NRNODES),XT(NSAMPLE),UPPER(NRNODES),
1    V(NSAMPLE),UT(NSAMPLE),RSNODECOST(NRNODES),
1    YL(NSAMPLE),TGINI(MDIM)

     CALL ZERV(NODESTATUS,NRNODES)
     CALL ZERV(NODESTART,NRNODES)
     CALL ZERV(NODEPOP,NRNODES)
     CALL ZERVR(AVNODE,NRNODES)

     DO N=1,NSAMPLE
         UT(N)=0
         JDEX(N)=N
     END DO

     NCUR=1
     NODESTART(1)=1
     NODEPOP(1)=NLS
     NODESTATUS(1)=2

     AV=0
     SS=0
     DO N=1,NLS
         D=Y(JDEX(N))
         SS=SS+(N-1)*(AV-D)*(AV-D)/N
         AV=((N-1)*AV+D)/N
     END DO !N
     AVNODE(1)=AV
     RSNODECOST(1)=SS/NLS

     DO 30 KBUILD=1,NRNODES
         IF (KBUILD.GT.NCUR) GOTO 50
         IF (NODESTATUS(KBUILD).NE.2) GOTO 30

         NDSTART=NODESTART(KBUILD)
         NDEND=NDSTART+NODEPOP(KBUILD)-1
         NODECNT=NODEPOP(KBUILD)
         SUMNODE=NODECNT*AVNODE(KBUILD)
```

```
      JSTAT=0
      CALL FINDBESTSPLIT(X,XT,UT,JDEX,Y,MDIM,NSAMPLE,
1     NDSTART,NDEND,MSPLIT,DECSPLIT,UBEST,NCASE,NDENDL,
1     JSTAT,V,MTRY,IP,NLIM,SUMNODE,NODECNT,YL,JJCAT)

      IF (JSTAT.EQ.1) THEN
            NODESTATUS(KBUILD)=-1
            GO TO 30
      ELSE
            MBEST(KBUILD)=MSPLIT
            UPPER(KBUILD)=UBEST
            BESTCRIT(KBUILD)=DECSPLIT
      END IF

   TGINI(MSPLIT)=TGINI(MSPLIT)+DECSPLIT
   NODEPOP(NCUR+1)=NDENDL-NDSTART+1
   NODEPOP(NCUR+2)=NDEND-NDENDL
   NODESTART(NCUR+1)=NDSTART
   NODESTART(NCUR+2)=NDENDL+1

   AV=0
   SS=0
   DO N=NDSTART,NDENDL
      D=Y(JDEX(N))
      K=N-NDSTART
      SS=SS+K*(AV-D)*(AV-D)/(K+1)
      AV=(K*AV+D)/(K+1)
   END DO !N
   AVNODE(NCUR+1)=AV
   RSNODECOST(NCUR+1)=SS/NLS
   AV=0
   SS=0
   DO N=NDENDL+1,NDEND
      D=Y(JDEX(N))
      K=N-NDENDL-1
      SS=SS+K*(AV-D)*(AV-D)/(K+1)
      AV=(K*AV+D)/(K+1)
```

```
      END DO !N
      AVNODE(NCUR+2)=AV
      RSNODECOST(NCUR+2)=SS/NLS

      NODESTATUS(NCUR+1)=2
      NODESTATUS(NCUR+2)=2
      IF (NODEPOP(NCUR+1).LE.NTHSIZE)
     1     NODESTATUS(NCUR+1)=-1
      IF (NODEPOP(NCUR+2).LE.NTHSIZE)
     1     NODESTATUS(NCUR+2)=-1

      IITREEMAP(1,KBUILD)=NCUR+1
      IITREEMAP(2,KBUILD)=NCUR+2
      IIPARENT(NCUR+1)=KBUILD
      IIPARENT(NCUR+2)=KBUILD
      NODESTATUS(KBUILD)=1
      NCUR=NCUR+2

      IF (NCUR.GE.NRNODES) GOTO 50

30    CONTINUE
50    CONTINUE

      END
C     END OF THE BUILDTREE SUBROUTINE

      SUBROUTINE FINDBESTSPLIT(X,XT,UT,JDEX,Y,MDIM,
     1     NSAMPLE,NDSTART,NDEND,MSPLIT,DECSPLIT,UBEST,
     1     NCASE,NDENDL,JSTAT,V,MTRY,IP,NLIM,
     1     SUMNODE,NODECNT,YL,JJCAT)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION NCASE(NSAMPLE),JDEX(NSAMPLE),IP(MDIM),
     1     NCAT(32),ICAT(32),JJCAT(MDIM)
      DIMENSION X(MDIM,NSAMPLE),UT(NSAMPLE),XT(NSAMPLE),
     1     V(NSAMPLE),Y(NSAMPLE),YL(NSAMPLE),
     1     SUMCAT(32),AVCAT(32),TAVCAT(32)
```

```
      CRITMAX=0
200   CALL ZERV(IP,MDIM)
      NON=0
      DO MT=1,MTRY
            CRITVAR=0
100         KV=INT(RAND(1)*MDIM)+1
            IF(IP(KV).EQ.1) GOTO 100
            IP(KV)=100
            LC=JJCAT(KV)
            IF(LC.EQ.1) THEN
                  DO N=NDSTART,NDEND
                  XT(N)=X(KV,JDEX(N))
                  YL(N)=Y(JDEX(N))
      END DO
      ELSE
            CALL ZERVR(SUMCAT,32)
            CALL ZERV(NCAT,32)
            DO N=NDSTART,NDEND
                  L=NINT(X(KV,JDEX(N)))
                  D=Y(JDEX(N))
                  SUMCAT(L)=SUMCAT(L)+D
                  NCAT(L)=NCAT(L)+1
            END DO
            DO J=1,LC
                  IF(NCAT(J).GT.0) THEN
                        AVCAT(J)=SUMCAT(J)/NCAT(J)
                  ELSE
                        AVCAT(J)=0
                  END IF
            END DO
            DO N=1,NSAMPLE
                  XT(N)=AVCAT(NINT(X(KV,JDEX(N))))
                  YL(N)=Y(JDEX(N))
            END DO
      END IF

      DO N=NDSTART,NDEND
```

```
        V(N)=XT(N)
END DO
DO N=1,NSAMPLE
        NCASE(N)=N
END DO
CALL QUICKSORT(V,NCASE,NDSTART,NDEND,NSAMPLE)
IF(V(NDSTART).GE.V(NDEND))THEN
        NON=NON+1
        IF(NON.GE.3*MDIM) THEN
                JSTAT=1
                RETURN
        END IF
        GOTO 100
END IF

SUML=0
SUMR=SUMNODE
NPOPL=0
NPOPR=NODECNT

DO NSP=NDSTART,NDEND-1
        D=YL(NCASE(NSP))
        SUML=SUML+D
        SUMR=SUMR-D
        NPOPL=NPOPL+1
        NPOPR=NPOPR-1
        IF (V(NSP).LT.V(NSP+1)) THEN
                CRIT=(SUML*SUML/NPOPL)+(SUMR*SUMR/NPOPR)
                IF (CRIT.GT.CRITVAR) THEN
                        UBESTT=(V(NSP)+V(NSP+1))/2.0
                        CRITVAR=CRIT
                        NBESTT=NSP
                ENDIF
        END IF
END DO

IF(CRITVAR.GT.CRITMAX) THEN
```

```
                UBEST=UBESTT
                NBEST=NBESTT
                MSPLIT=KV
                CRITMAX=CRITVAR
                DO N=NDSTART,NDEND
                        UT(N)=XT(N)
                END DO
                IF (JJCAT(KV).GT.1) THEN
                        IC=JJCAT(KV)
                        DO J=1,IC
                                TAVCAT(J)=AVCAT(J)
                        END DO
                END IF
        END IF
    END DO

        NL=NDSTART-1
                DO NSP=NDSTART,NDEND
                        IF(UT(NSP).LE.UBEST) THEN
                                NL=NL+1
                                NCASE(NL)=JDEX(NSP)
                        END IF
                END DO

        NDENDL=MAX0(NL,NDSTART+1)
        NR=NDENDL
        DO NSP=NDSTART,NDEND
                IF(UT(NSP).GT.UBEST) THEN
                        NR=NR+1
                        IF(NR.GT.NSAMPLE) GOTO 765
                        NCASE(NR)=JDEX(NSP)
                END IF
        END DO
765     CONTINUE

        IF(NDENDL.GE.NDEND) NDENDL=NDEND-1
```

```
      DO N=NDSTART,NDEND
            JDEX(N)=NCASE(N)
      END DO
      LC=JJCAT(MSPLIT)
      IF(LC.GT.1) THEN
            DO J=1,LC
                  IF(TAVCAT(J).LT.UBEST) THEN
                        ICAT(J)=1
                  ELSE
                        ICAT(J)=0
                  END IF
            END DO
            CALL PACKLB(LC,ICAT,NUBEST)
            UBEST=REAL(NUBEST)
      END IF

      DECSPLIT=CRITMAX-(SUMNODE*SUMNODE/NODECNT)
END
C     END OF THE FINDBESTSPLIT SUBROUTINE

SUBROUTINE TESTREEBAG(X,NSAMPLE,MDIM,IITREEMAP,NODESTATUS,
C     USES:     DUMINF, FCN
1     NRNODES,NDBIGTREE,YTREE,UPPER,AVNODE,MBEST,JJCAT)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(MDIM,NSAMPLE),
1     UPPER(NRNODES),AVNODE(NRNODES),YTREE(NSAMPLE)

      DIMENSION IITREEMAP(2,NRNODES),NODESTATUS(NRNODES),
1     MBEST(NRNODES),JJCAT(MDIM),ICAT(32)
      DO N=1,NSAMPLE
      KT=1
      DO K=1,NDBIGTREE
      IF(NODESTATUS(KT).EQ.-1) THEN
          YTREE(N)=AVNODE(KT)
          GOTO 100
      END IF
      M=MBEST(KT)
```

```
      LC=JJCAT(M)
      IF(LC.EQ.1) THEN
          IF (X(M,N).LE.UPPER(KT)) THEN
                KT=IITREEMAP(1,KT)
          ELSE
                KT=IITREEMAP(2,KT)
          ENDIF
      ELSE
      MM=NINT(UPPER(KT))
      CALL UNPACKLB(LC,MM,ICAT)
      J=NINT(X(M,N))
          IF(ICAT(J).EQ.1) THEN
                KT=IITREEMAP(1,KT)
          ELSE
                KT=IITREEMAP(2,KT)
          ENDIF
      END IF
      END DO
100   CONTINUE
      END DO
END
C     END OF THE TESTREEBAG SUBROUTINE

SUBROUTINE PERMOBMR(MR,X,TP,TX,JIN,NSAMPLE,MDIM)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(MDIM,NSAMPLE),TP(NSAMPLE),TX(NSAMPLE),
1     JIN(NSAMPLE)
      KOUT=0
      CALL ZERVR(TP,NSAMPLE)
      DO N=1,NSAMPLE
      IF(JIN(N).EQ.0) THEN
          KOUT=KOUT+1
          TP(KOUT)=X(MR,N)
      END IF
      END DO !N
      CALL PERM1(KOUT,NSAMPLE,TP)
      IOUT=0
```

```
      DO N=1,NSAMPLE
      TX(N)=X(MR,N)
      IF(JIN(N).EQ.0) THEN
        IOUT=IOUT+1
        X(MR,N)=TP(IOUT)
      END IF
      END DO
END
C     END OF THE PERMOBMR SUBROUTINE

SUBROUTINE ZERV(IX,M1)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION IX(M1)
      DO 10 N=1,M1
      IX(N)=0
10    CONTINUE
END
C     END OF THE ZERV SUBROUTINE

SUBROUTINE ZERVR(RX,M1)
C     INPUT:    A VALUE
C     USES:     DUMINF, FCN
C     OUTPUT:
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION RX(M1)
      DO 10 N=1,M1
      RX(N)=0.0D0
10    CONTINUE
END
C     END OF THE ZERVR SUBROUTINE

SUBROUTINE ZERM(MX,M1,M2)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION MX(M1,M2)
      DO 10 I=1,M1
          DO 20 J=1,M2
                MX(I,J)=0
```

```
20      CONTINUE
10  CONTINUE
END
C   END OF THE ZERM SUBROUTINE

SUBROUTINE PACKLB(L,ICAT,NPACK)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    INTEGER ICAT(32)
    NPACK=0
    DO 10 K=1,L
        NPACK=NPACK+ICAT(K)*(2**(K-1))
10  CONTINUE
END
C   END OF THE PACKLB SUBROUTINE

SUBROUTINE UNPACKLB(L,NPACK,ICAT)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    INTEGER ICAT(32)
    CALL ZERV(ICAT,32)
    N=NPACK
    ICAT(1)=MOD(N,2)
    DO 10 K=2,L
        N=(N-ICAT(K-1))/2
        ICAT(K)=MOD(N,2)
10  CONTINUE
END
C   END OF THE UNPACKLB SUBROUTINE

SUBROUTINE QUICKSORT(V,IPERM,II,JJ,KK)
    IMPLICIT DOUBLE PRECISION(A-H,O-Z)
    DIMENSION IPERM(KK),V(KK),IU(32),IL(32)
    M=1
    I=II
    J=JJ
10  IF (I.GE.J) GO TO 80
20  K=I
    IJ=(J+I)/2
```

```
      IIT=IPERM(IJ)
      VT=V(IJ)
      IF (V(I).LE.VT) GO TO 30
      IPERM(IJ)=IPERM(I)
      IPERM(I)=IIT
      IIT=IPERM(IJ)
      V(IJ)=V(I)
      V(I)=VT
      VT=V(IJ)
30    L=J
      IF (V(J).GE.VT) GO TO 50
      IPERM(IJ)=IPERM(J)
      IPERM(J)=IIT
      IIT=IPERM(IJ)
      V(IJ)=V(J)
      V(J)=VT
      VT=V(IJ)
      IF (V(I).LE.VT) GO TO 50
      IPERM(IJ)=IPERM(I)
      IPERM(I)=IIT
      IIT=IPERM(IJ)
      V(IJ)=V(I)
      V(I)=VT
      VT=V(IJ)
      GO TO 50
40    IPERM(L)=IPERM(K)
      IPERM(K)=IITT
      V(L)=V(K)
      V(K)=VTT
50    L=L-1
      IF (V(L).GT.VT) GO TO 50
      IITT=IPERM(L)
      VTT=V(L)
60    K=K+1
      IF (V(K).LT.VT) GO TO 60
      IF (K.LE.L) GO TO 40
      IF (L-I.LE.J-K) GO TO 70
```

```
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GO TO 90
70    IL(M)=K
      IU(M)=J
      J=L
      M=M+1
      GO TO 90
80    M=M-1
      IF (M.EQ.0) RETURN
      I=IL(M)
      J=IU(M)
90    IF (J-I.GT.10) GO TO 20
      IF (I.EQ.II) GO TO 10
      I=I-1
100   I=I+1
      IF (I.EQ.J) GO TO 80
      IIT=IPERM(I+1)
      VT=V(I+1)
      IF (V(I).LE.VT) GO TO 100
      K=I
110   IPERM(K+1)=IPERM(K)
      V(K+1)=V(K)
      K=K-1
      IF (VT.LT.V(K)) GO TO 110
      IPERM(K+1)=IIT
      V(K+1)=VT
      GO TO 100
END
C     END OF THE QUICKSORT SUBROUTINE

SUBROUTINE PERM1(NP,NS,TP)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION TP(NS)
      J=NP
```

```
11   RND = RAND(1)
     K=INT(J*RND)+1
     TX=TP(J)
     TP(J)=TP(K)
     TP(K)=TX
     J=J-1
     IF(J.GT.1) GO TO 11
     END
C    END OF THE PERM1 SUBROUTINE

FUNCTION RAND(J)
     IMPLICIT DOUBLE PRECISION(A-H,O-Z)
     SAVE DSEED
     DATA DSEED /17395/
     CALL LRND(DSEED,U)
     RAND=U
END
C    END OF THE RAND FUNCTION

SUBROUTINE LRND(DSEED,U)
     IMPLICIT DOUBLE PRECISION(A-H,O-Z)
     DATA D31M1 /2147483647/
     DSEED=DMOD(16087*DSEED,D31M1)
     U=DSEED/D31M1
RETURN
END
C    END OF THE LRND SUBROUTINE

FUNCTION RAND1(J)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
     SAVE DSEED1
     DATA DSEED1 /17395/
     CALL LRND1(DSEED1,U)
     RAND1=U
END
C    END OF THE RAND1 FUNCTION
SUBROUTINE LRND1(DSEED1,U)
```

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DATA D31M1 /2147483647.D0/
      DSEED1=DMOD(16087.0D0*DSEED1,D31M1)
      U=DSEED1/D31M1
RETURN
END
C     END OF THE LRND1 SUBROUTINE

FUNCTION RNORM(J)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      U=RAND(1)
      V=RAND(1)
      RNORM=DSQRT(-2.0D0*DLOG(U))*DCOS(6.28318531D0*V)
END
C     END OF THE RNORM FUNCTION
```

```
SUBROUTINE BERCRITW(GRMAT,CRIT1,CRIT2)
C    CALCULATES THE VARIATION RATIO AND THE DISTANCE BETWEEN MEANS
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
     DIMENSION XM(NNPMM,IP),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
     CRIT1=TELLER/ANOEMER
     CRIT2=TELLER
     RETURN
END
C    END OF THE BERCRITW SUBROUTINE

SUBROUTINE BERCRIT12(GRMAT,CRIT)
C    CALCULATES THE SUM OF DISSIMILARITIES IN THE KERNEL MATRIX
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
     DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
```

```
      S3=0.0D0
      DO 15 I=1,NN
         DO 14 J=NN+1,NNPMM
                  S3=S3+GRMAT(I,J)
14       CONTINUE
15    CONTINUE
      CRIT=S3
RETURN
END
C     END OF THE BERCRIT12 SUBROUTINE

SUBROUTINE BERCRITCC(GRMAT,CRIT)
C     CALCULATES THE CUT COST SELECTION CRITERION
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)

      S1=0.0D0
      DO 15 I=1,NN
         DO 14 J=NN+1,NNPMM
                  S1=S1+2.0D0*GRMAT(I,J)
14       CONTINUE
15    CONTINUE

      S2=0.0D0
      DO 17 I=1,NNPMM
         DO 16 J=1,NNPMM
                  S2=S2+GRMAT(I,J)**2
16       CONTINUE
17    CONTINUE
      CRIT=DABS(S1/NNPMM*DSQRT(S2))
RETURN
END
C     END OF THE BERCRITCC SUBROUTINE
```

```
SUBROUTINE BERCRITR(GRMAT1,EENP,EENM,ALPHA,CPAR,CRIT)
C    CALCULATES THE RAYLEIGH QUOTIENT
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
     DIMENSION GRMAT1(NNPMM,NNPMM),EENP(NNPMM),EENM(NNPMM)
     DIMENSION B(NNPMM),H(NNPMM,NNPMM),ALPHA(NNPMM)
     DIMENSION AKP(NNPMM),AKM(NNPMM)
     DIMENSION AM(NNPMM,NNPMM),ALPHAN(NNPMM),ALPHAM(NNPMM)

     DO 62 J=1,NNPMM
        S1=0.0D0
        S2=0.0D0
        DO 61 I=1,NNPMM
              S1=S1+GRMAT1(I,J)*EENM(I)
              S2=S2+GRMAT1(I,J)*EENP(I)
61      CONTINUE
        AKM(J)=S1/NN
        AKP(J)=S2/MM
        B(J)=AKP(J)-AKM(J)
62   CONTINUE

     DO 65 J1=1,NNPMM
        DO 64 J2=1,NNPMM
              S=0.0D0
              DO 63 I=1,NNPMM
                    S=S+GRMAT1(I,J1)*GRMAT1(I,J2)
63            CONTINUE
              H(J1,J2)=(S-MM*AKP(J1)*AKP(J2)-NN*AKM(J1)*AKM(J2))/NNPMM
64      CONTINUE
        H(J1,J1)=H(J1,J1)+CPAR
65   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
      CRIT=S1/S2
RETURN
END
C     END OF THE BERCRITR SUBROUTINE

FUNCTION ALIGNMENTTRANS(J,XM)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,IPTEL=4,NN=100,MM=100,NNPMM=NN+MM,IPP1=IP+1)
      DIMENSION XM(NNPMM,IP)
      S1=0.0D0
      S2=0.0D0
      S3=0.0D0
      S4=0.0D0
      DO 5 I1=1,NN
         DO 4 I2=1,NN
                  S=(XM(I1,J)-XM(I2,J))*(XM(I1,J)-XM(I2,J))
                  S1=S1+(2.0D0*DEXP(-S)-1.0D0)
```

```
                S4=S4+(2.0D0*DEXP(-S)-1.0D0)**2.0D0
4       CONTINUE
5   CONTINUE
    DO 10 I1=NN+1,NNPMM
        DO 9 I2=NN+1,NNPMM
                S=(XM(I1,J)-XM(I2,J))*(XM(I1,J)-XM(I2,J))
                S2=S2+(2.0D0*DEXP(-S)-1.0D0)
                S4=S4+(2.0D0*DEXP(-S)-1.0D0)**2.0D0
9       CONTINUE
10  CONTINUE
    DO 15 I1=1,NN
        DO 14 I2=NN+1,NNPMM
                S=(XM(I1,J)-XM(I2,J))*(XM(I1,J)-XM(I2,J))
                S3=S3+(2.0D0*DEXP(-S)-1.0D0)
                S4=S4+2.0D0*(2.0D0*DEXP(-S)-1.0D0)**2.0D0
14      CONTINUE
15  CONTINUE
    S3=2.0D0*S3
    ALIGNMENTTRANS=(S1+S2-S3)/((1.0D0*NNPMM)*DSQRT(S4))
RETURN
END
C   END OF THE ALIGNMENTTRANS FUNCTION

SUBROUTINE GRAMMATL(GAMPAR,XM,NV,INDVEK,GRMAT)
C   CALCULATES THE KERNEL MATRIX BASED ON THE TRANSFORMED GAUSSIAN KERNEL
C   FUNCTION
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
    DIMENSION INDVEK(IP)

    DO 10 I=1,NNPMM-1
        GRMAT(I,I)=1.0D0
        DO 5 J=I+1,NNPMM
                S=0.0D0
                DO 3 K=1,NV
                        KK=INDVEK(K)
```

```
                    S=S+2.0D0*(XM(I,KK)-XM(J,KK))*(XM(I,KK)-XM(J,KK))-1.0D0
3               CONTINUE
        GRMAT(I,J)=DEXP(-GAMPAR*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
C   END OF THE GRAMMATL SUBROUTINE

SUBROUTINE BERCRITA0(GRMAT,CRIT)
C   CALCULATES THE ZERO ORDER ALIGNMENT CRITERION
C    INPUT:    THE KERNEL MATRIX CALCULATED ON THE TRAINING INPUT PATTERNS
C    USES:     FUNCTION DSQRT
C    OUTPUT: THE VALUE OF THE ZERO ORDER ALIGNMENT
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)

    S1=0.0D0
    S2=0.0D0
    S3=0.0D0
    DO 10 I=1,NN
        DO 5 J=1,NN
                S1=S1+GRMAT(I,J)
                S3=S3+GRMAT(I,J)**2.0D0
5       CONTINUE
10  CONTINUE
    DO 20 I=NN+1,NNPMM
        DO 15 J=NN+1,NNPMM
                S1=S1+GRMAT(I,J)
                S3=S3+GRMAT(I,J)**2.0D0
```

```
15      CONTINUE
20  CONTINUE
    DO 30 I=1,NN
        DO 29 J=NN+1,NNPMM
                S2=S2+2.0D0*GRMAT(I,J)
                S3=S3+2.0D0*(GRMAT(I,J)**2.0D0)
29      CONTINUE
30  CONTINUE
    CRIT=(S1-S2)/(NNPMM*DSQRT(S3))
RETURN
END
C   END OF THE BERCRITA0 SUBROUTINE

SUBROUTINE BERCRITA1(KK,XM,GRMAT,GAM,CRIT)
C   CALCULATES THE FIRST ORDER ALIGNMENT CRITERION
C   INPUT:  THE INDEX OF THE VARIABLE W.R.T WHICH DIFFERENTIATION TAKES PLACE
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
    S1=0.0D0
    S2=0.0D0
    S3=0.0D0
    S4=0.0D0
    S5=0.0D0
    DO 10 I=1,NN
        DO 5 J=1,NN
                S1=S1+GRMAT(I,J)
                S3=S3+GRMAT(I,J)**2.0D0
                S4=S4+((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J)
                S5=S5+((XM(I,KK)-XM(J,KK))**2.0D0)*(GRMAT(I,J)**2.0D0)
5       CONTINUE
10  CONTINUE
    DO 20 I=NN+1,NNPMM
        DO 15 J=NN+1,NNPMM
                S1=S1+GRMAT(I,J)
                S3=S3+GRMAT(I,J)**2.0D0
                S4=S4+((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J)
```

```
                S5=S5+((XM(I,KK)-XM(J,KK))**2.0D0)*(GRMAT(I,J)**2.0D0)
15      CONTINUE
20  CONTINUE
    DO 30 I=1,NN
        DO 29 J=NN+1,NNPMM
                S2=S2+2.0D0*GRMAT(I,J)
                S3=S3+2.0D0*(GRMAT(I,J)**2.0D0)
                S4=S4-2.0D0*((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J)
                S5=S5+2.0D0*((XM(I,KK)-XM(J,KK))**2.0D0)*(GRMAT(I,J)**2.0D0)
29      CONTINUE
30  CONTINUE
    A1=S1-S2
    A2=NNPMM*DSQRT(S3)
    AFGA1=-2.0D0*GAM*S4
    AFGA2=(-2.0D0*GAM*S5)*NNPMM/DSQRT(S3)
    CRIT=DABS((A2*AFGA1-A1*AFGA2)/(A2*A2))
RETURN
END
C   END OF THE BERCRITA1 FUNCTION

SUBROUTINE BERCRITW0(GRMAT,CRIT)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),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
C     END OF THE BERCRITW0 SUBROUTINE

SUBROUTINE BERCRITW1(KK,XM,GRMAT,GAM,CRIT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
      S1=0.0D0
      S2=0.0D0
      S3=0.0D0
      S4=0.0D0
      S5=0.0D0
      S6=0.0D0
      DO 10 I=1,NN
            DO 5 J=1,NN
                  S1=S1+GRMAT(I,J)
                  S4=S4+((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J)
5           CONTINUE
10    CONTINUE
      DO 20 I=NN+1,NNPMM
            DO 15 J=NN+1,NNPMM
                  S2=S2+GRMAT(I,J)
                  S5=S5+((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J)
15          CONTINUE
20    CONTINUE
      DO 30 I=1,NN
            DO 29 J=NN+1,NNPMM
```

```
                S3=S3+2.0D0*GRMAT(I,J)
                S6=S6+2.0D0*((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J)
29        CONTINUE
30   CONTINUE
     W1=S1/(NN*NN)+S2/(MM*MM)-(2.0D0*S3)/(NN*MM)
     T1=S4/(NN*NN)+S5/(MM*MM)-(2.0D0*S6)/(NN*MM)
     AFGW1=-2.0D0*GAM*T1
     W2=NNPMM-S1/NN-S2/MM
     AFGW2=(2.0D0*GAM*S4)/NN+(2.0D0*GAM*S5)/MM
     CRIT=DABS((W2*AFGW1-W1*AFGW2)/(W2*W2))
RETURN
END
C    END OF THE BERCRITW1 SUBROUTINE

SUBROUTINE BERCRITW(GRMAT,CRIT1,CRIT2)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
     DIMENSION XM(NNPMM,IP),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
      CRIT1=TELLER/ANOEMER
      CRIT2=TELLER
RETURN
END
C     END OF THE BERCRITW SUBROUTINE

SUBROUTINE BERCRITG1(KK,XM,GRMAT,GAM,CRITG1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)

      S1=0.0D0
      DO 4 I=1,NN
          DO 3 J=1,NN
                S1=S1+(-2.0D0*GAM*((XM(I,KK)-XM(J,KK)**2.0D0)*GRMAT(I,J)))
3         CONTINUE
4     CONTINUE
      S2=0.0D0
      DO 6 I=NN+1,NNPMM
          DO 5 J=NN+1,NNPMM
                S2=S2+(-2.0D0*GAM*((XM(I,KK)-XM(J,KK)**2.0D0)*GRMAT(I,J)))
5         CONTINUE
6     CONTINUE
S3=0.0D0
DO 15 I=1,NN
      DO 14 J=NN+1,NNPMM
          S3=S3+(-2.0D0*GAM*((XM(I,KK)-XM(J,KK)**2.0D0)*GRMAT(I,J)))
14    CONTINUE
15    CONTINUE
      CRITG1=DABS((1.0D0/(NN**2))*S1+(1.0D0/(MM**2))*S2
     & -(2.0D0/(NN**MM))*S3)
RETURN
END
C     END OF THE BERCRITG1 SUBROUTINE
```

```
SUBROUTINE BERCRIT12(GRMAT,CRIT)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)

    S3=0.0D0
    DO 15 I=1,NN
        DO 14 J=NN+1,NNPMM
                S3=S3+GRMAT(I,J)
14      CONTINUE
15  CONTINUE
    CRIT=S3
RETURN
END
C   END OF THE BERCRIT12 SUBROUTINE

SUBROUTINE BERCRITET1(KK,XM,GRMAT,GAM,CRIT)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)

    S=0.0D0
    DO 15 I=1,NN
        DO 14 J=NN+1,NNPMM
                S=S+(-2.0D0*GAM*(XM(I,KK)-XM(J,KK)**2)*GRMAT(I,J))
14      CONTINUE
15  CONTINUE
    CRIT=DABS(S)
RETURN
END
C   END OF THE BERCRITET1 SUBROUTINE

SUBROUTINE BERCRITN1(KK,GAM,CPARS,YV,XM,NV,INDVEKN1,CRIT)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),XMN(NNPMM,NNPMM),YV(NNPMM)
    DIMENSION GRMATN(NNPMM,NNPMM),GRMAT(NNPMM,NNPMM)
```

```
      DIMENSION AL(NNPMM)
      DIMENSION INDVEKN1(IP)

      CALL GRAMMAT(GAM,XM,NV,INDVEKN1,GRMAT)
      CALL DOENSVM(YV,XM,GRMAT,CPARS,GAM,NV,INDVEKN1,AL,BOPT)
      S=0.0D0
      DO 10 I=1,NNPMM
          DO 9 J=1,NNPMM
                S=S+(-2.0D0*GAM*AL(I)*AL(J)*YV(I)*YV(J)*
               &((XM(I,KK)-XM(J,KK))**2.0D0)*GRMAT(I,J))
9         CONTINUE
10    CONTINUE
      CRIT=DABS(S)
RETURN
END
C     END OF THE BERCRITN1 SUBROUTINE

SUBROUTINE BERCRITR(GRMAT1,EENP,EENM,ALPHA,CPAR,CRIT)
C     CALCULATES THE ZERO-ORDER RAYLEIGH CRITERION
 IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
     DIMENSION GRMAT1(NNPMM,NNPMM),EENP(NNPMM),EENM(NNPMM)
     DIMENSION B(NNPMM),H(NNPMM,NNPMM),ALPHA(NNPMM)
     DIMENSION AKP(NNPMM),AKM(NNPMM)
     DIMENSION AM(NNPMM,NNPMM),ALPHAN(NNPMM),ALPHAM(NNPMM)

     DO 62 J=1,NNPMM
     S1=0.0D0
     S2=0.0D0
     DO 61 I=1,NNPMM
     S1=S1+GRMAT1(I,J)*EENM(I)
     S2=S2+GRMAT1(I,J)*EENP(I)
61    CONTINUE
     AKM(J)=S1/NN
     AKP(J)=S2/MM
     B(J)=AKP(J)-AKM(J)
62    CONTINUE
```

```
      DO 65 J1=1,NNPMM
      DO 64 J2=1,NNPMM
      S=0.0D0
      DO 63 I=1,NNPMM
      S=S+GRMAT1(I,J1)*GRMAT1(I,J2)
63    CONTINUE
      H(J1,J2)=(S-MM*AKP(J1)*AKP(J2)-NN*AKM(J1)*AKM(J2))/NNPMM
64    CONTINUE
      H(J1,J1)=H(J1,J1)+CPAR
65    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
      CRIT=S1/S2
```

```
      RETURN
      END
C     END OF THE BERCRITR SUBROUTINE

      SUBROUTINE BERCRITR1(KK,XM,GRMAT,GAM,ALPHA,CRIT)
C     CALCULATES THE FIRST ORDER RAYLEIGH CRITERION
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=10,NN=100,MM=100,NNPMM=NN+MM)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM),ALPHA(NNPMM)
      DIMENSION RM1(NNPMM),RM2(NNPMM),RMMAT(NNPMM,NNPMM)
      DIMENSION RNMAT(NNPMM,NNPMM)
      DIMENSION DRMMAT(NNPMM,NNPMM),DRNMAT(NNPMM,NNPMM)

      DO 5 I=1,NNPMM
      S=0.0D0
      DO 2 J=1,NN
      S=S+GRMAT(I,J)
2     CONTINUE
      RM1(I)=S/NN
5     CONTINUE
      DO 10 I=1,NNPMM
      S=0.0D0
      DO 7 J=NN+1,NNPMM
      S=S+GRMAT(I,J)
7     CONTINUE
      RM2(I)=S/MM
10    CONTINUE

      DO 15 I=1,NNPMM
      DO 14 J=1,NNPMM
      RMMAT(I,J)=(RM1(I)-RM2(I))*(RM1(J)-RM2(J))
14    CONTINUE
15    CONTINUE

      DO 25 I=1,NNPMM
      DO 24 J=1,NNPMM
      S1=0.0D0
```

```
      S21=0.0D0
      S22=0.0D0
      DO 16 II=1,NN
      S1=S1+GRMAT(I,II)*GRMAT(J,II)
      S21=S21+GRMAT(I,II)
      S22=S22+GRMAT(J,II)
   16 CONTINUE
      S2=S21*S22/NN
      S3=0.0D0
      S41=0.0D0
      S42=0.0D0
      DO 17 II=NN+1,NNPMM
      S3=S3+GRMAT(I,II)*GRMAT(J,II)
      S41=S41+GRMAT(I,II)
      S42=S42+GRMAT(J,II)
   17 CONTINUE
      S4=S41*S42/MM
      RNMAT(I,J)=S1-S2+S3-S4
   24 CONTINUE
   25 CONTINUE

      S1=0.0D0
      S2=0.0D0
      DO 30 I=1,NNPMM
      DO 29 J=1,NNPMM
      S1=S1+ALPHA(I)*RMMAT(I,J)*ALPHA(J)
      S2=S2+ALPHA(I)*RNMAT(I,J)*ALPHA(J)
   29 CONTINUE
   30 CONTINUE
      DELER1=S1
      DELER2=S2

      DO 45 L=1,NNPMM
      DO 44 K=1,NNPMM
      S1=0.0D0
      DO 35 J=1,NN
      S1=S1-2.0D0*GAM*((XM(L,KK)-XM(J,KK))**2.0D0)*GRMAT(L,J)
```

```
35   CONTINUE
     S1=S1/NN
     S2=0.0D0
     DO 36 J=NN+1,NNPMM
     S2=S2-2.0D0*GAM*((XM(L,KK)-XM(J,KK))**2.0D0)*GRMAT(L,J)
36   CONTINUE
     S2=S2/MM
     S3=0.0D0
     DO 37 J=1,NN
     S3=S3-2.0D0*GAM*((XM(K,KK)-XM(J,KK))**2.0D0)*GRMAT(K,J)
37   CONTINUE
     S3=S3/NN
     S4=0.0D0
     DO 38 J=NN+1,NNPMM
     S4=S4-2.0D0*GAM*((XM(K,KK)-XM(J,KK))**2.0D0)*GRMAT(K,J)
38   CONTINUE
     S4=S4/MM
     DRMMAT(L,K)=(S1-S2)*(RM1(K)-RM2(K))+(RM1(L)-RM2(L))*(S3-S4)
44   CONTINUE
45   CONTINUE

     DO 95 I=1,NNPMM
     DO 94 J=1,NNPMM
     S1=0.0D0
     DO 55 II=1,NN
     S1=S1-2.0D0*GAM*((XM(I,KK)-XM(II,KK))**2.0D0)*GRMAT(I,II)
    & *GRMAT(J,II)
55   CONTINUE
     S2=0.0D0
     DO 56 II=1,NN
     S2=S2-2.0D0*GAM*((XM(J,KK)-XM(II,KK))**2.0D0)*GRMAT(J,II)
    & *GRMAT(I,II)
56   CONTINUE
     S3=0.0D0
     DO 58 II=NN+1,NNPMM
     S3=S3-2.0D0*GAM*((XM(I,KK)-XM(II,KK))**2.0D0)*GRMAT(I,II)
    & *GRMAT(J,II)
```

```
58   CONTINUE
     S4=0.0D0
     DO 59 II=NN+1,NNPMM
     S4=S4-2.0D0*GAM*((XM(J,KK)-XM(II,KK))**2.0D0)*GRMAT(J,II)
    & *GRMAT(I,II)
59   CONTINUE
     S51=0.0D0
     S52=0.0D0
     S61=0.0D0
     S62=0.0D0
     DO 61 II=1,NN
     S51=S51-2.0D0*GAM*((XM(I,KK)-XM(II,KK))**2.0D0)*GRMAT(I,II)
     S52=S52+GRMAT(J,II)
     S62=S62-2.0D0*GAM*((XM(J,KK)-XM(II,KK))**2.0D0)*GRMAT(J,II)
     S61=S61+GRMAT(I,II)
61   CONTINUE
     S5=S51*S52/NN
     S6=S61*S62/NN
     S71=0.0D0
     S72=0.0D0
     S81=0.0D0
     S82=0.0D0
     DO 63 II=NN+1,NNPMM
     S71=S71-2.0D0*GAM*((XM(I,KK)-XM(II,KK))**2.0D0)*GRMAT(I,II)
     S72=S72+GRMAT(J,II)
     S82=S82-2.0D0*GAM*((XM(J,KK)-XM(II,KK))**2.0D0)*GRMAT(J,II)
     S81=S81+GRMAT(I,II)
63   CONTINUE
     S7=S71*S72/MM
     S8=S81*S82/MM
     DRNMAT(I,J)=S1+S2+S3+S4-S5-S6-S7-S8
94   CONTINUE
95   CONTINUE

     S1=0.0D0
     S2=0.0D0
     DO 150 I=1,NNPMM
```

```
      DO 149 J=1,NNPMM
      S1=S1+ALPHA(I)*DRMMAT(I,J)*ALPHA(J)
      S2=S2+ALPHA(I)*DRNMAT(I,J)*ALPHA(J)
149   CONTINUE
150   CONTINUE
      TELLER1=S1
      TELLER2=S2

C     CRIT=DABS((TELLER1/DELER1)-(TELLER2/DELER2))
      CRIT=DABS(1.0D0/(DELER2**2)*(DELER2*TELLER1-DELER1*TELLER2))
      RETURN
      END
C     END OF THE BERCRITR1 SUBROUTINE

SUBROUTINE BERCRITA(GRMAT,CRIT)
C     CALCULATES THE ZERO ORDER ALIGNMENT CRITERION
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
      DIMENSION XM(NNPMM,IP),GRMAT(NNPMM,NNPMM)
      S1=0.0D0
      S2=0.0D0
      S3=0.0D0
      DO 10 I=1,NN
          DO 5 J=1,NN
                S1=S1+GRMAT(I,J)
                 S3=S3+GRMAT(I,J)**2.0D0
5         CONTINUE
10    CONTINUE
      DO 20 I=NN+1,NNPMM
          DO 15 J=NN+1,NNPMM
                S1=S1+GRMAT(I,J)
                S3=S3+GRMAT(I,J)**2.0D0
15        CONTINUE
20    CONTINUE
      DO 30 I=1,NN
          DO 29 J=NN+1,NNPMM
                S2=S2+2.0D0*GRMAT(I,J)
```

```
                S3=S3+2.0D0*(GRMAT(I,J)**2.0D0)
29      CONTINUE
30  CONTINUE
CRIT=(S1-S2)/(NNPMM*DSQRT(S3))
RETURN
END

SUBROUTINE BERCRITW(GRMAT,CRIT1,CRIT2)
C   CALCULATE THE VARIATION RATIO CRITERION AND THE DISTANCE BETWEEN MEANS
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
    DIMENSION XM(NNPMM,IP),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
    CRIT1=TELLER/ANOEMER
    CRIT2=TELLER
    RETURN
    END
SUBROUTINE CROSSVAL(XM,YV,MODELMAT,IBESTDIM)
```

```
C   IMPLEMENTS CROSS-VALIDATION TO DETERMINE THE SVM NUMBER OF INPUT
C   VARIABLES
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
      PARAMETER (NCV=5)
      PARAMETER (NNUIT=NN/NCV,MMUIT=MM/NCV,NNIN=NN-NNUIT,MMIN=MM-MMUIT)
      PARAMETER (NNPMMIN=NNIN+MMIN,NNPMMUIT=NNUIT+MMUIT)
      PARAMETER (LDH=NNPMMIN)
      DIMENSION XM(NNPMM,IP),YV(NNPMM)
      DIMENSION XMIN(NNPMMIN,IP),YVIN(NNPMMIN)
      DIMENSION XMUIT(NNPMMUIT,IP),YVUIT(NNPMMUIT)
      DIMENSION GRMAT(NNPMMIN,NNPMMIN),GRNUUT(NNPMMUIT,NNPMMIN)
      DIMENSION B(NNPMMIN),H(NNPMMIN,NNPMMIN),HINV(NNPMMIN,NNPMMIN)
      DIMENSION AL(NNPMMIN)
      DIMENSION FOUTVEK(IP),MODELMAT(IP,IP),INDVEK(IP)

      DO 2 I=1,IP
         FOUTVEK(I)=0.0D0
2     CONTINUE
      DO 500 ICR=1,NCV

C     OBTAIN HOLD-OUT DATA
      NBEG=(ICR-1)*NNUIT+1
      NEND=ICR*NNUIT
      MBEG=NN+(ICR-1)*MMUIT+1
      MEND=NN+ICR*MMUIT

      IF(ICR.EQ.1) THEN
      DO 10 I=NBEG,NEND
         YVUIT(I)=YV(I)
         DO 9 J=1,IP
                XMUIT(I,J)=XM(I,J)
9        CONTINUE
10    CONTINUE
      DO 15 I=MBEG,MEND
         YVUIT(NNUIT+I-MBEG+1)=YV(I)
         DO 14 J=1,IP
```

```
                XMUIT(NNUIT+I-MBEG+1,J)=XM(I,J)
14      CONTINUE
15  CONTINUE
    DO 20 I=NEND+1,NN
        YVIN(I-NEND)=YV(I)
        DO 19 J=1,IP
                XMIN(I-NEND,J)=XM(I,J)
19      CONTINUE
20  CONTINUE
    DO 25 I=MEND+1,NNPMM
        YVIN(NNIN+I-MEND)=YV(I)
        DO 24 J=1,IP
                XMIN(NNIN+I-MEND,J)=XM(I,J)
24      CONTINUE
25  CONTINUE
    ENDIF

    IF((ICR.GT.1).AND.(ICR.LT.NCV)) THEN
    DO 30 I=NBEG,NEND
        YVUIT(I-NBEG+1)=YV(I)
        DO 29 J=1,IP
                XMUIT(I-NBEG+1,J)=XM(I,J)
29      CONTINUE
30  CONTINUE
    DO 35 I=MBEG,MEND
        YVUIT(NNUIT+I-MBEG+1)=YV(I)
        DO 34 J=1,IP
                XMUIT(NNUIT+I-MBEG+1,J)=XM(I,J)
34      CONTINUE
35  CONTINUE
    DO 40 I=1,NBEG-1
        YVIN(I)=YV(I)
        DO 39 J=1,IP
                XMIN(I,J)=XM(I,J)
39      CONTINUE
40  CONTINUE
    DO 45 I=NEND+1,NN
```

```
      YVIN(I-NNUIT)=YV(I)
      DO 44 J=1,IP
            XMIN(I-NNUIT,J)=XM(I,J)
44    CONTINUE
45  CONTINUE
    DO 50 I=NN+1,MBEG-1
      YVIN(NNIN+I-NN)=YV(I)
      DO 49 J=1,IP
            XMIN(NNIN+I-NN,J)=XM(I,J)
49    CONTINUE
50  CONTINUE
    DO 55 I=MEND+1,NNPMM
      YVIN(NNIN+MBEG-NN-1+I-MEND)=YV(I)
      DO 54 J=1,IP
            XMIN(NNIN+MBEG-NN-1+I-MEND,J)=XM(I,J)
54    CONTINUE
55  CONTINUE
    ENDIF

    IF(ICR.EQ.NCV) THEN
    DO 60 I=NBEG,NEND
    YVUIT(I-NBEG+1)=YV(I)
    DO 59 J=1,IP
      XMUIT(I-NBEG+1,J)=XM(I,J)
59  CONTINUE
60  CONTINUE
    DO 65 I=MBEG,MEND
      YVUIT(NNUIT+I-MBEG+1)=YV(I)
      DO 64 J=1,IP
            XMUIT(NNUIT+I-MBEG+1,J)=XM(I,J)
64    CONTINUE
65  CONTINUE
    DO 70 I=1,NBEG-1
      YVIN(I)=YV(I)
      DO 69 J=1,IP
            XMIN(I,J)=XM(I,J)
69    CONTINUE
```

```
70  CONTINUE
    DO 75 I=NN+1,MBEG-1
        YVIN(NNIN+I-NN)=YV(I)
        DO 74 J=1,IP
                XMIN(NNIN+I-NN,J)=XM(I,J)
74      CONTINUE
75  CONTINUE
    ENDIF

C   TRAIN AN SVM ON THE IN-DATA AND TEST ON THE HOLD-OUT DATA
    DO JJ=1,IP
        DO I=1,JJ
                INDVEK(I)=MODELMAT(JJ,I)
        END DO
        GAM=1.0D0/JJ
        CALL DEELGRAMMAT(GAM,JJ,INDVEK,XMIN,GRMAT)
        CALL DEELGRAMNUUT(GAM,JJ,INDVEK,XMIN,XMUIT,GRNUUT)
        CALL DOENSVMIN(YVIN,XMIN,GRMAT,JJ,INDVEK,AL,BOPT)
        CALL BERFOUTSVMUIT(YVIN,GRNUUT,YVUIT,AL,BOPT,FOUT)
        FOUTVEK(JJ)=FOUTVEK(JJ)+FOUT
    END DO
500 CONTINUE

C   DETERMINE THE NUMBER OF INPUT VARIABLES
    AMIN=1.0D15
    DO 510 I=1,IP
    IF (FOUTVEK(I).LT.AMIN) THEN
        AMIN=FOUTVEK(I)
        IBESTDIM=I
    ENDIF
510 CONTINUE
    RETURN
    END
C   END OF THE CROSSVAL SUBROUTINE
```

```
      SUBROUTINE DEELGRAMMAT(GAM,NV,INDVEK,XMIN,GRMAT)
C     CALCULATES THE KERNEL MATRIX ON THE IN- (TRAINING) DATA
C     USED IN SUBROUTINE CROSSVAL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM,IPP1=IP+1)
      PARAMETER (NCV=5)
      PARAMETER (NNUIT=NN/NCV,MMUIT=MM/NCV,NNIN=NN-NNUIT,MMIN=MM-MMUIT)
      PARAMETER (NNPMMIN=NNIN+MMIN,NNPMMUIT=NNUIT+MMUIT)
      DIMENSION XMIN(NNPMMIN,IP),GRMAT(NNPMMIN,NNPMMIN)
      DIMENSION INDVEK(IP)

      DO 10 I=1,NNPMMIN-1
         GRMAT(I,I)=1.0D0
         DO 5 J=I+1,NNPMMIN
         S=0.0D0
               DO 3 K=1,NV
                     KK=INDVEK(K)
                     S=S+(XMIN(I,KK)-XMIN(J,KK))*(XMIN(I,KK)-XMIN(J,KK))
3              CONTINUE
         GRMAT(I,J)=DEXP(-GAM*S)
5        CONTINUE
10    CONTINUE
      GRMAT(NNPMMIN,NNPMMIN)=1.0D0

      DO 20 I=2,NNPMMIN
         DO 15 J=1,I-1
               GRMAT(I,J)=GRMAT(J,I)
15       CONTINUE
20    CONTINUE
      RETURN
      END
C     END OF THE DEELGRAMMAT SUBROUTINE
```

```
SUBROUTINE DEELGRAMNUUT(GAMPAR,NV,INDVEK,XMIN,XMUIT,GRNUUT)
C    CALCULATES THE KERNEL MATRIX ON THE OUT- (TEST) DATA
C    USED IN SUBROUTINE CROSSVAL
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
     PARAMETER (NCV=5)
     PARAMETER (NNUIT=NN/NCV,MMUIT=MM/NCV,NNIN=NN-NNUIT,MMIN=MM-MMUIT)
     PARAMETER (NNPMMIN=NNIN+MMIN,NNPMMUIT=NNUIT+MMUIT)
     DIMENSION XMIN(NNPMMIN,IP),XMUIT(NNPMMUIT,IP)
     DIMENSION GRNUUT(NNPMMUIT,NNPMMIN)
     DIMENSION INDVEK(IP)

     DO 10 I=1,NNPMMUIT
          DO 5 J=1,NNPMMIN
          S=0.0D0
          DO 3 K=1,NV
                    KK=INDVEK(K)
                    S=S+(XMUIT(I,KK)-XMIN(J,KK))*(XMUIT(I,KK)-XMIN(J,KK))
3         CONTINUE
          GRNUUT(I,J)=DEXP(-GAMPAR*S)
5         CONTINUE
10   CONTINUE
RETURN
END

SUBROUTINE DOENSVMIN(YV,XM,GRMAT,NVER,INDVEK,AL,BOPT)
C    TRAINS AN SVM ON THE IN- (TRAINING) DATA
C    USED IN SUBROUTINE CROSSVAL
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
     PARAMETER (NCV=5)
     PARAMETER (NNUIT=NN/NCV,MMUIT=MM/NCV,NNIN=NN-NNUIT,MMIN=MM-MMUIT)
     PARAMETER (NNPMMIN=NNIN+MMIN,NNPMMUIT=NNUIT+MMUIT)
     PARAMETER (CPAR=NNPMMIN/10.0D0)
     PARAMETER (NVAR=NNPMMIN,NCON=2*NVAR+1,NEQ=1,LDA=NCON,LDH=NVAR)
     DIMENSION XM(NNPMMIN,IP),YV(NNPMMIN),GRMAT(NNPMMIN,NNPMMIN)
     DIMENSION XV(IP),XVV(IP)
```

```
      DIMENSION A(NCON,NVAR),B(NCON),G(NVAR),H(NVAR,NVAR)
      DIMENSION SOL(NVAR),ALAM(NVAR),AL(NNPMMIN)
      DIMENSION FW(NNPMMIN),FWR(NNPMMIN),YVR(NNPMMIN)
      DIMENSION IACT(NVAR),IPERM(NNPMMIN)
      DIMENSION INDVEK(IP)

      EP=1.0D-8

      DO 28 I=1,NVAR
         A(1,I)=YV(I)
28    CONTINUE

      DO 30 I=1,NVAR
         DO 29 J=1,NVAR
               A(I+1,J)=0.0D0
               A(NVAR+I+1,J)=0.0D0
29       CONTINUE
         A(I+1,I)=1.0D0
         A(NVAR+I+1,I)=-1.0D0
30    CONTINUE
      B(1)=0.0D0
      DO 35 I=1,NVAR
         B(I+1)=0.0D0
         B(NVAR+I+1)=-1.0D0*CPAR
35    CONTINUE
      DO 36 I=1,NVAR
         G(I)=-1.0D0
36    CONTINUE
      DO 40 I=1,NVAR
         DO 39 J=1,NVAR
               H(I,J)=YV(I)*YV(J)*GRMAT(I,J)
39       CONTINUE
40    CONTINUE
      CALL DQPROG(NVAR,NCON,NEQ,A,LDA,B,G,H,LDH,DIAG,SOL,NACT,
     & IACT,ALAM)
      DO 45 I=1,NVAR
         IF (DABS(SOL(I)).LT.EP) SOL(I)=0.0D0
```

```
        IF (DABS(SOL(I)-CPAR).LT.EP) SOL(I)=CPAR
    AL(I)=SOL(I)
45  CONTINUE

    DO 200 J=1,NNPMMIN
        IPERM(J)=J
        S=0.0D0
                DO 199 I=1,NNPMMIN
                        S=S+AL(I)*YV(I)*GRMAT(I,J)
199             CONTINUE
        FW(J)=S
200 CONTINUE
    CALL DSVRGP(NNPMMIN,FW,FWR,IPERM)
    DO 205 I=1,NNPMMIN
        YVR(I)=YV(IPERM(I))
205 CONTINUE

    BPAR=-FWR(1)+1.0D0
    BOPT=BPAR
    NFOUTE=NNIN
    NFOUTEOPT=NFOUTE
    NTEL=0
210 NTEL=NTEL+1
    BPAR=-(FWR(NTEL)+FWR(NTEL+1))/2.0D0
    IF (YVR(NTEL).LE.0.0D0) NFOUTE=NFOUTE-1
    IF (YVR(NTEL).GT.0.0D0) NFOUTE=NFOUTE+1
    IF (NFOUTE.LT.NFOUTEOPT) THEN
        NFOUTEOPT=NFOUTE
        BOPT=BPAR
    ENDIF
    IF (NTEL.LE.NNPMMIN-2) GOTO 210
RETURN
END
```

```
SUBROUTINE BERFOUTSVMUIT(YVIN,GRNUUT,YVUIT,ALPHA,BOPT,FOUT)
C    TRAINS AN SVM ON THE OUT- (TEST) DATA
C    USED IN SUBROUTINE CROSSVAL
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     PARAMETER (IP=50,NN=100,MM=100,NNPMM=NN+MM)
     PARAMETER (NCV=5)
     PARAMETER (NNUIT=NN/NCV,MMUIT=MM/NCV,NNIN=NN-NNUIT,MMIN=MM-MMUIT)
     PARAMETER (NNPMMIN=NNIN+MMIN,NNPMMUIT=NNUIT+MMUIT)
     DIMENSION GRNUUT(NNPMMUIT,NNPMMIN),ALPHA(NNPMMIN)
     DIMENSION YVIN(NNPMMIN),YVUIT(NNPMMUIT)
     FOUT=0.0D0
     DO 10 I=1,NNPMMUIT
          TOETS=1.0D0
          S=BOPT
          DO 5 J=1,NNPMMIN
               S=S+ALPHA(J)*YVIN(J)*GRNUUT(I,J)
5         CONTINUE
     IF (S.LT.0.0D0) TOETS=-1.0D0
     IF (DABS((YVUIT(I)-TOETS)).GT.0.1D0) FOUT=FOUT+1.0D0
10   CONTINUE
     FOUT=FOUT/NNPMMUIT
RETURN
END
C    END OF THE BERFOUTSVMUIT SUBROUTINE
```