Statistical Classification Procedures for Analysing Functional Data

Chané Orsmond

Report presented in partial fulfilment
of the requirements for the degree of
MCom Mathematical Statistics
at the University of Stellenbosch

Supervisor: Prof.S.J.Steel

December 2016

Plagiarism Declaration

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. I also understand that direct translations are plagiarism.
4. Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. I declare that the work contained in this assignment, except otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Signature	
Initials and Surname	**Date**
C.L. Orsmond	December 2016

Abstract

Functional data are obtained through the measurement of one or more variables at a set of discrete evaluation points over a continuum such as time, wavelength or values of a spatial variable. Functional extensions of traditional statistical methods are considered in the analyses of such data sets, which are typically comprised of a sample of functions.

Linear discriminant analysis for functional data and functional support vector machines are investigated in this thesis as binary functional classification procedures. To address the high correlations which typically exist amongst the input features of a functional data set, the fused lasso, which selects contiguous intervals of variables, is discussed. In addition, a sparse equivalent of partial least squares (SPLS), which achieves simultaneous variable selection and dimension reduction, is considered in a functional context.

An infrared spectroscopy data set is considered for practical implementation of the fore-mentioned functional data analysis techniques. The procedures are compared in terms of classification accuracy and variable selection properties, reported in the results of an empirical study.

Key words: Functional data, functional data analysis (FDA), linear discriminant analysis (LDA) for functional data, functional support vector machines (FSVM), fused lasso, sparse partial least squares (SPLS), spectroscopy data.

Table of Contents

List of Figures

Chapter 1

Introduction

Rapid technological advances and sophisticated data collection techniques have allowed capturing of a vast magnitude of data occurring in extremely varied forms. In particular, data that can be represented as a set of continuous, smooth functions varying over a continuum forms the focus area of this study. The collection of statistical techniques utilised in the analysis of such *functional* data sets is known as *functional data analysis* (FDA).

More specifically, functional data are obtained through the measurement of one or more variables at usually a large number of discrete evaluation points over a continuum. The continuum is often time, but can also be other measures, such as wavelength or values of a spatial variable. The observed values in a functional context can therefore be ordered in a natural way, a distinguishing property of functional data that is induced by the evaluation points as measured over the continuum. Furthermore, as a result of the smoothness of the curves representing the sample items, information provided by their derivatives can also be utilised, a property which is unique to functional data (Ramsay, 2013).

Since many classical statistical techniques can be extended for use in a functional context, FDA is a field of study with a large potential scope. In particular, functional support vector machines (FSVM) and linear discriminant analysis (LDA) as applied to functional data are investigated as binary (functional) classification techniques in this study.

Another important property of functional data is the high correlations which typically occur amongst the input features, especially those that are closely situated on the underlying continuum. Consequently, feature selection is an important step in the analysis of such data to address multicollinearity, an observation that is strengthened by the fact that functional data sets can easily be wide, *i.e.* have more input features than sample data cases. In this study focus is restricted to variable selection using the fused lasso as well as a sparse version of partial least squares (SPLS), which incorporates feature selection into the dimension reduction procedure of standard PLS.

The analysis of a practical data set is considered as part of this study. In particular, functional data consisting of near-infrared (NIR) spectroscopic measurements on a sample of table grapes intended for export are available. A problem of commercial importance is to predict discolouration based on spectral measurements made soon after harvesting. It would be greatly beneficial to the grape export industry to be able to predict which grapes are likely to discolour, before they are exported. This would make it possible to channel grapes deemed fairly likely to discolour to a local market, thereby limiting financial losses to the industry as a consequence of discolouration. The fore-mentioned functional binary classification methods are implemented for this purpose.

In addition, identifying spectral variables that are good predictors of discolouration provides another important objective of the practical analysis. The performance of the fused lasso and sparse partial least squares are compared as feature selection procedures. Reducing the number of wavelengths at which NIR measurements are required to achieve accurate classification has the added advantage of reducing the time and cost of data collection (personal communication by an expert in the field).

Chapter 2

Basic Concepts in Functional Data Analysis

The focus of this chapter is on basic concepts that are of fundamental importance in an analysis of functional data. The discussion is based largely on Ramsay and Silverman (2005).

2.1 Introduction and notation

Functional data arise when the observed values can be ordered in a natural way. In many cases these values correspond to distinct time points, but the ordering can also arise because the values correspond to distinct values of some other continuum, such as wavelength, weight or values of a spatial variable.

A crucial assumption in FDA is that the observed values, which will be denoted by y, are obtained from an underlying function which will be denoted by $x(\ )$, or simply x. The function x is assumed to be smooth (for example, at least its first few derivatives exist), and the observations represent discretely sampled, noise contaminated values of this function corresponding to distinct evaluation points t. Consequently, the relationship between y and x is frequently expressed as

$$y = x(t) + \varepsilon . \qquad (2.1)$$

In this expression ε is a noise or error term and its presence causes the discrete observations to appear non-smooth or spiky. The form of the relationship in Equation (2.1) implies that the mechanism generating the observed data can be decomposed into a signal and error component.

Normality of the error component may often be considered in an analysis of functional data, i.e. $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma} = \mathbf{I})$. Here, the vector $\boldsymbol{\varepsilon}$ contains the error terms corresponding to a vector $\mathbf{t}$ of evaluation points. The stated distribution implies that the errors are infact independently and identically distributed normal random variables. However, according to Ramsay and Silverman (2005), the Gaussian assumption may be too simple for most functional data sets and more complex distributions, in particular those which allow the covariance matrix $\mathbf{\Sigma}$ to vary over the continuum t, may be considered more suitable. More specifically, we may assume $\mathbf{\Sigma} = \sigma_t^2 \mathbf{I}_p$, implying that the variance of the error term changes with t.

The assumptions of the existence of an underlying function x from which the observed, discrete data are generated, as well as the smoothness of this function, are central in characterising data that fall into a functional framework, distinguishing this from the (simply) multivariate case.

Functional data containing only a single sample item, or data case, and the p evaluation points $t_1, t_2, ..., t_p$, consist of the $p-$ dimensional vector of discrete observations $\mathbf{y} = \left[y_1, y_2, \ldots, y_p\right]^T$ obtained according to the relationship $y_j = x\left(t_j\right) + \varepsilon_j$, $j = 1, 2, \ldots, p$. The notation y_j is used rather than $y\left(t_j\right)$ since we do not view $y(.)$ as a function. These data should be viewed as a single entity, *i.e.* discrete observations of the single underlying function $x(\ )$, rather than a sequence of individual data values.

Now consider a functional data set, within the same framework, corresponding to N sample items which can be viewed as N realisations of an underlying function $x(\ )$. The observations for the i^{th} sample item are contained in a vector $\mathbf{y}_i = \left[y_{i1}, y_{i2}, \ldots, y_{ip}\right]^T$, $i = 1, 2, ..., N$. This represents the most simple functional case. However, it may be that the discrete observations for different sample items correspond to different numbers and/or entirely different sequences of evaluation points $\mathbf{t}_i = \left[t_{i1}, t_{i2}, \ldots, t_{ip_i}\right]^T$, $i = 1, 2, \ldots, N$. To further complicate matters, the interval over which the data are collected may vary between the N sample items. In such cases, a functional analysis should include registration (see Section 2.7). Moving forward, notation is restricted to the first, simpler case.

The observed data set can be summarised in an $N \times p$ data matrix $\mathbf{Y} = \left[y_{ij} \right]$, with rows corresponding to the sample items and columns corresponding to the evaluation points. In many practical applications we have $p > N$, when $\mathbf{Y}$ is a wide data matrix (more columns than rows).

2.2 Objectives of FDA

One of the primary objectives of an analysis of any functional data set is estimation of the unknown, underlying function $x(\ )$. In broad terms this task is achieved by smoothing the discrete observations contained in the vector $\boldsymbol{y}$. In addition, it may also be of interest to estimate the first few derivatives of $x(\ )$. Often, the slopes and curvatures of the functions in the data set, as reflected by their derivatives, may provide more insight into the process generating the data than would be made available by an analysis only at the level of the functions themselves. Consequently, curve derivative estimation can play a critical role in FDA. These derivatives will be denoted by $D^m x = \dfrac{d^m x}{dt^m}$, $m = 1, 2, \ldots$.

A common objective in many statistical analyses is using input variable information in order to explain a response. In the functional case, this goal amounts to using the observed y_{ij}-values, or possibly their smoothed versions, later denoted by $x_i\left(t_j\right)$, as predictor or response values in a supervised learning context. In a classification problem this can be accomplished by observing a response variable of which the value indicates the class to which a sample item belongs. In regression scenarios different possibilities exist. The (linear) relationship between a functional response and a categorical independent variable, or that between a scalar response and a functional independent variable, or even between a response and independent variable which are both functions can also be studied. In addition, multivariate extensions of these options may also be considered, e.g. a possibly functional response and several functional inputs.

Finally, functional data may be analysed using functional equivalents of well-known statistical techniques, such as principal components analysis (PCA) and canonical correlation analysis (CCA).

2.3 Smoothing functional data

In order to progress with an analysis of functional data it is necessary to convert the observed values, summarised in the matrix $\mathbf{Y}$, to smooth estimates of the functions $x_i(\ )$, where the index refers to the i^{th} sample item for $i = 1,2,\dots,N$. In the absence of noise, this task can be achieved simply through an interpolation of the observed data (Ramsay *et al.*, 2009, p.13). However, in the case of noise contamination of the observations, which is far more frequent in a functional data context, smoothing of the discrete observations is necessary. This task, the extraction of a smooth, stable estimate of the underlying function from discrete observations that exhibit a degree of noise, is one of the challenges faced in an FDA. Sufficient data for the execution of this task is dependent on the size of the signal to noise ratio (SNR) as well as the sampling rate, or resolution, of the data. The latter is a measure that compares the number of evaluation points and the (hypothesised) degree of curvature of the underlying function x, emphasising that a large number p of discrete observations $\boldsymbol{y} = \left[y_1, y_2, \dots, y_p\right]^T$ does not necessarily imply sufficient data for the task of estimation in a functional context.

Functional data are often stored and analysed in terms of a set of basis functions. There are several reasons for this. Firstly, we would like to view and work with each data item as a continuous function x, rather than a set of discretely sampled observations. Secondly, using a basis expansion makes it possible to deal with cases where the different sample items correspond to different sets of evaluation points. Thirdly, a basis expansion can be used to obtain and manipulate derivatives of x. Fourthly, using a basis expansion enables one to smooth the discrete observations. Finally, a basis expansion can be used to reduce the size of the data matrix that needs to be stored, sometimes dramatically so.

Denote the postulated basis functions by $\varphi_1(t), \varphi_2(t), \dots, \varphi_M(t)$, where typically $M < p$. A basis function expansion for $x(t)$ is of the form

$$x(t) = \sum_{m=1}^{M} \xi_m \varphi_m(t) . \tag{2.2}$$

The values of the coefficients $\xi_1, \xi_2, \ldots, \xi_M$ in this expansion are unknown and have to be estimated from the data. Using estimated values of $\xi_1, \xi_2, \ldots, \xi_M$, we can compute an approximate value of x at any value t from Equation (2.2), since the set of basis functions $\boldsymbol{\varphi}(t)$ is known. In this way, the above basis expansion achieves an approximate representation of the infinite-dimensional function $x(\ )$ in terms of the finite dimensional vector of coefficients $\boldsymbol{\xi}$. The dimensionality is reduced to M , which corresponds to the number of basis functions considered in the expansion. Furthermore, the value of this parameter determines the degree of smoothing applied to the discretely sampled observations $\boldsymbol{y} = \left[y_1, y_2, \ldots, y_p \right]^T$ in order to produce the smooth estimate of $x(\ )$.

In particular, for the evaluation points $\mathbf{t} = \left[t_1, t_2, \ldots, t_p \right]^T$ we obtain the smoothed version of the vector $\mathbf{y}$, *viz.*

$$\hat{\mathbf{x}}(\mathbf{t}) = \hat{x}(\mathbf{t}) = \begin{bmatrix} \hat{x}(t_1) \\ \hat{x}(t_2) \\ \vdots \\ \hat{x}(t_p) \end{bmatrix} = \begin{bmatrix} \sum_{m=1}^{M} \xi_m \varphi_m(t_1) \\ \sum_{m=1}^{M} \xi_m \varphi_m(t_2) \\ \vdots \\ \sum_{m=1}^{M} \xi_m \varphi_m(t_p) \end{bmatrix} = \mathbf{B}\boldsymbol{\xi} , \tag{2.3}$$

where $\mathbf{B}$ is a $p \times M$ matrix with elements $B_{jm} = \varphi_m(t_j)$, and $\boldsymbol{\xi} : M \times 1$ is a vector which has to be estimated from the data. Note that the rows of $\mathbf{B}$ therefore correspond to the evaluation points, while the columns correspond to the basis functions.

If $M < p$ (which usually is the case), least squares can be used to determine $\xi_1, \xi_2, \ldots, \xi_M$. In fact, the least squares estimates of $\xi_1, \xi_2, \ldots, \xi_M$ can be found from:

$$\begin{aligned}
\hat{\xi} &= \underset{\xi}{\arg\min}\, \| \mathbf{y} - \mathbf{B}\xi \|^2 \\
&= \underset{\xi}{\arg\min} \sum_{j=1}^{p} \left[y_j - \sum_{m=1}^{M} \xi_m \varphi_m \left(t_j \right) \right]^2 \\
&= \underset{\xi}{\arg\min} \left(\mathbf{y} - \mathbf{B}\xi \right)^T \left(\mathbf{y} - \mathbf{B}\xi \right) \\
&= \left(\mathbf{B}^T \mathbf{B} \right)^{-1} \mathbf{B}^T \mathbf{y}
\end{aligned} \quad . \qquad (2.4)$$

For notational simplicity we will write ξ instead of $\hat{\xi}$ and x (or $\mathbf{x}$) instead of $\hat{x}$ (or $\hat{\mathbf{x}}$), but it should be borne in mind that $\mathbf{B}\xi$ represents an estimated smooth of the observed data vector $\mathbf{y} = \left[y_1, y_2, \ldots, y_p \right]^T$.

Naturally every y_i has its own vector of coefficients, $\xi_i = \left[\xi_{i1}, \xi_{i2}, \ldots, \xi_{iM} \right]^T$ and we can write $\mathbf{y}_i = \mathbf{B}\xi_i + \boldsymbol{\varepsilon}_i$, $i = 1, 2, \ldots, N$. We therefore have the following approximation for the observed data matrix:

$$\mathbf{X} : N \times p = \Xi \mathbf{B}^T , \qquad (2.5)$$

where the i^{th} row of the matrix $\mathbf{X}$ is given by the transpose of the vector $\mathbf{x}_i(t) = \mathbf{B}\xi_i$. In this expression the $N \times M$ matrix Ξ has the vectors $\xi_1^T, \xi_2^T, \ldots, \xi_N^T$ as rows, and we now need to store only this matrix instead of $\mathbf{Y}$, which can be a substantial saving when $M \ll p$. The value of M determines the degree of smoothing of the observed data, with a small value of M implying a large amount of smoothing. If $M = p$, we interpolate the data.

Functional data do not always arise from discrete observations corresponding to the same set of evaluation points $\mathbf{t} = \left[t_1, t_2, \ldots, t_p \right]^T$ for the entire sample and it may be the case that each sample item considers a different set of evaluation points, i.e. $\mathbf{t}_i = \left[t_{i1}, t_{i2}, \ldots, t_{ip_i} \right]^T$. In such a case, each of the N sample items corresponds to its own matrix $\mathbf{B}_i$, with rows corresponding to the sample specific evaluation points $\mathbf{t}_i$, and the set of basis functions

$\varphi_1, \varphi_2, \ldots, \varphi_M$, which are the same for each item, as columns. The least squares coefficients for the i^{th} sample item can then be obtained from $\boldsymbol{\xi}_i = \left(\mathbf{B}_i^T \mathbf{B}_i\right)^{-1} \mathbf{B}_i^T \mathbf{y}_i$. It should be noted that in such a scenario the result in Equation (2.5) does not hold, since each item corresponds to a different matrix $\mathbf{B}_i$, $i = 1, 2, \ldots, N$.

In this way, the use of a basis expansion for the underlying function $x(\ )$ makes it possible to deal with cases where the different sample items correspond to different sets of evaluation points, since the resulting estimate $x_i(t) = \sum_{m=1}^{M} \xi_{im} \varphi_m(t)$ can be evaluated at any time t for any of the N sample items.

2.4 Smoothing functional data using regularisation

In some cases, the least squares coefficient estimates may produce an estimated function $\hat{x} = \sum_{m=1}^{M} \hat{\xi}_m \varphi_m$ that is not sufficiently smooth. Regularisation is one approach that can be utilised in order to address this problem to produce an estimated function that provides a more satisfactory fit to the data with regard to the requirement of smoothness. Furthermore, this method allows explicit control of the degree of smoothness of the approximation. It may also be noted that least squares estimation can only be considered in cases where $M < p$, a requirement that is not needed when regularisation is applied, which may even consider the extreme case where $M = p$.

Broadly, the regularisation approach achieves smoothness of the estimated function by restricting its complexity. The optimization criterion within this framework is obtained by the addition of a penalty term to the usual sum of squares considered by least squares estimation. The resulting criterion is known as the penalised sum of squares:

$$PENSSE(\boldsymbol{\xi}) = \sum_{j=1}^{p} \left[y_j - \sum_{m=1}^{M} \xi_m \varphi_m(t_j) \right]^2 + \lambda \int \left[L_x(t) \right]^2 dt, \qquad (2.6)$$

where L_x is a functional penalising the roughness, or complexity, of $x(\ )$, and λ is known as the regularisation parameter.

One measure of the complexity or roughness of the target function $x(\ )$ is provided by its curvature. Mathematically, the curvature of a function x at the argument value t is measured by $D^2x(t) = \frac{d^2x}{dt^2}$. This definition is intuitively acceptable since a linear function, which has no curvature, has a zero second derivative (Ramsay *et al.*, 2009, p. 62). Consequently, a reasonable choice of the penalty functional reflecting the curvature, or roughness, of the function $x(\ )$ is provided by

$$L_x(t) = D^2x(t). \tag{2.7}$$

Estimation of the underlying function according to the penalised criterion $PENSSE(\xi)$ involves a trade-off between goodness of fit, as measured by the first term of the criterion, and smoothness of the fitted function. These two goals represent conflicting objectives of curve estimation. The value of the regularization parameter determines the degree of smoothing and, consequently, controls this trade-off.

A large value of the smoothing parameter implies a more severe penalty on the curvature of the estimated $x(\ )$ resulting in a smoother estimated function. For $\lambda = 0$, no regularisation takes place and $\hat{x}$ will be a jagged function that interpolates the data. In contrast, as $\lambda \to \infty$ the estimated function approaches the least squares fit, which has a zero value of the penalty term, *i.e.* $L_x(t) = D^2x(t) = 0$. An appropriate value of λ may be chosen subjectively or, more commonly, by a data-dependent method such as cross-validation.

Following the definition of the penalty functional in Equation (2.7), estimates of the coefficients considered in the basis function expansion of $x(\ )$, and consequently an estimate of the function itself, can be obtained by minimisation of the penalised sum of squares:

$$\begin{aligned}
\hat{\boldsymbol{\xi}} &= \underset{\boldsymbol{\xi}}{\arg\min}\left\{\sum_{j=1}^{p}\left[y_j - \sum_{m=1}^{M}\xi_m\varphi_m\left(t_j\right)\right]^2 + \lambda\int\left[D^2x(t)\right]^2 dt\right\} \\
&= \underset{\boldsymbol{\xi}}{\arg\min}\left\{\sum_{j=1}^{p}\left[y_j - \sum_{m=1}^{M}\xi_m\varphi_m\left(t_j\right)\right]^2 + \lambda\int\boldsymbol{\xi}^T\left[D^2\boldsymbol{\varphi}(t)\right]\left[D^2\boldsymbol{\varphi}(t)\right]^T\boldsymbol{\xi}\,dt\right\}, \qquad (2.8) \\
&= \underset{\boldsymbol{\xi}}{\arg\min}\left\{\sum_{j=1}^{p}\left[y_j - \sum_{m=1}^{M}\xi_m\varphi_m\left(t_j\right)\right]^2 + \lambda\boldsymbol{\xi}^T\int\left[D^2\boldsymbol{\varphi}(t)\right]\left[D^2\boldsymbol{\varphi}(t)\right]^T dt\,\boldsymbol{\xi}\right\}
\end{aligned}$$

based on the expansion $x(t) = \sum_{m=1}^{M}\xi_m\varphi_m(t) = \boldsymbol{\xi}^T\boldsymbol{\varphi}(t)$. Defining the $M \times M$ matrix Ω_B with jk^{th} element given by $\int\varphi_j''(t)\,\varphi_k''(t)\,dt$, the above criterion can equivalently be represented in matrix notation as follows:

$$\begin{aligned}
\hat{\boldsymbol{\xi}} &= \underset{\boldsymbol{\xi}}{\arg\min}\left\{\left\|\mathbf{y} - \mathbf{B}\boldsymbol{\xi}\right\|^2 + \lambda\boldsymbol{\xi}^T\Omega_B\,\boldsymbol{\xi}\right\} \\
&= \underset{\boldsymbol{\xi}}{\arg\min}\left\{\left(\mathbf{y} - \mathbf{B}\boldsymbol{\xi}\right)^T\left(\mathbf{y} - \mathbf{B}\boldsymbol{\xi}\right) + \lambda\boldsymbol{\xi}^T\Omega_B\,\boldsymbol{\xi}\right\}
\end{aligned}, \qquad (2.9)$$

the solution of which is obtained as $\hat{\boldsymbol{\xi}} = \left(\mathbf{B}^T\mathbf{B} + \lambda\Omega_B\right)^{-1}\mathbf{B}^T\mathbf{y}$.

2.5 Supervised learning with functional data

We now consider supervised learning in a functional data context in more detail. Throughout this discussion it should be borne in mind that we assume a sample of N data cases, indexed by $i = 1, 2, \ldots, N$. In a supervised context we therefore also require N observations of a possibly functional response and corresponding, possibly functional observations of input variables. Several different cases can be distinguished. It should be noted that all of the discussed cases, with the exception of the first classification case, consider functional linear models. In a functional context, a model is referred to as linear when no effect of a transformation of the function $x(\ )$ on the response is considered. The use of such a model is reasonable since a transformation of the underlying function, such as $\log\left[x(\ )\right]$ or $x^2(\ )$, simply defines a different function.

2.5.1 Classification

Firstly, it is possible to consider a classification problem where the response, which we will denote by Z, is qualitative, assuming values in a set $\{1,2,\ldots,U\}$ of labels. If $U=2$, we have a binary classification problem. The objective here is to develop a procedure which will take a p-vector $\mathbf{y}$ as input, possibly smooth the values in this vector, and produce an output from $\{1,2,\ldots,U\}$. In such a scenario each of the training data cases can be represented as $(\mathbf{y}_i, z_i)$.

Functional extensions of traditional classification techniques, such as linear discriminant analysis and support vector machines, which can be used to classify new curves, are discussed in Chapter 3.

2.5.2 Categorical input and a functional response

Secondly, it is possible to study the relationship between a functional response variable and a categorical input variable. Therefore the response is now the function x, while the input variable, say g, will be qualitative, assuming values from a set $\{1,2,\ldots,G\}$, with $G \geq 2$. An example of such a scenario is where we investigate the precipitation $x(t)$ over time t in each of G regions. The assumed model is

$$x_{ig}(t) = \mu(t) + \alpha_g(t) + \varepsilon_{ig}(t), \tag{2.10}$$

with $g=1,2,\ldots,G$ and $i=1,2,\ldots,N_g$. Here $N_1+N_2+\ldots+N_G = N$. The sample cases here correspond to weather stations, with N_g stations in region g. Note that we have effectively switched around the input and response from the first scenario. Also, in fitting (2.10) we use the smoothed y-values as response observations, and we estimate $\mu(t)$ and $\alpha_g(t)$ by computing appropriate averages. Fitting this model is therefore very similar to fitting a 1-way ANOVA model. Note however that both $\mu(t)$ and $\alpha_g(t)$ will be estimated in terms of basis function expansions (so that we can compute values of these functions at any argument t).

In particular, an estimate of the function $\mu(t)$ can be obtained from pointwise averaging of the smooth curves $x_i(\ )$, $i = 1, 2, \ldots, N$:

$$\begin{aligned}\hat{\mu}(t) &= \frac{1}{N}\sum_{i=1}^{N} x_i(t) \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} \xi_{im}\varphi_m(t) \\ &= \sum_{m=1}^{M}\varphi_m(t)\left\{\frac{1}{N}\sum_{i=1}^{N}\xi_{im}\right\} \\ &= \sum_{m=1}^{M}\varphi_m(t)\,\bar{\xi}_m\end{aligned}. \tag{2.11}$$

Here, the vector $\bar{\xi}_m$ reflects the average coefficient estimate corresponding to the m^{th} basis function over the N sample items. A similar result holds for estimation of the function $\alpha_g(t)$, which reflects the deviation of the g^{th} group mean from the overall mean:

$$\hat{\alpha}_g(t) = \frac{1}{N_g}\sum_{i=1}^{N_g} x_{ig}(t) - \hat{\mu}(t). \tag{2.12}$$

2.5.3 Functional input and a scalar response

Thirdly, it may be the case that each functional input, $x_i(\ )$ for the i^{th} sample item, corresponds to some value of a scalar response variable z_i, $i = 1, 2, \ldots, N$. A linear effect of the independent variable on the response may be modelled by:

$$z_i = \mu + \int_{T_1}^{T_2} x_i(t)\beta(t)\,dt + \varepsilon_i\ , \quad \text{for } i = 1,2,\ldots,N\ . \tag{2.13}$$

Here, μ and ε are the respective intercept and (data case specific) error terms and the values of T_1 and T_2 are determined by the response definition. As an example, the functional independent variable may be the temperature $x(t)$ at time t, with t measured in days, for each of N weather stations and the response z_i may be defined as the total annual amount of precipitation for the i^{th} weather station, implying the integration limits $T_1 = 0$ and $T_2 = 365$:

$$\begin{aligned} z_i &= total\ precipitation \\ &= \int_0^{365} z_i(t)\,dt \\ &= \mu + \int_0^{365} \beta(t)\,x_i(t)\,dt + \varepsilon_i \end{aligned} \quad . \qquad (2.14)$$

The above model considers $z(t)$, the precipitation at time t, to be reasonably explained by the functional covariate temperature $x(t)$.

The unknown coefficients $\boldsymbol{\beta}$ considered in a multiple regression setting are replaced in the functional context by the regression coefficient function $\beta(\ )$, the functional parameter of interest. This connection can easily be seen when comparing the multiple linear regression model $z_i = \beta_0 + \sum_{j=1}^{p} \beta_j y_{ij} + \varepsilon_i$, $i = 1,2,...,N$, to the model (Equation (2.13)) considered in a functional analysis of the same (discrete) data.

Since the space of functions satisfying the model in Equation 2.13 is of infinite dimensionality, it is necessary to incorporate smoothing into the estimation process of the unknown coefficient function. In addition to the basis function expansion of the underlying functions $x(t) = \sum_{m=1}^{M} \xi_m \varphi_m(t) = \boldsymbol{\xi}^T \boldsymbol{\varphi}(t)$, for each sample item, smoothing can be achieved in this framework by considering a basis function expansion of the unknown coefficient function $\beta(\ )$. This approach reduces the dimensionality of the problem by restricting the form of the candidate (coefficient) function.

Denoting these basis functions for representing $\beta(t)$ by $\theta_1(t),...,\theta_K(t)$, a basis function expansion of $\beta(t)$ has the form:

$$\beta(t) = \sum_{k=1}^{K} b_k \theta_k(t) = \boldsymbol{b}^T \boldsymbol{\theta}(t), \tag{2.15}$$

where $\boldsymbol{b} = [b_1, b_2, ..., b_K]^T$ are the coefficients of the expansion. This result implies that the assumed model can be written as

$$\begin{aligned} z_i &= \mu + \int_{T_1}^{T_2} \xi_i^T \boldsymbol{\varphi}(t) \boldsymbol{b}^T \boldsymbol{\theta}(t)\, dt + \varepsilon_i \\ &= \mu + \sum_{m=1}^{M} \sum_{k=1}^{K} \xi_{mi} b_k \int_{T_1}^{T_2} \varphi_m(t) \theta_k(t)\, dt + \varepsilon_i \end{aligned} , \tag{2.16}$$

for the i^{th} sample item.

It may be reasonable to consider the same system of basis functions in the expansion of both the underlying and coefficient functions. Furthermore, it is expected that $K > M$ since it is likely that a greater degree of smoothing will be required to obtain a smooth estimate of the underlying function $x(\ )$ in comparison to the coefficient function $\beta(\ )$.

The basis function expansion of the underlying function $x(\ )$, at evaluation point t, for each of the N sample items under consideration implies the following result:

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_N(t) \end{bmatrix} = \begin{bmatrix} \xi_1^T \boldsymbol{\varphi}(t) \\ \xi_2^T \boldsymbol{\varphi}(t) \\ \vdots \\ \xi_N^T \boldsymbol{\varphi}(t) \end{bmatrix} = \boldsymbol{\Xi} \boldsymbol{\varphi}(t). \tag{2.17}$$

Defining $\boldsymbol{z} = [z_1, z_2, ..., z_N]^T$, the vector of responses, and applying the result above, the assumed linear model can be written in the form:

$$\begin{aligned} \boldsymbol{z} &= \mu\boldsymbol{1} + \int_{T_1}^{T_2} \Xi\boldsymbol{\varphi}(t)\boldsymbol{\theta}^T(t)\boldsymbol{b}dt + \boldsymbol{\varepsilon} \\ &= \mu\boldsymbol{1} + \Xi J_{\varphi\theta}\boldsymbol{b} + \boldsymbol{\varepsilon} \end{aligned} \quad , \tag{2.18}$$

where $J_{\varphi\theta}$ is the $M \times K$ matrix given by $J_{\varphi\theta} = \int_{T_1}^{T_2} \boldsymbol{\varphi}(t)\boldsymbol{\theta}^T(t)dt$, with $(m,k)^{th}$ element $\int_{T_1}^{T_2} \varphi_m(t)\theta_k(t)dt$. Notation can be further simplified by defining the vector $\boldsymbol{b}^* = \begin{bmatrix} \mu \\ \boldsymbol{b} \end{bmatrix}$ and the coefficient matrix $W = \begin{bmatrix} \boldsymbol{1} & \Xi J_{\varphi\theta} \end{bmatrix}$. The N predicted scalar response values $\hat{z}_i$, $i = 1,2,...,N$, can then be obtained from the relationship:

$$\hat{\boldsymbol{z}} = \hat{W}\hat{\boldsymbol{b}}^* , \tag{2.19}$$

after completion of the following estimation steps:

1. The estimated coefficient matrix $\hat{\Xi}$, implied by the basis function expansion of the underlying functions $x(\ )$ for each sample item, is obtained based on an ordinary least squares approach or from the inclusion of a regularisation penalty. This then gives the estimated matrix $\hat{W}$, after evaluation of the matrix $J_{\varphi\theta}$.

2. Estimates of the coefficients $\boldsymbol{b}$ considered in the basis function expansion of the coefficient function $\beta(\ )$ are obtained from the equation $\hat{\boldsymbol{b}} = \left(\hat{W}^T\hat{W}\right)^{-1}\hat{W}^T\boldsymbol{z}$. Further, an estimate of the intercept term in the assumed model is given by $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} z_i$, producing the estimated vector $\hat{\boldsymbol{b}}^* = \begin{bmatrix} \hat{\mu} \\ \hat{\boldsymbol{b}} \end{bmatrix}$.

Estimating the unknown coefficient function $\beta(\ )$ by considering roughness penalties provides an alternative and more flexible regularisation approach than using the above described method of restricted basis functions. Within this framework, the problem of infinite dimensionality is addressed by imposing a constraint on the complexity of the function of interest, rather than assuming it to be of a specific form.

An estimate of the coefficient function $\beta(\ )$, as well as the intercept μ, can therefore also be obtained by minimisation of a penalised sum of squares

$$PENSSE(\mu,\beta)=\sum_{i=1}^{N}\left[z_i-\mu-\int_{T_1}^{T_2}x_i(t)\beta(t)dt\right]^2+\lambda\int\left[L\beta(t)\right]^2 dt\ , \tag{2.20}$$

where λ is the regularisation parameter and L is a linear differential operator that penalises $\beta(t)$ for complexity, the choice of which will be determined by the nature of the functional data under consideration (Ramsay & Silverman, 2005, p.266). As usual, the value of the parameter λ can be determined using cross-validation or may be selected in a more subjective manner.

Minimisation of the above penalised criterion with respect to μ, keeping the value of $\beta(t)$ fixed, yields the estimate $\hat{\mu}=\frac{1}{N}\sum_{i=1}^{N}z_i$ which corresponds to the estimate of this parameter used in the restricted basis function smoothing approach. Further computational details are addressed by Ramsay and Silverman (2005).

2.5.4 Functional input and response

As a fourth case, it is possible to consider a supervised learning problem where both the response and independent variable are functional in nature. In such a case, the input is the function $x(\ )$ and the response, denoted by z, is also considered to be a function of the continuum t . The assumed linear model is of the form

$$z_i(t)=\mu(t)+x_i(t)\beta(t)+\varepsilon_i(t)\ ,\quad \text{for } i=1,2,...,N\ , \tag{2.21}$$

where $\beta(\ )$ is again an unknown regression coefficient function that requires estimation. Within this framework, the vector of discrete observations $\boldsymbol{y}_i = \left[y_{i1}, y_{i2}, ..., y_{ip}\right]^T$, obtained at the set of evaluation points $\boldsymbol{t} = \left[t_1, t_2, ..., t_p\right]^T$, as well as discrete observations of the response $\boldsymbol{z}_i = \left[z_{i1}, z_{i2}, ..., z_{ip_z}\right]^T$ will be recorded for each of the N sample items. Here, the subscript p_z is used so that the set of evaluation points corresponding to the response observations is not restricted to be the same as that of the discrete input observations. A smooth estimate of the function $z_i(\ )$ is obtained from the discrete observations z_i by means of a basis function expansion.

Due to the functional nature of the response, the usual intercept term μ now also varies with t and can be reasonably approximated by the mean function $\hat{\mu}(t) = \bar{x}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t) = \sum_{m=1}^{M} \bar{\xi}_m \varphi_m(t)$. It is important to note that the mean function considers the smooth, estimated functions $x_i(\ )$, $i = 1,2,...,N$, rather than the discrete observations $\boldsymbol{y}_i$ so that its value at any argument t may be evaluated.

A regularised estimation approach using roughness penalties, similar to that discussed in the previous case of a scalar response paired with a functional independent variable, can be utilised to obtain an estimate of the unknown coefficient function $\beta(\ )$. Minimisation of the following penalised criterion is considered:

$$PENSSE(\beta) = \sum_{i=1}^{N} \int \left[z_i(t) - \hat{\mu}(t) - x_i(t)\beta(t)\right]^2 dt + \lambda \int \left[L\beta(t)\right]^2 dt \quad , \qquad (2.22)$$

the computational details of which are discussed by Ramsay and Silverman (2005).

The linear model given in Equation (2.21) is referred to as point-wise or concurrent, since the response evaluated at time t is assumed to depend only on the independent variable $x(\ )$

evaluated at the same time point. Other options are possible, such as a linear model allowing for a time-lag with regard to the effect of the independent variable on the response.

Each of the above four cases may be extended to consider multivariate functional data. In such a context, for example, discrete observations generated by two separate underlying functions, denoted by $x_1(\ )$ and $x_2(\ )$, are obtained for each of the N sample items. As an example, consider the scalar response z_i, the total annual amount of precipitation for the i^{th} weather station, explained by the two functional independent variables, viz. temperature $x_{i1}(t)$ and wind speed $x_{i2}(t)$, measured over time t (in days). A suitable linear model may have the form:

$$\begin{aligned} z_i &= total\ precipitation \\ &= \int_0^{365} z_i(t)\,dt \\ &= \mu + \int_{T_1}^{T_2} x_{i1}(t)\beta_1(t)\,dt + \int_{T_1}^{T_2} x_{i2}(t)\beta_2(t)\,dt + \varepsilon_i \end{aligned} \quad , \qquad (2.23)$$

for $i = 1,2,...,N$. Here, the precipitation at time t, $z(t)$, for each sample item is modelled in terms of the functional covariates temperature and wind speed, as opposed to the univariate case which considers only the temperature covariate to explain precipitation.

It should be noted, although it is not the focus of this study, that instead of considering a sample of N independent data cases $x_1(\ ),x_2(\ ),...,x_N(\ )$, functional data may also arise from a single long record.

2.6 Basis function systems

We turn to the choice of basis functions to be used. In order to obtain a useful approximation of the underlying function, basis functions that exhibit features matching those of the function $x(\ )$ being estimated should be considered and, consequently, the choice of the basis system to be used in an analysis is largely dependent on the nature of the functional data under consideration.

2.6.1 Fourier basis system

The Fourier basis system provides a natural choice for periodic data, which are characterised by repeated patterns or cycles in the data that occur at regular intervals known as periods. This system consists of the basis functions:

$$\varphi(t) = \left[1, sin(\omega t), cos(\omega t), sin(2\omega t), cos(2\omega t), \ldots, sin\left(\frac{M-1}{2}\omega t\right), \cos\left(\frac{M-1}{2}\omega t\right)\right]^T \quad (2.24)$$

for an expansion that considers M basis functions. The period of the data, determined by the frequency parameter ω, is defined to be $\frac{2\pi}{\omega}$. Since sine and cosine functions are simple to derive, use of the Fourier basis system in an analysis of functional data makes it particularly straightforward to obtain estimates of derivatives of the underlying function.

2.6.2 B-spline basis system

The B-spline basis system provides the most common choice for non-periodic functional data, a topic which first requires some theory on splines. Broadly, the use of splines enables flexible estimation of an unknown function which, in a functional context, is the underlying function $x(\ )$.

Flexibility is introduced by firstly dividing the interval of the continuum t over which evaluation of the function $x(\ )$ is of interest into sub-intervals, by specifying a sequence of knots, and then fitting a separate (low degree) polynomial of specified order q in each (James et al., 2013, p.271). Adjacent polynomial functions that join at any particular knot are constrained to have equal function values at this junction point as well as equal derivatives up to order $q - 2$, to ensure sufficient smoothness of the resulting estimated function.

The degree of a polynomial function refers to its highest power, whilst its order q is the number of coefficients defining the polynomial, one more than the polynomial degree. The order of the overall spline fit to the data corresponds to the order of the polynomial fitted in each sub-interval, with a higher order implying a smoother fit. The most commonly used orders are $q = 1,2,4$ corresponding to piecewise constant functions, piecewise linear and piecewise cubic splines respectively (Hastie *et al.*, 2011, p.144).

The system of B-splines consists of a set of basis functions $\boldsymbol{\varphi}(t)$ which are themselves spline functions that are piecewise polynomials as defined by a knot sequence and an order q. Any linear combination of these (spline) basis functions is again a spline function. Consequently, obtaining an estimate of the underlying function using a basis function expansion $x(t) = \sum_{m=1}^{M} \xi_j \varphi_j(t)$ and a B-spline basis system produces a spline fit to the observed, discrete data points $\boldsymbol{y}$.

Construction of any spline function requires specification of the number of knots as well as their placement. The first problem can be addressed using the well-known method of cross-validation. Alternatively, the number of knots can be selected in a more subjective manner, for example, by comparing results and graphical representations for different choices of the value of this parameter. Further, many applications consider equally spaced knots throughout the interval of interest. However, this approach may not be adequate in cases where the data are not equally spaced. In such cases, alternative approaches include positioning knots at quantiles of the data or manually placing more knots in regions where the underlying function $x(\ )$ is believed to contain a higher degree of curvature.

Given a specified knot sequence, B-spline basis functions of order q are, by definition, non-zero over no more than q sub-intervals, a property of this basis system which is referred to as compact support. This property is crucial for efficient computation since it implies sparsity of the matrix $\boldsymbol{B}$ containing the values of each of the basis functions evaluated at the set of points $\boldsymbol{t}$. As seen before, estimation of the underlying function $x(\ )$ using a basis function expansion amounts to estimation of the basis coefficients $\xi_1, \xi_2, ..., \xi_M$. This task can be achieved using a least squares approach from the equation $\hat{\boldsymbol{\xi}} = \left(\boldsymbol{B}^T\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\boldsymbol{y}$, or by incorporating a degree of regularisation from $\hat{\boldsymbol{\xi}} = \left(\boldsymbol{B}^T\boldsymbol{B} + \lambda\Omega_B\right)^{-1}\boldsymbol{B}^T y$. Execution of these approaches, which both involve matrix multiplication, will require far less computation time due to the sparsity of the matrix $\boldsymbol{B}$, especially in the case of a very large number of evaluation points.

Figure 2.1 illustrates a system of B-spline basis functions of order $q = 4$, corresponding to cubic piecewise polynomials giving rise to each of the $M = 8$ spline basis functions. A total of four interior knots were placed uniformly over the interval $t \in [0,1]$. More generally, this set corresponds to the interval of interest of the continuum t considered in a functional data set. Furthermore, the compact support property is easy to see in the plot, and each basis function is seen to be non-zero over at most $q = 4$ of the sub-intervals.

The above discussion concerns estimation of the underlying function $x(\ )$ by means of a basis function expansion. More generally, the Fourier and B-spline basis systems can be used in estimation of other unknown functions of interest, examples of which include the coefficient function $\beta(\ )$ and the response function $z(\ )$ in the case of a functional response variable.

The choice of basis function system considered in an FDA is not restricted to the Fourier and B-spline bases. Other options include wavelets, exponential and power bases, polynomial bases as well as the simplest choice, the constant basis. Details on these topics are provided by Ramsay and Silverman (2005).

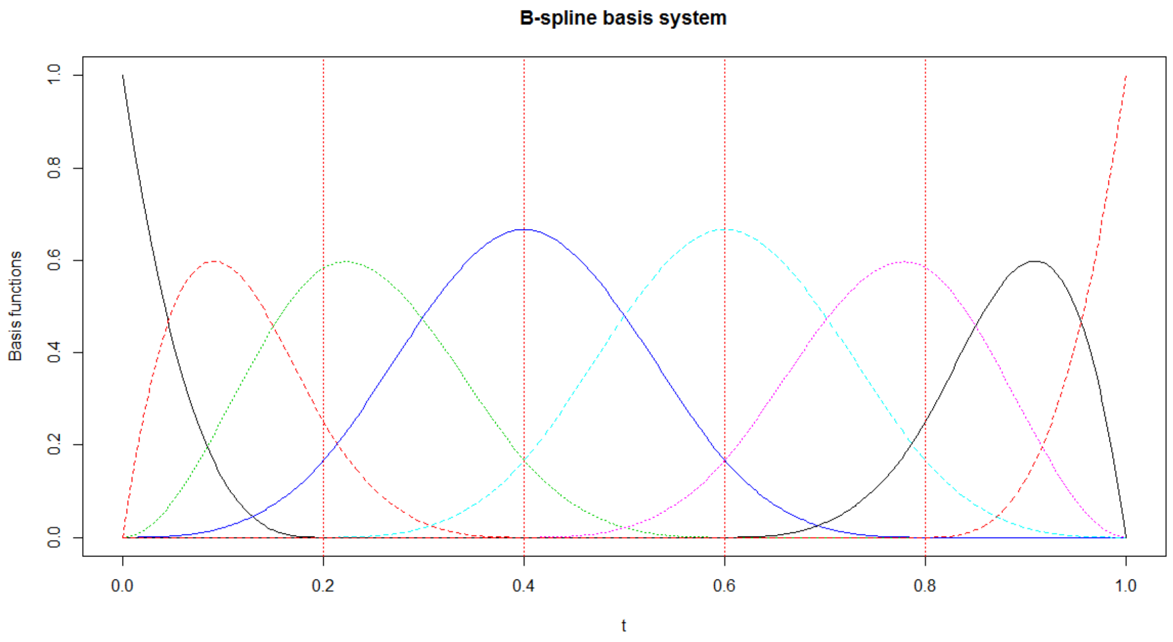

Figure 2.1

2.7 Curve registration

The estimated functions $x_i(\ ), i = 1,2,\ldots,N$, obtained from smoothing the discrete observations $\boldsymbol{y}_i$ in a functional data sample typically exhibit two different types of variation. The first is known as phase variation and refers to variation in the location of curve features along the horizontal axis (Ramsay et al., 2009, p.118). Secondly, the curves may also exhibit amplitude variation, seen in different sizes of the curves as measured on the vertical axis. Graphical representations of these two important types of variation are given in Figure 2.2.

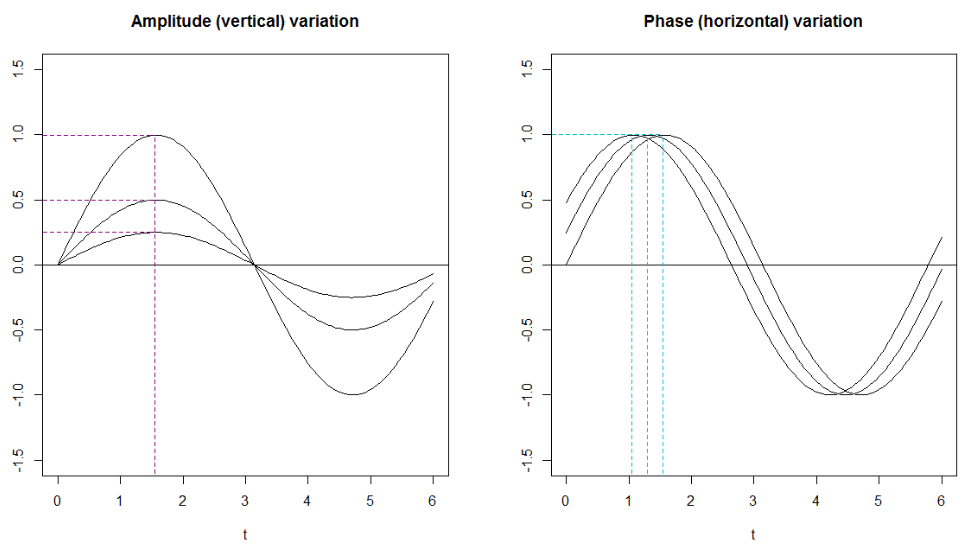

Figure 2.2

The functional equivalents of statistical methods, such as means and variances as well as more complex procedures such as principal components and canonical correlation analyses, are designed only to model amplitude variation (Ramsay et al., 2009, p.118). Consequently, aligning the estimated curves of each sample item so that they differ only in amplitude, a process which is known as curve registration, is a crucial step of any functional data analysis. The major part of the following discussion of this topic is based on Ramsay and Li (1998).

Ramsay et al. (2009) illustrate an example providing strong motivation for curve registration. In this functional sample, the curves $x(\ )$ represent the heights of 10 girls measured over time. Interest is in the acceleration curves, i.e. D^2x.

Comparison of registered and unregistered estimated acceleration curves

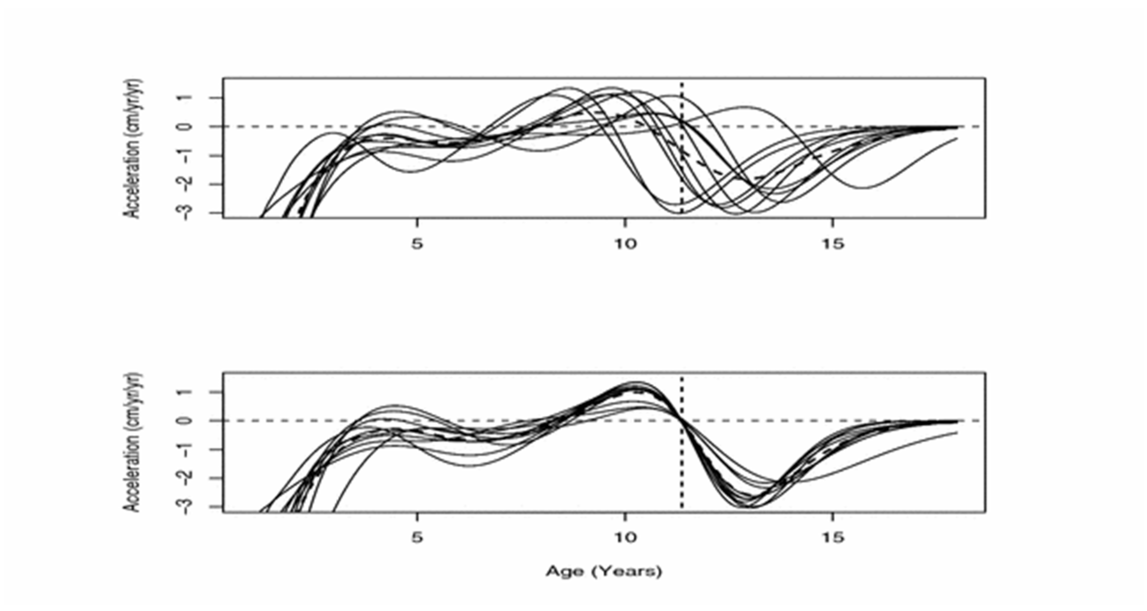

Figure 2.3

Figure 2.3 illustrates the estimated acceleration curves for each girl in the sample along with the mean curve, denoted by the dashed line, obtained by pointwise averaging of the (estimated) sample curves both before (top panel) and after (bottom panel) registration had been applied to the data.

For the unregistered curves, the mean curve does not resemble any of the sample curves. In general, if registration to remove phase variation is not performed the mean curve tends to underestimate the amplitude of both local minima and maxima. After alignment of important features in each of the sample curves, a procedure later referred to as landmark registration, the resulting mean curve is a much more realistic representation of the common structural pattern exhibited by the sample curves.

Consider the N smooth curves $x_1(\ ), x_2(\ ), \ldots, x_N(\ )$. It is assumed that the discrete observations for the i^{th} sample item correspond to a set of evaluation points over the interval $[0,T_i]$, allowing for a possibly different set of points $\boldsymbol{t}_i = \left[t_{i1}, t_{i2}, \ldots, t_{ip_i}\right]^T$ for each item, $i = 1,2,\ldots,N$. Further, let $[0,T]$ denote the common interval over which the N curves are to be aligned, with the requirement that the vector of discrete observations $\boldsymbol{y}_i$ for each sample item corresponds to a set of evaluation points contained within this interval.

Broadly, registration achieves curve alignment by applying a transformation to the continuum t for each sample item. This transformation is specified by a time warping function $h_i(t)$, $i = 1,2,...,N$, with common domain $[0,T]$ and varying range $[0,T_i]$. The individual intervals $[0,T_i]$ are aligned over the common interval $[0,T]$ and, consequently, the warping functions are subject to the boundary conditions $h_i(0) = 0$ and $h_i(T) = T_i$, *i.e.* $h_i : [0,T] \to [0,T_i]$. Analyses of any functional data set are then based on the registered curves $x_i^*(t) = x_i[h_i(t)]$, which exhibit only amplitude variation, rather than simply the smooth curves $x_i(t)$, $i = 1,2,...,N$. The problem of registering the curves of a sample consequently amounts to estimation of the functions $h_1(\ ), h_2(\ ),...,h_N(\ )$, which capture the phase variation in the original, unregistered curves.

We proceed with a discussion of three types of registration, viz. shift registration, landmark registration and continuous registration.

2.7.1 Shift registration

Shift registration, a method proposed by Silverman (1995), aligns the curves over the common interval $[0,T]$ by considering a linear transformation of the continuum t. In particular, the time warping function, $h_i(\ )$ for the i^{th} sample item , applies a time shift transformation, i.e. $h_i(t) = t + \delta_i$, $i = 1,2,...,N$. Following this definition, the registered curves can be obtained according to:

$$x_i^*(t) = x_i[h_i(t)] = x_i(t + \delta_i), i = 1,2,...,N. \tag{2.25}$$

Within this framework, the registration problem of estimating the time warping functions $h_i(\ )$ amounts to estimation of the unknown shift parameters δ_i, $i = 1,2,...,N$, a task achieved by minimisation of a global registration criterion:

$$REGSSE = \sum_{i=1}^{N} \int_{0}^{T} \left[x_i\left(t+\delta_i\right) - \hat{\mu}\left(t\right) \right]^2 dt \ , \tag{2.26}$$

where $\hat{\mu}\left(t\right) = \frac{1}{N}\sum_{i=1}^{N} x_i\left(t\right)$, the estimated mean function of the unregistered curves. This criterion considers the integrated (global) sum of the squared differences between the registered curves $x_i\left(t+\delta_i\right)$ and the estimated mean curve (Ramsay & Silverman, 2005, p.131). However, since the mean function in any FDA should be based on the registered curves, an iterative estimation procedure is considered in which the mean function is updated based on updated estimates of the shift parameters $\delta_1, \delta_2, ..., \delta_N$, at each step of the algorithm.

In particular, a modified Newton-Raphson algorithm for minimising $REGSSE$ is considered (Ramsay & Silverman, 2005, p.143). After initialising the values $\delta_i^{(0)}$, where $\delta_i^{(0)} = 0, \ i = 1, 2, ..., N$, provides a common choice, the estimated mean function of the shifted curves is computed, i.e. $\hat{\mu}^{(0)}\left(t\right) = \frac{1}{N}\sum_{i=1}^{N} x_i\left(t+\delta_i^{(0)}\right)$. The v^{th} step of the algorithm for $v = 1, 2, ..., n_{iter}$, where n_{iter} is the number of iterations until convergence, consists of two parts:

1. Update the shift parameter estimates from the previous iteration $\delta_i^{(v-1)}$ according to:

$$\delta_i^{(v)} = \delta_i^{(v-1)} - \alpha \ \frac{\frac{\partial}{\partial \delta_i} REGSSE}{\frac{\partial^2}{\partial \delta_i^2} REGSSE} \ , \tag{2.27}$$

   where $\alpha > 0$ is a step-size parameter, and often set equal to one. Furthermore, the derivatives in this expression, which follow from Equation (2.26), are explicitly specified by Ramsay & Silverman (2005).

2. Update the estimate of the mean function: $\hat{\mu}^{(v)}\left(t\right) = \frac{1}{N}\sum_{i=1}^{N} x_i\left(t+\delta_i^{(v)}\right)$.

Practical applications have shown that the algorithm usually converges within one or two iterations, but the speed and the reliability of the convergence improves with better initial estimates $\delta_i^{(0)}, i = 1,2,...,N$ (Ramsay & Silverman, 2005, p.143). After completion of this estimation procedure, the registered curves $x_i^*(t) = x_i\left(t + \delta_i^{(n_{iter})}\right), i = 1,2,...,N$, can easily be obtained. In addition, an estimate of the mean function that is adequate for the purposes of a functional data analysis based on the registered curves can be computed, a quantity known as the estimated structural average of $x_1(\ ), x_2(\ ),..., x_N(\ )$, based on $h_i(\ ), i = 1,2,...,N$, at time point t (Kneip & Gasser, 1992, p.1279):

$$\hat{\mu}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i^*(t) = \frac{1}{N}\sum_{i=1}^{N} x_i\left(t + \delta_i^{(n_{iter})}\right) . \tag{2.28}$$

2.7.2 Landmark registration

It is common for each of the smooth, estimated curves $x_i(\ ), i = 1,2,...,N$ of a functional data set to exhibit a shared structural pattern. However, this typical shape often varies in dynamics and intensity between the N sample items (Kneip & Gasser, 1992, p.1266). In particular, specified landmarks may have different locations for each item in the sample. A landmark refers to an important feature or characteristic of a curve that can be associated with a specific argument value (or location) t, of which maxima, minima and crossings of fixed thresholds provide common examples (Ramsay & Silverman, 2005, p.132). Landmarks may be identified at the level of the curves themselves or even at the level of some derivative.

Unless all of the curves in the sample exhibit a systematic pattern of landmarks over time, shift registration, which considers only a simple location shift of the continuum t, will not achieve satisfactory alignment of curve features. A curve alignment method known as landmark registration may be applied as an alternative. It should be noted that the shift registration procedure has the advantage that no specification of landmarks is required.

Landmark registration aligns the curves in a functional data sample by considering a non-linear transformation of t via a time warping function $h_i(t)$, $i = 1,2,...,N$, so that specified landmarks for each sample item occur at the same (transformed) times or locations. The process requires the specification of L landmarks of interest as well as their locations along the horizontal axis, the continuum t, for each curve:

$$\boldsymbol{\pi}_i = \left[\pi_{1i}, \pi_{2i}, \ldots, \pi_{Li}\right]^T, i = 1, 2, \ldots, N \quad , \tag{2.29}$$

where $\pi_{\ell i}$ is the location of the ℓ^{th} landmark feature for the i^{th} sample curve.

The time warping functions transform the location of the ℓ^{th} landmark for each sample item to its average location $\bar{\pi}_\ell = \frac{1}{N}\sum_{i=1}^{N}\pi_{\ell i}$, i.e.

$$h_i\left(\bar{\pi}_\ell\right) = \pi_{\ell i} \text{ , } i = 1,2,\ldots,N \text{ and } \ell = 1,2,\ldots,L \text{ .} \tag{2.30}$$

Here, the time warping functions are constrained to be continuous and, since the locations of the landmark features should occur in the same order in transformed time as in observed time for each sample item, it is also required that $h_i(\ )$ is strictly monotone increasing, i.e. $h_i(t_1) > h_i(t_2)$ for $t_1 > t_2$, $i = 1,2,\ldots,N$.

The computational details of estimation of the warping functions $h_1(\ ), h_2(\ ), \ldots, h_N(\ )$, from which the registered curves can be obtained, are discussed by Kneip and Gasser (1992). However, the authors note that this estimation procedure, subject to satisfying Equation (2.30), only fixes the values of the warping function $h_i(\ )$ at the average landmark locations $\bar{\pi}_1, \bar{\pi}_2, \ldots, \bar{\pi}_L$. In between these points, the estimated function is subject to the requirements of smoothness and monotonicity. Consequently, linear interpolation of the fixed points $h_i(\bar{\pi}_\ell)$, $\ell = 1,2,\ldots,L$,to produce the estimated function $\hat{h}_i(\ )$ will not be sufficient since the condition of smoothness will not be met. As a result, an interpolating procedure that guarantees strict monotonicity and smoothness of the estimated functions $\hat{h}_1(\ ), \hat{h}_2(\ ), \ldots, \hat{h}_N(\ )$ is discussed by these authors.

Following this estimation procedure, the remaining variation in the registered curves $\hat{x}_i^*(\ ) = x_i\left[\hat{h}_i(\ )\right]$, denoted simply by $x_i^*(\ )$, $i = 1,2,...,N$, will be attributable mainly to amplitude, and elimination of phase variation present in the unregistered curves will have been achieved.

A first possible drawback of a landmark registration approach to curve alignment is difficulty in determining the estimated location of each landmark of interest, since this process is based on the estimated curves $x_i(\ ), i = 1,2,...,N$. Secondly, not all of the specified landmark features may be present in each sample curve, resulting in possibly missing feature locations. Furthermore, landmark registration only achieves the alignment of curves according to the identified features and it may be the case that the resulting curves $x^*(\ )$ are still unregistered at other locations of the transformed, common interval $[0,T]$.

As an alternative approach, providing a solution to the last shortcoming of landmark registration, Ramsay and Li (1998) proposed a continuous registration procedure that aims to achieve a more satisfactory alignment of the sample curves in their entirety, rather than just alignment at specified points (landmarks). In addition, continuous registration can be applied in cases where variable shift and scaling factor transformations in the time domain are required to align the sample curves and it is consequently a more broadly applicable method than shift registration, which only achieves a constant shift transformation of t across the whole curve.

2.7.3 Continuous registration

Continuous registration can be viewed as a non-parametric approach to curve alignment. Broadly, the time warping functions, which again consider non-linear transformations of t, are assumed to belong to a family of smooth, monotone functions allowing flexibility of the estimates $\hat{h}_1(\ ), \hat{h}_2(\ ),..., \hat{h}_N(\ )$ while still maintaining computational efficiency.

This curve registration approach requires the specification of a target function $r(\ )$, defined over the interval $[0,T]$, belonging to the same class as the sample functions $x_i(\ )$. The purpose of continuous alignment is to align the features of the registered curves

$x_i^*(\ ) = x_i\left[h_i(\ )\right], i = 1,2,...,N$, with those of the target. Assuming that the target is defined as a continuous function, e.g. $r(t) = \sin\left(\frac{t^2}{\pi}\right)$, the following model is implied:

$$r(t) = x_i\left[h_i(t)\right] + \varepsilon_i\ , i = 1,2,...,N\ , \tag{2.31}$$

where ε_i reflects the deviation of the i^{th} registered curve from the target function, and is assumed to be small relative to x_i and roughly centred around 0. Alternatively, if the target is defined by the discrete values $r_1, r_2, ..., r_J$, the assumed model can be written in the form:

$$r_j = x_i\left[h_i(t_j)\right] + \varepsilon_{ij}\ , i = 1,2,...,N \text{ and } j = 1,2,...,J\ . \tag{2.32}$$

Now, consider the general problem of estimating an unknown function f which is constrained to be strictly monotone increasing and positive. Due to the monotonicity requirement, the first derivative of the function is necessarily strictly positive, i.e. $Df > 0$, implying, amongst many other possibilities, the form

$$Df(t) = e^{W(t)}\ , \tag{2.33}$$

where $W(\ )$ is an unconstrained function. Integrating both sides of Equation (2.33) gives the solution:

$$f(t) = C_0 + \int_0^t exp\left[W(u)\right]du \quad . \tag{2.34}$$

Here, integration up to t following an exponential transformation of $W(\ )$ ensures that the function of interest f is both positive and monotone increasing, as required. Further, C_0 is an unknown constant to be estimated from the data.

Alternatively, based on Equation (2.33), the appropriate linear differential equation for the class of monotone functions can be expressed as

$$D^2 f = wDf \quad , \tag{2.35}$$

where $w = DW$. The solution of this differential equation has the form

$$f(t) = C_0 + C_1 \int_0^t exp\left[\int_0^u w(v)\,dv\right] du \quad , \tag{2.36}$$

where $w = \frac{D^2 f}{Df}$, the relative curvature of f, and C_1 is a positive constant. Defining

$$W(u) = \int_0^u w(v)\,dv + \log C_1 \quad , \tag{2.37}$$

leads to equivalence of Equations (2.34) and (2.36).

Within the continuous registration framework, the time warping functions $h_i(\ )$ for the sample items are assumed to belong to the monotone family defined by Equation (2.34), or equivalently by Equation (2.36). This has the advantage of simplifying the problem of estimating the function $h(\ )$, which is constrained to be monotone, to one of estimating the unconstrained function $w(\ )$.

Consider a single sample item corresponding to the unknown time warping function h. Based on a specified target function $r(\ )$, estimation of $h(\ )$ is achieved by minimising a penalised squared error criterion:

$$F_\lambda(h) = \int_0^T \left\{r(t) - x\left[h(t)\right]\right\}^2 dt + \lambda \int_0^T \left[w(t)\right]^2 dt\,, \tag{2.38}$$

where $h(\ )$ is of the form specified in Equation (2.34). The first term of the fitting criterion provides a measure of the goodness of fit of the registered function $x\left[h(\ )\right]$ to the target function $r(\ )$, and this quantity can be viewed as a measure of the similarity of the shapes of the two functions. In contrast, landmark registration considers the timings of a set of landmarks as a measure of similarity (Ramsay & Silverman, 2002, p.110). In order to constrain the flexibility of the time warping function, a penalty term is added which controls the flexibility of h, through penalisation of its curvature w. If the target is defined discretely, the first integral in Equation (2.34) is replaced by a sum of squared errors.

Substituting the assumed form of the function $h(\ )$, the regularisation criterion $F_\lambda(h)$ becomes

$$F_\lambda(w) = \int_0^T \left\{ r(t) - x\left[C_0 + C_1 \int_0^t exp\left[\int_0^u w(v)\,dv \right] du \right] \right\}^2 dt + \lambda \int_0^T \left[w(t) \right]^2 dt \quad , \tag{2.39}$$

where the function $w(\ )$ and the constants C_0 and C_1 are the only unknown quantities. Consequently, estimation of the time warping function is achieved by estimating its relative curvature. The degree of smoothness imposed on the relative curvature of $h(\ )$ is controlled by the regularisation parameter λ, where larger values shrink the relative curvature towards zero and the warping function towards linearity, i.e. $h(t) = t$. The values $\lambda = 10^{-4}, 10^{-3}$ and 10^{-2} have produced good results over a range of applications (relative curvature is a scale free measure).

The above discussion holds for any individual data case in the functional sample. Consider now the i^{th} case. Then, subject to the boundary conditions $h_i(0) = 0$ and $h_i(T) = T_i$, estimates of the unknown coefficients in the expression of the time warping function, as per Equation (2.36), can be obtained as $C_0 = 0$ and

$$\begin{aligned} h_i(T) &= T_i \\ &= \int_0^T exp\{W(u)\}du \quad \text{as per Equation (2.34) with } C_0 = 0 \qquad (2.40) \\ &= \int_0^T exp\left\{\int_0^u w(v)dv + logC_{1i}\right\}du \quad \text{from Equation (2.37)} \end{aligned}$$

implying the result $C_{1i} = T_i\left[\int_0^T exp\left\{\int_0^u w(v)dv\right\}du\right]^{-1}$.

However, the criterion $F_\lambda(w)$ to be optimised is of infinite dimensionality. By restricting the form of the candidate functions $w(\ )$, a reduction in the dimensionality of the problem is achieved. In particular, a basis function expansion of the curvature function is considered. Following this assumption, minimisation of the criterion can be achieved by numerical integration to produce the estimated function $\hat{w}(\ )$, and consequently an estimate of the time warping function. Alternatively, if an order 1 B-spline basis function system is considered, the criterion $F_\lambda(w)$ can be integrated explicitly and a closed form expression of the estimate $\hat{h}(\ )$ is obtained.

The above estimation procedure is performed separately for each of the items in the functional data sample, producing the estimated transformation functions $\hat{h}_i(\ )$ and consequently the registered curves $x_i^*(\ ) = x_i\left[\hat{h}_i(\ )\right], i = 1,2,...,N$, which will exhibit aligned salient curve features.

Finally, the Procrustes method, a special case of continuous registration, considers the mean function of the unregistered curves, after alignment of salient curve features via shift or landmark registration, as the target $r(\ )$. Once estimates of the time warping functions $h_i(\ ), i = 1,2,...,N$ have been obtained, the estimated mean curve (target) is recalculated and the process continues iteratively. In addition, the fitting criterion $F_\lambda(w)$ can be extended to achieve registration at the level of some derivative $D^m x, m = 1, 2, \ldots$, rather than at the level of the curves themselves. Details on this extension are considered by Ramsay and Li (1998).

Chapter 3

Functional Binary Classification

3.1 Introduction

It may be the case that a supervised learning problem considers a functional input variable together with a qualitative response, a situation referred to as functional classification. Within this framework, each training data item in the sample can be represented as a pair $(\mathbf{y}_i, z_i)$, $i = 1,2,...,N$. Here, $\boldsymbol{y}_i$ is the vector of discrete observations obtained at the set of evaluation points $\boldsymbol{t} = \left[t_1, t_2, ..., t_p\right]^T$. It corresponds to a qualitative response z_i assuming values in a set $\{1,2,\ldots,U\}$ of labels. In this chapter, attention is restricted to the binary case, *i.e.* $U = 2$.

Focus is placed on extensions of traditional classification techniques to consider samples where the data are curves, making them appropriate for application in a functional data analysis (FDA). More specifically, we require a procedure taking the *p*-dimensional vector $\boldsymbol{y}$ as input, and generating an output in the set $\{1,2\}$, by means of a classification rule. It is necessary for an intermediate step of this process to involve smoothing of the discrete input vector in order to produce an estimate of the underlying continuous function $x(\ )$.

3.2 Linear discriminant analysis for functional data

Linear discriminant analysis (LDA) was proposed by Fisher (1936) for implementation on finite-dimensional data sets. However, any FDA considers data sets of infinite dimensionality since the curves $x(\ )$ can be sampled at an infinite number of evaluation points over the continuum t . Consequently, LDA cannot be applied directly in the functional case, and some adaptations of the classification procedure are first required. James and Hastie (2001) consider two such adapted approaches, based on regularisation and filtering methods.

3.2.1 The general model

Let $x(\ )$ represent the underlying function of a sample item in the functional data set. It is assumed that $x(\ )$ is distributed as a Gaussian process with

$$\begin{aligned} E\left[x(t)\right] &= \mu_u(t) \\ Cov\left[x(t),x(t')\right] &= \omega(t,t') \end{aligned} \quad , \tag{3.1}$$

if $x(\ )$ belongs to class u, $u = 1,2$. This distributional assumption is in line with the general formulation of the usual LDA approach, in particular a class specific mean vector and common covariance matrix, both of which have functional equivalents when the data are curves.

However, the underlying function $x(\ )$ is unknown and comes in the form of a noise contaminated vector of discrete observations $\boldsymbol{y}$, satisfying the relationship $y = x(t) + \varepsilon$. It is assumed that the measurement errors have zero expectation and constant variance, *i.e.* $Var(\varepsilon) = \sigma^2$. For a set of p evaluation points $\boldsymbol{t} = \left[t_1, t_2, \ldots, t_p\right]^T$, the distributional assumptions imply the result

$$\boldsymbol{y} \sim N\left(\mu_u(\boldsymbol{t}), \Sigma(\boldsymbol{t})\right) \quad , \tag{3.2}$$

where the vector $\boldsymbol{y} = \left[y_1, y_2, \ldots, y_p\right]^T$ is observed according to the relationship $y_j = x(t_j) + \varepsilon_j$, $j = 1,2,\ldots,p$, and

$$\mu_u(\boldsymbol{t}) = \left[\mu_u(t_1), \mu_u(t_2), \ldots, \mu_u(t_p)\right]^T, \quad u = 1,2 \ . \tag{3.3}$$

Furthermore, the p x p matrix $\Sigma(\boldsymbol{t})$ is modelled as

$$\Sigma(\boldsymbol{t}) = \Omega(\boldsymbol{t}) + \sigma^2 I_p \ , \tag{3.4}$$

and consequently the ij^{th} element of this matrix is given by $\sigma(t_i, t_j) = \omega(t_i, t_j) + \sigma^2 \delta_{ij}$, for $i, j = 1,2,...,p$ and $\delta_{ij} = \begin{cases} 0 \ \text{if} \ i \neq j \\ 1 \ \text{if} \ i = j \end{cases}$.

Following these definitions and assumptions, a sample curve $x(\ )$ is classified to the class

$$\underset{u}{\arg\max}\{L_u\} \quad \text{for } u = 1,2 , \tag{3.5}$$

known as the Bayes classification rule. The function to be maximized is referred to as the (linear) discriminant function for class u, given by

$$L_u(\boldsymbol{t}) = x(\boldsymbol{t})^T \Sigma(\boldsymbol{t})^{-1} \mu_u(\boldsymbol{t}) - \frac{1}{2}\mu_u(\boldsymbol{t})^T \Sigma(\boldsymbol{t})^{-1} \mu_u(\boldsymbol{t}) + \log(\pi_u) \quad , \tag{3.6}$$

where π_u is the prior probability of a sample item belonging to the u^{th} class, $u = 1,2$.

However, the discriminant function involves the functions $\mu_u(\ )$ and $\Sigma(\ )$, determined by $\Omega(\ )$ and the constant variance σ^2, all of which are unknown, as well as the unknown prior probability π_u. Consequently, before the class membership of any (test or training) sample item can be determined using the Bayes classification rule, estimation of these unknown quantities is required.

3.2.2 Estimation using the filtering method

Both the filtering and regularisation approaches can be viewed as methods of fitting the described general functional model. We restrict attention to the filtering approach. Broadly, this method entails the use of basis functions to obtain estimates of the unknown quantities $\mu_u(t)$, as well as σ^2 and $\omega(t,t')$, implying estimates of $\mu_u(\boldsymbol{t})$ and $\Sigma(\boldsymbol{t})$.

As before, smoothing of the discrete observations $\boldsymbol{y}$ is achieved by considering a basis function expansion of the underlying function, *i.e.* $x(t) = \sum_{m=1}^{M} \xi_m \varphi_m(t)$. After estimation of the basis coefficients, the computational details of which are discussed in Sections 2.3 and 2.4, the estimated smooth function $\hat{x}(t) = \sum_{m=1}^{M} \hat{\xi}_m \varphi_m(t)$, denoted simply by $x(t)$, can be obtained.

As in the usual linear discriminant approach to classification, the prior probability π_u can be estimated by the proportion of sample items falling into the class, *i.e.*

$$\hat{\pi}_u = \frac{N_u}{N} \text{ for } u = 1,2 \ , \qquad (3.7)$$

where N_u is the number of sample items in class u and N denotes the total number of items in the sample, with $N = N_1 + N_2$.

Now, according to the filtering approach, the sample mean function of the curves in the u^{th} class, $\bar{x}_u(t)$, provides a reasonable estimate of the unknown, class specific function $\mu_u(t)$. Based on the basis function expansion of $x(t)$ this quantity may be expressed mathematically as:

$$\begin{aligned}\mu_u(t) = \bar{x}_u(t) &= \frac{1}{N_u}\sum_{i=1}^{N_u} x_i(t) \\ &= \frac{1}{N_u}\sum_{i=1}^{N_u}\left\{\sum_{m=1}^{M}\xi_{im}\varphi_m(t)\right\} \\ &= \sum_{m=1}^{M}\left\{\frac{1}{N_u}\sum_{i=1}^{N_u}\xi_{im}\right\}\varphi_m(t) \\ &= \sum_{m=1}^{M}\bar{\xi}_m^u\varphi_m(t)\end{aligned} \quad , \tag{3.8}$$

where $\bar{\xi}_m^u$ represents the average of the m^{th} basis coefficients over the N_u sample items in class u. Replacing the basis coefficients by their (least squares or regularised) estimates $\hat{\xi}_i = \left[\hat{\xi}_{i1},\hat{\xi}_{i2},...,\hat{\xi}_{iM}\right]^T$ for $i = 1,2,...,N_u$ in Equation (3.8) yields the estimated function:

$$\hat{\mu}_u(t) = \sum_{m=1}^{M}\bar{\hat{\xi}}_m^u\varphi_m(t) \quad . \tag{3.9}$$

By the assumption in Equation (3.4), estimation of the unknown quantity $\Sigma(\boldsymbol{t})$ reduces to the problem of estimating each of its components, i.e. the function $\omega(t,t')$ and the constant σ^2. An estimate of the measurement error variance can be obtained from averaging the estimated deviations $\varepsilon_{ij} = y_{ij} - x_i(t_j)$ over the p evaluation points $t_1,t_2,...,t_p$ and all N items (curves) in the sample:

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{1}{p}\sum_{j=1}^{p}\left[y_{ij} - x_i(t_j)\right]^2\right\} \quad . \tag{3.10}$$

Furthermore, the function $\omega(t,t')$ can be estimated by the sample covariance function $cov_x(t,t')$, as defined by Ramsay and Silverman (2005), which is common to both classes in the binary classification problem. Mathematically, this can be expressed as:

$$\begin{aligned}\omega(t,t') &= cov_x(t,t') \\ &= \frac{1}{N-1}\sum_{i=1}^{N}\{x_i(t)-\overline{x}(t)\}\{x_i(t')-\overline{x}(t')\} \\ &= \frac{1}{N-1}\sum_{i=1}^{N}\left\{\sum_{m=1}^{M}\xi_{im}\varphi_m(t)-\overline{x}(t)\right\}\left\{\sum_{m=1}^{M}\xi_{im}\varphi_m(t')-\overline{x}(t')\right\}\end{aligned} \quad , \qquad (3.11)$$

where $\overline{x}(t)=\frac{1}{N}\sum_{i=1}^{N}x_i(t)=\sum_{m=1}^{M}\overline{\xi}_m\varphi_m(t)$, the overall sample mean function. After replacing the basis coefficients by their estimated values, the estimated function $\hat{\omega}(t,t')$ is obtained, implying the estimated p x p matrix $\hat{\Omega}(\boldsymbol{t})$. Finally, the estimated quantities imply the result:

$$\hat{\Sigma}(\boldsymbol{t})=\hat{\Omega}(\boldsymbol{t})+\hat{\sigma}^2 I_p \quad . \qquad (3.12)$$

Consequently, the use of basis functions in the estimation of $\mu_u(t)$ and $\omega(t,t')$, according to the filtering method, is achieved implicitly through a basis function expansion of the underlying function $x(\;)$.

3.2.3 Classification using the estimated discriminant function

Following the estimation procedures described in Section 3.2.2, the estimated discriminant function can be obtained by replacing the unknown quantities in L_u in Equation (3.6) by their respective estimated values:

$$\hat{L}_u = x(\boldsymbol{t})^T\,\hat{\Sigma}(\boldsymbol{t})^{-1}\,\hat{\mu}_u(\boldsymbol{t})-\frac{1}{2}\hat{\mu}_u(\boldsymbol{t})^T\,\hat{\Sigma}(\boldsymbol{t})^{-1}\,\hat{\mu}_u(\boldsymbol{t})+log(\hat{\pi}_u) \; . \qquad (3.13)$$

Here $x(\boldsymbol{t})=\left[x(t_1),x(t_2),...,x(t_p)\right]^T$ denotes the p-dimensional vector obtained by evaluating the estimated function $x(t)=\sum_{m=1}^{M}\hat{\xi}_m\varphi_m(t)$ at the set of points $\boldsymbol{t}=\left[t_1,t_2,...,t_p\right]^T$. These values are used instead of the discrete observations contained in the vector $\boldsymbol{y}$ since the latter effectively represent noise contaminated versions of the values in $x(\boldsymbol{t})$.

Classification of a new sample item $x(\ )$ can then be achieved by substituting L_u in the classification rule by the estimated discriminant function, i.e. classify to

$$\underset{u}{\arg\max}\left\{\hat{L}_u\right\} \quad \text{for } u = 1,2 \quad . \tag{3.14}$$

3.2.4 Further remarks

In the case of sparse functional data sets, where only a portion of each sample curve has been observed, the regularisation and filtering approaches may result in a poor fit of the general functional model. As an alternative, James and Hastie (2001) propose a method known as functional linear discriminant analysis (FLDA) which can be applied more successfully in the case of sparsity. Briefly, this classification approach models the basis coefficients ξ_m , $m = 1,2,...,M$, in the expansion of $x(\ )$ using a Gaussian distribution. Further modelling and computational details are discussed by these authors.

3.3 Support vector machines for functional data

3.3.1 Introduction

Classification using support vector machines, a procedure designed for finite-dimensional inputs, can be cast into a more general framework allowing extension of its use to functional (binary) classification problems.

According to the methodology presented by Rossi and Villa (2005), we consider a training set of N i.i.d. realisations of the pair (X,Z) of random variables, *i.e.* $(x_1,z_1),(x_2,z_2),...,(x_N,z_N)$. Here, the variable X is assumed to take values in a Hilbert space χ and the variable Z denotes the corresponding class membership of the input, where $Z \in \{-1,1\}$. For a problem where, instead, the response Z^* assumes values in the set $\{0,1\}$, a simple recoding via the transformation $Z = 2Z^* - 1$ can be applied.

For functional data sets, the input X is a function-valued random variable and, consequently, the Hilbert space χ is of infinite dimension. In the context of functional classification, specifically the application of support vector machines to functional data, the authors consider two approaches to address this problem of infinite dimensionality of the input. Firstly, regularity constraints can be imposed on the functions in the relevant Hilbert space χ. Alternatively, the same functions can instead be projected onto finite-dimensional functional spaces. These approaches lead to the application of linear and functional kernels, respectively, within the usual SVM procedure.

3.3.2 Functional support vector machines (FSVM)

Relating the functional case to the general SVM framework, the unknown, underlying function $x(\ )$, or simply x, for each sample item in the sample provides the N realisations of the function valued variable X assuming values in the Hilbert space χ.

While it is rarely the case for multivariate data, functional data are often linearly separable due to the high dimensionality of the space within which they are defined. Consequently, a linear decision boundary separating the two classes, given by the hyperplane

$$F(x) = \langle w,x \rangle_{\chi} + b \ , \tag{3.15}$$

is sought. Here, $w \in \chi$ is a coefficient function, $b \in \Re$ and $\langle .,. \rangle_{\chi}$ denotes the inner product in the Hilbert space χ. Furthermore, $F(\ )$ is a functional taking functions defined in χ as arguments. A curve x can then be classified according to the rule

$$sign\{F(x)\} \ . \tag{3.16}$$

For linearly separable data, a function w and constant b exist such that $z_i F(x_i) > 0 \ \forall i$. The fore-mentioned condition implies that the separating hyperplane results in correct classifications, and hence a positive margin $z_i F(x_i)$, for each of the N sample items. However, such a function is not uniquely defined and consequently the hyperplane that maximizes the margin between the training points for the two classes is considered optimal. This approach leads to the optimization problem

$$\begin{gathered}\underset{b,w}{minimize}\left\{\langle w,w\rangle_{\chi}\right\} \\ \text{subject to } z_i\left(\langle w,x_i\rangle_{\chi}+b\right)\geq 1 \ for\ i=1,2,...,N\end{gathered} \quad . \tag{3.17}$$

To allow some misclassifications (on the training set), and consequently extend support vector machines to the case of linearly non-separable data, slack variables, denoted by $\zeta_i, i=1,2,...,N$, can be incorporated into the optimization criterion, *viz.*

$$\begin{gathered}\underset{b,w}{minimize}\left\{\langle w,w\rangle_{\chi}+C\sum_{i=1}^{N}\zeta_i\right\} \\ \text{subject to } z_i\left(\langle w,x_i\rangle_{\chi}+b\right)\geq 1-\zeta_i \quad , \\ \zeta_i\geq 0\ , for\ i=1,2,...,N\end{gathered} \tag{3.18}$$

where $C \geq 0$ is a penalisation parameter placing an upper bound on the permitted number of misclassifications. As previously noted, functional data are often linearly separable. Consequently, the introduction of slack variables into the optimization criterion in Equation (3.17) is not theoretically required since a linear decision boundary exists that gives zero misclassification error on the training data. However, we consider the regularised criterion in Equation (3.18) with the purpose of preventing the FSVM procedure from overfitting the functional data.

The criterion to be optimised, as specified in Equation (3.18), has an equivalent dual form that is obtained through the use of Lagrange multipliers, *viz.*

$$\begin{gathered}\underset{\boldsymbol{\alpha}}{minimize}\left\{\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j z_i z_j\langle x_i,x_j\rangle_{\chi}\right\} \\ \text{subject to} \sum_{i=1}^{N}\alpha_i z_i=0 \text{ and } 0\leq\alpha_i\leq C, for\ i=1,2,...,N\end{gathered} \quad . \tag{3.19}$$

This finite, N-dimensional form of the optimization criterion offers an advantage over the expression in Equation (3.18), which requires an optimization procedure in the infinite dimensional space χ.

Let $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_N]^T$ denote the solution to the dual form of the optimization criterion (Equation (3.19)). The resulting linear decision boundary, or separating hyperplane, is given by

$$\hat{F}(x) = \sum_{i=1}^{N} \hat{\alpha}_i z_i \langle x_i, x \rangle_{\chi} + \hat{b} \quad , \tag{3.20}$$

and a functional input x can be classified according to

$$sign\left\{\hat{F}(x)\right\} \ . \tag{3.21}$$

The estimate of the constant b is obtained from the Karush-Kuhn-Tucker boundary conditions. More specifically, the equation

$$\hat{\alpha}_i \left[z_i \left(\langle x_i, \hat{w} \rangle_{\chi} + b \right) - 1 \right] = 0 \ , \tag{3.22}$$

is solved for b for each of the support points, *i.e.* the training inputs x_i for which $\hat{\alpha}_i > 0$. Here, an estimate of the coefficient function is obtained from the solution vector $\hat{\boldsymbol{\alpha}}$ as $\hat{w} = \sum_{i=1}^{N} \hat{\alpha}_i z_i x_i$ (Hastie *et al.*, 2011). Typically, an average is taken over the solutions corresponding to each of the support points to produce the final estimate $\hat{b}$.

By examination of the classification rule in Equation (3.21), it is evident that all calculations required to classify an input function x are done through the inner product of curves in the Hilbert space χ, rather than basing calculations on the curves themselves. The key to performing these computations in the infinite-dimensional space χ, specifically optimizing the criterion in Equation (3.19), is provided by replacing the inner product in this space by a symmetric and positive definite kernel function $K(.,.)$ taking functions in χ as arguments. We consider two options that are particularly useful for application in a functional data analysis, namely linear and functional kernels.

3.3.3 Linear kernels

The use of linear kernels within the FSVM procedure arises from imposing regularity constraints on the functions $x_i \in \chi, i = 1,2,...,N$, in order to address the infinite dimensionality of the Hilbert space χ , as discussed by Park et al. (2008).

Let $\boldsymbol{y}_i = \left[y_{i1}, y_{i2}, ..., y_{ip} \right]^T$ denote the p-dimensional vector of discrete observations corresponding to the set of evaluation points $\boldsymbol{t} = \left[t_1, t_2, ..., t_p \right]^T$ for the i^{th} functional sample item x_i. The training set can then be expressed as $\left\{ \left(\boldsymbol{y}_i, z_i \right) \right\}_{i=1}^{N}$. As an initial step, a basis function expansion of the functional inputs x_i is considered, achieving regularisation and, consequently, a reduction in dimensionality by considering only a finite set of M basis functions in the expansion, *i.e.*

$$x_i = \sum_{m=1}^{M} \xi_{im} \varphi_m = \boldsymbol{\xi}_i^T \boldsymbol{\varphi} \ . \tag{3.23}$$

After estimation of the basis coefficients $\xi_{i1}, \xi_{i2}, ..., \xi_{iM}$ for a given basis system, of which the B-spline and Fourier systems provide common choices for non-periodic and periodic data respectively, estimates of the N curves in the sample can be obtained from $x_i = \sum_{m=1}^{M} \hat{\xi}_{im} \varphi_m$. This process may be viewed as a pre-smoothing of the discrete data $\boldsymbol{y}_i$, $i = 1,2,...,N$.

Classification is subsequently performed by applying the functional SVM procedure to the smoothed data, i.e. the estimated functions $x_1, x_2, ..., x_N$. A linear kernel function taking the estimated functions as arguments is defined by

$$\begin{aligned} K_{lin}\left(x_i,x_j\right) &= \left\langle x_i,x_j\right\rangle_{\chi} \\ &= \int x_i\left(t\right)x_j\left(t\right)dt \\ &= \int \hat{\xi}_i^T\varphi\left(t\right)\hat{\xi}_j^T\varphi\left(t\right)\ dt \ \text{ from Equation (3.23) }, \\ &=\hat{\xi}_i^T\int\varphi\left(t\right)\varphi^T\left(t\right)dt\,\hat{\xi}_j \\ &= \hat{\xi}_i^T\Phi\hat{\xi}_j \end{aligned} \tag{3.24}$$

where Φ is an MxM basis matrix with ij^{th} element given by $\Phi_{ij}=\int\varphi_i\left(t\right)\varphi_j\left(t\right)dt$, $i,j=1,2,...,M$. Such an inner product in the functional space χ can be computed using classical quadrature schemes, to perform numerical integration, or approximated through application of Monte Carlo methods (Rossi and Villa, 2005, p.637).

Classifications are made by applying the standard FSVM procedure with all inner products in χ replaced by the linear kernel defined in Equation (3.24). This gives rise to the optimization criterion

$$\begin{gathered} \underset{\boldsymbol{\alpha}}{minimize}\left\{\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j z_i z_j K_{lin}\left(x_i,x_j\right)\right\} \\ \text{subject to}\sum_{i=1}^{N}\alpha_i z_i=0 \ \text{ and } 0\le\alpha_i\le C,\ for\ i=1,2,...,N \end{gathered} \quad , \tag{3.25}$$

and the decision boundary

$$\hat{F}\left(x\right)=\sum_{i=1}^{N}\hat{\alpha}_i z_i K_{lin}\left(x_i,x\right)+\hat{b}\,, \tag{3.26}$$

based on the solution vector $\hat{\boldsymbol{\alpha}}$ and the estimate $\hat{b}$.

The functional SVM procedure using the linear kernel function $K_{lin}\left(.,.\right)$ is expected to perform well on data sets with fewer evaluation points p than sample items N . In the reverse scenario, which is more common in the functional context, the use of non-linear kernels such as the radial basis function kernel may result in better performance (in terms of classification errors on a test set). Application of such a non-linear kernel in cases where $p < N$ may result in over-smoothing (Park et al., 2008). As an alternative to non-linear kernels, we may consider the application of functional kernels within the FSVM procedure for functional data sets with $p > N$.

3.3.4 Functional kernels

As an alternative approach to reducing the dimensionality of the classification problem in the functional context, the functions x in the space χ can be projected onto a finite-dimensional functional space.

More specifically, let P_M denote the projection of the original data, defined on χ, onto a finite, M-dimensional space H spanned by an orthonormal basis $\{\psi_m\}_{m=1}^{M}$, achieving a reduction in the dimensionality of the problem. A kernel function in the space χ can then be defined in terms of a kernel function defined on the transformed, finite-dimensional inputs $P_M(x)$, *i.e.*

$$K_M(x_i, x_j) = K\left(P_M(x_i), P_M(x_j)\right) , \tag{3.27}$$

where $K(.,.)$ is any standard SVM kernel defined on $\Re^M$, e.g. the Gaussian kernel.

Following this definition, classifications can be obtained by replacing all inner products in χ appearing in the FSVM procedure by the kernel function $K_M(.,.)$. The resulting criterion to be optimised then only requires computations in the finite-dimensional space H . This approach is analogous to that of functional support vector machines using linear kernels, with the functional kernel replacing $K_{lin}(.,.)$.

As an alternative to working with functional kernels in functional support vector machines, Biau *et al.* (2005) established an equivalent procedure:

1. Express each function x_i in the sample as a series expansion of a complete orthonormal basis $\{\psi_m\}_{m=1}^{\infty}$ of χ :

$$x_i = \sum_{m=1}^{\infty} \xi_{im}\psi_m, \quad i = 1,2,...,N \,. \tag{3.28}$$

2. Approximate each function by the finite sum

$$x_i \approx \sum_{m=1}^{M} \xi_{im}\psi_m = \boldsymbol{\xi}_i^T \boldsymbol{\psi} \,, \quad i = 1,2,...,N \;. \tag{3.29}$$

3. Apply an ordinary SVM in $\Re^M$ using the basis function coefficients as inputs, implying the training set $\{(\xi_i, z_i)\}_{i=1}^{N}$. In practice, this process will be based on the estimated coefficients $\hat{\xi}_i$, $i = 1,2,...,N$.

Furthermore, if linear kernels are applied within the FSVM approach to binary classification using an orthonormal basis $\{\psi_m\}_{m=1}^{\infty}$ to achieve pre-smoothing of the data, as in Equation (3.28), the basis matrix Φ reduces to the identity matrix, *i.e.*

$$\Phi_{i,j} = \int \varphi_i(t)\varphi_j^T(t)\,dt = \delta_{ij} \;. \tag{3.30}$$

This result implies that the procedure is equivalent to an ordinary SVM applied to the basis coefficients and is consequently equivalent to functional support vector machines using functional kernels (Park et al., 2008, p.2580).

While the B-spline basis does not formally meet the requirements of the orthonormal basis required in the projection of the data for use in functional kernels, this system often gives good results in practice and can consequently be applied in FSVM (Rossi and Villa, 2005, p.638). Often, derivatives of functions in the data set may provide more insight into the process generating the data than would be made available by an analysis only at the level of the functions themselves. As a further motivation for this choice of orthonormal basis, the B-spline system allows easy derivative estimation of the input functions in the space χ, allowing kernels to be constructed based on the derivatives $D^q x$, $q = 1,2,\ldots$. As an example, in the field of spectroscopy the curvature of the functions, quantified by their second derivatives, is frequently more useful for class prediction than the curves themselves.

3.3.5 Implementation

The implementation of functional support vector machines in practice, using both linear and functional kernels, requires specification of the following quantities:

1. The dimension M of the projection P_M when using functional kernels, or the number of basis functions in pre-smoothing the data to apply linear kernels.

2. The penalisation parameter $C \geq 0$.

3. The parameters of the chosen kernel function $K(.,.)$ defined for the finite-dimensional, transformed inputs $P_M(x)$ or, equivalently, for the basis coefficients ξ, when using functional kernels.

A procedure known as a data splitting device can be used for these purposes, the computational details of which are discussed by Biau *et al.* (2005).

3.4 Further remarks

While the focus of this chapter is restricted to functional extensions of linear discriminant analysis and support vector machines, work on extending many traditional classification techniques to consider samples where the data are curves has been completed. Functional neural networks, proposed by Conan-Guez and Rossi (2004), as well as *K*-nearest neighbour classification for functional data (Biau *et al.*, 2005) provide examples of alternative procedures that can be considered in a (binary) functional classification problem.

Chapter 4

Variable Selection

4.1 Introduction

Consider a data set of size N within a supervised learning framework, where each item in the sample can be represented as a pair $\{(\boldsymbol{y}_i, z_i)\}_{i=1}^{N}$. The response variable, denoted by Z, can assume either numerical values or can be equal to 0 or 1, giving rise to regression and binary classification problems, respectively. In the functional case, the input vector $\boldsymbol{y}_i = \left[y_{i1}, y_{i2}, ..., y_{ip} \right]^T$ represents discrete, noise-contaminated observations of an underlying function $x_i(\ )$ sampled at the set of evaluation points $\boldsymbol{t} = \left[t_1, t_2, ..., t_p \right]^T$. In contrast, for non-functional data, a multivariate analysis of the discrete observations $\boldsymbol{y}_i$, $i = 1, 2, ..., N$, viewed as N realisations of the input variables $Y_1, Y_2, ..., Y_p$, is considered.

It is known that data sets with $p > N$ typically exhibit a high degree of multicollinearity. This property especially manifests in the functional case due to the natural ordering of the features $Y_1, Y_2, ..., Y_p$ induced by the evaluation points $t_1 < t_2 < ... < t_p$. Consequently, an analysis based on only a subset of the p discrete observations (inputs) for each sample item may be of interest and often prove to be beneficial. In the case of a functional data analysis, this process amounts to eliminating some of the evaluation points in the set $\boldsymbol{t}$ at which the underlying function $x(\ )$ is sampled.

Applying a variable selection method to obtain this restricted set of the p inputs or, equivalently, the p evaluation points in a functional context, may achieve improved prediction accuracy. Furthermore, a reduction in the cost of data collection and reduced computation time provide two additional advantages.

We proceed with a discussion of two techniques for variable selection and dimension reduction that are considered particularly useful in situations where the data are curves, namely the fused lasso and sparse partial least squares (SPLS).

4.2 Variable selection using the fused lasso

4.2.1 The standard lasso

Consider the standard linear model

$$z_i = \sum_{j=1}^{p} \beta_j y_{ij} + \varepsilon_i \ , \ i = 1, 2, ..., N \ , \tag{4.1}$$

where the errors ε_i have zero mean and constant variance σ^2. The inputs are assumed to have been suitably standardised leading to omission of an intercept term from the model. The task of model fitting, effectively obtaining estimates of the regression coefficients $\beta_1, \beta_2, ..., \beta_p$, can be performed using a least squares approach which entails minimization of the well-known sum of squares criterion.

Any statistical analysis of a wide data set, *i.e.* $p > N$, should incorporate some method of regularisation in order to counteract the curse of dimensionality and prevent overfitting. In the case of fitting the above linear model to possibly functional data, regularisation can be achieved by penalising the size of the coefficient estimates, $\hat{\boldsymbol{\beta}}^{LS}$, resulting from the least squares estimation procedure.

In particular, these restricted coefficient estimates can be found by minimising a penalised sum of squares

$$\sum_{i=1}^{N} \left[z_i - \sum_{j=1}^{p} \beta_j y_{ij} \right]^2 + \lambda J(\boldsymbol{\beta}) \ , \tag{4.2}$$

with respect to the vector $\boldsymbol{\beta}$. Here, $J(\ )$ denotes a penalty function applied to the coefficients to constrain their size. The tuning parameter λ, the value of which is generally determined by cross-validation, controls the degree of penalisation. If $\lambda = 0$, no regularisation is performed and the usual least squares coefficient estimates are obtained.

The special case of the lasso arises from the choice of the L_1 penalty within the above-described regularisation framework. This specific penalty function has the form

$$J(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left|\beta_j\right| , \tag{4.3}$$

with the effect that, for a sufficiently large value of the tuning parameter λ, some of the parameter estimates $\hat{\beta}_j^{lasso}$, $j = 1,2,...,p$, are constrained to be exactly zero. In such a way, variable selection is achieved by identifying a reduced set of the p inputs $Y_1,Y_2,...,Y_p$ corresponding to the non-zero coefficient estimates. In the functional case, the subset of evaluation points is obtained as the points along the continuum t at which this reduced set of inputs are measured.

However, the lasso does not lead to a closed form expression for the coefficient solution vector $\hat{\boldsymbol{\beta}}^{lasso}$. This disadvantage can be overcome by consideration of the least angle regression (LAR) algorithm, proposed by Efron *et al.* (2004), for fitting linear regression models to high-dimensional data. The LAR algorithm can be applied with a simple modification in order to obtain the entire lasso path corresponding to all possible values of λ (Hastie *et al*., 2011, p.76).

4.2.2 Further penalty functions

Another common choice of the penalty term in supervised learning problems is the L_2 penalty, *viz.*

$$J(\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j^2 , \tag{4.4}$$

giving rise to the well-known shrinkage method of ridge regression, which results in a closed form expression for the estimates $\hat{\boldsymbol{\beta}}^{ridge}$. As the value of the tuning parameter increases, the coefficient estimates become more constrained towards zero and move further away from their least squares values. However, the penalty term does not constrain the estimates to be exactly zero and, consequently, no variable selection is achieved.

A penalty functional known as the elastic net, a linear combination of the L_1 and L_2 penalties, can also be considered within the penalised sum of squares criterion, *i.e.*

$$J(\boldsymbol{\beta}) = \sum_{j=1}^{p}\left[\alpha\left|\beta_j\right| + (1-\alpha)\beta_j^2\right] \ . \tag{4.5}$$

Here, the parameter α determines the mix of the penalties and is commonly chosen by cross-validation (Hastie *et al*., 2011, p.663).

4.2.3 The fused lasso

Choice of the L_1 penalty to regularise the sum of squares criterion has several limitations. For high-dimensional data sets with $p > N$, the lasso produces a solution with at most N non-zero coefficients (Tibshirani *et al*., 2005, p.93). This property may be viewed as a limiting feature for a variable selection method, especially in the functional context where often $p >> N$. Furthermore, functional data sets are known to exhibit a high degree of multicollinearity. The lasso tends to select only a single variable from a group of highly correlated variables and discard the rest (Zou and Hastie, 2005, p.303). Due to the ordering of the features, it seems desirable that intervals of contiguous evaluation points, and consequently inputs, should be selected rather than relatively isolated points spread out over the full interval of the continuum of interest.

Potentially the most significant drawback of the lasso when applied in a functional data analysis is that it does not account for the natural ordering in the observed values. As a solution to this disadvantage, the fused lasso, a method proposed by Tibshirani *et al*. (2005), can be considered as an alternative variable selection approach.

Addition of the penalty function, as defined in Equation (4.3), to the sum of squares criterion has the consequence of producing a sparse solution, *i.e.* some of the components of the vector of estimated lasso coefficients $\hat{\boldsymbol{\beta}}^{lasso}$ are zero. The fused lasso coefficient estimates are obtained by minimisation of the penalised criterion considered by the lasso with the addition of a further penalty term, *i.e.*

$$\hat{\boldsymbol{\beta}}^{fused\ lasso} = \underset{\boldsymbol{\beta}}{argmin}\left\{\sum_{i=1}^{N}\left[z_i - \sum_{j=1}^{p}\beta_j y_{ij}\right]^2 + \lambda_1\sum_{j=1}^{p}\left|\beta_j\right| + \lambda_2\sum_{j=2}^{p}\left|\beta_j - \beta_{j-1}\right|\right\} . \qquad (4.6)$$

Consequently, the fused lasso requires the specification of the two regularisation parameters λ_1 and λ_2. If $\lambda_2 = 0$, the standard lasso variable selection procedure is obtained. Conversely, for $\lambda_1 = 0$, a method known as variable fusion, developed by Land and Friedman (1996), is performed.

As in the case of the lasso, the first penalty term encourages sparsity in the coefficient estimates. The second penalty term encourages sparsity in their differences, resulting in the property that the selected variables occur in contiguous groups. In addition, for high-dimensional data with $p > N$, the fused lasso potentially selects a greater number of the p evaluation points (or inputs) than variable selection using the standard lasso. This result may be attributed to the fact that variable selection using the fused lasso produces a piecewise constant solution with at most N groups of identical non-zero coefficients (Tibshirani *et al.*, 2005, p.101).

4.2.4 Implementation

Implementation of the fused lasso in practice requires specification of the tuning parameters λ_1 and λ_2, which determine the achieved degree of penalisation. As described by Tibshirani *et al.* (2005), cross-validation over a grid of values of the two parameters can be utilised for this purpose. However, the process may become computationally quite expensive.

For selected values of the penalty parameters, the fused lasso criterion in Equation (4.6) is a quadratic programming problem. We consider the *genlasso* package in *R* (Arnold and Tibshirani, 2015) in order to perform the required computations. The standard *genlasso* function is applied with response vector $\boldsymbol{z} \in \Re^N$ and N x p input matrix of observed values $Y = \left[y_{ij} \right]$. A re-parameterization of the penalty parameters, via $\lambda = \lambda_1$ and $\gamma = \frac{\lambda_2}{\lambda_1}$, is considered. A further modification in terms of the penalty matrix D is required, *viz.*

$$\tilde{D} = \begin{bmatrix} D : (p-1) \text{ x } p \\ \gamma I : p \text{ x } p \end{bmatrix}, \tag{4.7}$$

where D is the matrix of first differences.

4.2.5 Functional classification

Once variable selection has been performed, classification of the curves in a functional data set can be achieved based on the identified reduced set of evaluation points to potentially improve prediction accuracy compared to using the full set $\boldsymbol{t}$. The fused lasso can be treated as a filtering approach to pre-process the data, through the process of variable selection, before the application of functional (binary) classification procedures.

In particular, the functional extension of linear discriminant analysis (LDA) will be based on the estimated discriminant function

$$\hat{L}_u = x(\boldsymbol{t})^T \hat{\Sigma}(\boldsymbol{t})^{-1} \hat{\mu}_u(\boldsymbol{t}) - \frac{1}{2} \hat{\mu}_u(\boldsymbol{t})^T \hat{\Sigma}(\boldsymbol{t})^{-1} \hat{\mu}_u(\boldsymbol{t}) + log(\hat{\pi}_u) \ , \tag{4.8}$$

as described in Section 3.2.3. However, the described estimation and classification procedures will only consider the selected subset of evaluation points identified by the fused lasso rather than the set of all p evaluation points.

Feature selection can also be considered as an initial step before the application of support vector machines in the classification of a functional sample. Obtaining the basis expansion coefficient estimates $\hat{\xi}_1, \hat{\xi}_2, ..., \hat{\xi}_M$, on which functional SVMs using both linear and functional kernels are based, using only the reduced set of evaluation points identified by the fused lasso, provides one example of possible implementations of this approach. In effect, the continuous function estimates $x_i(\ )$ are then obtained based on smoothing only a subset of the p, originally observed values $y_{i1}, y_{i2}, ..., y_{ip}$ for each sample item, $i = 1,2,...,N$.

4.3 Simultaneous variable selection and dimension reduction using sparse partial least squares (SPLS)

4.3.1 Standard partial least squares (PLS)

Partial least squares is a well-known dimension reduction technique, developed by Wold (1966), for application in high-dimensional regression problems that exhibit the added complication of multicollinearity among the predictors $Y_1, Y_2, ..., Y_p$. As a result, this technique naturally lends itself to an analysis of functional data.

Although originally proposed for dimension reduction in regression problems, PLS can be extended for use in functional classification, as is our focus, by treating the (binary) response Z in a supervised context as a continuous variable, using a $\{0,1\}$ dummy coding. Mathematical justification of this choice is provided by Chung and Keleş (2010).

As a first step in the application of PLS to a possibly functional data set, standardisation of the observed predictor values contained in the $N \times p$ data matrix Y is required. As before, in the functional case, the rows of this matrix correspond to the discrete observations $\boldsymbol{y}_i$ obtained at the evaluation points $t_1, t_2, ..., t_p$ for each of the N sample items. Partial least squares then constructs K linear combinations of the original predictors $\boldsymbol{Y} = \left[Y_1, Y_2, ..., Y_p\right]^T$, *viz.*

$$T_k = \boldsymbol{w}_k^T \boldsymbol{Y} \quad \text{for}\ k = 1, 2, ..., K, \tag{4.9}$$

in a supervised manner. The direction vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_K$ are found to capture variance in the input space, while simultaneously maximising correlation with the response Z. The tuning parameter of the process, K, which denotes the number of linear combinations or latent variables constructed, can be selected using cross-validation. It should be noted that these latent components cannot be directly observed. Fewer linear combinations than predictors are considered, *i.e.* $K < p$, in order to achieve a reduction in dimensionality.

As discussed by Chung and Keleş (2010), an estimate of the k^{th} p-dimensional direction vector $\mathbf{w}_k$ can be obtained as the solution of the optimization problem

$$\max_{\boldsymbol{w}} \left\{ \boldsymbol{w}^T M \boldsymbol{w} \right\} \text{ subject to } \boldsymbol{w}^T \boldsymbol{w} = 1 \text{ and } \boldsymbol{w}^T S \hat{\boldsymbol{w}}_l = 0, \quad l = 1, ..., k-1 \, , \tag{4.10}$$

where $M = Y^T \boldsymbol{z}\boldsymbol{z}^T Y$, with $\boldsymbol{z}$ denoting the observed, binary response vector. Here, S represents the sample covariance matrix of the observed predictors.

Optimisation of the PLS criterion, as given in Equation (4.10), to obtain estimates of the direction vectors $\hat{\boldsymbol{w}}_k$, $k = 1, 2, ..., K$, can be achieved through application of a non-linear iterative least squares (NIPALS) algorithm, as discussed by Wold (1975).

After estimation of the direction vectors, classification of sample items in a functional data set can subsequently be performed using the resulting latent variables, $\hat{T}_k = \hat{\boldsymbol{w}}_k^T \boldsymbol{Y}$, $k = 1, 2, ..., K$, as inputs into an off-the-shelf classifier, with logistic regression and linear discriminant analysis providing common examples, since the dimensionality of the data has been reduced from p to K and, consequently, the problem of high dimensionality has been addressed.

4.3.2 Sparse partial least squares (SPLS)

While partial least squares achieves an effective reduction in the dimensionality of the data, no variable selection is performed since each of the latent components is a linear combination of the full set of p inputs $Y_1, Y_2, ..., Y_p$. Furthermore, while this method is suited to the high-dimensional case, i.e. $p > N$, Chun and Keleş (2010) illustrate that a large number of irrelevant variables in the input set $\boldsymbol{Y} = \left[Y_1, Y_2, ..., Y_p\right]^T$, in terms of their effect on the response Z, will ultimately deteriorate the performance of the partial least squares procedure.

To address this drawback, variable filtering approaches are often utilised in order to pre-process the data, effectively reducing noise, before partial least squares is applied. Two-sample *t*-tests provide a common example of such a procedure in the binary classification setting. However, the choice of filtering method is arbitrary, and tuning may become computationally involved. In addition, commonly used variable filtering approaches often do not account for the correlations that exist among variables (Chung and Keleş, 2010, p.3).

As a solution, Chun and Keleş (2010) propose a sparse version of partial least squares regression (SPLS) which incorporates feature selection into the dimension reduction procedure. This provides the opportunity of improving interpretability, induced by sparsity, through the use of a computationally efficient procedure, while still addressing the correlations amongst the predictors using the PLS framework.

Sparse PLS constructs latent components $T_1, T_2, \ldots, T_K$ that are linear combinations of only a subset of the original p input variables $Y_1, Y_2, \ldots, Y_p$. This variable selection property is achieved, while simultaneously performing dimension reduction, by imposing sparsity on the PLS direction vectors $\mathbf{w}_k$, $k = 1, 2, ..., K$.

In particular, as discussed by these authors, sparsity of the direction vector $\boldsymbol{w} = \left[w_1, w_2, ..., w_p\right]^T$ is achieved by including an L_1 penalty, imposed on a surrogate of the direction vector $\boldsymbol{c} = \left[c_1, c_2, ..., c_p\right]^T$, in the formulation of the optimization criterion, while keeping the vectors $\mathbf{w}$ and $\mathbf{c}$ close to each other. In addition, an L_2 penalty is imposed on the surrogate vector in order to address the potential singularity of the matrix M. A more detailed discussion of previous work motivating the inclusion of these specific penalty terms in the sparse formulation of partial least squares is provided by Chun and Keleş (2010).

An estimate of the k^{th} SPLS direction vector can be obtained by solving the optimization problem,

$$\begin{aligned} &\underset{\mathbf{w},\mathbf{c}}{minimize}\left\{-\kappa\mathbf{w}^T M\mathbf{w}+(1-\kappa)(\mathbf{c}-\mathbf{w})^T M(\mathbf{c}-\mathbf{w})+\lambda_1\sum_{j=1}^{p}|c_j|+\lambda_2\sum_{j=1}^{p}c_j^2\right\} \\ &\qquad\qquad subject\ to\ \mathbf{w}^T M\mathbf{w}=1 \qquad\qquad , \\ &\qquad\qquad and\ \mathbf{w}^T S\hat{\mathbf{w}}_l=0, l=1,\ldots,k-1 \end{aligned} \tag{4.11}$$

where $\kappa < 1$.

4.3.3 Implementation

Implementation of the sparse PLS procedure in practice requires specification of the set of tuning parameters κ, λ_1, λ_2 and K. Chun and Keleş (2010) regard λ_1 and the number of latent components K to be the two key tuning parameters of the process.

The solution of the SPLS optimization criterion for a univariate response Z, i.e. the binary classification setting, is not influenced by the value of κ. As for standard partial least squares, the number of linear combinations considered can be selected using the method of cross-validation. Furthermore, the authors discuss both soft and hard thresholding approaches for the selection of the tuning parameter λ_1. Of the two approaches, implementation of hard thresholding is regarded as computationally more efficient.

Optimization of the SPLS criterion given in Equation (4.11), as described by these authors, entails alternatively iterating between solving for $\boldsymbol{w}$ for a fixed value of the surrogate direction vector $\mathbf{c}$, and solving for $\mathbf{c}$ after fixing $\boldsymbol{w}$, which often requires a large value of the parameter λ_2. As a result, SPLS is typically fit with the choice $\lambda_2 = \infty$. The final estimated direction vector, as used in the construction of the sparse PLS latent components, is given by the estimate $\hat{c}$. Typically, this vector is rescaled to have norm 1.

The necessary selection and optimization procedures required in the application of sparse partial least squares to a functional data set can be performed using the *spls* package available in *R*. The optimisation criterion (Equation 4.11) can be represented equivalently in terms of a parameter η which controls the thresholding, as does λ_1. Reformulation in this manner presents interpretive advantages, since η only assumes values in the set $\{0,1\}$ (Chung and Keleş, 2010, p.6). Furthermore, when $\eta = 0$, SPLS reduces to the standard partial least squares procedure. For the N x p input matrix of observed values $Y = \left[y_{ij} \right]$, the response vector $\boldsymbol{z} \in \Re^N$ and a selected value of the parameter κ , the function *cv.spls()* can be used to perform cross-validation to select optimal values of the parameters K and η. The *spls* package implements soft thresholding for selection of the tuning parameter λ_1. Further fitting, for values of the tuning parameters selected by cross-validation, can be executed using the *spls()* function.

4.3.4 Functional classification

As in the case of ordinary partial least squares, classification can be performed after construction of the latent components $\hat{T}_1, \hat{T}_2, \ldots, \hat{T}_K$. In particular, functional classification can be achieved by regarding these linear combinations as inputs in a classification method suitable for application in a low-dimensional setting, since it will typically be the case that $K < N$. Variable selection is embedded in this procedure and can be contrasted with variable selection as a filtering approach, with the fused lasso providing an example, before the application of functional classification procedures.

Linear classifiers are often utilised due to their simple interpretation. For example, if the latent components are considered as inputs in the discriminant analysis classification procedure, a method termed SPLS discriminant analysis (SPLSDA), developed by Chung and Keleş (2010), is obtained. Logistic regression provides another example of a linear classifier that is commonly used in this context.

4.3.5 Further remarks

The simultaneous variable selection and dimension reduction property of SPLS offers the possibility of improved classification accuracy and interpretability when compared to the standard PLS procedure, especially in the presence of a large number of irrelevant variables. In addition, it allows a low-dimensional representation of potentially infinite-dimensional data through plots of the (typically first few) direction vectors.

However, variable selection using the method of sparse partial least squares does not explicitly account for the natural ordering of the variables $Y_1, Y_2, ..., Y_p$, induced by the evaluation points $t_1, t_2, ..., t_p$, that exists in a functional context. Consequently, further research is required to address this drawback. A possible approach may involve the incorporation of an additional penalty term in the SPLS optimization criterion (Equation (4.11)) so that selected variables occur in contiguous groups, making the procedure more suitable for application in a functional data analysis.

Chapter 5

Empirical Study

5.1 Data and problem description

Table grapes (*Vitis vinifera L.)* are commonly affected by brown discolouration of a significant proportion of individual grape berries, with the consequence of reducing the appeal of the product. Several different types of berry browning can occur, including that of internal flesh and/or external skin of the affected berries. This discolouration can be evident at harvest, develop during cold storage and, in some instances, only become apparent during the shelf life of the fruit (Crisosto *et al*., 1994; Fourie, 2009). With products destined for long-distance markets, it would be beneficial to the grape export industry to be able to predict, before they are exported, which grapes are likely to discolour. This would, for example, make it possible to channel grapes deemed fairly likely to discolour to a local market.

Infrared spectroscopy data are available on table grapes intended for export. A statistical analysis of these data is considered in an attempt to address the problem of predicting discolouration. Specifically, near-infrared (NIR) spectroscopy measurements in the 800-2900 nm range were made of small surface areas on unaffected individual grape berries shortly after the grapes were harvested. This resulted in measurement of the wavelength-dependent absorption, reflection and scattering of the NIR radiation, which in turn relate to the chemical composition and internal micro-structure of the fruit (Nicolaï *et al*., 2007). These grapes were kept in cold storage for 6 weeks, mimicking the exportation process, leading to discolouration of some of the berries. NIR measurements were then repeated on the same berries subjected to measurement before cold storage.

Within a statistical learning framework, these data represent a binary classification problem. The response variable of interest, denoted by Z, assumes values in the set $\{0,1\}$, with $Z=0$ indicative of the absence of discolouration after cold storage. The feature vectors considered are observations of spectral variables $Y_1, Y_2, ..., Y_p$, representing infra-red absorption values at wavelengths $\omega_1 < \omega_2 < ... < \omega_p$. In the supervised problem under consideration, measurements were made at $p = 2203$ wavelengths on $N = 407$ individual grape berries. In $N_0 = 111$ cases, no discolouration of the grape berry was observed, i.e. $Z = 0$. Consequently, the outcome of the experiment saw a total of $N_1 = 296$ discoloured grape berries ($Z = 1$) after cold storage, with $N = N_0 + N_1$.

Two objectives are especially significant to exporters in the analysis of this data set. Firstly, determining whether it is possible to successfully predict, soon after they are harvested, which of the grape berries will discolour. In addition, it is of interest to conclude whether accurate prediction can be achieved if measurements are restricted to a subset of the $p = 2203$ wavelengths in the sample, i.e. the task of variable or feature selection. The two fore-mentioned objectives are intertwined: prediction accuracy can frequently be improved by considering only a subset of the available features. Reducing the number of wavelengths at which observations have to be made for the purpose of accurate classification has the added advantage of reducing the cost of data collection, which is substantial in this application.

Due to the natural ordering of the features, induced by the wavelengths $\omega_1 < \omega_2 < ... < \omega_p$, the studied classification problem falls within the area of functional data analysis. The wavelengths at which the infra-red absorption values are observed, represent the continuum t of the functional data set. Furthermore, it is assumed that the observed spectral variables $\boldsymbol{y}_i = \left[y_{i1}, y_{i2}, ..., y_{ip} \right]^T$ represent discretely sampled, noise-contaminated observations of an unknown, underlying function $x_i(\ )$ at the evaluation points (or wavelengths) $t_1, t_2, ..., t_p$, for each of the sample items in the data set, $i = 1, 2, ..., N$.

The following points have to be borne in mind in carrying out this functional data analysis. The features $Y_1, Y_2, ..., Y_p$ are highly correlated, since absorption variables are measured at wavelengths which are very close together. The resulting high degree of autocorrelation, characteristic of functional data, implies a need for regularisation to be incorporated into the analysis. In addition, observing more features than sample items puts the data in a high-dimensional setting, i.e. $p > N$. Furthermore, due to the natural ordering of the wavelengths, it is desirable that feature, or equivalently wavelength, selection should result in selected intervals of contiguous wavelengths rather than relatively isolated features spread out over the full interval.

5.2 Exploratory data analysis

We proceed with a preliminary analysis of the data. As an initial step, in line with analyses of other similar data in the field of chemometrics, the observed values of the p spectral variables $Y_1, Y_2, ..., Y_p$ for each of the N sample items are log-transformed before any further computations are carried out.

For each of the items in the data set, the discrete observations $\boldsymbol{y}_i$ are smoothed by making use of a basis function expansion to produce the continuous function estimate $x_i(\ ), i = 1,2,...,N$. Due to the non-periodic nature of the data, a B-spline basis system is utilised for this purpose. In particular, $M = 12$ basis functions $\varphi_1, \varphi_2, ..., \varphi_M$ which are comprised of cubic piecewise polynomials are considered, *i.e.* order four splines $(q = 4)$. Furthermore, the resulting eight interior knots, obtained according to the relationship

$$\textit{Number of interior knots} = M - q\ , \tag{5.1}$$

a property of the B-spline basis system, are spaced equally over the 800-2900 nm wavelength range.

Estimates of the basis coefficients $\xi_1, \xi_2, ..., \xi_M$ for each of the N items in the data set are obtained using a least squares approach. Since this procedure produces a satisfactory degree of smoothness of the resulting curve estimates $x_1(\ ), x_2(\ ), ..., x_N(\ )$, it is deemed unnecessary to incorporate regularisation, as described in Section 2.4, into the estimation procedure.

Rather than specifying the number of basis functions subjectively, the value of M used in the functional data analysis can be selected data-dependently to maximise the criterion

$$\frac{1}{M}\sum_{m=1}^{M}\left(\bar{\hat{\xi}}_{m}^{0}-\bar{\hat{\xi}}_{m}^{1}\right)^{2}, \tag{5.2}$$

which may be regarded as a simple measure of separation between the groups $Z = 0$ and $Z = 1$. Binary classification, at the most basic level, involves maximisation of some measure of separation between the two response groups of interest, providing motivation for following this methodology. Here, $\bar{\hat{\xi}}_{m}^{u} = \frac{1}{N_u}\sum_{i=1}^{N_u}\hat{\xi}_{im}$ represents the average of the m^{th} basis coefficient estimate over the N_u sample items in the u^{th} class, for $u = 0,1$ and $m = 1,2,\ldots,M$.

Values of the parameter M in the set $[4,50]$ were considered. By the relationship in Equation (5.1), the number of basis functions M has a minimum value of $q = 4$. A value of M that is substantially smaller than $p = 2203$ is considered as the upper limit of the parameter search so that data smoothing, achieved by the use of a basis function expansion, achieves substantial dimension reduction in terms of data representation, *i.e.* storing the N x M matrix Ξ rather than the N x p matrix Y, as well as a significant reduction in the degree of noise present in the data.

The criterion in Equation (5.2) attained a global maximum at the lower endpoint of the search interval, *i.e.* $M = 4$. Furthermore, a second (local) maximum was observed for $M = 5$. Consequently, to achieve a compromise between maximisation of Equation (5.2) and avoiding undersmoothing that would result from using too few basis functions in the smoothing of the data, the third maximum ($M = 12$) was selected.

The discrete observations for each of the $N = 407$ sample items were measured at the same set of $p = 2203$ evaluation points (wavelengths) over the 800-2900 nm range. In addition, the curve estimates exhibit a shared structural pattern as well as an alignment of salient curve features, specifically minima and maxima for this functional sample, evident from Figure 5.1.

Consequently, no curve registration is required as a preliminary, or pre-processing, step in the analysis of this data set.

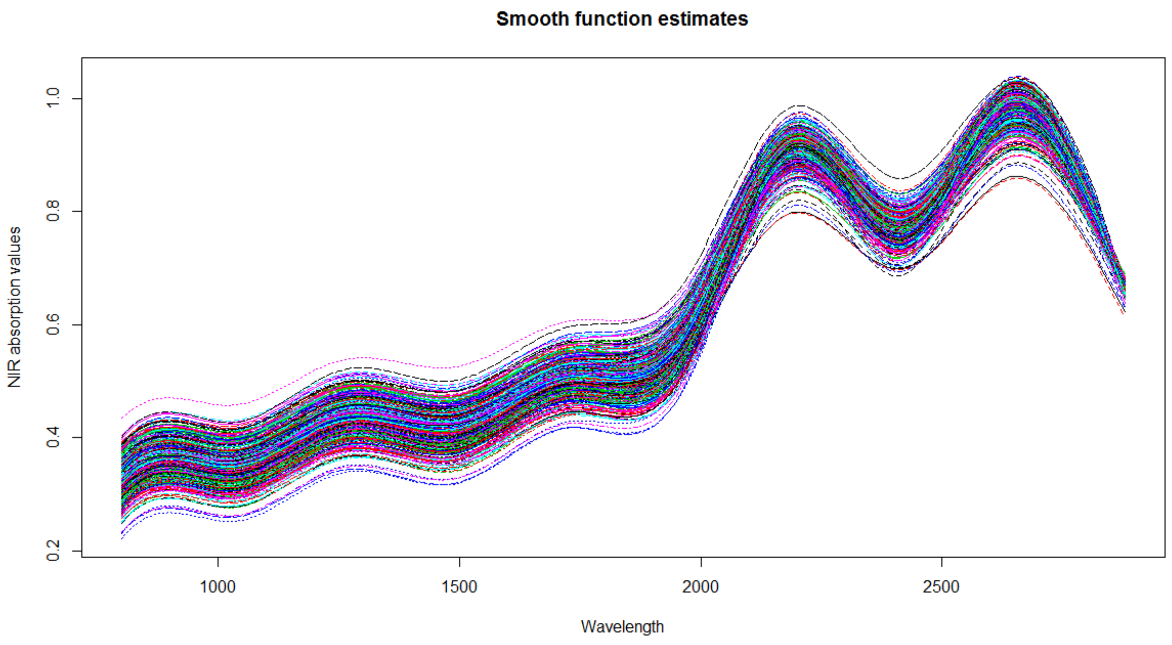

Figure 5.1

The estimated sample mean curve, obtained by pointwise averaging of the curve estimates shown in Figure 5.1 at the set of p wavelengths, is displayed in Figure 5.2 separately for each of the two response groups, $Z = 0$ and $Z = 1$, *i.e.*

$$\bar{x}_u(t) = \frac{1}{N_u}\sum_{i=1}^{N_u} x_i(t) \ , \tag{5.3}$$

for $t = t_1, t_2, \ldots, t_p$ and $u = 0,1$.

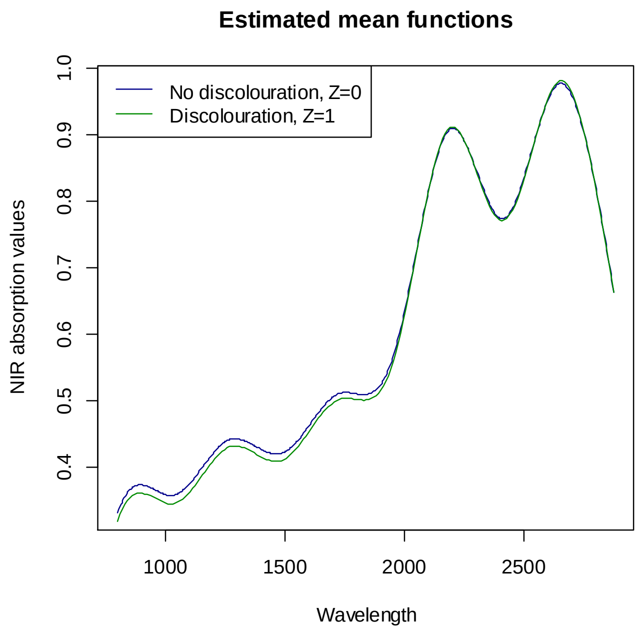

Figure 5.2

A slight difference between the mean curves is observed at low wavelength values, suggesting that it might be possible to use the information provided by these spectral variables to distinguish between the two response groups. Consequently, it is expected, at an initial level, that low wavelength variables will be more important in the classification process than those observed at high wavelength values.

The estimated sample correlation function,

$$corr\left[x(t),x(t')\right]=\frac{cov\left[x(t),x(t')\right]}{\sqrt{var\left[x(t)\right]var\left[x(t')\right]}}\quad , \tag{5.4}$$

evaluated at the same set of p wavelengths, is shown in Figure 5.3. Here, $cov(t,t')$ and $\mathrm{var}(t)$ respectively denote the estimated covariance and variance functions.

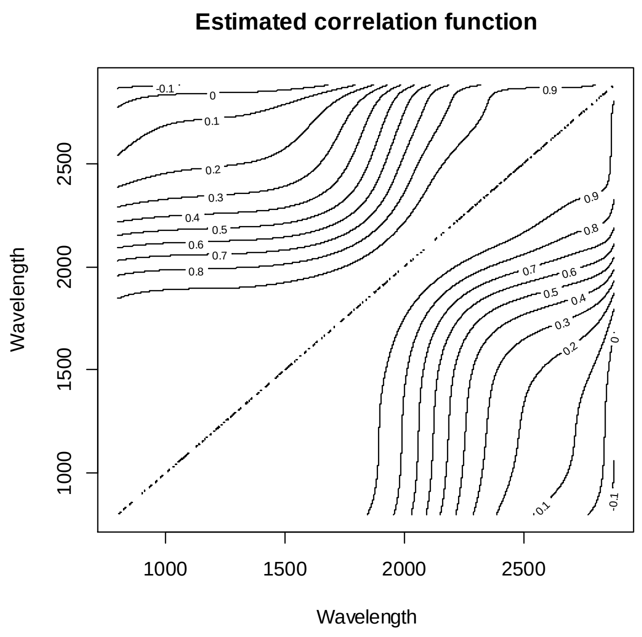

Figure 5.3

In Figure 5.3, the diagonal running from the lower left to the upper right corners of the plot contains unit values, i.e. the correlations between identical wavelength values. The shape of the contour plot is in line with expectations, showing high correlations between evaluation points that are closely situated along the wavelength continuum. Consequently, the sample correlation function provides a graphical representation of the high degree of autocorrelation present in the functional data set. This property is especially observed at short wavelengths. The close proximity of the contours at higher wavelengths indicates a steep gradient, and consequently a more rapid decline in correlation than that observed at low wavelength values.

5.3 Experimental design

The observed data ($N = 407$ sample items) were repeatedly, randomly split into 70% training and 30% test cases. Due to the unbalanced nature of the data, with approximately 73% of the cases showing evidence of discolouration (Z=1), proportional splits were undertaken in order to preserve this property in both the test and training subsets. A total of 30 splits were carried out, and results are reported as averages over these repetitions.

An empirical study was conducted to compare linear discriminant analysis (LDA) for functional data and functional support vector machines (FSVM) as binary classification methods. In addition, the performance of the latter procedure based on only a subset of the wavelengths, identified by feature selection using the fused lasso, was investigated. Furthermore, sparse partial least squares discriminant analysis (SPLSDA) was applied to the data set in order to investigate the classification accuracy achieved by incorporating simultaneous variable selection and dimension reduction into the standard LDA procedure in a functional setting.

The procedures are evaluated in terms of the following criteria. The overall test error, averaged over the 30 repetitions, is reported for each of the above-mentioned methods. However, due to the unbalanced nature of the data, it is insufficient to consider only the overall error rate of a procedure as an indication of model performance, since an overall test error of approximately 27% can be achieved by simply classifying all the input cases into the majority group, *i.e.* $Z = 1$. As a result, the average misclassification error rates on the test cases per group, $Z = 0$ and $Z = 1$, for each of the procedures are also reported.

As mentioned in Section 5.2, the observed values of the p spectral variables $Y_1, Y_2, ..., Y_p$ for each of the N sample items were log-transformed as a data pre-processing step. Furthermore, before any feature selection or classification procedures were applied to the data, the inputs were standardised. This process is performed at each repetition, *i.e.* random split of the full data set into training and test sets, by computing the means and standard deviations of the p discretely observed feature variables $Y_1, Y_2, ..., Y_p$ in the training set, and using these values in the standardisation of each training and test case.

In the following sections details are provided for the different techniques that were applied to the data set, with the corresponding *R* code given in the Appendix.

5.3.1 Sparse partial least squares (SPLS)

Sparse partial least squares, as described in Section 4.3, was applied to the data set in order to simultaneously achieve dimension reduction and variable selection. A subjective choice of the parameter $0 \le \eta \le 1$ was made, *viz.* $\eta = 0.8$ (equivalent to specification of the parameter λ_1), while the corresponding number of latent variables K for each repetition was selected by applying 10-fold cross-validation, using the function *cv.spls()*, to the set of training data resulting from each specific split. Consequently, different splits of the data into training and test subsets resulted in the selection of a different number of linear combinations. Since the SPLS solution in a binary classification setting is not influenced by the value of the parameter κ, the default value of 0.5 was used. In addition, as discussed in Section 4.3.3, fitting was carried out with the parameter choice $\lambda_2 = \infty$.

The SPLS latent components, $T_1, T_2, ..., T_K$, for each sample item in the training set could then be obtained in *R* using the function *splsda()*, with the above-described parameter specifications.

5.3.2 Sparse partial least squares discriminant analysis (SPLSDA)

Following construction of the SPLS latent components $T_1, T_2, ..., T_K$, representing K linear combinations of a subset of the features $Y_1, Y_2, ..., Y_p$, potentially different for each repetition, these components were considered as inputs in the standard linear discriminant analysis classification procedure. Consequently, the SPLSDA classification procedure of Chung and Keleş (2010) was applied to the functional data set in order to obtain class predictions for the observations in the test set.

However, the unbalanced nature of the data makes it important to adjust the SPLSDA procedure in an attempt to balance the classification error rates for the two groups. LDA uses a threshold value of 0.5 in the classification of a sample item to class $Z = 0$, i.e. a sample item with input (vector of latent components) $\boldsymbol{T} = [T_1, T_2, ..., T_K]^T$ will be classified to group $Z = 0$ for a posterior probability $P\left(Z = 0 \middle| \boldsymbol{T}\right) > 0.5$ and to group $Z = 1$ otherwise. Hastie *et al.* (2011) suggest choosing this threshold, or cut-point, to empirically minimize the training error for the specific data set under consideration. Application of this recommendation to balance the group error rates on the training data yielded an adjusted threshold value of $P\left(Z = 0 \middle| \boldsymbol{T}\right) > 0.2709$ for a class $Z = 0$ prediction. This value was selected to (approximately) balance the two group error rates on the basis of several training data splits.

The adjusted posterior probability threshold value corresponds to an updated LDA classification rule, whereby cases are classified to the response group $Z = 0$ if

$$\hat{L}_o > \hat{L}_1 - 0.99 \quad . \tag{5.5}$$

Here, $\hat{L}_u$ denotes the estimated discriminant function for class $u = 0,1$.

The adjusted threshold value of 0.2709 was obtained by expanding the discriminant functions in the above classification rule as per their definitions in Equation (3.6). The resulting equation could then be expressed in terms of the posterior probability of a group $Z = 0$ classification, *i.e.* $P(Z = 0 \mid \boldsymbol{T}) = \pi_0 f_0(\boldsymbol{T})$, with $f_0(\ )$ denoting the density function of the Gaussian process defined in Equation (3.2) for $u = 0$.

5.3.3 Functional linear discriminant analysis

Linear discriminant analysis as extended for use in functional data applications was considered as a second classification procedure in the empirical study. As per the discussion in Section 3.2, estimates of unknown quantities in the linear discriminant functions L_u were obtained using the filtering approach. Specifically, this estimation procedure is based on continuous (underlying) function estimates constructed by smoothing the discrete observations for each sample item using a basis function expansion comprised of $M = 12$ B-spline basis functions, as selected in Section 5.2, over the full wavelength range, *i.e.* 800-2900 nm.

As in the case of SPLSDA it is necessary to adjust LDA for functional data to account for the unbalanced nature of the data. This was achieved, as before, by adjusting the threshold value for the posterior probability of a $Z = 0$ class prediction to balance the resulting group error rates on the training data. This process yielded the classification rule

$$\hat{L}_o > \hat{L}_1 - 1.0505 , \tag{5.6}$$

implying the adjusted posterior probability threshold value $P\left(Z=0\middle|\hat{x}(\boldsymbol{t})\right)>0.2591$, for a sample item to be classified to the response group $Z=0$. This value was obtained from Equation (5.6) by following the same procedure outlined for SPLSDA.

5.3.4 Functional support vector machines (FSVM)

As per the discussion in Section 3.3.4, the application of functional support vector machines using both functional and linear kernels is equivalent to implementation of the standard SVM procedure in $\mathbb{R}^M$, with the M basis function coefficient estimates for each sample item as inputs, implying the training set, $\left\{\left(\hat{\xi}_i, z_i\right)\right\}_{i=1}^{N}$. The performance of functional support vector machines as a binary functional classification method, in terms of accuracy in the classification of discoloured grape berries, was investigated as part of the empirical study.

As an initial modelling step, in line with the theory of support vector machines, the response variable $Z\in\{0,1\}$ was recoded as $Z^* = 2Z-1\in\{-1,1\}$. Further implementation, as discussed in Section 3.3.5, required specification of the following unknown quantities:

- The number of basis functions: selected as $M=12$, as per the discussion in Section 5.2.

- The kernel to be applied to the basis function coefficient estimates: the radial kernel function was selected for this purpose, i.e.

$$K\left(\xi_i,\xi_j\right)=exp\left\{-\frac{1}{\gamma}\left\|\xi_i-\xi_j\right\|^2\right\}. \tag{5.7}$$

Application of this kernel function required specification of the unknown parameter γ. In addition, a suitable value of the penalisation parameter $C\geq 0$ in the SVM procedure needed to be specified. Estimation of both these unknown quantities can be achieved simultaneously by making use of the function *train()*, available in the *R* package *caret*, with the specification *method="svmRadial"* or, equivalently, *method="svmRadialSigma"*.

The fore-mentioned function sets up a grid of values of the tuning parameters of interest (C and γ in the case of classification using support vector machines) and evaluates, using resampling, their effect on model performance. The optimal values of the tuning parameters, out of those specified in the grid, are reported as output. In the application of the function *train()* to the grapes data set, a sequence of 10 equally spaced values of the cost parameter between $C = 0.0001$ and $C = 1$, as well as 50 values, also equally spaced, of the kernel function parameter γ between a minimum and maximum value obtained using the function *sigest(),* were considered for tuning. Function *sigest()* estimates the range of values of the radial kernel function parameter that would return good results when SVM fitting is to be carried out using the function *ksvm*(). Both of the fore-mentioned functions are available in the package *kernlab.* The 10% and 90% quantiles of the hyper-parameter, returned as output by the function *sigest(),* were considered as suitable values for the range of γ in the parameter search.

The above procedure for obtaining suitable estimates of γ and C was applied to the vectors of $M = 12$ basis function coefficient estimates for all $N = 407$ sample items in the full set as input data. Consequently, the same estimates, *viz.* $\hat{C} = 0.0001$ and $\hat{\gamma} = 0.3039255$, were used in the SVM fitting procedure for each of the 30 splits into training and test sets.

After specification of the above unknown quantities, fitting the FSVM model was done using the *ksvm*() function. However, as before, it was necessary to adjust the classification procedure to account for the unbalanced nature of the data. In the case of obtaining class predictions using (functional) support vector machines, this problem can be addressed through an adjustment of the intercept $\hat{b}$ in the classification rule,

$$sign\left\{\hat{F}\left(\xi\right)\right\} = sign\left\{\sum_{i=1}^{N}\hat{\alpha}_i z_i K\left\langle \xi_i, \xi \right\rangle + \hat{b}\right\}, \tag{5.8}$$

for a sample item with input vector ξ.

An adjusted intercept, denoted by $\hat{b}^{adjust}$, was determined for each repetition, *i.e.* a different value was obtained for each split of the full data set into test and training subsets. This is a consequence of the variation in the value of $\hat{b}^{adjust}$ required to balance the group errors across different splits, making it unsuitable to apply a single adjusted value over all 30 repetitions. More specifically, the following steps were performed for each split:

1. Obtain the estimated values $\hat{F}_j = \hat{F}(\xi_j) = \sum_{i=1}^{N} \hat{\alpha}_i z_i K\left\langle \xi_i, \xi_j \right\rangle + \hat{b}$ for each sample item in the training set, $j = 1, 2, ..., N_{train}$, based on the fitted SVM model for the specific repetition.

2. Obtain the corresponding vector of order statistics, with elements $\hat{F}_{(1)} < \hat{F}_{(2)} < ... < \hat{F}_{(N_{train})}$.

3. Compute the vector of split points $\boldsymbol{s}$, with i^{th} element

$$s_i = \frac{\hat{F}_{(i)} + \hat{F}_{(i+1)}}{2}, \text{ for } i = 1, 2, ..., (N_{train} - 1) . \tag{5.9}$$

4. For each split point s_i, compute the two class-specific group errors on the training data with classifications made according to the rule

$$\hat{F}(\xi) < s_i \Rightarrow \hat{z} = 0 \left(\hat{z}^* = -1 \right), \tag{5.10}$$

for $i = 1, 2, ..., N_{train}$.

Select the optimal value s^* as the split point that results in the minimum absolute difference in the group $Z = 0$ and $Z = 1$ training misclassification error rates, out of all considered values $s_1, s_2, ..., s_{N_{train}}$.

5. Determine the adjusted threshold for the specific repetition, given by

$$\hat{b}^{adjust} = \hat{b} - s^* . \tag{5.11}$$

Class predictions for sample items in the corresponding test data set, *i.e.* the same split for which $\hat{b}^{adjust}$ was determined on the training data, can subsequently be obtained according to the adjusted SVM rule

$$sign\left\{\hat{F}^{adjust}(\xi)\right\} = sign\left\{\sum_{i=1}^{N}\hat{\alpha}_i z_i K\langle\xi_i,\xi\rangle + \hat{b}^{adjust}\right\}. \qquad (5.12)$$

5.3.5 Variable selection using the fused lasso

It is of interest to compare the performance of binary functional classification using functional support vector machines evaluated on a subset of the $p = 2203$ wavelengths (evaluation points) to the classification accuracy achieved on the full set of wavelengths. The fused lasso was implemented for this purpose due to the property of selected variables occurring in contiguous intervals, taking account of the natural ordering of features characteristic to functional data. Consequently, a reduction in computation time as well as a potential improvement in classification accuracy compared to FSVM applied to the full set of wavelengths was expected.

Details of implementation of the fused lasso in *R* using the function *genlasso()* are discussed in Section 4.2.4. Full specification of the penalty matrix $\tilde{D}$ requires an estimated value of the unknown parameter γ. The subjective choice $\gamma = 1$ was made, implying the relationship $\lambda_1 = \lambda_2$ for the two regularisation parameters in the fused lasso optimisation problem (Equation 4.6).

Following specification of the parameter γ, the *genlasso()* function subsequently performs a one-dimensional optimisation over $\lambda = \lambda_1$. Output is given in the form of the complete solution path for the generalised lasso problem (equivalent to the fused lasso with the specification $D = \tilde{D}$), *i.e.* the values of λ at which the solution path changes slope. In addition, the residual sum of squares (*RSS*) and degrees of freedom (*df*) for the model fit corresponding to each value of the parameter λ are reported. It should be noted that since the matrix of discrete input data Y has more columns than rows, *i.e.* $p > N$, a warning is given to indicate that a small ridge penalty has been added to the generalized lasso criterion before the path is computed, as per R documentation on the *genlasso* package.

In order to proceed with feature selection using the fused lasso, it was necessary to select a single, optimal value of the parameter λ in the solution path reported by *genlasso()*. Tibshirani (2011) discusses a C_p-type statistic, defined as

$$C_p(\lambda) = RSS_\lambda - N\sigma^2 + 2\sigma^2 \mathrm{df}_\lambda \ , \tag{5.13}$$

that can be utilised for the purpose of model selection. Here, RSS_λ and df_λ denote the residual sum of squares and degrees of freedom respectively of the fused lasso fit corresponding to λ. Since the response variable Z only assumes values in the set $\{0,1\}$, an estimate of the unknown quantity $\sigma^2 = \mathrm{var}[Z] = \mathrm{var}[\varepsilon]$ is obtained for each split as

$$\hat{\sigma}^2 = \hat{P}(Z=1)\left[1 - \hat{P}(Z=1)\right], \tag{5.14}$$

the variance of a Bernoulli distribution. Here, the probability of a class $Z=1$ membership, $P(Z=1)$, is reasonably approximated by the proportion of class $Z=1$ items in the training data for each repetition.

Consequently the optimal value $\hat{\lambda} = \hat{\lambda}_1 \left(= \hat{\lambda}_2\right)$ for each training split is selected as

$$\hat{\lambda} = \underset{\lambda}{\mathrm{argmin}} \left\{\hat{C}_p(\lambda)\right\}, \tag{5.15}$$

where $\hat{C}_p(\lambda)$ denotes the estimated C_p statistic based on the estimate $\hat{\sigma}^2$ obtained in Equation (5.14).

The p-dimensional, estimated fused lasso coefficient vector $\hat{\boldsymbol{\beta}}^{\textit{fused lasso}}$ corresponding to the estimated parameter value $\hat{\lambda}$ can then be obtained using the function *coef()*. The reduced set of features, or wavelengths, selected by the fused lasso correspond to the non-zero elements

$\hat{\beta}^{fused\ lasso}$. Since the selected value of $\hat{\lambda}$ differs for each training data split, each repetition will result in a different number of selected variables in the fused lasso wavelength subset.

5.3.6 Functional support vector machines (FSVM) using the fused lasso wavelength subset

After selection of a subset of the full set of $p = 2203$ wavelengths for a specific repetition using the fused lasso, functional support vector machines based on this identified subset was applied to identify discoloured grape berries. The number of reduced features, which was specific to each split, is denoted by $p^{fused\ lasso}$.

The same methodology outlined in Section 5.3.4 for the fitting and adjustment of the functional SVM procedure using the full set of evaluation points was followed, with the exception of the following modifications:

- Since the fused lasso selects features in contiguous groups, it may be the case that these groups for a specific repetition are situated relatively far apart over the 800-2900 nm wavelength range. As a result of the compact support property of the B-spline basis system, evaluating the set of M=12 basis functions at each of the $p^{fused\ lasso}$ selected wavelengths, in the process of smoothing the discrete data, may produce a sparse basis matrix $\boldsymbol{B}$ with possibly entire column(s) of zeros. Consequently, the inverse matrix $\left(\boldsymbol{B}^T\boldsymbol{B}\right)^{-1}$ is not defined and, as a result, the basis function coefficient estimates that are considered as inputs into the standard SVM procedure cannot be determined.

  In order to address this problem, the number of basis functions considered in smoothing of the discrete data using a basis function expansion was reduced from $M = 12$ to the minimum value of $M^* = 4$, for all repetitions. Consequently, the vector of basis function coefficient estimates regarded as input into the SVM classification procedure for a specific repetition was obtained as

$$\hat{\xi}^{fl} = \left[\left(\mathbf{B}^{fl}\right)^T \mathbf{B}^{fl}\right]^{-1}\left(\mathbf{B}^{fl}\right)^T \mathbf{y}^{fl}, \tag{5.16}$$

where the vector $\boldsymbol{y}^{fl}$ contains the discrete NIR absorption values for a sample item (grape berry) measured at only the subset of wavelengths identified by the fused lasso. In addition, the $p^{fused\ \text{lasso}} \times M^*$ matrix $\boldsymbol{B}^{fl}$ contains the values of the set of basis functions evaluated at only the reduced wavelength set, *i.e.* $B_{jm} = \varphi_m\left(t_j\right)$, for $m = 1, 2, 3, 4$ and $j \in J$, where the set J contains the indices of the selected variables (for a specific repetition).

It should be noted that continuous estimates of the underlying functions are obtained by smoothing the discrete observations for each sample item using a basis function expansion comprised of $M^* = 4$ B-spline basis functions evaluated over the subset of wavelengths identified by the fused lasso, rather than the full 800-2900 nm wavelength range (as in the case of fitting an FSVM model based on the full set of evaluation points).

- The vector of basis function coefficient estimates $\hat{\xi}^{fl}$, as per Equation (5.16), is only defined for $p^{fused\ lasso} > M^*$. As a result, no binary functional classification was carried out for repetitions where the fused lasso selected fewer than $M^* = 4$ variables.

- The SVM parameter estimates $\hat{C}$ and $\hat{\gamma}$ reported in Section 5.3.4 are based on underlying function estimates for each sample item constructed using $M = 12$ B-spline basis functions, and are consequently inappropriate for application in FSVM model fitting based on a reduced feature set. In order to obtain more suitable estimates, the described tuning procedure using the functions *train()* and *sigest()* was repeated using the basis function coefficient estimate vectors $\hat{\xi}^{fl}$ rather than $\hat{\xi}$ for each of the $N = 407$ grape berries as input data. The subsequent parameter estimates were obtained as $\hat{\gamma}^* = 0.6804165$ and $\hat{C}^* = \hat{C} = 0.0001$.

5.4 Results

5.4.1 Model performance: test errors

The results of the empirical study are reported in Table 5.1. The overall test errors as well as the response group $Z = 0$ (no discolouration) and $Z = 1$ (discolouration) misclassification errors on the test data, averaged over the 30 repetitions, are used as performance measures. The test errors resulting from application of the following procedures are reported, following discussion of their implementation in Section 5.3:

Procedure	**Description**
A	Sparse partial least squares discriminant analysis (SPLSDA).
B	Linear discriminant analysis (LDA) for functional data.
C	Functional support vector machine (FSVM) based on the full set of $p = 2203$ evaluation points (wavelengths).
D	Functional support vector machine (FSVM) based on a subset of the full wavelength range identified by feature selection using the fused lasso. The number of selected variables, denoted by $p^{fused\ lasso}$, differs for each repetition.

To provide an indication of the variation in the performance of each method across the 30 splits into test and training subsets, the standard errors of each of the reported test errors for procedures A to D are also given in Table 5.1.

It should be noted that the fused lasso resulted in the selection of fewer than $M^* = 4$ wavelengths for repetitions 1, 3 and 10. Consequently, classification using functional support vector machines on the subset of $p^{fused\ lasso}$ wavelengths was not carried out for these splits. As a result, the test errors of procedure D are reported as averages over 27 rather than 30 outputs, contributing slightly to the higher degree of variation observed for this method.

Table 5.1: Average (overall and class specific) test and standard errors for each procedure

	Procedure							
	A		B		C		D	
Test error	**Mean**	**Standard error**	**Mean**	**Standard error**	**Mean**	**Standard error**	**Mean**	**Standard error**
Overall	0,331	0,052	0,326	0,050	0,445	0,031	0,465	0,068
Class Z=0	0,402	0,094	0,332	0,081	0,594	0,099	0,547	0,149
Class Z=1	0,304	0,077	0,323	0,069	0,388	0,045	0,434	0,119

To aid comparison of the performances of the considered classification and variable selection procedures, boxplots of the achieved test errors over the 30 repetitions are given in Figure 5.4.

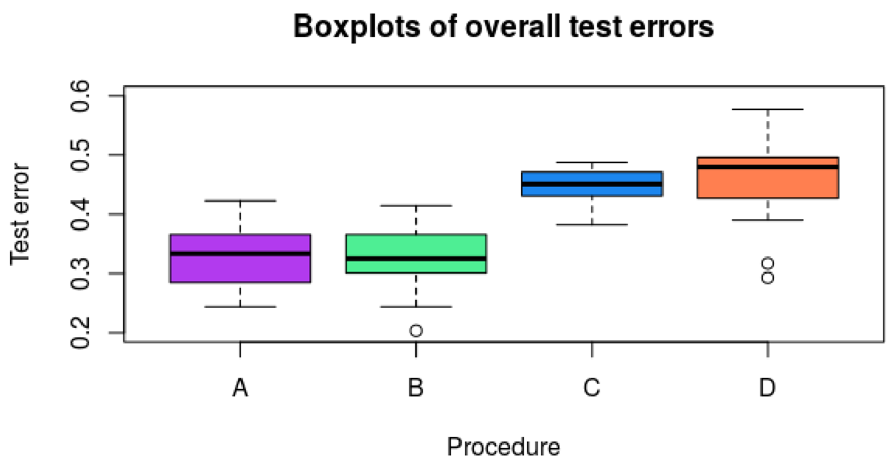

Figure 5.4

SPLS discriminant analysis and LDA for functional data achieve the lowest average misclassification errors over the 30 repetitions with a similar, and relatively low, degree of variation in the reported test errors across the splits. In addition to being the most accurate classification method in distinguishing between discoloured and non-discoloured grape berries, the latter procedure also achieves the most balanced errors across the two response groups.

A substantial deterioration in performance in terms of classification accuracy is observed for functional support vector machines based on the full set of p evaluation points. Furthermore, fine-tuning of the SVM procedure did not result in balanced class error rates. A further increase in the overall test error is observed for fitting based on the reduced set of the p wavelengths identified by the fused lasso. However, incorporating feature selection into the FSVM procedure did achieve more balanced group errors.

The performance of procedure D is the most dependent on the specific set of sample items included in the respective test and training data sets for a repetition since it exhibits the largest standard errors out of all the procedures applied to the functional data set. Procedures A to C exhibit relatively similar variation across the 30 splits.

5.4.2 **Variable selection**

Wavelengths that are selected more frequently by the SPLS and fused lasso procedures indicate variables that are deemed important by these methods in the identification of a discoloured grape berry. Frequency plots for both methods applied to the practical functional data set over 30 repetitions are given in given in Figures 5.5 and 5.7 respectively.

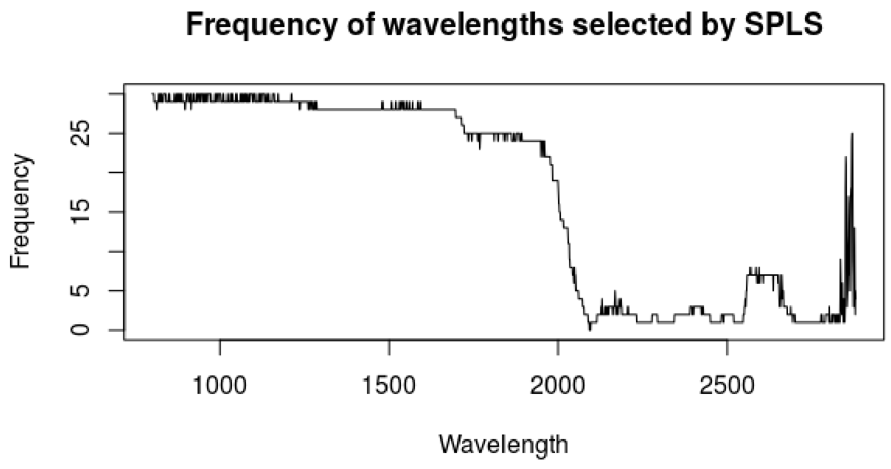

Figure 5.5

From Figure 5.5 it is seen that low wavelengths, those that fall into the 800-1200 nm range, are selected by the feature selection component of the SPLS procedure in almost every repetition. Wavelengths between approximately 1200 and 2000 nm are seen to be selected slightly less frequently, although still with a high frequency. Furthermore, a sharp decline in selection frequency is observed around 2000 nm, after which the high-wavelength variables are selected relatively infrequently by SPLS, with the exception of a spike in the frequency of selection at the upper end of the wavelength interval.

It was seen in Figure 5.2 that the low wavelength variables are likely to provide the most information in terms of class separation, based on the behaviour of the plotted estimated mean curves. This initial expectation is confirmed by the frequency curve in Figure 5.5, from which it is clear that the low wavelengths are selected most frequently across the 30 repetitions.

The horizontal fragments of the frequency curve in Figure 5.5 identify variables that are selected in blocks. Consequently, the application of sparse partial least squares to this specific functional data set resulted in the selection of individual wavelengths as well as a small number of contiguous intervals of features.

As discussed in Section 5.3.1, the number of latent components K considered as inputs into the SPLSDA procedure are selected for each repetition using the function *cv.spls()*. Figure 5.6 gives the frequency of the number of linear combinations selected over the 30 repetitions, the value of which ranges between $K = 1$ and $K = 6$.

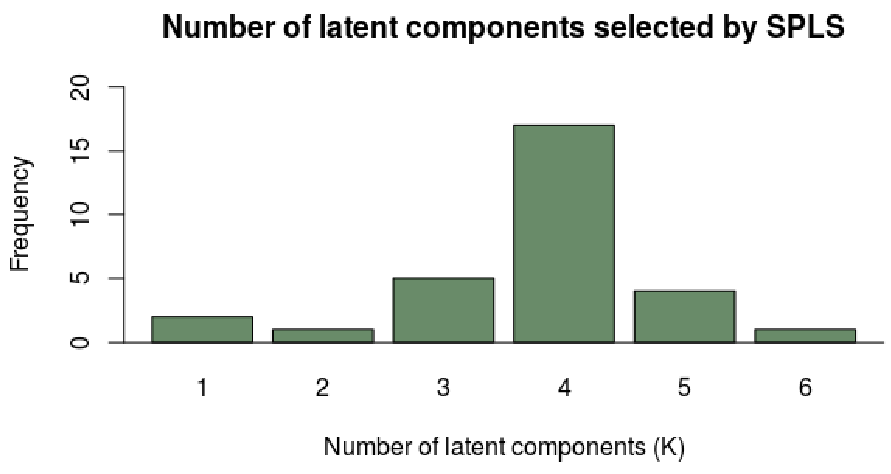

Figure 5.6

It is evident that fitting of the SPLSDA procedure is based on $K \in \{3, 4, 5\}$ latent components for a majority of the repetitions. A clear mode is observed at $K = 4$, accounting for more than half of the splits.

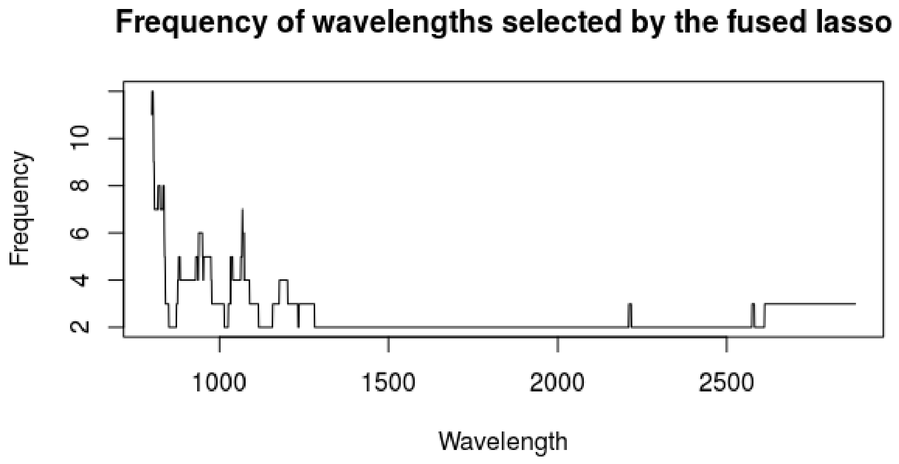

Figure 5.7

The property of the fused lasso that selected wavelengths occur in contiguous intervals rather than as isolated points spread out over the full continuum is easily identified by the horizontal fragments of Figure 5.7. As a part of the fused lasso fitting procedure, the number of contiguous intervals selected for each of the 30 splits was recorded. Only a single interval of variables was selected in each of the repetitions, of different lengths situated at different locations along the wavelength interval, with the exception of four out of the total 30 repetitions for which 2 contiguous intervals were chosen.

Low wavelengths are seen to be selected with a higher frequency than variables at larger wavelength values, a consistently observed trend in the results of the empirical study. However, a low frequency of selection is observed over the entire wavelength range, especially for variables at wavelengths larger than $\pm$ 1250 nm. This suggests that the fused lasso, implemented using the function *genlasso()* and selection of the tuning parameter λ based on minimisation of the estimated statistic $\hat{C}_p(\lambda)$, results in a subset comprised of too few variables. This is a conclusion that is strengthened by the poor performance of Procedure D in terms of classification accuracy.

5.4.3 Model interpretability

In order to further investigate the extent of separation between the two response groups achieved by sparse partial least squares and to aid interpretation of the procedure, scatterplots of the K SPLS component scores for the full set of N sample items can be constructed. For this purpose, the estimation and model fitting procedures outlined in Section 5.3.1 were followed, with the N x p matrix of discrete observations Y as input.

Cross-validation, using the function *cv.spls()*, selected $K = 3$ SPLS latent components on the basis of the full data set. Furthermore, the feature selection property of the sparse partial least squares procedure resulted in selection of the first 1212 wavelengths in the full set of $p = 2203$ evaluation points, as well as ω_{1215} and ω_{1216}. This result is again in line with the initial expectation that the low wavelength variables provide the most information in terms of class separation.

The component scores plotted in Figure 5.8 were computed according to

$$T_{ik} = \hat{\boldsymbol{w}}_k^T \boldsymbol{y}_i \text{ for } k = 1,2,3,\ i = 1,2,...,N\ , \tag{5.17}$$

where $\hat{\boldsymbol{w}}_k$ denotes the k^{th} estimated direction vector. The vector $\boldsymbol{y}_i$ contains the observed NIR absorption values for the i^{th} sample item, *i.e.* observed values of the original predictor vector $\boldsymbol{Y} = \left[Y_1, Y_2, ..., Y_p\right]^T$, as measured at each of the $p = 2203$ wavelengths.

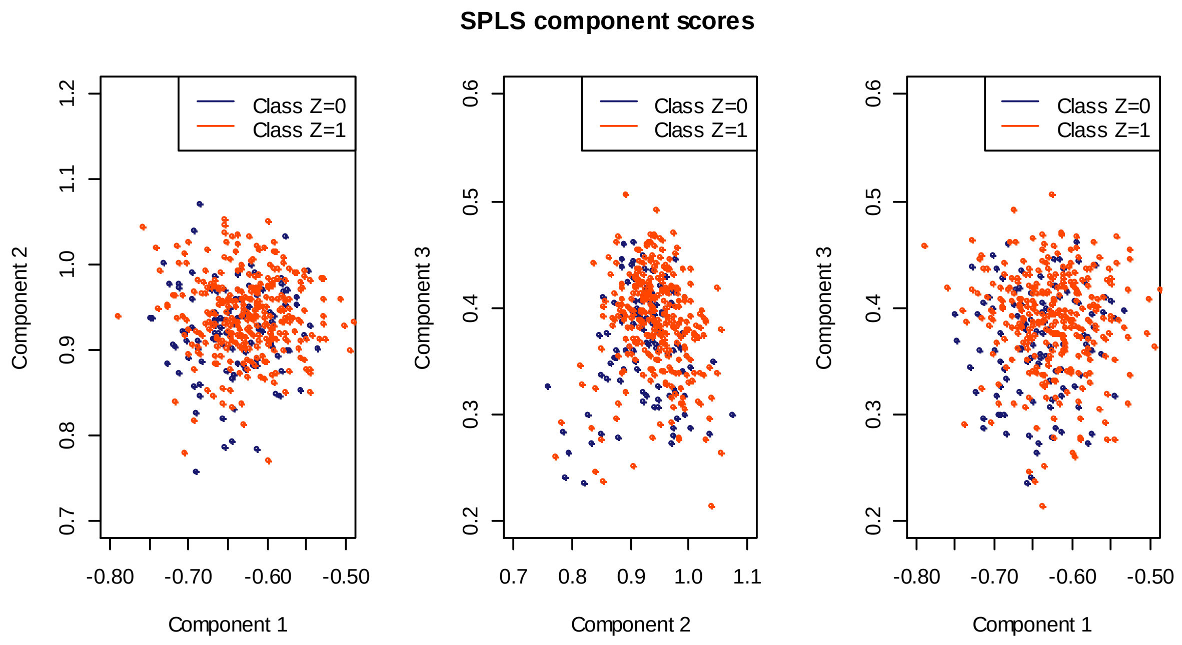

Figure 5.8

The left hand panel of Figure 5.8 gives a scatterplot of the first and second component scores, i.e. T_1 and T_2 for each of the N individual grape berries, representing the two directions of greatest variation in the data. It is evident that the scores for the two response groups overlap. A similar result is observed in both scatterplots of component scores given in the middle and right panels of the figure. Consequently, the SPLS fit does not seem to achieve a high degree of class separation in terms of location. However, by using these SPLS scores as inputs into the standard linear discriminant analysis classification method, *i.e.* application of the SPLSDA procedure, improved class separation can potentially be achieved translating to greater classification accuracy.

Furthermore, SPLS discriminant analysis constructs an (estimated) linear decision function $\hat{\delta}^{SPLSDA}(\boldsymbol{T})$ that depends on the SPLS latent components, i.e.

$$\hat{\delta}^{SPLSDA}(\boldsymbol{T}) = \hat{\beta}_1^{SPLSDA} T_1 + \hat{\beta}_2^{SPLSDA} T_2 + \hat{\beta}_3^{SPLSDA} T_3 \ . \qquad (5.18)$$

Standardisation of the inputs $T_k, k = 1, 2, ..., K$, translates to omission of the intercept term from $\hat{\delta}^{SPLSDA}(\boldsymbol{T})$. For the functional data set under consideration, the elements of the vector $\hat{\boldsymbol{\beta}}^{SPLSDA}$ were obtained as

$$\begin{aligned} \hat{\beta}_1^{SPLSDA} &= 10.961 \\ \hat{\beta}_2^{SPLSDA} &= 11.682 \; . \\ \hat{\beta}_3^{SPLSDA} &= 13.548 \end{aligned} \tag{5.19}$$

To aid interpretation of the fitted SPLSDA model, the coefficient estimates $\hat{\boldsymbol{\beta}}$ for the implied linear decision function, corresponding to $\hat{\delta}^{SPLSDA}(\boldsymbol{T})$, as a function of the original $p = 2203$ features $Y_1, Y_2, ..., Y_p$ rather than the latent components, can be computed. In particular, as per the discussion by Chung and Keleş (2010), the coefficient vector $\hat{\boldsymbol{\beta}}$ is computed according to the equation

$$\hat{\boldsymbol{\beta}} = \hat{W}\hat{\boldsymbol{\beta}}^{SPLSDA}, \tag{5.20}$$

where $\hat{W} = [\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \hat{\boldsymbol{w}}_3]$ denotes the p x K matrix of estimated direction vectors. The coefficient estimates $\hat{\boldsymbol{\beta}} = \left[\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_p\right]^T$ obtained for the functional data set with grape berries comprising the sample items, corresponding to the estimates given in Equation (5.19) for the latent components, are plotted as a function over the 800-2900 nm wavelength range in Figure 5.9.

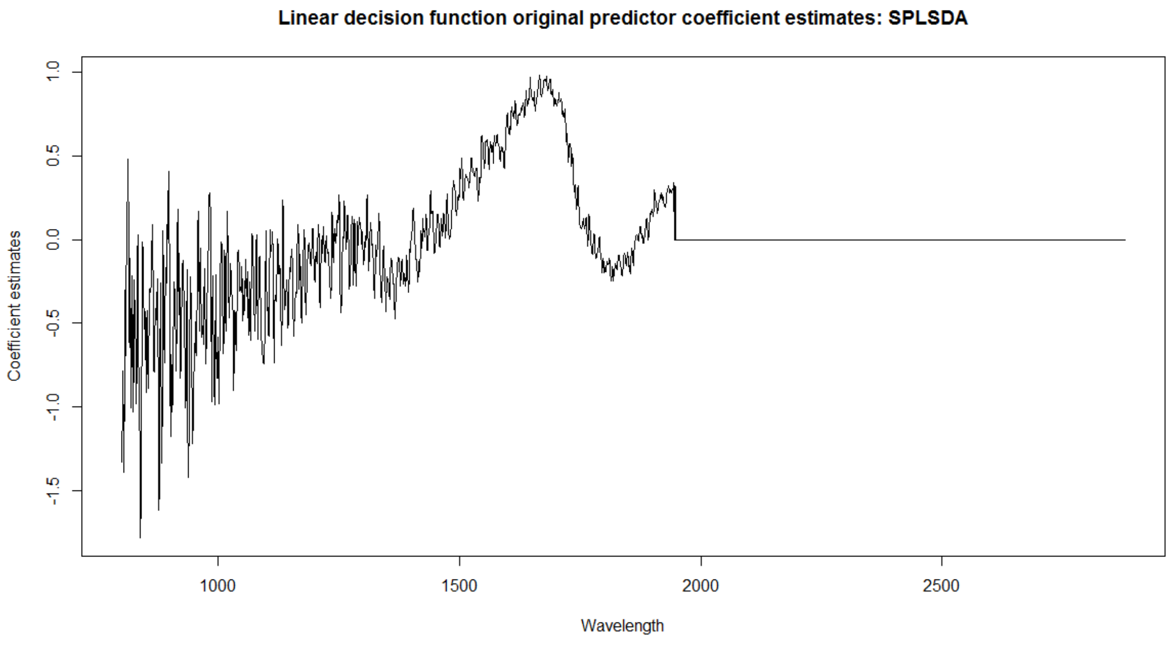

Figure 5.9

Since only low wavelength variables were selected by sparse partial least squares, the elements of the vector $\hat{\beta}$ corresponding to the unselected, high wavelengths are zero. Consequently, information provided by absorption values measured at the latter, unselected set of evaluation points is not utilised in classifying sample items. As a result, a reduction in both time and potentially the cost of data collection can be achieved by measuring NIR absorption values at only the set of low wavelength variables identified by feature selection.

Furthermore, the plotted function exhibits mostly negative coefficient estimates at wavelengths in the 800-1500 nm range as well as consecutive positive estimated values over a majority of the wavelengths between $\omega = 1500$ nm and $\omega = 2000$ nm. As a result, a sample item with large observed absorption values in the former, low wavelength range will tend to be classified to class $Z = 0$. Similarly, absorption measurements that are large in magnitude over the 1500-2000 nm range identify a discoloured grape berry, i.e. classification to the response group $Z = 1$. The practical implications of these observed results include further reducing the set of evaluation points, after the removal of wavelengths larger than (approximately) 2000 nm, by omitting all wavelengths in the measurement process other than a small set of consecutive wavelengths between the two ranges, *viz.* 800-1500 nm and 1500-2000 nm.

While linear discriminant analysis for functional data achieves the lowest overall and most balanced group test error rates out of the procedures applied to the practical data set, classification accuracy should not be considered in isolation. Model interpretability forms an additional important facet of model performance.

By considering the $p = 2203$ discrete wavelength measurements for each of the $N = 407$ sample items, the LDA (for functional data) linear decision function $\delta^{FLDA}(\boldsymbol{t})$ based on the full data set can be obtained by following the model fitting procedures outlined in Section 5.3.3. More specifically, the linear decision function is given by

$$\delta^{FLDA}(\boldsymbol{t}) = L_1(\boldsymbol{t}) - L_0(\boldsymbol{t}), \qquad (5.21)$$

where $L_0(\boldsymbol{t})$ and $L_1(\boldsymbol{t})$ are the respective class $Z = 0$ and $Z = 1$ discriminant functions, as defined in Equation 3.6. While the discriminant functions, and consequently $\delta^{FLDA}(\boldsymbol{t})$, consider the values $\hat{x}(\boldsymbol{t})$ as inputs, dependence on the underlying function estimate $\hat{x}(\ )$ is supressed in the notation for the sake of simplicity.

Consequently, the estimated decision boundary is obtained as

$$\hat{\delta}^{FLDA}(\boldsymbol{t}) = \hat{L}_1(\boldsymbol{t}) - \hat{L}_0(\boldsymbol{t}) = \hat{\beta}_0{}^{FLDA} + x(\boldsymbol{t})^T \hat{\boldsymbol{\beta}}^{FLDA} , \qquad (5.22)$$

with intercept

$$\hat{\beta}_0{}^{FLDA} = log\left(\frac{\hat{\pi}_1}{\hat{\pi}_0}\right) - \frac{1}{2}\left[\hat{\mu}_1(\boldsymbol{t}) - \hat{\mu}_0(\boldsymbol{t})\right]^T \hat{\Sigma}(\boldsymbol{t})^{-1}\left[\hat{\mu}_1(\boldsymbol{t}) - \hat{\mu}_0(\boldsymbol{t})\right]. \qquad (5.23)$$

It should be noted that a value $\hat{\delta}^{FLDA}(\boldsymbol{t}) < 1.0505$ corresponds to a response group $Z = 0$ classification, based on the adjusted classification rule in Equation (5.6).

The elements of the p-dimensional vector of estimated coefficients

$$\hat{\boldsymbol{\beta}}^{FLDA} = \hat{\Sigma}(\boldsymbol{t})^{-1}\left[\hat{\mu}_1(\boldsymbol{t}) - \hat{\mu}_0(\boldsymbol{t})\right], \tag{5.24}$$

are plotted as a function of wavelength over the 800-2900 nm range in Figure 5.10.

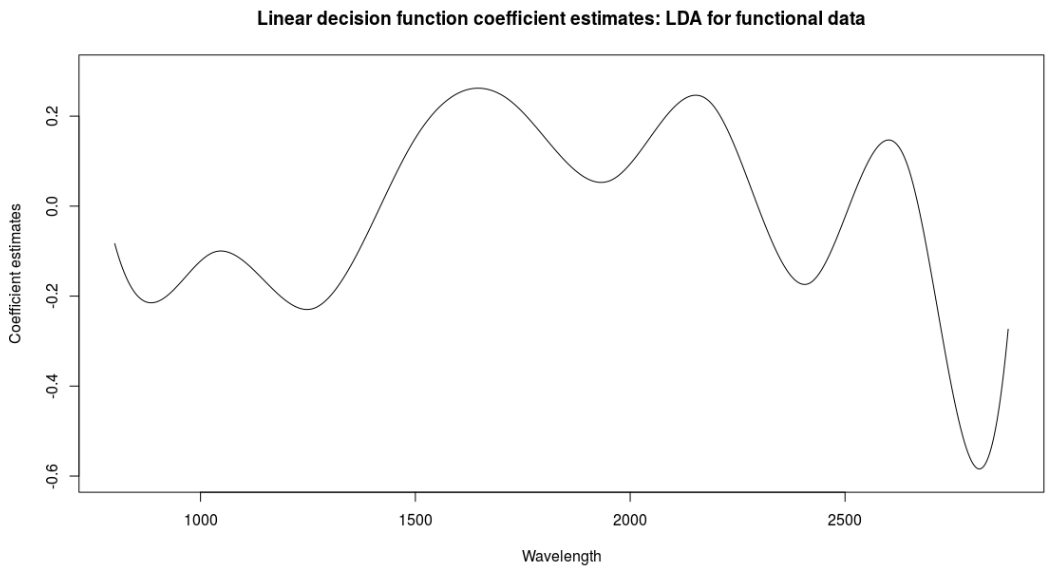

Figure 5.10

The SPLSDA linear decision function coefficient estimates plotted in Figure 5.9 correspond to the original features $\boldsymbol{Y} = \left[Y_1, Y_2, \ldots, Y_p\right]^T$, which are measured at the set of discrete evaluation points $\boldsymbol{t} = \left[t_1, t_2, \ldots, \mathrm{t}_p\right]^T$ along the wavelength continuum. In contrast, the estimates plotted in Figure 5.10 correspond to the underlying function estimate $\hat{x}(\ )$, obtained using a B-spline basis function expansion, evaluated at the discrete set of wavelengths $\boldsymbol{t} = \left[t_1, t_2, \ldots, \mathrm{t}_p\right]^T$. Consequently, $\hat{x}(t_1), \hat{x}(t_2), \ldots, \hat{x}(t_p)$ represent smoothed versions of the discrete variables $Y_1, Y_2, \ldots, Y_p$, accounting for the difference in the degrees of smoothness of the two sets of coefficient functions.

The coefficient estimate function in Figure 5.10 exhibits largely the same behaviour as observed for SPLSDA. Variables $\hat{x}(t)$ for $t \in (800 \text{ nm}, 1500 \text{ nm})$ correspond to negative estimates while those in the higher wavelength interval from 1500 nm to just over 2000 nm correspond to consistently positive coefficient estimates. Since no feature selection is performed by LDA for functional data, the coefficient function is non-zero over the 2000 + nm range as well. In particular, a small region of large negative coefficient estimates is observed around approximately 2700 nm. This behaviour is difficult to explain and could potentially be attributed to an end-point effect. The estimation could be repeated with a truncated dataset, wavelengths restricted to shorter than 2500 nm, to explore this result.

Successful application of the FSVM procedure to obtain class predictions required a substantial amount of fine tuning without achieving a corresponding improvement in classification performance in comparison to the test errors achieved by the other methods considered in the empirical study. This result was observed for fitting based on the full set as well as the fused lasso subset of evaluation points, bringing into question the usefulness of support vector machines as a functional classification procedure and, in addition, highlighting the danger of blind application of such a black-box method.

Chapter 6

Summary and Avenues for Further Research

Considering data sets that can be represented as a set of continuous, smooth functions varying over a continuum places the focus of the study on the field of functional data. In particular, a practical data set obtained through the measurement of near-infrared (NIR) absorption values at a discrete set of $p = 2203$ evaluation points over a 800-2900 nm wavelength range, for a sample of $N = 407$ individual grape berries, was considered for analysis. Accurate prediction of class $Z = 0$ and $Z = 1$ memberships for the sample items, with the latter class indicative of a discoloured grape berry, formed the main goal of the empirical part of the study.

While many classical statistical techniques can be extended for use in a functional context, attention was restricted to functional binary classification using functional support vector machines (FSVM) and linear discriminant analysis (LDA) for functional data. Feature selection using the fused lasso and a sparse equivalent of partial least squares (PLS) were considered to address the problem of multicollinearity that is widely recognised as a concern in functional data.

In the empirical study, the functional extension of LDA achieved an overall test error of around 33%, the highest observed classification accuracy in identifying discoloured grape berries out of all the procedures applied in the analysis. Based on absorption measurements made soon after harvesting, grapes that are likely to discolour can subsequently (potentially) be identified and channelled to a local market, translating to a financial saving for the grape export industry. It is likely that the financial implications of wrongly predicting discolouration may be greater than the cost associated with a misclassification to response group $Z = 0$, since a higher profit can presumably be earned in an international market.

Incorporating sparsity into the traditional method of partial least squares and using the resulting latent components as inputs into the LDA procedure had the effect of only a slight deterioration in model performance, as quantified by test errors. Consequently, SPLS maintained more or less the same degree of accuracy based on a reduced set of wavelengths. By identifying spectral variables that are good predictors of discolouration, a reduction in both the time and cost of data collection can be achieved since absorption measurements at fewer wavelengths are required. In particular, the low wavelength variables were seen to provide the most information in terms of class separation, a conclusion consistently observed in the exploratory analysis as well as the results of the empirical study.

Further improvements in the performance of SPLS as functional data analysis procedure can possibly be achieved by adjusting its variable selection component to choose variables in contiguous intervals rather than (mostly) isolated features over the continuum, thereby taking into account the natural ordering in the variables that is characteristic of data sets of this type. This could potentially be achieved by incorporating an additional, fused lasso-type penalty term in the criterion to be optimised (see Equation (4.11)).

Functional support vector machines performed relatively poorly on the practical data set, reflected in relatively large test errors, despite substantial fine-tuning of the procedure. Basing estimation of the SVM cost parameter C as well as the radial kernel function parameter γ on the specific set of training data for each repetition would be likely to improve the overall classification performance of the procedure and potentially achieve more balanced group errors. The reported results are based on fitting using the same estimates $\hat{C}$ and $\hat{\gamma}$ for each repetition, as estimated separately on the full set of $N = 407$ observations. This was done with the intention of achieving a saving in computation time.

Performance of the FSVM procedure was worsened by reducing the full set of wavelengths to a subset, specific to each repetition, identified by the fused lasso. This may be attributed to the selection of too few variables to provide sufficient information for accurate class separation, a consequence of basing the choice of the regularisation parameter λ from the output of the *genlasso()* function to minimise the (estimated) statistic $\hat{C}_p(\lambda)$. Alternative approaches for the specification of the value of λ, which controls the degree of regularisation, may result in improved predictive power. In addition, the fused lasso was implemented using the parameter specification $\gamma = 1$, implying the relationship $\lambda_1 = \lambda_2$. The effect of other values of γ on classification performance could be investigated.

Furthermore, alternative, and possibly more advanced, methods for selecting the number of basis functions M used in the smoothing of the discrete observations for each sample item may improve the performance of functional classification using FSVM and LDA for functional data. These methods respectively consider the resulting basis function coefficient estimates and the function estimates evaluated at discrete evaluation points as inputs. In addition, regularisation could be incorporated into smoothing of the discrete observations, as discussed in Section 2.4.

A problem encountered in the fitting of functional support vector machines based on the fused lasso subset was the non-invertibility of the matrix $\boldsymbol{B}^T\boldsymbol{B}$. The inverse of this matrix is required in the computation of the basis function coefficient estimates. As a solution, the number of basis functions was reduced to the minimum value of $M^* = 4$, as discussed in section 5.3.6. Alternative approaches to addressing this complication could be explored.

As a result of the smoothness of the curves representing the sample items of a functional data set, information provided by their derivatives can be utilised in classification, a property which is unique to functional data (Ramsay, 2013). Furthermore, in some spectrometric problems, the curvature of the spectrum is more relevant for class prediction than the spectrum itself (Rossi and Villa, 2005, p.641). Consequently, the functional data analysis could be extended to consider classifiers built on derivatives of the underlying function estimates, i.e. $D^m x = \frac{d^m x}{dt^m}, m = 1,2,\ldots$.

References

Arnold, T. & Tibshirani, R. 2011. *Introduction to the genlasso package* [Online]. Available: https://cran.r-project.org/web/packages/genlasso/vignettes/article.pdf [2016, September 5].

Biau, G., Bunea, F. & Wegkamp, M. 2005. Functional classification in Hilbert spaces. *IEEE Transactions on Information Theory*. 51(6), 2163-2172.

Chung, D. & Keleş, S. 2010. Sparse Partial Least Squares Classification for High Dimensional Data. *Statistical Applications in Genetics and Molecular Biology,* 9(1), Article 17.

Chun, H. & Keleş, S. 2010. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society Series B (Statistical methodology)*. 72 (1), 3-25.

Conan-Guez., B. & Rossi., F. 2004. Phoneme discrimination with functional multilayer Perceptron. *In IFCS 2004, Classification, Clustering, and Data Mining Applications*. Le Chesnay Cedex, France. 157-165.

Crisosto, C., Garner D. & Crisosto, G. 2002. High carbon dioxide atmospheres affect stored ‘Thompson Seedless’ table grapes. *Horticultural Science*. 37, 1074–1078.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. 2004. Least angle regression (with discussion). *Annals of Statistics.* 32(2), 407-499.

Fisher, R. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics.* 7,179-188.

Fourie, J. 2009. Browning of Table Grapes. *South African Fruit Journal*. 8(1), 52-53.

Hastie,T., Tibshirani, R. & Friedman, J. 2011. *The Elements of Statistical Learning: Data Mining, Inference and Prediction,* Second edition. New York: Springer.

James, A., Witten, D., Hastie, T. & Tibshirani, R. 2013. *An Introduction to Statistical Learning with Applications in R.* New York: Springer Science+Business Media.

James, G. & Hastie, T. 2001. Functional Linear Discriminant Analysis for Irregularly Sampled Curves. *Journal of the Royal Statistical Society. Series B (Statistical Methodology).* 63 (3), 533-550.

Kneip, A. & Gasser, T. 1992. Statistical tools to analyse data representing a sample of curves. *The Annals of Statistics.* 20 (3), 1266-1305.

Land,S. & Friedman,J. 1996. Variable fusion: a new method of adaptive signal regression. *Tehnical Report*. Department of Statistics, Stanford University, Stanford.

Nicolaï, B., Beullens, K., Bobelyn, E., Peirs, A., Theron, K. & Lammertyn, J. 2007. Non-destructive measurement of fruit and vegetable quality by means of NIR spectroscopy. *Postharvest Biology and Technology*. 46, 99–118.

Park, C., Koo, J., Kim, S., Sohn, I. & Lee, J. 2008. Classification of gene functions using support vector machine for time-course gene expression data. *Computational Statistics & Data Analysis.* 52, 2578-2587.

Ramsay, J. 2013. *Functional Data Analysis* [Online]. Available: http://www.psych.mcgill.ca /misc/fda/ [2016, September 16].

Ramsay, J., Hooker, G. & Graves, S. 2009. *Functional Data Analysis with R and MATLAB.* New York: Springer Science+Business Media.

Ramsay, J. & Li, X. 1998. Curve registration. *Journal of the Royal Statistical Society Series B (Statistical Methodology).* 60 (2), 351-363.

Ramsay, J. & Silverman, B. 2002. *Applied Functional Data Analysis: Methods and Case Studies.* New York: Springer Science+ Business Media.

Ramsay, J. & Silverman, B. 2005. *Functional Data Analysis.* New York: Springer Science+ Business Media.

Rossi, F. & Villa, N. 2005. Classification in Hilbert spaces with support vector machines, in *ASMDA 2005, International Symposium on Applied Stochastic Models and Data Analysis.* Brest, France. May. 635–642.

Rossi, F. & Villa, N. 2006. Support vector machine for functional data classification. *Neurocomputing.* 69(7-9), 730-742.

Silverman, B. 1995. Incorporating parametric effects into functional principal components analysis. *Journal of the Royal Statistical Society Series B (Methodological).* 57 (4). 673-689.

Tibshirani, R. 2011. The solution path of the generalised lasso. Ph.D. Stanford University.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J. & Knight, K. 2005. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B (Statistical Methodology).* 67, 91-108.

Wold, H. 1966. Estimation of Principal Components and Related Models by Iterative Least Squares. In *Multivariate Analysis*, Krishnaiah, P (ed.). New York: Academic Press.

Wold, H. 1975. *Path models with latent variables: The NIPALS approach*. New York: Academic Press.

Zou, H. & Hastie, T. 2005.Regularisation and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B (Statistical Methodology).* 67(2), 301-320.

Appendix

```
P.output<-function (Xmat,Yvek,eval.points,nmc)

{
  # Xmat=matrix of observed y values (log-transformed)
  # Yvek=vector of 0/1 responses
  # eval.points=vector of p evaluation points
  # nmc=number of repetitions, i.e. test and training splits


  p = length(eval.points)
  # Number of evaluation points
  n1 = sum(Yvek == 0)
  n2 = sum(Yvek == 1)
  n12 = n1 + n2
  ni = c(n1, n2)
  # Group sizes
  ngroepe = 2


  ## Penalty matrix for fused lasso ##

  gamma<-1
  Dmat = matrix(0,nrow=2*p-1,ncol=p)
  for (i in 1:(p-1)) {
  Dmat[i,i] = -1
  Dmat[i,i+1] = 1  }
  for (i in p:(2*p-1)) Dmat[i,i-p+1] = gamma


#### DATA SPLITTING ###

  ttr = 0.7
  # Training fraction
  pos = c(0, cumsum(ni))

  traingrt = floor(ttr * ni)
  # Number of training cases for each group
  testgrt = ni - traingrt
  # Number of test cases for each group
  ntrain = sum(traingrt)
  # Total number of training cases
  ntest = sum(testgrt)
  # Total number of test cases
  trainpos = c(0, cumsum(traingrt))
  testpos = c(0, cumsum(testgrt))
  train.label = c(rep(Yvek[1], traingrt[1]),rep(Yvek[n12],traingrt[2]))
  # Set up the response vector for the training cases
  test.label = c(rep(Yvek[1], testgrt[1]), rep(Yvek[n12], testgrt[2]))
  # Set up the response vector for the test cases

#### INITIALISE OUTPUT MATRICES/VECTORS ####

  test.error.splsda<-NULL
  # SPLSDA test errors
  error.0.splsda<-NULL
  # SPLSDA group 0 misclassification error (test data)
  error.1.splsda<-NULL
  # SPLSDA group 1 misclassification error (test data)
  K.cv<-NULL
  # Number of latent components used in SPLS
  vars.select.spls<-matrix(0,nrow=nmc,ncol=p)
  # Indicies of variables selected by SPLS

  test.error.flda<-NULL
  # Functional LDA test error
  error.0.flda<-NULL
  # Functional LDA group 0 misclassification error (test data)
  error.1.flda<-NULL
  # Functional LDA group 1 misclassification error (test data)
  test.error.fsvm<-NULL
  # Functional SVM test errors
  error.0.fsvm<-NULL
  # Functional SVM group 0 misclassification error (test data)
  error.1.fsvm<-NULL
  # Functional SVM group 1 misclassification error (test data)

  vars.select.fl<-matrix(0,nrow=nmc,ncol=p)
  # Indicies of variables selected by the fused lasso
  no.contig<-NULL
  # Number of contiguous intervals of variables selected by the fused lasso

  test.error.fsvm.sub<-NULL
  # Functional SVM on fused lasso subset test error
  error.0.fsvm.sub<-NULL
  # Functional SVM on fused lasso subset group 0 misclassification error
  (test data)
  error.1.fsvm.sub<-NULL
  # Functional SVM on fused lasso subset group 1 misclassification error
  (test data)


#### START OF SIMULATION LOOP ####

  for (jjj in 1:nmc){
  print(jjj)
  Traingr = NULL
  TrainInd = NULL

  ## Sample training and test cases from the two groups ##

  for (i in 1:ngroepe)
  {
      Traingr[[i]] = sample(((pos[i] + 1):(pos[i + 1])), traingrt[[i]])
      TrainInd = c(TrainInd, Traingr[[i]])
  }

  train.data.raw = Xmat[TrainInd, ]
  test.data.raw = Xmat[-TrainInd, ]


  ## Scale the training and the test data ##

  gemx = matrix(apply(train.data.raw, 2, mean))
  # Mean for each variable (columns) on training data
  sdx = matrix(apply(train.data.raw, 2, sd))
  # Standard deviation for each variable (columns) on training data
  train.data = scale(train.data.raw)
  # The scaled training data
  test.data = scale(test.data.raw, center = gemx, scale = sdx)
  # The scaled test data


#### SPLS ####

  ## Variable Selection using SPLS ##

  crossval<-cv.spls(x=train.data,y=train.label,fold=10,K=1:10,eta=0.8,
                    kappa=0.5,select="simpls",fit="simpls",scale.x=FALSE,
                    scale.y=FALSE, plot.it=FALSE)
  #Select tuning parameter K (number of latent components)
  K.cv[jjj]<-crossval$K.opt
  splsda.out<splsda(x=as.matrix(train.data),y=train.label,K=K.cv[jjj],
                    eta=0.8,kappa=0.5,scale.x=FALSE)
  vars.index<-splsda.out$A
  #Indicies of selected variables
  vars.select.spls[jjj,vars.index]= 1

  ## SPLDA ##

  index0<-which(train.label==0)
  # Indicies of the group zero training observations
  spls.coefs<-splsda.out$W
  # Matrix of estimated direction vectors (only non-zero for variables
    selected by SPLS)
  dir.vec<-matrix(0,nrow=p,ncol=K.cv[jjj])
  dir.vec[vars.index,]<-spls.coefs
  dir.vec

  lin.combs.train<-as.matrix(train.data)%*%dir.vec
  # Linear combinations (from SPLS) on the training data used as inputs
    into LDA
  lin.combs.test<-as.matrix(test.data)%*%dir.vec
  # Linear combinations (from SPLS) on the test data used as inputs into
    LDA

  ## LDA on latent components ##

  lin.combs.train0<-as.matrix(lin.combs.train[index0,])
  # Group 0 linear combinations (training data)
  lin.combs.train1<-as.matrix(lin.combs.train[-index0,])
  # Group 1 linear combinations (training data)
  cov.mat<-cov(lin.combs.train)
  # Common covariance matrix
  mean.vec0<-apply(lin.combs.train0,2,mean)
  # Class 0 mean vector
  mean.vec1<-apply(lin.combs.train1,2,mean)
  # Class 1 mean vector
  pi1<-sum(train.label)/ntrain
  # Class proportions (on training data)
  pi0<-1-pi1
  cov.mat.inv<-solve(cov.mat)
  L0<-function(input) {t(input)%*%cov.mat.inv%*%mean.vec0-
      0.5*t(mean.vec0)%*%cov.mat.inv%*%mean.vec0+log(pi0)}
  # Group 0 Discriminant function
  L1<-function(input) {t(input)%*%cov.mat.inv%*%mean.vec1-
      0.5*t(mean.vec1)%*%cov.mat.inv%*%mean.vec1+log(pi1)}
  # Group 1 Discriminant function

  ## Classify test data ##

  thresh1<-0.99
  # Used in adjusted LDA classification rule

  test.pred.splsda<-NULL
  for(i in 1:ntest)
  {
      eval.L0<-L0(input=lin.combs.test[i,])
      eval.L1<-L1(input=lin.combs.test[i,])
      test.pred.splsda[i]<-ifelse(eval.L0>eval.L1-thresh1,0,1)
  }
  test.pred.splsda

  test.error.splsda[jjj]<- sum(test.label!=test.pred.splsda)/ntest
  error.splsda<-error.calc2(predicted=test.pred.splsda,observed=test.label)
  error.0.splsda[jjj]<-error.splsda[[1]]
  error.1.splsda[jjj]<-error.splsda[[2]]


#### FUNCTIONAL LDA: FULL DATA SET ####

  ## Data smoothing: training data ##

  M<-12
  # Number of basis functions (selected separately on all the data)
  B.basis<-create.bspline.basis(rangeval=c(min(eval.points),
  max(eval.points)),nbasis=M)
  B.mat<-eval.basis(eval.points,B.basis)
  # Basis matrix
  train.0<-train.data[index0,]
  # Group 0 training cases
  train.1<-train.data[-index0,]
  # Group 1 training cases
  coefs.g0<-solve(crossprod(B.mat),crossprod(B.mat,t(train.0)))
  # Group 0 basis function coefficient estimates
  coefs.g1<-solve(crossprod(B.mat),crossprod(B.mat,t(train.1)))
  # Group 1 basis function coefficient estimates
  fun.g0<-fd(coef=coefs.g0,basisobj=B.basis)
  # Group 0 smooth function estimates
  fun.g1<-fd(coef=coefs.g1,basisobj=B.basis)
  # Group 1 smooth function estimates

  coefs.train<- cbind(coefs.g0,coefs.g1)
  # Basis coefficient estimates: all training cases
  fun.train<-fd(coef=coefs.train,basisobj=B.basis)
  # Smooth function estimates: all training cases

  ## Group specific mean functions (training data) ##

  fun.g0.val<-t(coefs.g0)%*%t(B.mat)
  # Group 0 smooth function estimate values at the p evaluation points
```

```
mean.g0.val<-apply(fun.g0.val,2,mean)
# Group 0 mean function estimate values at the p evaluation points
fun.g1.val<-t(coefs.g1)%*%t(B.mat)
mean.g1.val<-apply(fun.g1.val,2,mean)


## Common covariance function (training data) ##

covar.fun<-var.fd(fdobj=fun.train,fdobj2=fun.train)
covar.mat<-eval.bifd(sevalarg=eval.points,tevalarg=eval.points,
           bifd=covar.fun)
# Common covariance function evaluated at the p wavelengths
fun.train.val<-t(coefs.train)%*%t(B.mat)
# Function estimate values at the p evaluation points: all training cases
error<-(train.data-fun.train.val)^2
# Observed discrete values (y)- smooth estimated function values (xhat
  at the p evaluation points)
ave.p<-apply(error,1,mean)
# Average over the p evaluation points for each sample item
sigma.hat<-mean(ave.p)
# Average over the N sample items
Cov.mat<- covar.mat+sigma.hat*diag(p)
Cov.mat.inv<-solve(Cov.mat)


## Data smoothing: test data ##

coefs.test<-solve(crossprod(B.mat),crossprod(B.mat,t(test.data)))
fun.test.val<-t(coefs.test)%*%t(B.mat)


## Classify test cases ##

discr0.test<-NULL
# Group 0 discriminant function (evaluated for test cases)
for(i in 1:ntest)
{
    discr0.test[i]<-(t(fun.test.val[i,])%*%Cov.mat.inv%*%mean.g0.val)
                     -0.5*t(mean.g0.val)%*%Cov.mat.inv%*%mean.g0.val
                     +log(pi0)
}
discr0.test


discr1.test<-NULL
# Group 1 discriminant function (evaluated for test cases)
for(i in 1:ntest)
{
    discr1.test[i]<-(t(fun.test.val[i,])%*%Cov.mat.inv%*%mean.g1.val)
                      -0.5*t(mean.g1.val)%*%Cov.mat.inv%*%mean.g1.val
                      +log(pi1)
}
discr1.test

thresh2<-1.0505
# Used in adjusted LDA classification rule

test.pred.flda<-ifelse(discr0.test>discr1.test-thresh2,0,1)
# Test data class predictions
```

```
  test.error.flda[jjj]<- sum(test.label!=test.pred.flda)/ntest
  error.flda<-error.calc2(predicted=test.pred.flda,observed=test.label)
  error.0.flda[jjj]<-error.flda[[1]]
  error.1.flda[jjj]<-error.flda[[2]]

#### FUNCTIONAL SVM: FULL DATA SET ####

  train.label.svm<-2*train.label-1
  # Recode training labels from 0/1 to -1/1
  test.label.svm<-2*test.label-1
  # Recode test labels from 0/1 to -1/1


  ## Fit SVM on training data (Functional SVM: use basis coefficients as
     inputs) ##

  opt.gam<-0.3039255
  # FSVM parameter estimates (determined separately)
  opt.cost<-1e-04

  svmfit<-ksvm(x=t(coefs.train),y=train.label.svm,type="C-
               svc",kernel="rbfdot",

  kpar=list(sigma=opt.gam),C=opt.cost,prob.model=FALSE,scaled=TRUE)


  sv.index<-unlist(alphaindex(svmfit)[[1]])
  # Indicies of support vectors
  b<--1*unlist(b(svmfit))
  # Estimated intercept
  coefs.svm<-as.matrix(unlist(coef(svmfit)))
  # Coefficients (alpha hats) times -1/1 class lables for the
    support vectors
  coefVol<-rep(0,ntrain)
  coefVol[sv.index]<-coefs.svm
  rbf<-rbfdot(sigma=opt.gam)
  # Radial kernel function


  ## Adjusted SVM intercept(balance group training error rates) ##

  # Predict training data #

  gmat.train<-kernelMatrix(kernel=rbf,x=t(coefs.train),y=t(coefs.train))
  # Matrix of kernel function values on training data
  q.train<-matrix(t(as.matrix(coefVol))%*%gmat.train,ncol=1)

  fhats<-q.train+b
  fhats.sort<-sort(x=fhats,decreasing=FALSE)

  splits<-NULL
  for(i in 1:(ntrain-1))
  {
     splits[i]=(fhats.sort[i]+fhats.sort[i+1])/2
  }
  splits

  truez0<-which(train.label.svm==-1)
  # Indicies of true class 0 cases
```

```
truez1<-which(train.label.svm==1)
# Indicies of true class 1 cases

diff<-NULL
# Difference between group 1 and group 0 error for each split value
for(i in 1:(ntrain-1))
{
  class0<-which(fhats<splits[i])
  class1<-which(fhats>splits[i])

  zero.error<-0
  for(j in 1:length(truez0))
  {
    zero.error<-zero.error+any(class1==truez0[j])
  }
  zero.error
  zero.err<-zero.error/length(truez0)

  one.error<-0
  for(j in 1:length(truez1))
  {
    one.error<-one.error+any(class0==truez1[j])
  }
  one.error
  one.err<-one.error/length(truez1)

  diff[i]<-abs(zero.err-one.err)

}
diff

diff.min<-which(diff==min(diff))
# Index of minimum difference in group training errors
opt.split<-splits[diff.min]
# Optimal split on training data


## Predict test data ##

b.star<-b-opt.split
# Adjusted svm intercept value on the specific training data split
gmat.test<-kernelMatrix(kernel=rbf,x=t(coefs.train),y=t(coefs.test))
# Matrix of kernel function values on test data
q.test<-matrix(t(as.matrix(coefVol))%*%gmat.test,ncol=1)

pred.test.svm<-sign(q.test+b.star)
# Test data class predictions

test.error.fsvm[jjj]<- sum(test.label.svm!=pred.test.svm)/ntest
test.error.out<error.calc1(predicted=pred.test.svm,
                            observed=test.label.svm)
error.0.fsvm[jjj]<-test.error.out[[1]]
error.1.fsvm[jjj]<-test.error.out[[2]]
```

```
#### FUSED LASSO ####

  ## Fit fused lasso model ##

  fl.out<-genlasso(y=train.label,X=train.data,D=Dmat,
                    verbose=FALSE,maxsteps=2000,minlam=0)

  rss<-summary(fl.out)[,3]
  # RSS for each lambda value
  fl.lambda<-summary(fl.out)[,2]
  # Vector of lambda values in the full genlasso solution path
  df<-summary(fl.out)[,1]
  # df for each lambda value

  ## Cp statistic to select lambda ##

  estimated.sigma<-pi1*(1-pi1)
  Cp<-rss+2*estimated.sigma*df

  min.ind<-which(Cp==min(Cp))
  lambda.min<-fl.lambda[min.ind[1]]
  # Value of lambda minimising Cp


  fl.coefs<-round(coef(fl.out,lambda=lambda.min)$beta,digits=4)
  # Genlasso fit for a specific value of lambda
  variables.ind<-which(fl.coefs!= 0)
  # Indicies of non-zero coefficients (variables selected by fused lasso)
  vars.select<-rep(0,p)
  vars.select[variables.ind]=1
  vars.select.fl[jjj,]<- vars.select

  s <- split(variables.ind, cumsum(c(0, diff(variables.ind) != 1)))
  run.info <- list(lengths = unname(sapply(s, length)), values = unname(s))
  no.contig[jjj]<-length(run.info$lengths)
  # Number of contiguous intervals selected by the fused lasso


#### FUNCTIONAL SVM: FUSED LASSO SUBSET ####

  ## Subsets selected by the fused lasso ##

  eval.points.sub<-eval.points[variables.ind]
  train.sub<-train.data[,variables.ind]
  test.sub<-test.data[,variables.ind]
  p.sub<-length(variables.ind)

  M.sub<-4
  # Adjusted number of basis functions


  ## Data smoothing: subset of training data ##
  # Classification is only carried out if M.sub<p.sub

  if (M.sub<p.sub) {

      B.basis.sub<-create.bspline.basis(rangeval=c(min(eval.points),
                    max(eval.points)),nbasis=M.sub)
      B.mat.sub<-eval.basis(eval.points.sub,B.basis.sub)
```

```
mat<-solve(t(B.mat.sub)%*%B.mat.sub)%*%t(B.mat.sub)
coefs.train.sub<-mat%*%t(train.sub)
coefs.test.sub<-mat%*%t(test.sub)

# Fit SVM #

opt.gam.sub<-0.6804165
svmfit2<-ksvm(x=t(coefs.train.sub),y=train.label.svm,type="C-
              svc",kernel="rbfdot",kpar=list(sigma=opt.gam.sub),
              C=opt.cost,prob.model=FALSE,scaled=TRUE)

sv.index2<-unlist(alphaindex(svmfit2)[[1]])
b2<--1*unlist(b(svmfit2))
coefs.svm2<-as.matrix(unlist(coef(svmfit2)))
coefVol2<-rep(0,ntrain)
coefVol2[sv.index2]<-coefs.svm2

# Adjusted SVM intercept(balance group training error rates) #

# Predict training data #

rbf.sub<-rbfdot(sigma=opt.gam.sub)

gmat.train.sub<-kernelMatrix(kernel=rbf.sub,x=t(coefs.train.sub),
                             y=t(coefs.train.sub))
q.train.sub<-matrix(t(as.matrix(coefVol2))%*%gmat.train.sub,ncol=1)

fhats2<-q.train.sub+b2
fhats2.sort<-sort(x=fhats2,decreasing=FALSE)

splits2<-NULL
for(i in 1:(ntrain-1))
{
  splits2[i]=(fhats2.sort[i]+fhats2.sort[i+1])/2
}
splits2


diff2<-NULL
for(i in 1:(ntrain-1))
{
  c0<-which(fhats2<splits2[i])
  c1<-which(fhats2>splits2[i])

  zero.e<-0
  for(j in 1:length(truez0))
  {
    zero.e<-zero.e+any(c1==truez0[j])
  }
  zero.e
  zero.er<-zero.e/length(truez0)

  one.e<-0
  for(j in 1:length(truez1))
  {
    one.e<-one.e+any(c0==truez1[j])
  }
  one.e
  one.er<-one.e/length(truez1)
```

```
        diff2[i]<-abs(zero.er-one.er)

      }
      diff2

      diff2.min<-which(diff2==min(diff2))
      opt.split2<-splits2[diff2.min]


      # Predict test data#

      b.star2<-b2-opt.split2

      gmat.test.sub<-kernelMatrix(kernel=rbf.sub,x=t(coefs.train.sub),
                                   y=t(coefs.test.sub))
      q.test.sub<-matrix(t(as.matrix(coefVol2))%*%gmat.test.sub,ncol=1)

      pred.test.svm.sub<-sign(q.test.sub+b.star2)
      # Test data class predictions

      test.error.fsvm.sub[jjj]<-sum(test.label.svm!=pred.test.svm.sub)
                                  /ntest
      error.svm.sub<-error.calc1(predicted=pred.test.svm.sub,
                                   observed=test.label.svm)
      error.0.fsvm.sub[jjj]<-error.svm.sub[[1]]
      error.1.fsvm.sub[jjj]<-error.svm.sub[[2]]

    }  else {

      test.error.fsvm.sub[jjj]<-NA
      error.0.fsvm.sub[jjj]<-NA
      error.1.fsvm.sub[jjj]<-NA

      # No classification errors for splits with M.sub>p.sub

    }

  }

  output<- list("test.error.splsda"=test.error.splsda,"error.0.splsda"
  =error.0.splsda,"error.1.splsda"=error.1.splsda,"K"=K.cv,
  "vars.select.spls"=vars.select.spls, "test.error.flda"=test.error.flda,
  "error.0.flda"=error.0.flda,"error.1.flda"=error.1.flda,"test.error.fsvm"
  =test.error.fsvm,"error.0.fsvm"=error.0.fsvm,"error.1.fsvm"=error.1.fsvm,
  "vars.select.fl"=vars.select.fl,"no.contig"=no.contig,
  "test.error.fsvm.sub"=test.error.fsvm.sub,"error.0.fsvm.sub"
  =error.0.fsvm.sub,"error.1.fsvm.sub"=error.1.fsvm.sub)

  return(output)

}
```