Classification Using A Functional Partial Least Squares Logistic Regression Method

by

Aaron McAtee

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 8, 2016

Keywords: classification, partial least squares, functional data analysis

Copyright 2016 by Aaron McAtee

Approved by

Nedret Billor, Professor of Mathematics and Statistics
Peng Zeng, Associate Professor of Mathematics and Statistics
Guanqun Cao, Assistant Professor of Mathematics and Statistics
George Flowers, Dean, Graduate School

Abstract

Statistical analysis of functional data has been explored extensively over the last decade and functional partial least squares regression has emerged as a popular choice for classification problems. In partial least squares algorithms, uncorrelated components are derived iteratively by finding linear combinations of the predictors that maximize the variance between the predictors and the response. In this paper, we will develop a method to extract the components that explicitly considers the predictive power of the individual predictors. If an individual predictor does not display high predictive power as well as high covariance with the response, then their coefficients will be set to zero. This modified partial least squares method will be used to develop a set of uncorrelated latent variables, called mPLS components. The mPLS components will be used as the predictor variables in the logistic regression model. The efficacy of our algorithm will be assessed using fractional anisotropy data.

Acknowledgments

I would like to thank my advisor, Dr. Nedret Billor. You have provided me with many opportunities to hone my skills as a statistician and you have allowed me to explore different aspects of the field. Your willingness to adapt to my changing career goals is much appreciated. I have benefited greatly from your guidance throughout my time at Auburn. Thank you to the rest of my committee, Dr. Peng Zeng and Dr. Guanqun Cao. I have learned a lot from both of you.

Thank you, Kathy! As I continue to wind my way through various careers, I know I can always count on your support. The journey down different paths is always enjoyable with you by my side. We are moving on to the next challenge.

Table of Contents

Abstract ii

Acknowledgments iii

List of Figures vi

List of Tables vii

1 Introduction 1

2 Functional Data Analysis 3

2.1 Basis Functions 4

2.2 Fitting a Function to Observations 5

2.3 Example 7

3 Logistic Regression 10

3.1 Multivariable Predictors 10

3.2 Functional Predictor 11

4 Functional PLS Logistic Regression 14

4.1 Partial Least Squares 14

4.2 Functional Partial Least Squares 16

4.3 Functional PLS Logistic Regression 16

4.4 Power of Prediction Statistics 18

4.5 PLS Components Extraction 20

4.6 Regressing Response on PLS Components 21

4.7 Regression Coefficients in Terms of Original Variables 22

5 Case Studies 24

5.1 Case Study 1: Two Class Curve Simulation 24

5.2 Case Study 2: Diffusion Tensor Imaging 26

5.2.1 Analysis of DTI - FA Data 27

6 Conclusions and Recommendations 30

Bibliography 31

Appendix 33

List of Figures

2.1 Basis Systems 4

2.2 Sample Curve 8

2.3 Sample Functions 9

2.4 Sample Functions Generated with Roughness Penalty 9

4.1 Receiver Operator Characteristic Curve 19

5.1 Simulated Curves 25

5.2 Logistic Regression Parameter Function 26

5.3 FA Tracts 27

List of Tables

5.1 Simulated Curves: Classification Error Rates 26

5.2 DTI-FA: Classification Error Rates Using B-spline Basis Functions 28

5.3 DTI-FA: Classification Error Rates Using Fourier Basis Functions 29

Chapter 1

Introduction

Statistical analysis of functional data has been explored extensively over the last decade. Generalized linear regression models have been developed by James[10], Cardot and Sarda [4] and Müler and StadtMüler [15]. Principal component regression by Aguilera et al. [1] and partial least squares regression by Preda and Saporta [16] propose non-parametric models for regression on functional data. Partial Least Squares (PLS) regression is an attractive alternative to Principal Component (PC) regression for classification problems because it takes into consideration the variance between the predictor and response when selecting regression components.

In this paper we are interested in using PLS logistic regression to classify a binomial response when the predictor is functional. Logistic regression using a PLS technique was developed in Bastien et al. [3]. In their paper, PLS components are derived iteratively by finding linear combinations of the predictors that maximize the variance between the predictors and the response. Escabias et al. [8] extended this method to include functional predictors. We will develop a method to extract the PLS components that explicitly considers the predictive power of the individual predictors. This modified partial least squares (mPLS) method will be used to develop a set of uncorrelated latent variables, called mPLS components. The mPLS components will be used as the predictor variables in the logistic regression model.

An overview of functional data analysis is given in Chapter 2, followed by the essentials of logistic regression in Chapter 3. A detailed account of the PLS logistic regression model, with our mPLS component extraction method, will be covered in Chapter 4. Two case studies will be undertaken in Chapter 5 to determine the efficacy of our method. The first study is conducted on simulated curve data. The second is conducted on Fractional Anisotropy (FA) data collected using Diffusion Tensor Imaging. The MRI/DTI data were collected at Johns Hopkins University and the Kennedy-Krieger Institute.

Chapter 2
Functional Data Analysis

Functional data can be defined as data that is accurately described by a continuous function. Although, by necessity, most data is collected at discrete points in time; there is a class of data where it is expected that the change in values from time t to time $t+1$ possess a certain level of smoothness. Examples of data that can be considered functional are the height of a child measured monthly, the ambient air temperature at a given location measured hourly, or a person's heart rate measured every minute during a 30 minute exercise period.

If we can adequately describe the functional form of the changes in data over time, then we can use the entire function as a single predictor variable. This is in contrast to the approach used in longitudinal data analysis, where every measurement is treated as a separate data point. Hopefully, by using a functional form of the data, we can exploit the inherent smoothness of the data when we conduct our statistical analysis.

Suppose we have n subjects and we measure a characteristic of each subject N times during an interval T. Let $\{x_{ij}\}$ be the set of discrete observations, where $i = 1, \ldots, n$ and $j = 1, \ldots, N$. We will assume that there are smooth functions $x_i(t)$ in L^2 and in the interval $\{0, 1\}$, such that

$$x_{ij} = x_i(t_j) + \varepsilon_{ij}, \quad i = 1, \ldots, n, j = 1, \ldots, N \tag{2.1}$$

where the ε_{ij}'s are measurement errors.

The first step in functional data analysis is to define a system to represent the functions $x_i(t)$. We will represent the functions $x_i(t)$ as a linear combination of simpler functional building blocks $\phi_1, \ldots, \phi_k$, called basis functions.

2.1 Basis Functions

Basis functions allow us to represent inherently complicated functions with relatively simpler, computationally efficient functions. Although there are many possible choices of basis functions, we will only consider two widely used sets of functions: Fourier and B-spline.

The Fourier basis system use the following trigonometric functions as its building blocks:

$$1, sin(\omega t), cos(\omega t), sin(2\omega t), cos(2\omega t), \ldots, sin(k\omega t), cos(k\omega t) \tag{2.2}$$

where ω is related to the period of the function (T), by the relation $\omega = 2\pi/T$.

These functions are ideal for describing data that is periodic and working with these functions is computationally efficient. A set of Fourier basis functions are fully defined by setting the number of basis functions K and the period T. The left panel of figure 2.1 shows a thirteen Fourier basis function system with a period of 1.

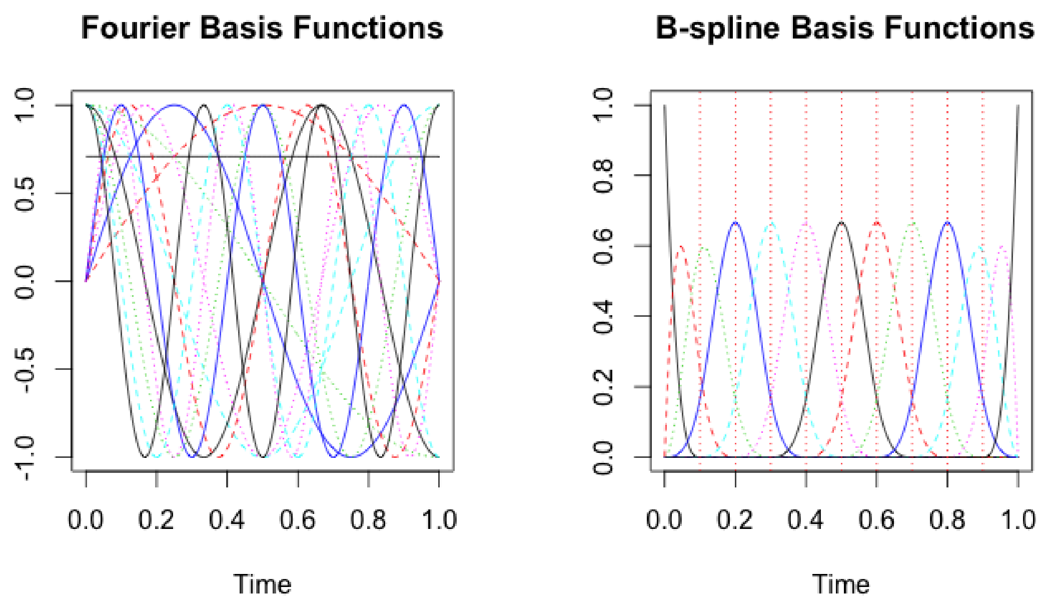

Figure 2.1: Basis Systems

Another popular choice is the B-spline basis system. B-splines are piecewise polynomials of degree n with C^{n-1} continuity at common points called knots. Requiring continuity at the knots ensures smoothness of the function. A B-spline basis system is defined by the range of validity, the number and placement of knots, and the order of the polynomials. In this thesis, 4^{th} degree polynomials with the knots equally spaced within the interval of observation are used. The right panel of figure 2.1 shows a thirteen B-spline basis function system with equally spaced knots defined on the interval $[0, 1]$.

Regardless of the basis function system that is chosen, the linear expansion of the observed covariate has the following form for K basis functions:

$$x_i(t) = \sum_{k=1}^{K} a_{ik}\phi_k(t) \tag{2.3}$$

where a_{ik} is the i^{th} subject's coefficient associated with the k^{th} basis function.

The number of basis functions K that are used to represent $x_i(t)$ is an important factor to consider. If $K = N$, then $x_i(t)$ will fit the discrete observations perfectly and the ε_{ij}'s in equation 2.1 will be equal to zero. In this case, we are modeling the measurement error in our function and it is unlikely that the function will be smooth. If we use too few basis functions, $K << N$, to represent $x_i(t)$, then we may smooth out important characteristics of the data.

2.2 Fitting a Function to Observations

After a basis system has been chosen, the number of basis functions and their corresponding basis coefficients must be determined. For ease of notation, since the basis coefficients for each subject are determined independently, we will drop the subscript i. Thus, equation 2.1 can be written

$$x_j = x(t_j) + \varepsilon_j = \sum_{k=1}^{K} a_k\phi_k(t) + \varepsilon_j, \;\; j = 1, \ldots, N \tag{2.4}$$

where j denotes the observation point, N is the total number of observations, K is the number of basis functions, $a_1, \ldots, a_K$ are the basis function coefficients, $\phi_1, \ldots, \phi_K$ are the basis functions, and ε_j is the error at observation point j.

This can be written in matrix form as

$$\mathbf{x} = \mathbf{x}(t) + \varepsilon = \mathbf{a}'\mathbf{\Phi} + \varepsilon \tag{2.5}$$

where $\mathbf{x} = (x_1, \ldots, x_N)'$, $\mathbf{x}(t) = (x(t_1), \ldots, x(t_N))'$, $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_N)'$, $\mathbf{\Phi} = (\phi_1(t), \ldots, \phi_K(t))$ and $\mathbf{a} = (a_1, \ldots, a_K)'$.

The simplest strategy for estimating the vector of coefficients, $\mathbf{a}$, for a given choice of K basis functions, is to assume the errors are independent and normally distributed and to minimize the least squares criterion:

$$SMSEE(\mathbf{x}|\mathbf{a}) = (\mathbf{x} - \mathbf{\Phi}\mathbf{a})'(\mathbf{x} - \mathbf{\Phi}\mathbf{a}) \tag{2.6}$$

This unweighted least squares criteria is only optimal if the residuals at all observation points are normally distributed with constant variance. A more accurate approach would be to use a weighted least squares fit. We can account for non stationary and autocorrelated errors by including a weighting matrix $\mathbf{W}$. In this case, the least squares criterion becomes:

$$SMSEE(\mathbf{x}|\mathbf{a}) = (\mathbf{x} - \mathbf{\Phi}\mathbf{a})'\mathbf{W}(\mathbf{x} - \mathbf{\Phi}\mathbf{a}), \tag{2.7}$$

where $\mathbf{W} = \Sigma_{\varepsilon}^{-1}$ is the inverse of the residuals covariance matrix. The estimate for the vector of coefficients is $\hat{\mathbf{a}} = (\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi})^{-1}\mathbf{\Phi}'\mathbf{W}\mathbf{x}$.

A third method of fitting a function to the discrete observations is via a roughness penalty. Using a high-dimensional basis with a roughness penalty will reduce the probability that important features of the data are missed, which can occur if we use a basis set that is too small for the application. Adding a roughness penalty to equation 2.5 gives us

$$\mathbf{x} = \mathbf{x}(t) + \lambda PEN_2(x) + \varepsilon = \mathbf{a}'\mathbf{\Phi} + \lambda PEN_2(x) + \varepsilon \tag{2.8}$$

Once choice for the penalty function is $PEN_2(x) = \int [D^2 x(s)]^2 ds$. This is the integrated squared second derivative of the function and λ is a smoothing parameter. The penalty term allows us to reduce the variation as much as we want to the solution of the differential equation $D^2 x(s)$. The best value for λ can be computed using cross-validation. The criterion to be minimized takes the following form:

$$PENSSE_\lambda(\mathbf{x}|\mathbf{a}) = (\mathbf{x} - \mathbf{\Phi}\mathbf{a})'\mathbf{W}(\mathbf{x} - \mathbf{\Phi}\mathbf{a}) + \lambda PEN_2(x) \tag{2.9}$$

The expression for the estimated coefficient vector is

$$\hat{\mathbf{a}} = (\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda PEN_2(\mathbf{x}))^{-1}\mathbf{\Phi}'\mathbf{W}\mathbf{x} \tag{2.10}$$

The R package "fda" [18] has the necessary functionality to create functional data objects from discrete observations using any of the three methods discussed above. In the following example, we look at the curve generation process using a set of simulated curve data. This simulated data will also be used in the first case study in chapter 5.

2.3 Example

1000 curves were simulated from two different classes. Each class contained 500 curves simulated using the following functions:

$$x_1(t) = uh_1(t) + (1-u)h_2(t) + \varepsilon(t) \tag{2.11}$$

$$x_2(t) = uh_1(t) + (1-u)h_3(t) + \varepsilon(t) \tag{2.12}$$

where u and $\varepsilon(t)$ are simulated uniform and standard normal random variables, respectively, and

$$h_1(t) = max[6 - |t-1|, 0], \;\; h_2(t) = h_1(t-4), \;\; h_3(t) = h_1(t+4) \tag{2.13}$$

The discrete observations that make up the curves were simulated at 101 equally spaced points on the interval $[1, 21]$. Figure 2.2 shows one of the simulated curves. This is merely the discrete observations joined piecewise, thus it lacks the smoothness that we expect to see in a function.

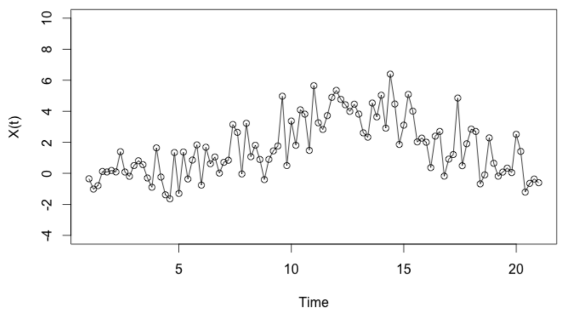

Figure 2.2: Sample Curve

Figure 2.3 shows functional curves for the same data after it is converted to a functional data object using the "fda" package in R. In the top left panel, the data was fitted with a set of 13 Fourier basis functions with a period equal to the interval of observation. In the right panel, the data was fitted with a set of 13 fourth-order B-spline basis functions. The knots were equally spaced at the observation times. The bottom left and right panels show the functional curves produced using 45 Fourier and 45 B-spline basis functions, respectively.

The B-spline and Fourier basis sets appear to fit the data equally well for this particular curve. The sum of squares of the residuals decreased from 1.1 to 0.96 when we increased the number of basis functions form 13 to 45. By increasing the number of basis functions, we

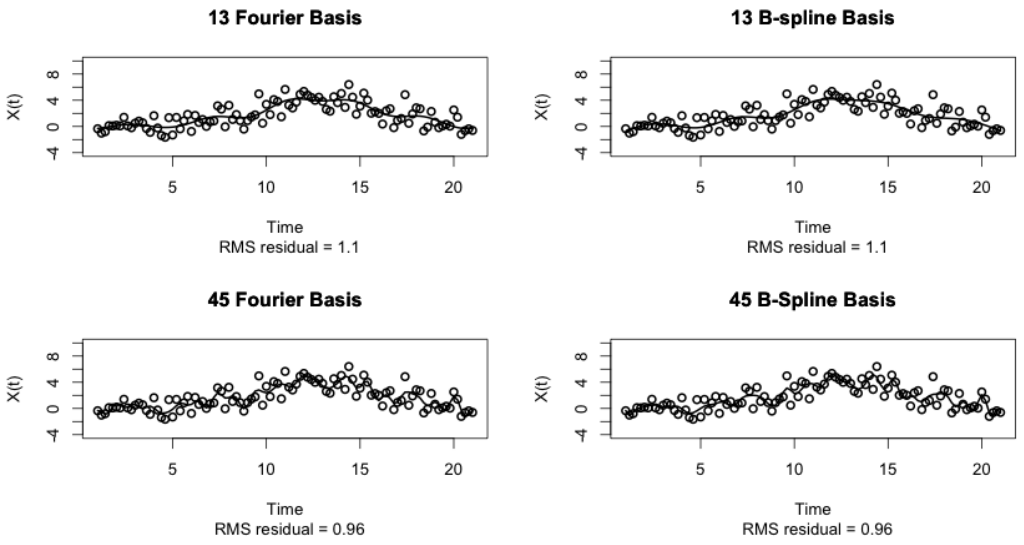

Figure 2.3: Sample Functions

have made a trade-off between the fit of the curve and the smoothness of the curve, and in doing so we have increased the chance of modeling measurement errors.

As a comparison, figure 2.4 shows the functional curves generated using 65 basis functions and a penalty parameter to ensure smoothness. The SSE is slightly reduced for both the Fourier and B-spline basis sets, but the effects of adding additional parameters will need to be considered when we use these functions for logistic regression.

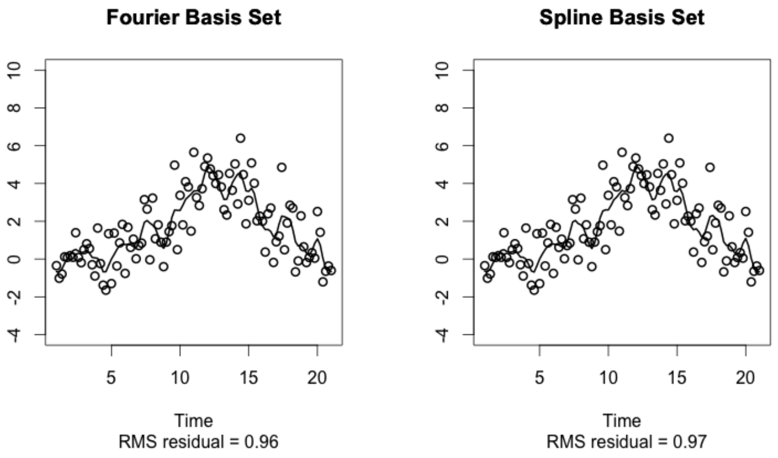

Figure 2.4: Sample Functions Generated with Roughness Penalty

Chapter 3

Logistic Regression

Logistic regression is a method of fitting a regression curve when the response Y takes one of only two possible values, i.e. the presence or absence of a characteristic of interest. In this case, we could define

$$y_i = \begin{cases} 1 & \text{if the } i^{th} \text{ subject has the characteristic,} \\ 0 & \text{if the } i^{th} \text{ subject does not have the characteristic.} \end{cases} \tag{3.1}$$

where y_i is a realization of a random variable Y_i that can take the values 1 and 0 with probabilities π_i and $1 - \pi_i$, respectively.

3.1 Multivariable Predictors

In multivariable linear regression, predictive models take the following form

$$Y_i = \sum_{j=0}^{N} X_{ij}\beta_j + \varepsilon_i, \;\; i = 1, \ldots, n \tag{3.2}$$

where n is the number of subjects and N is the number of predictor variables. Y_i is the response of subject i, X_{ij} is the i^{th} subjects value of predictor j, β_j is the j^{th} regression coefficient. The ε_i term allows for sources of variation considered extraneous, such as measurement error, unimportant additional causal factors, and sources of nonlinearity. These are assumed to be independently and identically distributed.

When Y is a binomial variable, instead of modeling Y_i directly, we want to model the $Pr(Y_i = 1|X_i)$, which will be denoted by π_i. Substituting π_i into the linear regression equation above gives the following:

$$\pi_i = \textstyle\sum_{j=0}^{p} X_{ij}\beta_j + \varepsilon_i \approx \sum_{j=0}^{p} X_{ij}\beta_j = \beta_0 + X\beta \qquad (3.3)$$

However, π_i only takes values on the interval $[0, 1]$ and linear functions are unbounded. To deal with this problem, we use the logit transform of π_i, i.e. $logit(\pi_i) = log(\pi_i/1 - \pi_i)$, as the response. This gives us the logistic regression model:

$$logit(\pi_i) = log\tfrac{\pi_i}{1-\pi_i} = \beta_0 + X\beta \qquad (3.4)$$

After logit regressing the π_i on the covariates X, we will reverse the transformation to obtain

$$\pi_i = \tfrac{exp^{\beta_0 + X\beta}}{1+exp^{\beta_0 + X\beta}} \qquad (3.5)$$

Finally, classification of subjects into groups is made using the following simple rule: when $\pi_i \geq 0.5$, we predict that $Y_i = 1$ and when $\pi_i < 0.5$, we predict that $Y_i = 0$.

3.2 Functional Predictor

Functional logistic regression is used when the response is a binary variable and at least one of the covariates is functional. In this case, we consider a functional analogue of multivariable logistic regression analysis and replace the observed scalar covariates with functional covariates. This also means that the parameter β will be a continuous function.

The conditional probability of the response Y_i given the functional covariate $X_i(t)$ is Bernoulli(π_i), giving us the following functional form of the logistic regression model:

$$\pi_i = P[Y_i = 1|X_i(t) = x_i(t)] = \tfrac{exp\{\beta_0 + \int_T x_i(t)\beta(t)dt\}}{1+exp\{\beta_0 + \int_T x_i(t)\beta(t)dt\}}, \quad i = 1, \ldots, n \qquad (3.6)$$

Taking the logit transformation gives the following equivalent expression:

$$l_i = log \tfrac{\pi_i}{1-\pi_i} = \beta_0 + \int_T x_i(t)\beta(t)dt \tag{3.7}$$

This is a functional generalized linear model. The following key assumptions were made to develop the model:

(1) The $x_i(t)'s$ are a random sample of observations of a functional variable $X_i(t)$

(2) X and Y are defined on the same sample space

(3) X is in a separable Hilbert space L^2_T with inner product $<\alpha, \beta>= \int_T \alpha(t)\beta(t)dt$

There are two issues of concern: (1) the functional form of $x_i(t)$ cannot be observed continuously and (2) estimating the parameter function $\beta(t)$ is not possible with a finite number of observations. These issues can be handled if we assume that $x_i(t)$ and $\beta(t)$ can be expressed in terms of the same basis functions that span the space that the sample paths $x_i(t)$ belong.[6] The basic concept is as follows:

(1) Let $\Phi = (\phi_1(t), \ldots, \phi_K(t))'$ be a vector of K basis functions. Then

(a) $x_i(t) = \mathbf{a}_i'\Phi$, where $\mathbf{a}_i = (a_{i1}, \ldots, a_{iK})'$ are the sample path basis coefficients.

(b) $\beta(t) = \beta'\Phi$, where $\beta = (\beta_1, \ldots, \beta_K)'$ are the parameter function basis coefficients.

(2) $l_i = \beta_0 + \int_T \mathbf{a}_i'\Phi\beta'\Phi dt = \beta_0 + \mathbf{a}_i'\psi_{jk}\beta$, where $\psi_{jk} =< \phi_j, \phi_k >$

(3) $L = (l_1, \ldots, l_n)' = \mathbf{1}\beta_0 + A\Psi\beta$ is the multiple logistic regression model where

(a) A is an (nxK) matrix of the sample path basis coefficients

(b) Ψ is an (KxK) matrix of the inner products of the basis functions.

(c) $A\Psi$ is the design matrix.

We can regress the log-odds of the response on the design matrix $A\Psi$ to find $\hat{\beta}$, which is an estimate of the parameter function $\beta(t)$. Then we can convert $\hat{\beta}$ into a function using the same basis functions used to describe X.

Another issue that must be dealt with, when logistic regression is performed, is the high multicollinearity of the covariates. This can cause inaccurate estimation of the parameter function $\beta(t)$. To mitigate the multicollinearity issue, a PLS logit model that uses the logit PLS components of the design matrix as covariates will be used. Details on the computation of PLS components are covered in the next chapter.

Chapter 4

Functional PLS Logistic Regression

Wold [20] introduced a PLS analysis for multivariate linear regression. In the analysis he obtained a set of uncorrelated latent variables, he called factor scores, that takes into account the relationship between the response and the predictor variables. He then fit a regression model using the latent variables as covariates. PLS regression is an alternative to principal component regression (PCR) for dealing with the multicollinearity issue. It is generally believed that PLS is better suited for classification problems than PCR, since the value of the response is considered in extracting PLS components.

Since Wold's introduction of PLS regression, there have been numerous studies that have looked to (1) improve PLS algorithms, (2) adapt PLS linear regression to the logistic regression model, and (3) extend PLS logistic regression to include functional covariates. Of particular interest to this thesis is the work of Escabias et al. [8], in which they propose a functional PLS logit regression model to forecast a binary response variable from a functional predictor.

4.1 Partial Least Squares

Consider the generalized linear model $Y = X\beta + \varepsilon$, where Y is an (n x 1) response vector, X is an (n x N) matrix of predictors, β is an (N x 1) vector of unknown coefficients, and ε is an (n x 1) vector of errors. The errors are assumed to be normally distributed with zero mean and constant variance σ^2.

The least squares solution for the coefficient matrix β is given by

$$\beta = (X'X)^{-1}X'Y \tag{4.1}$$

When there are more variables than observations, $X'X$ is singular. One way to handle this problem is by using a dimension reduction method, such as principal component regression or partial least squares regression (PLS).

In PLS regression, we try to find a set of uncorrelated vectors to simultaneously decompose the predictor matrix X and the response vector Y, with the constraint that these vectors must maximize the covariance between X and Y. These vectors form the columns of what is called the scores matrix T and they span the column space of X. The predictor matrix is decomposed as the product of T and a loading matrix P, such that $X = TP'$. Then the response Y is regressed on just the first few columns of T.

A popular PLS algorithm, SIMPLS by de Jong [7], is given below. Let $S_0 = X'Y$ and s be a non-negative integer less than or equal to the number of subjects n.

```
For $h = 1, \ldots, s$
    $q_h =$ dominant eigenvector of $S'_{h-1}S_{h-1}$
    $w_h = S_{h-1}q_h$
    $t_h = Xw_h$
    $t_h = t_h - \bar{t}_h$
    $w_h = w_h/\|t_h\|$
    $t_h = t_h/\|t_h\|$
    $p_h = X't_h$
    $q_h = Y'q_h$
    $u_h = Yq_h$
    $v_h = p_h$
    if $h > 1$, then
        $v_h = v_h - V(V'p_h)$
        $u_h = u_h - T(T'u_h)$
    end if
    $v_h = v_h/\|v_h\|$
    $S_h = S_{h-1} - v_h(v'_h S_{h-1}$
    $W = [W, w_h]$
        Append $t_h, p_h, q_h, u_h, v_h$ to $T, P, Q, U, V$ respectively
end for
$B = WQ'$
$B_0 = \bar{Y}$
```

4.2 Functional Partial Least Squares

The functional linear regression model to predict a scalar response with a functional predictor assumes that

$$Y_i = \int_T X_i(t)\beta(t)dt + \varepsilon_i, \quad i = 1, \ldots, n \tag{4.2}$$

The data that we observe for the i^{th} subject are $\{Y_i, X_i(t), t \in T\}, i = 1, \ldots, n$. $X(t)$ is a random curve which is observed for each subject and corresponds to a square integrable stochastic process on a real interval T. The response variable Y is a real valued random variable that is defined on the same probability space as $X(t)$. Although it may be continuous or discrete; in this thesis we consider the case were Y is a binomial random variable. $\beta(t)$ is a real-valued function and ε_i is the random error term.

Using the ordinary least squares method provides inconsistent estimates of $\beta(t)$, and thus functional regression on the principal components of $X(t)$ or the partial least squares components of $X(t)$ and Y have been developed as an alternative. As with multivariable PLS, the functional PLS method obtains a set of PLS components, $(t_1, \ldots, t_s)$ using an iterative procedure. The PLS components are linear combinations of the predictor variables that provide maximum covariance with the response. Details of an algorithm to find the functional PLS components for the special case of logistic regression are given in the next few sections.

4.3 Functional PLS Logistic Regression

Assume we have a data set of observations $\{Y_i, x_i(t)), t \in T\}$, where $x_i(t)$ is a random sample of a functional curve $X_i(t)$, each $Y_i \in \{0, 1\}$, and T is the interval of observations.

A primary goal of this thesis is to develop a modified functional PLS logistic regression algorithm that can be used to predict the response Y_i with the following functional logistic regression model:

$$\pi_i = P[Y_i = 1 | X_i(t) = x_i(t)] = \frac{exp\{\beta_0 + \int_T x_i(t)\beta(t)dt\}}{1 + exp\{\beta_0 + \int_T x_i(t)\beta(t)dt\}}, \quad i = 1, \ldots, n \tag{4.3}$$

where β is the parameter function, n is the number of subjects, and π_i is the conditional probability that subject i will have a response $Y_i = 1$, given the functional predictor $X_i(t)$.

Further assume that we have represented the functional predictor as a linear combination of basis functions $\{\phi_1, \ldots, \phi_K\}$ and the parameter function is in the space spanned by the set of basis functions. Then we have the following functional data components:

(1) $\Phi = (\phi_1(t), \ldots, \phi_K(t))'$, the vector of K basis functions

(2) A, the (nxK) matrix of sample path basis coefficients

(3) Ψ, the (KxK) matrix of inner products of the basis functions

Escabias et al. [8] showed that under these conditions, the functional logistic regression model is equivalent to the following multivariable logistic regression model

$$L = \mathbf{1}\beta_0 + A\Psi\beta = \mathbf{1}\beta_0 + H\beta \tag{4.4}$$

where $L = (l_1, \ldots, l_n)$ is an n x 1 vector, with l_i being the log-odds of the response of the i^{th} subject, $\beta = (\beta_1, \ldots, \beta_K)'$ is the vector of parameters for the multivariable logistic model, and H is the design matrix.

Our first goal is to extract PLS components, which we will denote by t_i, from the design matrix H. To do this, we must find uncorrelated linear combinations of H_i using a maximum covariance criteria. The algorithm we will use to extract the PLS components is a modified version of the method proposed by Escabias et al. [8]. Our modifications are based on power of prediction statistics that are discussed in the next section.

4.4 Power of Prediction Statistics

We considered modified functional PLS logistic regression algorithms that use the predictive power of the individual covariates, i.e. the columns of the design matrix H, to build the PLS components, in hopes of improving the correct classification rate.

Mittlböck and Schemper [13] reviewed 12 psuedo-R^2 measures and Menard [11] considered several others. The measures that we assessed in this thesis are McFadden's R^2, Efron's R^2, Tjur's coefficient of discrimination, and area under the receiver operator characteristic (ROC) curve.

McFadden's R^2 uses the ratio of log-likelihoods to show the level of improvement over the intercept-only model provided by the predictor. It is given by

$$R^2 = 1 - \frac{ln\hat{L}(M_{full})}{ln\hat{L}(M_{int})} \tag{4.5}$$

where M_{full} is the model with the predictor and M_{int} is the model without the predictor. If the full model has no predictive information about the response, the likelihood value will only be slightly larger than the likelihood of the intercept only model. Thus the ratio of the two log-likelihoods will be close to 1, and R^2 will be close to 0. If the full model explains nearly all of the variation in the response, then the likelihood value for the full model will be close to 1. In this case, the log-likelihood of the full model will be close to 0 and R^2 will be close to 1.

For Efron's R^2, the model residuals are squared, summed, and divided by the total variability in the response. This is equivalent to the squared correlation between the predicted values and the actual values. It is given by

$$R^2 = 1 - \frac{\sum(y_i - \pi_i)^2}{\sum(y_i - \hat{y})^2} \tag{4.6}$$

where π_i is the model predicted probability.

Tjur's coefficient of discrimination measures the difference between the expectations of the distributions of successes and failures, that is

$$D = \bar{\pi}_1 - \bar{\pi}_0 \tag{4.7}$$

where $\bar{\pi}_1$ and $\bar{\pi}_0$ denote the averages of fitted values for successes and failures, respectively.

The ROC curve is a plot of the true positive rate versus the false positive rate for the different possible cut-points of a diagnostic test. It shows the tradeoff between sensitivity and specificity and the closer the ROC curve comes to the diagonal of the ROC space, the less accurate the test. Figure 4.1 is an example ROC curve.

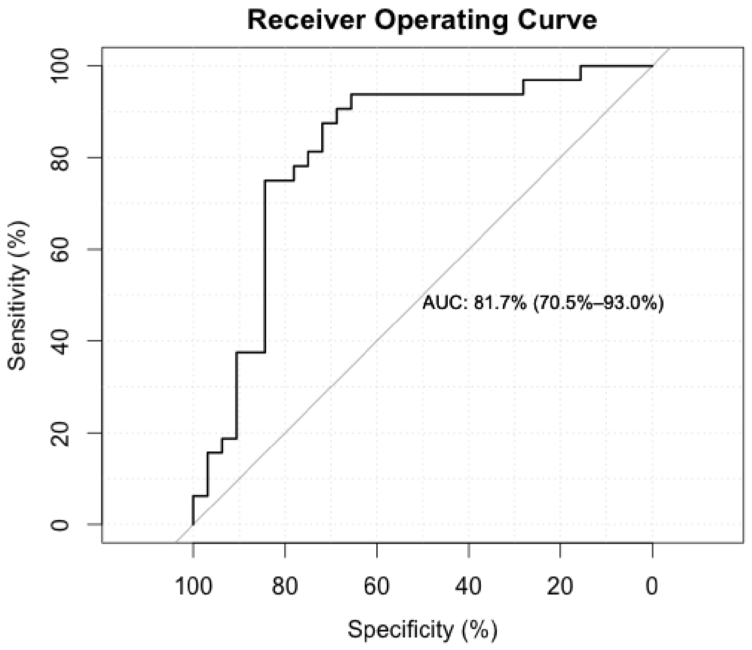

Figure 4.1: Receiver Operator Characteristic Curve

The area under the ROC curve (AUC) may be broadly interpreted as follows:

$$\begin{aligned} &AUC = 0.5 && \text{useless} \\ &0.7 \leq AUC < 0.8 && \text{acceptable} \\ &0.8 \leq AUC < 0.9 && \text{excellent} \\ &AUC \geq 0.9 && \text{outstanding} \end{aligned} \tag{4.8}$$

We chose to incorporate predictive power measures in our algorithm instead of goodness of fit measures because our goal was to improve correct classification rate. After extensive analysis, we found that all four power of prediction measure provided similar results in terms of the classification error rates. Therefore we will only report the results obtained using the AUC metric when we discuss the case studies in chapter 5.

4.5 PLS Components Extraction

As is the case for all PLS algorithms, the goal of component extraction is to find a set of uncorrelated vectors that are linear combinations of the predictive variables $X(t)$ and maximize the covariance between the predictor and the response. This is done iteratively.

Extraction of 1^{st} PLS component, t_1:

(1) Logit regress Y on each H_j, $j = 1, \ldots, K$

(a) Results of interest are the regression coefficients associated with H_j, denoted by a_{1j}

(b) If a_{1j} is not significant, as determined by the predictive power measure, then set a_{1j} equal to zero. This ensures that only those covariates that predict the response are used to build the PLS component.

(c) Form a (K x 1) vector of the regression coefficients; $a_1 = (a_{11} \ldots a_{1K})'$

(2) Set

$$t_1 = \frac{a_{11}H_1 + \ldots + a_{1K}H_K}{\|a_1\|} = \frac{Ha_1}{\|a_1\|} \tag{4.9}$$

Extraction of k^{th} PLS component, t_k, for $k > 1$:

(1) Logit regress Y on $t_1, \ldots, t_{k-1}$ and each H_j, $j = 1, \ldots, K$

(a) Results of interest are the regression coefficients associated with H_j, denoted by a_{kj}

(b) If a_{kj} is not significant, as determined by the predictive power measure, then set a_{kj} equal to zero.

(c) Form a (K x 1) vector of the regression coefficients; $a_k = (a_{k1} \ldots a_{kK})'$

(2) In order to find a PLS component that is orthogonal to all previous PLS components, linearly regress each significant H_j, identified in step (1), on $t_1, \ldots, t_{k-1}$

(a) Primary result of interest is the (n x 1) vector of residuals, denoted by $r_{k-1,j}$, which will be used to form the PLS component t_k in step (3).

(b) Also of interest are the regression coefficients associated with each t_i, denoted by $p_{ij}^{(k-1)}$. These coefficients will be needed to rewrite the final logit regression equation in terms of the original covariates.

(c) Form an (n x K) matrix of the residuals; $R_{k-1} = (r_{k-1,1}, \ldots, r_{k-1,K})'$

(3) Set

$$t_k = \frac{a_{k1} r_{k-1,1} + \ldots + a_{kK} r_{k-1,K}}{\|a_k\|} = \frac{R_{k-1} a_k}{\|a_k\|} \tag{4.10}$$

Note: Stop calculating PLS components when none of the a_{kj}'s are considered significant when Y logit regressed on $t_1, \ldots, t_{k-1}$ and x_j. This will result in a set of s uncorrelated PLS components, where $s < K$.

4.6 Regressing Response on PLS Components

The next step is to perform logistic regression of the log-odds of the response on the extracted PLS components. The result is a set of regression coefficients, $c_0, c_1, \ldots, c_s$, where $s \leq K$. The regression equation in terms of the PLS components can be written as follows:

$$\hat{L} = c_0 + c_1 t_1 + \ldots + c_s t_s \tag{4.11}$$

If we let $T = [t_1 \ \ t_2 \ldots t_s]$ and $c = (c_1 \ \ c_2 \ldots c_s)'$, then we can write the equation in matrix form

$$\hat{L} = \mathbf{1}c_0 + \mathbf{Tc} \tag{4.12}$$

4.7 Regression Coefficients in Terms of Original Variables

Equation 4.12 gives us the log-odds in terms of the PLS components. For clearer interpretability of the results and to allow prediction of class for similar data sets, we need to rewrite the regression coefficients in terms of the original variables. Recall that

$$t_1 = \frac{a_{11}H_1 + \ldots + a_{1p}H_K}{\|a_1\|} = \frac{Ha_1}{\|a_1\|} \tag{4.13}$$

is already written in terms of the original variables. For $k > 1$

$$t_k = \frac{a_{k1}r_{k-1,1} + \ldots + a_{kK}r_{k-1,K}}{\|a_k\|} = \frac{R_{k-1}a_k}{\|a_k\|} \tag{4.14}$$

where a_{kj}'s are the coefficients of H_j found by regressing the response on $t_1, \ldots, t_{k-1}$ and H_j, and $R_{k-1} = (r_{k-1,1}, \ldots, r_{k-1,p})'$ is the matrix of residuals.

We also know from step (2) of the algorithm, that

$$H_j = p_{1j}^{(k-1)}t_1 + p_{2j}^{(k-1)}t_2 + \ldots + p_{(k-1,j)}^{k-1}t_{k-1} + r_{k-1,j} \tag{4.15}$$

which can be rewritten as

$$r_{k-1,j} = H_j - p_{1j}^{(k-1)}t_1 - p_{2j}^{(k-1)}t_2 - \ldots - p_{(k-1,j)}^{k-1}t_{k-1} \tag{4.16}$$

This expression can be substituted into equation 4.14 to give the PLS components in terms of the original variables.

The result is a matrix of parameter function coefficients, denoted by $\hat{\beta}$. Finally, using the same basis functions that were used to convert the discrete observations into functional covariates $X(t)$, we can convert the $\hat{\beta}$'s into a parameter function, $\beta(t)$.

Chapter 5

Case Studies

5.1 Case Study 1: Two Class Curve Simulation

In this example, the simulated curves discussed in chapter 2 are used. Recall that we have 1000 curves from two different classes and each class has 500 curves simulated using the following functions:

$$x_1(t) = uh_1(t) + (1-u)h_2(t) + \varepsilon(t) \tag{5.1}$$

$$x_2(t) = uh_1(t) + (1-u)h_3(t) + \varepsilon(t) \tag{5.2}$$

where u and $\varepsilon(t)$ are simulated uniform and standard normal random variables, respectively, and

$$h_1(t) = max[6 - |t-1|, 0], \;\; h_2(t) = h_1(t-4), \;\; h_3(t) = h_1(t+4) \tag{5.3}$$

The sample curves were simulated at 101 equally spaced points on the interval $[1, 21]$. The response was a binary variable with $Y = 0$ for the first curve and $Y = 1$ for the second curve. The simulated curves are shown in figure 5.1. The simulated curves were randomly divided into a training set containing 800 curves (400 from each class) and a test set containing 200 curves.

In order to test the mfPLSLR algorithm, smooth curves through the discrete observations were generated using a set of 25 B-spline basis functions with knots equally spaced on the interval $[1, 21]$. Area under the ROC curve (AUC) was used as a criterion during extraction of the functional logit PLS components for the training data set. We set the coefficient of a predictor variable to zero if $AUC < 70$ in the PLS extraction step of our

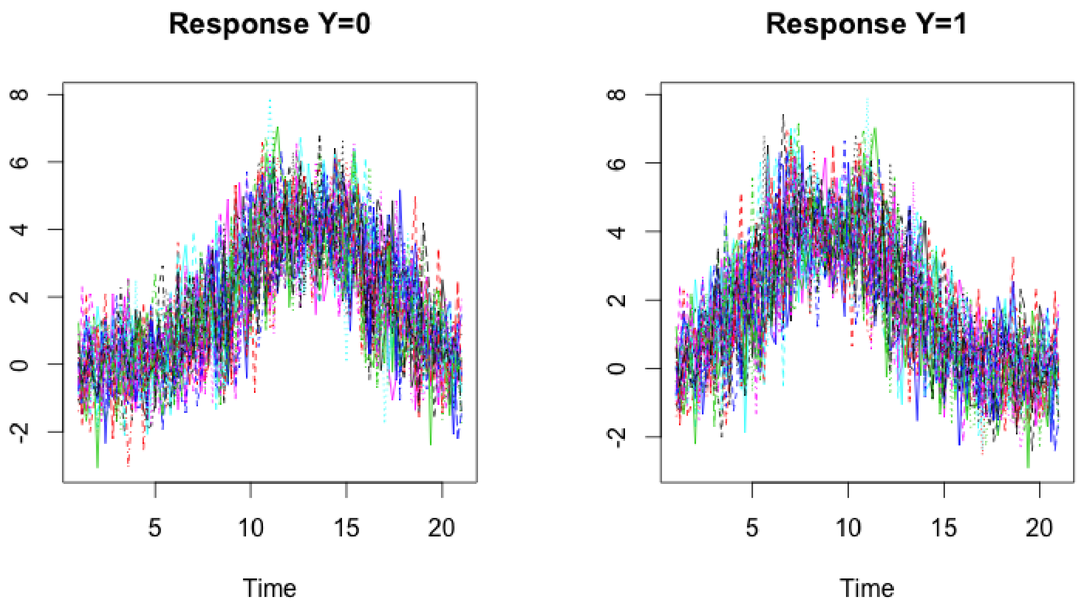

Figure 5.1: Simulated Curves

algorithm, thus ensuring that PLS components were linear combinations of only predictors with high predictive power. The extracted PLS components were used to find the logistic regression parameter function, $\beta(t)$, and the intercept. The parameter function is shown in figure 5.2.

Then smooth curves were generated for the test data and these were used with the intercept and $\beta(t)$ to predict the class of each observation in the test data set. This procedure was repeated 100 times and the mean error rate and standard deviation were calculated. Results for our modified functional PLS algorithm (mfPLSLR) are shown in table 5.1. Also included in the table for comparison purposes, are results obtained using an existing functional partial least squares logit regression algorithm (fPLSLR), a functional principal component logit regression algorithm (fPCLR), and a multivariable PLS logit regression algorithm (PLSLR).

The classification error rates are comparable for all of these methods, with the fPCLR performing the best. We observed a slight improvement with our mfPLSLR algorithm when compared to an existing fPLSLR algorithm.

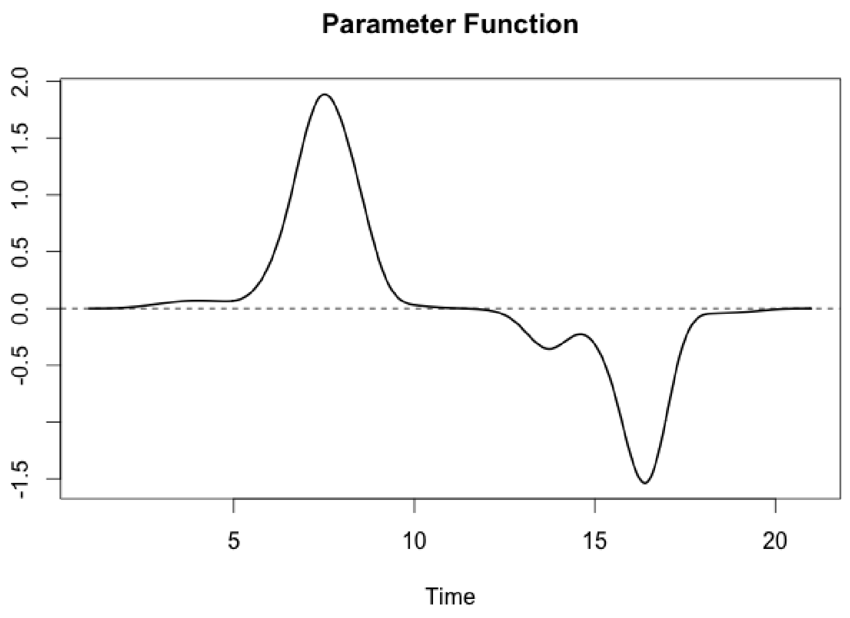

Figure 5.2: Logistic Regression Parameter Function

Method	Error Rate	Standard Deviation
mfPLSLR	0.025	0.083
fPLSLR	0.032	0.083
fPCLR	0.020	0.098
PLSLR	0.045	0.071

Table 5.1: Simulated Curves: Classification Error Rates

5.2 Case Study 2: Diffusion Tensor Imaging

Diffusion Tensor Imaging (DTI) of the brain is a method for characterizing microstructure changes or differences with neuropathology (Alexander et al. [2]). One common DTI measure is fractional anisotropy (FA), which indicates the degree of unequal diffusion properties along different axis. FA is a scalar value between zero and one, with a value of one meaning that diffusion occurs only along one axis and is fully restricted along all other directions. An FA value of zero means that diffusion is isotropic.

5.2.1 Analysis of DTI - FA Data

The data set used for this example contains FA tract profiles from DTI data collected at Johns Hopkins University and the Kennedy-Krieger Institute. The data set includes 382 DTI scans from a total of 142 subjects: 100 with multiple sclerosis (MS) and 42 healthy subjects. The FA tract profiles, measured along the corpus callosum, were sampled on a grid of 93 positions and they form relatively smooth functions. In addition to the tract profiles, each subjects MS status (coded 0 for healthy and 1 for MS) is included in the data. Figure 5.3 shows the FA tract profiles for the two groups. The black line is the mean curve for each group.

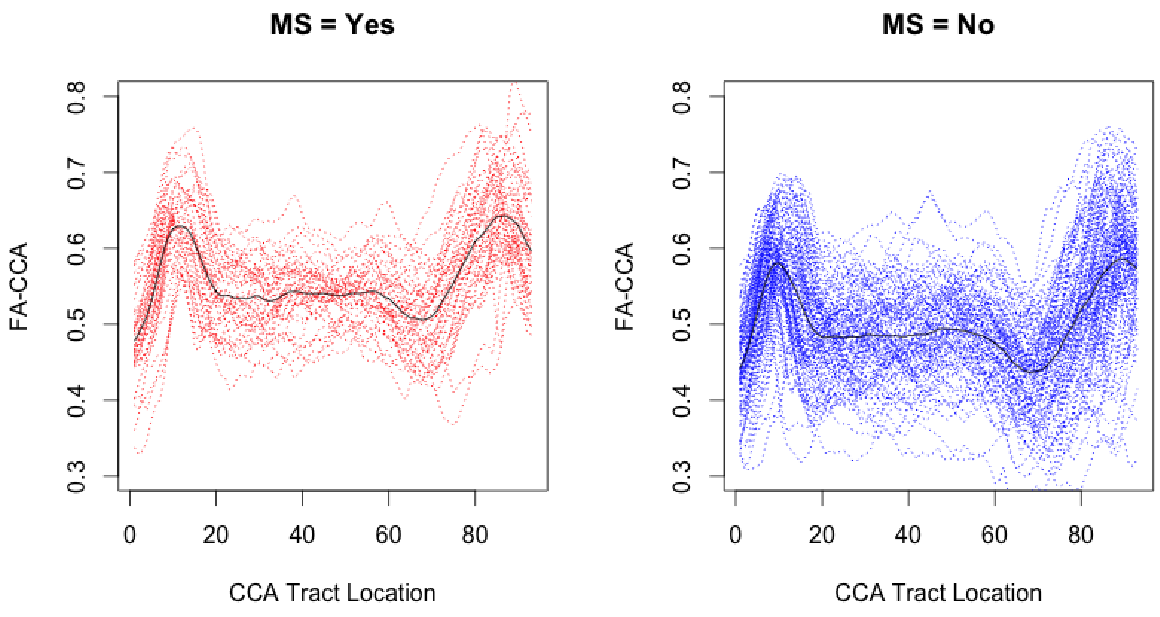

Figure 5.3: FA Tracts

Several of the subjects had multiple DTI scans during the study and there were more MS patients that healthy controls in the data set. In order to obtain a balanced design for our analysis, a random sample of 42 of the MS subjects were selected and only the first DTI scan from each subject was used. The resulting data set contained 84 DTI scans with each subjects MS status.

The goal was to properly classify each subject as either an MS subject or a healthy control using a functional form of the FA tract profiles as the predictor. In our analysis we assessed not only the effectiveness of our mfPLSLR algorithm; but also the effect that basis type, number of basis functions used, and the use of curve smoothing have on classification error rates. The results are shown in tables 5.2 and 5.3.

Method	Basis Functions	Smoothing	Error Rate	Standard Deviation
mfPLSLR	15	None	0.238	0.083
mfPLSLR	45	None	0.249	0.076
mfPLSLR	63	Roughness Penalty	0.239	0.083
fPLSLR	15	None	0.243	0.097
fPLSLR	45	None	0.237	0.082
fPLSLR	63	Roughness Penalty	0.248	0.068
fPCLR	15	None	0.265	0.076
fPCLR	45	None	0.280	0.081
fPCLR	63	Roughness Penalty	0.279	0.087

Table 5.2: DTI-FA: Classification Error Rates Using B-spline Basis Functions

For this data set, the lowest error rates were obtained when using B-splines for the basis functions and both fPLSLR models performed better than the fPCLR model. However, the differences in the error rates between our mfPLSLR algorithm and an existing fPLSLR algorithm were not statistically significant. It was also noted that neither the number of basis functions nor the implementation of smoothing via a roughness penalty had a significant effect on error rates.

Method	Basis Functions	Smoothing	Error Rate	Standard Deviation
mfPLSLR	15	None	0.276	0.073
mfPLSLR	45	None	0.277	0.075
mfPLSLR	63	Roughness Penalty	0.277	0.084
fPLSLR	15	None	0.283	0.075
fPLSLR	45	None	0.284	0.082
fPLSLR	63	Roughness Penalty	0.288	0.086
fPCLR	15	None	0.284	0.084
fPCLR	45	None	0.268	0.080
fPCLR	63	Roughness Penalty	0.269	0.087

Table 5.3: DTI-FA: Classification Error Rates Using Fourier Basis Functions

Chapter 6

Conclusions and Recommendations

In this thesis we have considered several modifications to the functional PLS logistic regression model in an attempt to decrease the error rate. We found that using power of prediction measures, such as McFadden's R^2 or AUC, as criteria in extracting the PLS components, reduced the number of components needed in the final model. However, the classification error rate for our modified fPLSLR model was not significantly different from the error rate obtained using existing fPLSR models.

Also of note is the fact that the various power of prediction measures all gave similar results. We found that the optimum cutoff criteria needed for each criteria were slightly different, but once the criteria were optimized for the each metric, the prediction error rates were similar.

Another goal was to analyze the effect that different types and numbers of basis functions used for curve generation would have on the classification error rate. We found that the use of B-spline basis functions yielded a slightly lower classification error rate for our fPLSLR algorithm, when compared to Fourier basis functions. The number of basis functions used and the use of a roughness penalty in creating the functional predictors did not have a significant effect on the classification error rate.

The data sets analyzed in this thesis are considered relatively smooth and do not contain outliers. Since our algorithm could be considered a form of variable selection, it would be interesting to see if outlying observations would be filtered out in the PLS component extraction process.

Bibliography

[1] Aguilera, A.M. and Escabias, M., "Principal Component Logistic Regression", Proceedings in Computational Statistics, 14th Symposium, 2000.

[2] Alexander, A.L., Lee, J.E., Lazar, M. and Field, A.S., "Diffusion Tensor Imaging of the Brain", Neurotherapeutics, 2007 July; 4(3): 316-329.

[3] Bastien, P., Vinzi, V. E. and Tenenhaus, M., "PLS generalised linear regression", Computational Statistics & Data Analysis, Volume 48, Pages 17 - 46, 2005.

[4] Cardot, H. and Sarda, P., "Estimation in Generalized Linear Model for Functional Data via Penalized Likelihood", Journal of Multivariate Analysis, 92, 24?41, 2005.

[5] Cox, D.R. and Snell, E.J., "The Analysis of Binary Data", 2nd ed. Chapman and Hall, London, 1989.

[6] Dardis, C. (2015). LogisticDx: Diagnostic Tests for Models with a Binomial Response. R package version 0.2. http://CRAN.R-project.org/package=LogisticDx

[7] de Jong, S., "SIMPLS, an Alternative Approach to Partial Least Squares Regression", Chemometrics and Intelligent Laboratory Systems 18(3): 251-263, February 1993.

[8] Escabias, M., Aguilera, A.M. and Valderrama, M.J., "Functional PLS logit regression model", Computational Statistics & Data Analysis, Volume 51, Pages 4891 - 4902, 2007.

[9] James, G. "Generalized linear models with functional predictors. Journal of the Royal Statistical Society", Series B 64, 411-432, 2002.

[10] McFadden, D., "Conditional Logit Analysis of Qualitative Choice Behavior", Zarembka P. (ed.) Frontiers in Economics. Academic Press, New York, 1974.

[11] Menard, S., "Coefficients of determination for multiple logistic regression analysis", The American Statistician 54: 17-24, 2000.

[12] Mevik, B., Wehrens, R. and Liland, K.H., (2015). pls: Partial Least Squares and Principal Component Regression. R package version 2.5-0. http://CRAN.R-project.org/package=pls

[13] Mittlböck, M. and Schemper, M. "Explained variation in logistic regression", Statistics in Medicine 15: 1996.

[14] Müller, H., Stadtmüller, U., "Generalized functional linear models", The Annals of Statistics, 33(2), 774-805, 2005.

[15] Preda, C. and Saporta, G., "PLS regression on a stochastic process", Computational Statistics Data Analaysis. 48 149-158, 2005.

[16] Ramsay, J.O. and Silverman, B.W., "Functional Data Analysis", 2nd ed., Springer-Verlag, New York, 2005.

[17] Ramsay, J.O., Hooker, G. and Graves, S., "Functional Data Analysis with R and MATLAB", Springer Science + Business Media, New York, 2009.

[18] Ramsay, J.O., Wickham, H., Graves, S. and Hooker, G. (2014). fda: Functional Data Analysis. R package version 2.4.4. http://CRAN.R-project.org/package=fda

[19] Tjur T., "Coefficients of Determination in Logistic Regression Models - A New Proposal: The Coefficient of Discrimination", The American Statistician, 63: 366-372, 2009.

[20] Wold, H., "Soft modelling by latent variables: The non-linear iterative partial least squares (NIPALS) approach", Perspectives in Probability and Statistics, Papers in Honour of M. S. Bartlett (J. Gani, ed.). Academic Press, London, 1975.

Appendix

```
# Classification using a functional PLS logistic regression algorithm

# Fractional anisotropy (FA) tract profiles from diffusion tensor imaging (DTI)
# Data collected at JHU and the Kennedy-Krieger Institute
# 382 DTI scans from 142 patients:
#    100 with multiple sclerosis (MS)
#    42 healthy controls
# FA profiles obtained along the corpus callosum (CCA)

library(fda)
library(pls)
library(LogisticDx)
library(pscl)
library(binomTools)

CCA <- read.table("~/Desktop/AUMasters/DTI/FAcca.txt", header=TRUE,sep="\t")
CCA <- t(t(CCA))
CCA <- na.omit(CCA)
index <- seq(1,93)
MS1 <- matrix(1,nrow=42)
MS0 <- matrix(0,nrow=99)
MS <- rbind(MS1,MS0)
profiles <- cbind(CCA,MS)

#par(mfrow=c(1,2))
#matplot(index,t(CCA[1:42,1:93]),type='l',main='MS = Yes',
#        xlab='CCA Tract Location',ylab='FA-CCA',col='red')
#matplot(index,t(CCA[43:141,1:93]),type='l',main='MS = No',
#        xlab='CCA Tract Location',ylab='FA-CCA',col='blue')
#par(mfrow=c(1,1))


X <- CCA[1:84,]
Y <- factor(MS[1:84])
nvar <- ncol(X)
X1 <- X[1:42,]
X0 <- X[43:84,]
```

```
Y1 <- Y[1:42]
Y0 <- Y[43:84]
nobs <- nrow(X)
obs <- nrow(X1)
########################################################################
Error <- NULL
Betacoefs <- NULL
# Calculate sampling distribution of CCR and Error rate
for (s in 1:100) {                                    # set number of sampling runs
Prob <- NULL
#Split data into training and test data
# 64 observations randomly assigned to training set
train <- sample(1:obs,32)
X0train <- NULL
X0test <- NULL
X1train <- NULL
X1test <- NULL
for(i in 1:obs) {
  if(i %in% train) {
    X0jl <- NULL
    X1jl <- NULL
    for(l in 1:nvar) {
      X0jl <- cbind(X0jl,X0[i,l])
      X1jl <- cbind(X1jl,X1[i,l])
    }
    X0train <- rbind(X0train,X0jl)
    X1train <- rbind(X1train,X1jl)
  }
  else{
    X0jl <- NULL
    X1jl <- NULL
    for(l in 1:nvar) {
      X0jl <- cbind(X0jl,X0[i,l])
      X1jl <- cbind(X1jl,X1[i,l])
    }
    X0test <- rbind(X0test,X0jl)
    X1test <- rbind(X1test,X1jl)
  }
}
Xtrain <- rbind(X0train,X1train)
Xtest <- rbind(X0test,X1test)
train <- rbind(cbind(X0train,0),cbind(X1train,1))
test <- rbind(cbind(X0test,0),cbind(X1test,1))
Ytrain <- factor(train[,94])
Ytest <- factor(test[,94])
```

```
################################################################################
# Generate functional data curves
# Each observation curve will be a linear combination of the basis functions
rangeval <- c(1,93)

# Use a roughness penalty
norder <- 4
deriv <- 2
lambda <- .01
nbasis <- length(index) + norder - 2
evalbasis = create.bspline.basis(rangeval, nbasis, norder, index)  # use B-spline basis
#evalbasis = create.fourier.basis(rangeval, nbasis, 93)            # use Fourier basis
basismat <- eval.basis(index,evalbasis)
curvefdPar = fdPar(evalbasis, deriv, lambda)
curvefd = smooth.basis(index,t(Xtrain),curvefdPar)$fd
#plotfit.fd(t(Xtrain)[,1], index, curvefd[1],lty=1, lwd=2,
#            main="Penalty B-Spline Basis",xlab='Time',ylab='X(t)')

A <- t(curvefd$coefs)         # sample path basis coefficients
PSI <- crossprod(basismat)    # coefficients of basis functions at knots
H <- A %*% PSI                # Design matrix
I <- diag(nbasis)
#########################################################################
# Compute the modified functional PLS logit regression (mFPLSLR) model
# Computation of a set of PLS components

# Step 1: First logit PLS component
delta <- NULL
gofp <- NULL
auc <- NULL
T <- NULL
Ti <- NULL
Tj <- NULL
z <- 0.01
for (j in 1:nbasis) {
  logit.fit <- glm(Ytrain ~ H[,j],family=binomial)
  # aucj <- gof(logit.fit,plotROC=TRUE)$auc[1]          # calculate AUC
  # auc <- rbind(auc,aucj)
  PseudoR2c <- Rsq(logit.fit)
  Tjurj <- PseudoR2c[8]                                 # calculate Tjur's coefficient
  Tjur <- rbind(Tjur,Tjurj)
  deltaj <- logit.fit$coefficients[2]
  sedeltaj <- summary(logit.fit)$coefficients[2,2]
  pvaluej <- summary(logit.fit)$coefficients[2,4]
  # if(pvaluej > z) deltaj <- 0                         # use pvalue criteria
  # if(aucj < 70) deltaj <- 0                           # use AUC criteria
  if(Tjurj < .2) deltaj <- 0                           # use Tjur criteria
  delta <- rbind(delta,deltaj)
}
V1 <- delta/norm(delta,type='F')
V <- V1
T1 <- H %*% V1
################################################################################
# 2nd logit PLS component
delta <- NULL
R <- NULL
P1 <- NULL
for (j in 1:nbasis) {
  logit.fit2 <- glm(Ytrain ~ H[,j] + T1,family=binomial)
  # aucj <- gof(logit.fit2,plotROC=TRUE)$auc[1]          # calculate AUC
  # auc <- rbind(auc,aucj)
  PseudoR2c <- Rsq(logit.fit2)
  Tjurj <- PseudoR2c[8]                                 # calculate Tjur's coefficient
  Tjur <- rbind(Tjur,Tjurj)
  deltaj <- logit.fit$coefficients[2]
  sedeltaj <- summary(logit.fit)$coefficients[2,2]
  pvaluej <- summary(logit.fit)$coefficients[2,4]
  deltaj <- logit.fit2$coefficients[2]
  sej <- sqrt(diag(vcov(logit.fit2)))
  sedeltaj <- sej[2]
  pvaluej <- summary(logit.fit2)$coefficients[2,4]
  # if(pvaluej > z) deltaj <- 0                         # use pvalue criteria
  # if(aucj < 70) deltaj <- 0                           # use AUC criteria
  if(Tjurj < .2) deltaj <- 0                           # use Tjur criteria
  delta <- rbind(delta,deltaj)
}
V2 <- delta/norm(delta,type='F')
V <- cbind(V,V2)
for (j in 1:nbasis) {
  linreg.fit <- lm(H[,j] ~ T1 - 1)
  Rj <- t(t(linreg.fit$residuals))
  p1j <- linreg.fit$coefficients[1]
  R <- cbind(R,Rj)
  P1 <- rbind(P1,p1j)
}
D1 <- I - V1 %*% t(P1)
T2 <- H %*% D1 %*% V2
T <- cbind(T1,T2)
```

```
####################################################################
# 3rd logit PLS component
delta <- NULL
R <- NULL
V3 <- NULL
P21 <- NULL
P22 <- NULL
for (j in 1:nbasis) {
  logit.fit3 <- glm(Ytrain ~ H[,j] + T1 + T2,family=binomial)
  #  aucj <- gof(logit.fit3,plotROC=FALSE)$auc[1]
  #  auc <- rbind(auc,aucj)
  PseudoR2c <- Rsq(logit.fit3)
  Tjurj <- PseudoR2c[8]                                 # calculate Tjur's coefficient
  Tjur <- rbind(Tjur,Tjurj)
  deltaj <- logit.fit3$coefficients[2]
  sej <- sqrt(diag(vcov(logit.fit3)))
  sedeltaj <- sej[2]
  pvaluej <- summary(logit.fit3)$coefficients[2,4]
  # if(pvaluej > z) deltaj <- 0                         # use pvalue criteria
  # if(aucj < 70) deltaj <- 0                           # use AUC criteria
  if(Tjurj < .2) deltaj <- 0                          # use Tjur criteria
  delta <- rbind(delta,deltaj)
}
V3 <- delta/norm(delta,type='F')
V <- cbind(V,V3)
for (j in 1:nbasis) {
  linreg.fit <- lm(H[,j] ~ T1 + T2 - 1)
  Rj <- t(t(linreg.fit$residuals))
  p21j <- linreg.fit$coefficients[1]
  p22j <- linreg.fit$coefficients[2]
  R <- cbind(R,Rj)
  P21 <- rbind(P21,p21j)
  P22 <- rbind(P22,p22j)
}
D21 <- I - V1 %*% t(P21)
D22 <- V2 %*% t(P22)
T3 <- H %*% (D21 - D1 %*% D22) %*% V3
T <- cbind(T,T3)

####################################################################
# Logit regression fitting of the response on the PLS components
logit.fit21 <- glm(Ytrain ~ T1,family=binomial)
int <- logit.fit21$coefficients[1]
t1coef <- logit.fit21$coefficients[2]
#t2coef <- logit.fit21$coefficients[3]
```

```
#t3coef <- logit.fit21$coefficients[4]
pred <- round(logit.fit21$fitted.values,digits=2)
pvalueT1 <- summary(logit.fit21)$coefficients[2,4]
pvalueT <- rbind(pvalueT,pvalueT1)

# Find logit regression coefficients in terms of orginal variables
 # Calculation of values of Xcoef depend on the number of PLS components used
Xcoef <- (t1coef*V1)
#Xcoef <- (t1coef*V1 + t2coef*D1 %*% V2)
#Xcoef <- (t1coef*V1 + t2coef*D1 %*% V2 + t3coef * (D21 - D1 %*% D22) %*% V3)

# Calculate parameter function beta(t)
betahat <- Xcoef
beta <- fd(betahat, evalbasis)
betacoefsj <- matrix(beta$coefs,nrow=1)
Betacoefs <- rbind(Betacoefs,betacoefsj)
plot(beta, lty=1, lwd=2, col=1,main="Parameter Function",xlab='CCA Tract Location',ylab=

# Predict response for Xtest
curvefdtest = smooth.basis(index,t(Xtest),curvefdPar)$fd  # Use with roughness penalty
#plotfit.fd(t(Xtest)[,1], index, curvefdtest[1],lty=1, lwd=2,
#           main="Penalty B-Spline Basis",xlab='Time',ylab='X(t)')

Atest <- t(curvefdtest$coefs)
Htest <- Atest %*% PSI
one <- matrix(1,nrow=nrow(Htest))
Lhat <- int*one + Htest %*% betahat
Prob1 <- 1/(1 + exp(Lhat))
Prob2 <- exp(Lhat)/(1 + exp(Lhat))

# Check classification accuracy
Ypredict <- factor(round(Prob2,digits=0))
CC <- 0
for (i in 1:length(Ytest)) {
  if(Ytest[i] == Ypredict[i]) CC <- CC + 1
}
CCR <- CC/length(Ytest)
ER <- 1 - CCR
Error <- rbind(Error,ER)
}
####################################################################
# Examine sampling distribution of Error
Errorrate <- mean(Error)
Errorsd <- sd(Error)
hist(Error)
```