Group Lasso for Functional Logistic Regression

by

Jessica Godwin

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
August 3, 2013

Keywords: functional data analysis, group lasso, functional logistic regression, fMRI

Abstract

Functional datasets are comprised of data that have been sampled discretely over a continuum, usually time. While the recorded data are discrete, it is assumed that there is a smooth, underlying curve describing the observations. In this thesis an attempt is made to develop a variable selection technique for the functional logistic regression model.

The functional logistic regression model is the functional analog of logistic regression. In this model, the responses are binary and represent two separate classes; the predictors are functional. Due to the nature of functional data, observations at neighboring time points are correlated, leading to redundant information within each observation and each functional predictor. In a dataset with many variables, it is necessary to be able to select a smaller subset of informative variables. In this thesis, we attempt to remove the autocorrelation between neighboring observations and perform variable selection. We do this by employing a principal component analysis on binary data with multiple functional predictors. The data are then subject to a variant of the group lasso, an L_1 regularization method that estimates the logistic model and selects variables simultaneously. We assess our method with a simulation study and an application to a real dataset.

Acknowledgments

I would like to thank my advisor, Dr. Nedret Billor. You have been an incredible mentor to me. From the first semester you taught me, you have encouraged me in the direction of research. Thank you to my friends and family for all of your support. To my parents, my deepest gratitude for supporting me throughout my life and teaching me to work hard. Cory, thank you for all the love, laughter and constant encouragement. Thank you to the rest of my committee, Dr. Dmitry Glotov and Dr. Ash Abebe. Finally, thank you to Dr. Gopikrishna Deshpande and his student, Yun Wang, for sharing your data with me. I could not have done any of this without any of you. War eagle!

Table of Contents

List of Figures

Chapter 1

Introduction

Functional data analysis (FDA) is a relatively new area within the discipline of statistics. The first in depth text on the subject, *Functional Data Analysis*, was published by as recently as 1997 [8]. Functional data are data that have been measured discretely over a continuum, usually time. Instead of treating the many discrete measurements as individual observations, one makes the assumption that these measurements represent a smooth, underlying curve. This curve, then, is considered as one observation.

Much of this work was motivated by Functional Magnetic Resonance Imaging (fMRI), a noninvasive imaging technique which allows an experimenter to take images of a subject's brain over time. These images are taken while the subject performs a task such as finger tapping or correctly identifying images of human faces amidst a series of images containing both human faces and objects. fMRI is currently being used to assess which areas of the brain are activated while performing certain tasks. This is done by dividing the brain into voxels, the three-dimensional analog of a pixel, and measuring brain activation. Depending on how one defines a voxel, a typical fMRI image has over 1,000,000 voxels. As fMRI studies usually have a small number of subjects, this results in datasets that are incredibly high-dimensional. High dimensionality is one of the biggest problems in statistical analysis of fMRI data [10].

Initial statistical analysis of fMRI data was univariate in nature [3]. This is obviously simplistic for data that take into account four dimensions: three spatial dimensions and one temporal dimension. The second decade of fMRI research focused on multivariate data analysis. Considering the fact that, the capturing of images happens over time, the next logical step in this progression is functional data analysis. In an fMRI session, images of a subject's brain are taken at discrete time points. However, one would expect the brain's response to a stimulus to be continuous in nature. This makes fMRI imaging data a great candidate for functional data analysis. In 2005 Viviani et al. published a paper using functional principal component analysis in fMRI. They showed the results to be much more interpretable than multivariate PCA [12]. Since then more statistical analysis of fMRI data has been functional in nature. There is a need to attack the problem of high dimensionality of brain imaging data. There is also a need for the development of better classification methods [10]. One of the best things about Functional Magnetic Resonance Imaging is its noninvasiveness. If statistical classification methods are improved, it could aid the advancement of noninvasive diagnostic techniques for mental illness or even degenerative diseases such as Alzheimer's. There is some research being done in this area, but there is room for more advancement [10].

1.1 Basic Functional Data Analysis

Consider sample curves of the form $\{\mathbf{x_i}(t),\, t \in T,\, i = 1, \ldots, n\}$, where T is an interval over which the observations were measured. The observations belong to the Hilbert space, $L_2(T)$, of square-integrable functions with the inner product

$$\langle f, g \rangle = \int_T fg dt, \qquad \forall f, g \in L_2(T). \tag{1.1}$$

A vector $\mathbf{x_i} = (x_{i1}, \ldots, x_{iN})$, represents the discrete measurements for the i^{th} subject of one variable, x, at N points in T. There are functional analogs of the traditional summary statistics. The functional sample mean, $\bar{\mathbf{x}}(t)$, is defined below:

$$\bar{x}(t) = n^{-1} \sum_{i=1}^{n} x_i(t). \tag{1.2}$$

The sample mean is computed point-wise at $t \in T$. Similarly, one can compute the covariance between measurements at two time points s and t

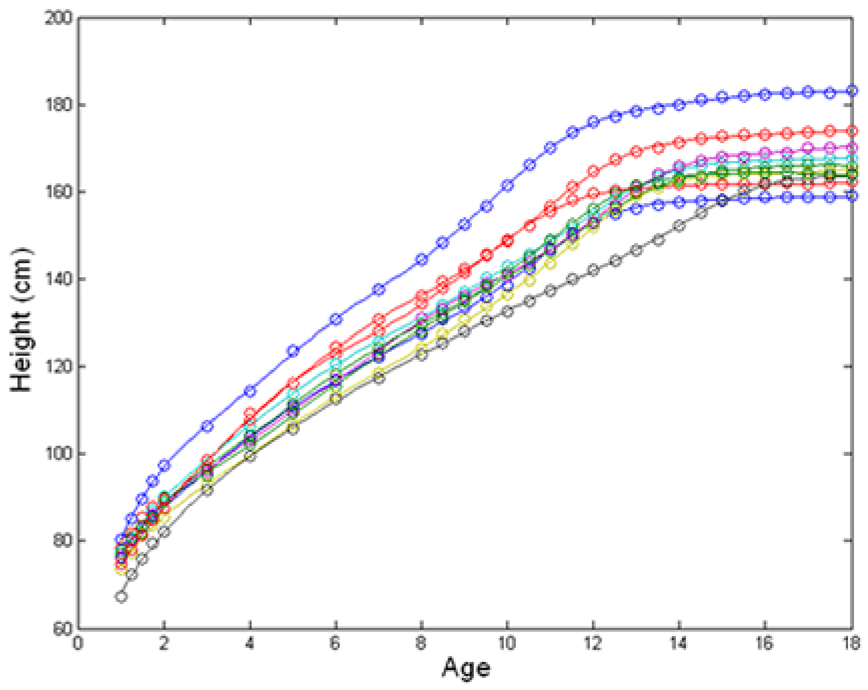


Figure 1.1: Berkeley Growth Study
These are the curves describing the height in cm of girls followed in the Berkeley Growth Study beginning in 1929. The circles mark the discrete height measurements over time.

$$cov(s,t) = (n-1)^{-1}\sum_{i=1}^{n}(x_i(s)-\bar{x}_i(s))(x_i(t)-\bar{x}_i(t)). \tag{1.3}$$

When $s = t$, this is simply the variance of x computed at that time point. If we further assume that the observations, $\mathbf{x}_i(t)$, belong to an L_2 space, we can define the inner product,

$$\langle x, y\rangle = \int x(t)y(t)dt. \tag{1.4}$$

Once we consider the observations as smooth functions, a natural extension would be to estimate the derivatives. In the case of the Human Growth Data in Figure 1.1, one might want to estimate the velocity or acceleration of the growth for inference as well [8].

1.2 Basis Expansion of Functional Data

Let observations $\{\mathbf{x_i}(t),\, t \in T,\, i = 1, \ldots, n\}$ belong a subspace of L_2 spanned by the p-dimensional basis system of independent functions, $\{\phi_1, \ldots, \phi_p\}$. The assumed smooth functional observation, or linear expansion, $x_i(t)$, can be expressed in terms of the sum

$$x_i(t) = \sum_{k=1}^{p} a_k \phi_k. \tag{1.5}$$

Here, each a_k is called a basis coefficient. When derivative estimation is required, one must estimate it separately. Taking the derivate of the smooth function $x_i(t)$ does not often give a good derivative estimate.

There are many different basis systems that can be used in a basis expansion. One of the most common systems is the Fourier basis system. A Fourier basis is defined by a Fourier series,

$$1,\, \sin(\omega t),\, \cos(\omega t),\, \sin(2\omega t),\, \cos(2\omega t),\, \sin(3\omega t),\, \cos(3\omega t),\, \ldots \quad .$$

Fourier bases are useful for data that are periodic, e.g. weather patterns [8]. Additionally, derivative estimation is easier with a Fourier basis as the derivatives of $\sin(t)$ and $\cos(t)$ are known and easy to compute.

For data that are not cyclical in nature, the most common choice is the B-spline basis system. They are computationally efficient and are flexible enough to approximate most non-periodic functions. Splines are defined over an interval T subdivided into subintervals at points called knots. A basis system with L knots contains $L + 1$ subintervals. Over each subinterval, a polynomial functional of order m is defined. These functions are constrained to be equal at the knots. The number of knots, L, and order of the splines, m, are two important parameters defining a B-spline basis system. In this work, only B-spline bases will be used.

After a basis system has been chosen, the basis coefficients must be estimated. One method of basis coefficient estimation is that of least squares estimation. Denote the discrete observations of a functional dataset y_j, $j = 1, \ldots, N$, where N is the number of time points at which observations were taken. Assume independently and identically distributed measurement errors, ε_j, with $\mathrm{E}[\varepsilon_j] = 0$ and $\mathrm{Var}(\varepsilon_j) = \sigma^2$. Then,

$$y_j = x(t_j) + \varepsilon_j. \tag{1.6}$$

Defining $x(t_j)$ in terms of (1.4), the sum of squared errors can be defined as

$$SSE(\mathbf{y}|\mathbf{a}) = \|\mathbf{y} - \mathbf{\Phi a}\|^2. \tag{1.7}$$

Taking the first derivative, the least squares solution minimizing SSE($\mathbf{y}|\mathbf{a}$)

$$\hat{\mathbf{a}} = (\mathbf{\Phi}'\mathbf{\Phi})^{-\mathbf{1}}\mathbf{\Phi}'\mathbf{y}. \tag{1.8}$$

However, least squares smoothing is inappropriate if the error assumptions are not true. In the case of functional data measured over time, observations at adjacent time points are likely correlated, violating the standard error assumptions.

A more common method of spline smoothing is smoothing by roughness penalty. This method is designed to estimate a curve that is rough enough to describe observed features of the data, but suppresses high-frequency features of the data, including noise. To find the coefficients of a smooth approximation of this type, the sum of squared errors is minimized with the added constraint of a roughness penalty. Roughness of a function is described by the curvature, or the squared second derivative. The quantity penalized is the integrated squared second derivative,

$$PEN_2(x) = \int [D^2 x(s)]^2 ds. \tag{1.9}$$

The corresponding sum of squared errors to be minimized is as follows:

$$PENSSE_\lambda(x|\mathbf{y}) = [\mathbf{y} - x(t)]'\mathbf{W}[\mathbf{y} - x(t)] + \lambda PEN_2(x), \tag{1.10}$$

where $\mathbf{W}$ is the matrix of weights describing the covariance structure of the errors. The smoothing parameter λ is chosen by the method of generalized cross validation developed by Craven and Wahba [1]. This is done by choosing λ such that it minimizes the following equation

$$GCV(\lambda) = \frac{n \times SSE}{(n - df(\lambda))^2}. \tag{1.11}$$

Here $df(\lambda) = \text{trace}(S_{\phi,\lambda})$, where

$$S_{\phi,\lambda} = \mathbf{\Phi}(\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi})^{\mathbf{-1}}\mathbf{\Phi}\mathbf{W} \tag{1.12}$$

is the hat matrix of the spline smoother.

Once you have smoothed functional observations, the statistical analysis can begin. As with traditional statistics, the beginning of FDA focused on univariate statistics. Our focus, however, is on a set of multiple functional predictors. In any functional data set with multiple functional predictors, and especially in fMRI, dimensionality is an issue. It may be important, for the sake of interpretation and computational expense, to select a smaller subset of important variables from the dataset. In any dataset, classification is often of interest. This particularly resonates within fMRI. Currently classification is used to explore brain functionality. For example, in a certain study subjects are given one of two stimuli. Does the brain behave differently in the presence of each stimulus to be able to predict which stimulus a subject was presented with? If so, it would be important to identify which areas of the brain are associated with this difference. As classification improves within the realm of fMRI, it could contribute to diagnosis of brain degeneration or mental illness in clinical settings. The rest of this thesis focuses on a method of dimension reduction and variable selection in a dataset with multiple functional predictors and a binary response.

Chapter 2

Functional Principal Component Logistic Regression

2.1 Principal Component Analysis

Traditional principal component analysis (PCA) is a method of data reduction for multivariate datasets [4]. Consider a data matrix $\mathbf{X}_{n \times m}$, where n is the sample size and m is the number of variables. Let $\mathrm{E}[\mathbf{X}] = 0$ and $\mathbf{C} = (\mathbf{X}'\mathbf{X})^{-1}$ be the variance-covariance matrix. PCA is performed on the covariance matrix or correlation matrix of $\mathbf{X}$. It reduces dimension by finding a linear combination of the variables that has the maximum variance; this linear combination is the first principal component (PC). The next PC is found by finding a linear combination of the variables that is independent of the first PC and has the next largest variance. This goes on until $\min\{N-1, m\}$ PCs have been found. This is equivalent to solving the following equation:

$$\mathbf{C}\mathbf{f} = \lambda \mathbf{f}. \tag{2.1}$$

The solutions to this equation are the eigenvalues, λ, and eigenvectors, $\mathbf{f}$, of the covariance or correlation matrix. Dimension reduction occurs when a number of principal components, $s \leq m$, is chosen. Often this is done by examining the amount of cumulative variance explained by each additional principal component.

2.2 Functional Principal Component Analysis

Consider a sample containing observations of one functional predictor. There is correlation between observations at adjacent points in T. In a linear model framework, this leads to the problem of high multicollinearity. Escabias et al. propose a method of functional principal component analysis (fPCA) for the logistic regression model with one functional predictor that alleviates this issue [6]. We will extend this method to a functional logistic regression model with multiple functional predictors.

As outlined by Ramsay and Silverman, fPCA is merely a functional analog of the traditional multivariate principal component analysis [8]. Assume functional observations of one variable $x_i(t) \in L_2$, where $i = 1, \ldots, n$, with the usual functional sample mean, $\bar{x}(t)$, and sample covariance function, $\hat{C}(s,t)$, $s, t \in T$. Without loss of generality, assume $\bar{x}(t)$=0. The functional principal components, ξ_j, are found by solving the following functional equivalent of (2.1)

$$\int_T \hat{C}(s,t) f(s) ds = \lambda f(t). \tag{2.2}$$

The solutions to (2.2) are the eigenvalues, λ, and eigenfunctions, $f(t)$, of the the covariance matrix $\mathbf{C}$. The number of eigenvalues is $N-1$. The i^{th} component of the j^{th} principal component, ξ_{ij}, is expressed as

$$\xi_{ij} = \int_T x_i(t) f_j(t) dt. \tag{2.3}$$

The solution of (2.2) cannot always be computed.

When the n sample functions of a functional predictor belong to the space $L_2(T)$ spanned by orthonormal bases $\{\phi_1, \ldots, \phi_p\}$, the functional PCs are equivalent to the multivariate PCs of the matrix $\mathbf{A\Psi}$ [2]. Here, $\mathbf{A}$ is the $n \times p$ matrix of coefficients of the basis expansions and $\mathbf{\Psi}$ is a $p \times p$ matrix whose components are defined as

$$\psi_{ij} = \int_T \phi_i \phi_j dt. \tag{2.4}$$

2.3 The Functional Logistic Regression Model

Escabias et al. develop a principal component functional logistic regression model for one functional variable [2]. We describe this method before extending it to the case with multiple functional predictors. Consider observations $\{(y_i,\ \mathbf{x_i}(t)),\ t \in T,\ i = 1,\ \ldots,\ n\}$, where $\mathbf{x_i}(t)$ is a functional predictor. Each $x_{ij}(t)$ is the i^{th} observation at the j^{th} time point, and each $y_i \in \{0,1\}$. The conditional distribution of $Y_i \mid \mathbf{X_i}(t)$ is Bernoulli(π_i), with

$$\pi_i = E[Y_i \mid \mathbf{X_i}(t)] = \frac{\exp\{\alpha + \int x_i(t)\beta(t)dt\}}{1 + \exp\{\alpha + \int x_i(t)\beta(t)dt\}} \qquad i = 1, \ldots, n, \tag{2.5}$$

where $\alpha \in \mathbb{R}$ and $\beta(t)$, the parameter, is a function. Making the logit transform, a generalized model is formed [7]:

$$l_i = \ln\left(\frac{\pi_i}{1 - \pi_i}\right) = \alpha + \int_T x_i(t)\beta(t)dt \qquad i = 1, \ldots, n. \tag{2.6}$$

Under the assumption that $\beta(t)$ belongs to the same L_2 space spanned by $\{\phi_1, \ldots, \phi_p\}$,

$$\beta(t) = \sum_{k=1}^{p} \beta_k \phi_k. \tag{2.7}$$

The l_i's can be expressed in terms of the $\mathbf{A\Psi}$ matrix,

$$\mathbf{L} = \alpha \mathbf{1}_{n\times 1} + \mathbf{A\Psi}\beta, \tag{2.8}$$

where β is a $p \times 1$ dimensional vector containing the coefficients β_k for the basis expansion of $\beta(t)$. The basis coefficients β_k can be estimated using a Newton-Raphson algorithm maximizing the likelihood equation

$$Y\prime(X - \Pi) = 0. \tag{2.9}$$

In (2.9), $Y = (y_1, \ldots, y_n)$, $\Pi = (\pi_1, \ldots, \pi_n)$ and X=(1| $\mathbf{A\Psi}$) [6]. Once the coefficients are estimated,

$$\hat{\beta}(t) = \sum_{k=1}^{p} \hat{\beta}_k \phi_k. \tag{2.10}$$

The vector $\mathbf{L}$ in (2.9) can be reexpressed in terms of the principal components of $\mathbf{A\Psi}$:

$$\mathbf{L} = \alpha \mathbf{1}_{n \times 1} + \mathbf{\Gamma V}' \beta = \mathbf{\Gamma} \gamma, \tag{2.11}$$

where $\mathbf{\Gamma}$=(ξ_{ij}) is the matrix of the p principal components and $\mathbf{V}$, the eigenvectors. This notation allows for estimation of the components of β,

$$\hat{\beta} = \mathbf{V} \hat{\gamma}. \tag{2.12}$$

Reduction of the effects of multicollinearity occurs when a number of PCs, $s \leq p$, is chosen.

2.4 Principal Component Functional Logistic Regression for Multiple Functional Predictors

The extension of PCA to the model with multiple functional predictors lies in the definition of the inner product [4]. Let $\{(y_i,\, \mathbf{x_i}^m(t)),\, t \in T,\, i = 1, \ldots, n,\, m = 1, \ldots, M\} \in L_2(T)$ spanned by $\{\phi_1, \ldots, \phi_p\}$, where each $\mathbf{x_i}^m(t)$ is a functional predictor and each $y_i \in$ {0,1}. The inner product, then, of two functional PCs can be defined as follows:

$$\langle \xi_i, \xi_j \rangle = \sum_{m=1}^{M} \int_T \xi_i^m \xi_j^m dt. \tag{2.13}$$

The functional logistic regression model with multiple predictors is still defined by (2.4). The definition of π_i, however, changes with the change in inner product:

$$\pi_i = E[Y_i \mid \mathbf{X_i}(t)] = \frac{\exp\{\alpha + \sum_{m=1}^{M} \int_T x_i^m(t)\beta^m(t)dt\}}{1+\exp\{\alpha + \sum_{m=1}^{M} \int_T x_i^m(t)\beta^m(t)dt\}} \qquad i = 1, \ldots, n. \tag{2.14}$$

Making the logit transform,

$$l_i = \alpha + \sum_{m=1}^{M} \int_T x_i^m(t)\beta^m(t)dt \qquad i = 1, \ldots, n. \tag{2.15}$$

To perform the dimension reducing PCA, we redefine the design matrix $\mathbf{A\Psi}$. Here,

$$\mathbf{A}_{n\times(Mp)} = \left[\begin{array}{c|c|c} \mathbf{A}^1 & \ldots & \mathbf{A}^M \end{array}\right], \tag{2.16}$$

and

$$\mathbf{\Psi}_{(Mp)\times(Mp)} = \begin{vmatrix} \mathbf{\Psi}^1 & \mathbf{0} & \ldots & \ldots \\ \mathbf{0} & \mathbf{\Psi}^2 & \mathbf{0} & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{0} & \ldots & \ldots & \mathbf{\Psi}^M \end{vmatrix}. \tag{2.17}$$

Each $\mathbf{A}^m$ is an $n \times p$ matrix of basis coefficients. $\mathbf{\Psi}$ is a diagonal block matrix with each $\mathbf{\Psi}^m$ having dimensions $p \times p$. Now, $\beta = (\beta_1',\ldots,\beta_M')'$. Redefining (2.7),

$$\mathbf{L} = \alpha\mathbf{1}_{n\times1} + \mathbf{A\Psi}\beta = \alpha\mathbf{1}_{n\times1} + \sum_{m=1}^{M} \mathbf{A}^m\mathbf{\Psi}^m\beta^m. \tag{2.18}$$

This definition depends on the assumption that each observation $x_i^m(t)$ and the respective parameter function, $\beta^m(t)$, can be defined by the same set of basis functions $\{\phi_1^m(t), \ldots, \phi_p^m(t)\}$.

Below, we express $\mathbf{L}$ in terms of the principal components

$$\mathbf{L} = \alpha\mathbf{1}_{n\times1} + \sum_{m=1}^{M} \mathbf{\Gamma}^m\mathbf{V}^{m\prime}\beta^m, \tag{2.19}$$

where $\mathbf{\Gamma}^m = (\xi_{ij}^m)_{n\times p}$ are the principal components of the $\mathbf{A\Psi}^m$, and $\mathbf{V}^m$ is the matrix of eigenvectors. The number of of PCs $s_m \leq p_m$, to be chosen for each of the M should be determined by cumulative variance. For simplicity, we chose the same number of PCs, s, for each of the M predictors. After the dimension has been reduced on the within-variable level, (2.12) becomes

$$\pi_{i(s)} = \frac{\exp\{\alpha_{(s)} + \sum_{m=1}^M \sum_{j=1}^s \xi_{ij(s)}^m f_{ij(s)}^m \beta_{j(s)}^m\}}{1 + \exp\{\alpha_{(s)} + \sum_{m=1}^M \sum_{j=1}^s \xi_{ij(s)}^m f_{ij(s)}^m \beta_{j(s)}^m\}} \qquad i = 1, \ldots, n. \tag{2.20}$$

We can re-express (2.17) as

$$\mathbf{L} = \alpha_{(s)}\mathbf{1}_{n\times 1} + \sum_{m=1}^{M} \mathbf{\Gamma}_{(s)}^m \mathbf{V}_{(s)}^{m\prime} \beta_{(s)}^m. \tag{2.21}$$

The vectors β^m can be estimated as in (2.9),

$$\hat{\beta}^m = \mathbf{V}^m \hat{\gamma}^m. \tag{2.22}$$

Although multicollinearity has been dealt with on a within variable basis, as M gets large there could be multiple predictors providing similar information. In the case of fMRI data, time series of adjacent voxels are expected to be similar. There is a need to reduce dimension on the multiple variable level by selecting only those functional predictors which are relevant to the response. This will alleviate another potential source of multicollinearity. The following section of this thesis will focus on a method of simultaneous model estimation and variable selection.

Chapter 3

Group Lasso for Functional Logistic Regression

3.1 Introduction

The principal component analysis allows for removal of redundant information on a within predictor basis. As stated before, this is necessary due to the autocorrelation of observations between time points. There is also a need to select only those predictors which provide relevant information to the model.

There are many methods of variable selection used in linear models and generalized linear models. Model selection techniques, such as stepwise and forward selection, cycle algorithmically through subsets of variables until certain criteria are met. The variables included in the various steps of the algorithm are determined by previously selected p-values. These methods are inherently subjective, as it is up to the person analyzing the model to choose the "best" model based on a set of criteria. The criteria that determine the quality of the model are also chosen by the analyst.

3.2 Lasso

A more objective method of variable selection, the lasso, was introduced by Tibshirani in 1996. The lasso is a method that simultaneously performs model selection and parameter estimation. It is an L_1 regularization technique that performs this variable selection by shrinking certain β coefficients to exactly 0, excluding those predictors from the model. The other, non-zero, coefficients represent variables that are relevant to the model. This is done by solving the least squares estimation subject to a constraint on the β coefficients. Assume a standard regression model with independent observations $\{(y_i,\ \mathbf{x}_i),\ i = 1, \dots, n\}$, where $\mathbf{x}_i$=(x_{i1},...,x_{ip}). The estimates of regression coefficients by the lasso method $(\hat{\alpha}, \hat{\beta})$ are

$$(\hat{\alpha}, \hat{\beta}) := \arg\min\{\sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{p}\beta_j x_{ij})^2\}, \tag{3.1}$$

under $\sum_j |\beta_j| \leq t$, where $t \geq 0$. Note that α is not penalized.

Tibshirani also applied the lasso to the logistic regression model [11]. Consider independent observations $\{(y_i, \mathbf{x}_i), i = 1, \dots, n\}$ where $y_i \in \{0, 1\}$. The variable selection and model estimation are performed by maximizing the loglikelihood function,

$$l(\beta) = \sum_{i=1}^{n} y_i(\mathbf{x}_i'\beta) - \ln(1 + \exp\{\mathbf{x}_i'\beta\}), \tag{3.2}$$

under $\sum_j |\beta_j| \leq t$, where $t \geq 0$. An iterated reweighted least squares algorithm is used to compute $\hat{\beta}$ under these conditions.

3.3 Group Lasso

Consider a linear model with multiple predictors, some of them categorical. A categorical predictor with l levels will be represented in the model by $l - 1$ variables. The lasso only has the ability to shrink individual regression coefficients to zero. In the case of the categorical predictor, this has little interpretation. If the categorical predictor is not relevant to the response, all $l - 1$ variables should be removed from the model.

Consider independent observations $\{(y_i, \mathbf{x}_i), i = 1, \dots, n\}$ where $\mathbf{x}_i = (\mathbf{x}_{i1}', \dots, \mathbf{x}_{iM}')'$. Each $\mathbf{x}_{im}$ represents a group of predictors. The linear regression model is defined as

$$\mathbf{Y} = \alpha + \sum_{m=1}^{M} \mathbf{x_m}\beta_{\mathbf{m}} + \varepsilon, \tag{3.3}$$

where $\alpha \in \mathbb{R}$ is the intercept, each $\beta_{\mathbf{m}}$ is a vector whose components are the regression coefficients for the m^{th} group of predictors and $\mathbf{Y}_{n\times 1}$ is the vector of responses. Yuan and

Lin [14] developed a method of variable selection called the group lasso considers each of the M groups of variables for inclusion or exclusion in the model. The coefficient estimates are defined as

$$\hat{\beta} = \arg\min(\|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{m=1}^{M} \|\beta_m\|_2), \tag{3.4}$$

where λ is a tuning parameter and $\beta = (\alpha, \beta_1', \ldots, \beta_M')'$. The penalty is a mixture of L_1 and L_2 regularization methods, the lasso and the ridge regression penalties.

3.4 Group Lasso for Functional Logistic Regression

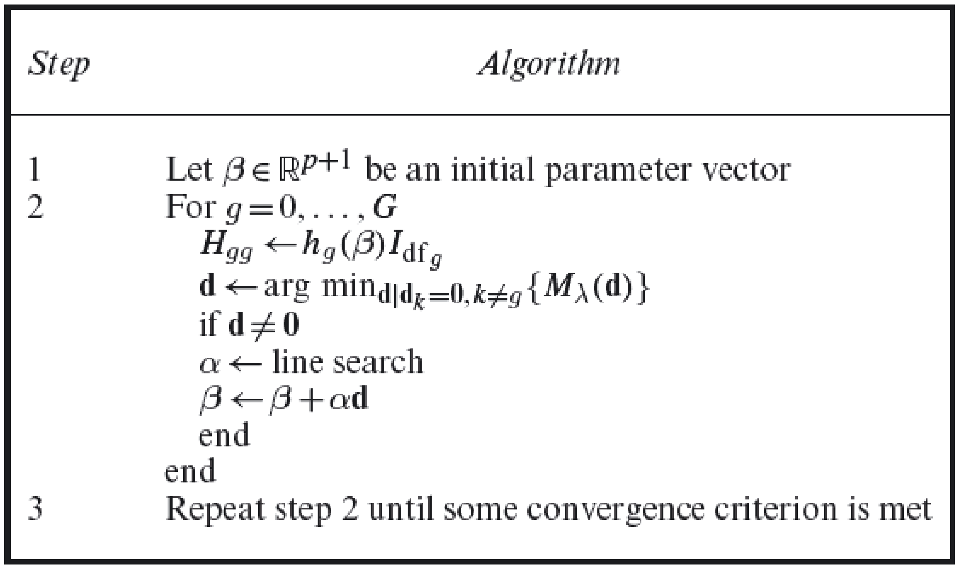
Meier et al. [6] describe a method of group lasso for the multivariate logistic regression model. Consider independent observations $\{(y_i, \mathbf{x}_i), i = 1, \ldots, n\}$ where $y_i \in \{0, 1\}$ and $\mathbf{x}_i = (\mathbf{x}_{i1}', \ldots, \mathbf{x}_{iM}')'$. Each $\mathbf{x}_i^m$ represents a group of predictors. Group lasso is performed by minimizing the following convex function:

$$S_\lambda(\beta) = -l(\beta) + \lambda \sum_{m=1}^{M} s(df_m)\|\beta^m\|_2, \tag{3.5}$$

where df_m is the degrees of freedom of the m^{th} group of predictors. The use of $s(df_m) = df_m^{1/2}$ is suggested [6]. The solution to this equation is the logistic group lasso estimator, $\hat{\beta}_\lambda$. It is found using a block co-ordinate gradient descent minimization algorithm. The algorithm uses a second-order Taylor series expansion,

$$S_\lambda(\hat{\beta}^{(t)} + \mathbf{d}) \approx -\{l(\hat{\beta}^{(t)}) + \mathbf{d}^T \nabla l(\hat{\beta}^{(t)}) + \frac{1}{2}\mathbf{d}^T H^{(t)}\mathbf{d}\} + \lambda \sum_{m=1}^{M} s(df_m)\|\hat{\beta}_m^{(t)} + \mathbf{d}_m\|_2 = M_\lambda^{(t)}(\mathbf{d}). \tag{3.6}$$

The algorithm begins by assuming an initial parameter vector, $\beta^{(0)}$. For each of the M groups of variables, the algorithm finds $\mathbf{d}$ that minimizes $M_\lambda(\mathbf{d})$. If this $\mathbf{d}$ is not identically 0, the estimate of β is updated. The updated estimate $\hat{\beta}^{(t+1)}$ is the previous estimate, $\beta^{(t)}$, plus



```
Step    Algorithm

1       Let $\beta \in \mathbb{R}^{p+1}$ be an initial parameter vector
2       For $g = 0, \ldots, G$
            $H_{gg} \leftarrow h_g(\beta) I_{\mathrm{df}_g}$
            $\mathbf{d} \leftarrow \arg\min_{\mathbf{d}|\mathbf{d}_k=0, k\neq g}\{M_\lambda(\mathbf{d})\}$
            if $\mathbf{d} \neq \mathbf{0}$
            $\alpha \leftarrow$ line search
            $\beta \leftarrow \beta + \alpha\mathbf{d}$
            end
        end
3       Repeat step 2 until some convergence criterion is met
```

Figure 3.1: Group Lasso Algorithm: Outline of the block coordinate gradient descent algorithm used to perform grouped variable selection in the logistic regression model [6].

a scalar times **d**. This algorithm proceeds for each group until some convergence criterion is met. The choice of λ is dependent upon n and the degrees of freedom of each of the M groups. In the multivariate model, group lasso performed on a dataset containing M groups of discrete predictors. A number of groups of predictors less than M is selected. This version of the group lasso is shown to be asymptotically consistent [6]. The minimization can be done in R, using the package grplasso written by Meier et al [6].

We apply this group lasso method to the functional logistic regression model with multiple functional predictors. Recall observations $\{(y_i, \mathbf{x_i}^m(t)), t \in T, i = 1, \ldots, n, m = 1, \ldots, M\} \in L_2^M(T)$ spanned by $\{\phi_1, \ldots, \phi_p\}$, where each $\mathbf{x_i}^m(t)$ is a functional predictor and each $y_i \in \{0,1\}$. Consider the model in (2.19), after s principal components have been chosen. The loglikelihood function, $l(\beta)$, of the functional logistic regression model is

$$l(\beta) = \sum_{i=1}^{n} y_i(\alpha_{(s)} + \sum_{m=1}^{M} \int x_i^m(t)\beta_{(s)}^m(t)dt) - \ln(1 + \exp\{\alpha_{(s)} + \sum_{m=1}^{M} \int x_i^m(t)\beta_{(s)}^m(t)dt\}). \quad (3.7)$$

Using the definition $l(\beta)$ in (3.7) we minimize the objective equation,

$$S_\lambda(\beta) = -l(\beta) + \lambda \sum_{m=1}^{M} s(df_m) \|\beta^m\|_2. \tag{3.8}$$

In the case of our method of principal component logistic regression with multiple functional predictors, the degrees of freedom in (3.3) is equivalent to the number of chosen PCs, s.

In the functional case, each of the M functional predictors is defined by a group of s coefficients. When one entire group of coefficients is shrunk to 0, it excludes the corresponding single functional predictor from the model. For any set of s coefficients that are not equal to zero, the corresponding functional predictor is included in the model. In essence, the group lasso performs single variable selection in the functional logistic regression model with multiple functional predictors.

Chapter 4

Application

4.1 Introduction

In a Functional Magnetic Resonance Imaging experiment, an experimenter aims to measure the amount of activation in each voxel of the brain. When a part of the brain is active, there is increased bloodflow to the area. fMRI measures the change in blood flow using the blood-oxygen-level-dependent (BOLD) contrast [3]. Assessing which parts of the brain are active during an fMRI experiment allows researchers to determine which parts of the brain respond to certain stimuli. There is a need for classification tools in the statistical analysis of fMRI. Logistic regression could be used to distinguish between brains at rest and those presented with stimuli. Another application would be to distinguish between subjects receiving one of two particular stimuli. For example, an experimenter may play pieces of music or speech to a subject [9]. Being able to classify which stimulus was presented allows one to learn more about the way the brain works. Classification also has an application in diagnosis of mental illness or degenerative disease.

4.2 Simulation Study

We assess our methodology using a simulation study. Using the R package neuRosim, we were able to simulated preprocessed fMRI data. The package can be used to simulated fMRI time series or complete 4D fMRI volumes. With neuRosim, one can define the onset and duration of stimuli. One can specify the effect size of the stimuli, TR and times of spatial and temporal noise [13].

We simulated preprocessed four dimensional fMRI data for an area of 4000 voxels containing two non-overlapping regions of activation. We used neuRosim to create block designs of a stimulus followed by rest. Design 1 presented an effect size that was larger in Region 1 than in Region 2. Design 2 created activation that was larger in Region 2 than in Region 1. Half of the observations in each simulation were simulated under Design 1, the other half under Design 2. Our goal was to use the developed method of group lasso for functional logistic regression with multiple functional predictors to classify the validation set into the proper groups. We simulated 15 datasets each at two levels of signal-to-noise ratio (SNR), 0.75 and 3.87, and two levels of subject size, 30 and 50. An observation simulated under Design 1 was given a y value of 1, otherwise $y_i = 0$. Two-fold cross validation was then used to form training and validation sets. All analysis was performed in R; the package grplasso was used to perform the final variable selection [6]. Observations with $\hat{\pi}_i$ >0.5 were classified as y_i=1, otherwise y_i=0.

For each of the four sets, two-fold cross validation resulted in 30 models. We report the number of voxels selected out of the initial 4000 voxels. We also report sensitivity, defined as the ratio of true positives to true positives plus false negatives; false positive rate, the ration of false positives to false positives plus true negatives; and the accuracy, defined as the ratio of true positives plus true negatives to the number of observations in the validation set. These findings can be seen in Figure 4.1. We did not report the number of principal components selected in the table. In the cases where $n = 50$, the original number of basis functions was 43. After PCA, 8, 9 or 10 principal components were selected every time. In the cases where $n = 30$, 7 or 8 PCs were chosen. The method classifies well, even after use of a small number of voxels. As expected, the cases with fewer subjects have lower sensitivity and accuracy and a higher false positive rate. In the two simulations with lower SNR classification appears to improve, which is surprising.

	No. Voxels Selected	Sensitivity	False Positive Rate	Accuracy
SNR = 3.87 n = 50	5.414(1.842)	0.935(0.096)	0.053(0.079)	0.936(0.060)
SNR = 0.75 n = 50	5.267(1.617)	0.953(0.099)	0.044(0.082)	0.949(0.670)
SNR = 3.87 n = 30	3.867(1.332)	0.825(0.188)	0.166(0.197)	0.840(0.158)
SNR = 0.75 n = 30	3.967(1.449)	0.874(0.153)	0.108(0.159)	0.871(0.106)

Figure 4.1: Simulation Results: The values reported are the means, followed by their standard deviations in parentheses.

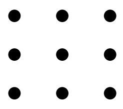


Figure 4.2: fMRI Nine Dot Experiment: Subjects were asked to use four and five lines to connect all nine dots in the figure above.

4.3 fMRI Example

To test this methodology on a real dataset, we used fMRI data collected by Auburn University's MRI Research Center. The data was collected from 6 subjects on a 7T MRI scanner, and each scanning session lasted 1000s. The data were preprocessed by the experimenters using SPM8. Slice timing correction was made. Spatial realignment, normalization, and smoothing were performed. And, finally, the data were detrended. The complete raw data voxel-wise time series were then extracted using MarsBaR.

The experimental design was a block design with two conditions.In one condition, subjects were asked to use four lines to connect all dots in Figure 4.2. In the other, subjects were asked to use five lines to connect all dots in Figure 4.2. Conditions were presented in a random sequence, and each condition was followed by a period of rest. All subjects were able to connect the nine dots using five lines, but only one (Subject 5) was able to solve the puzzle using four lines. Our aim was to use group lasso for functional logistic regression to classify the six subjects as having solved the four line puzzle, $y = 1$, or as being unable to

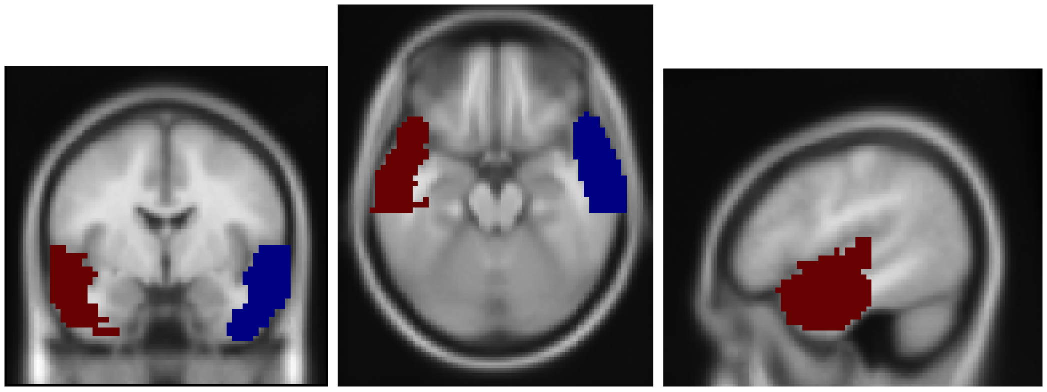


Figure 4.3: Voxel Mask of the Anterior Temporal Lobe: The right anterior temporal lobe is in red, the left in blue.

solve the puzzle, $y = 0$.

According to the experimenters, there were two important regions of interest (ROIs) in this study. These regions are the left and right anterior temporal lobes (ATL) which can be seen in Figure 4.3. They are associated with semantic memory, knowledge of objects and facts. From the right ATL, 7560 voxel time series were extracted. From the left ATL, 6584 voxel time series were extracted. This led to a total of time series from 14144 voxels. To perform PCA on the 14144 $\mathbf{A\Psi}$ matrices of the spline smooths, the number of basis functions, p, must be less than the number of subjects. We chose to use 5 basis functions. After performing the principal component analysis, 3 PCs were chosen. From the 14144 voxels, the group lasso procedure selected 11 voxels. From these 11 voxel time series, the classification procedure correctly selected Subject 5 as having solved the puzzle.

4.4 Conclusion

We have developed a viable method of variable selection for functional logistic regression by employing the group lasso in an interesting way. There are obvious limitations with small sample sizes. Being limited to a number of spline basis functions that is less than the sample size could lead to poor estimation of the functional observations. The method employed is also computationally expensive for a large number of functional variables and subjects. Each spline smooth procedure must be performed for all variables and subjects. Additionally, we would like to explore the statistical properties of the estimators of the parameter functions. The application in the field of Functional Magnetic Resonance Imaging is exciting. In future studies on real data, it would be interesting to study the neurological significance of the voxels selected by the group lasso.

Bibliography

[1] Craven, P., Wahba, G., 1979. Smoothing Noisy Data with Spline Functions. *Numerische Mathematik*, 31, 377-403.

[2] Escabias, M., Aguilera, A. M., Valderrama, M. J., 2004. Principal component estimation of functional logistic regression: discussion of two different approaches. *Nonparametric Statistics*, 16(3-4), 365-384.

[3] Huettel, S.A., Song, A.W., McCarthy, G., 2009. *Functional Magnetic Resonance Imaging* (2nd Edition). Sunderland, MA: Sinauer Associates.

[4] Joliffe, I. T., 2002. *Principal Component Analysis*, 2nd ed., Springer-Verlag, New York.

[5] Matsui, H., Konishi, S., 2011. Variable selection for functional regression models via the L1. *Computational Statistics and Data Analysis*, 55:3304-3310.

[6] Meier, L., van de Geer, S., Bühlmann, P., 2008. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B*, 70(Part 1), 53-71.

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

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

[9] Ryali, S., Kaustubh, S., Abrams, D., Menon, V., 2010. *Neuroimage*, June 51(2), 752-764.

[10] Tian, T.S, 2010. Functional data analysis in brain imaging studies. *Frontiers in Psychology*, 1(35), 1-11.

[11] Tibshirani, R., 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society B*, 58(1), 267-288.

[12] Viviani, R., Grön, G., Spitzer, M., 2005. Functional Principal Component Analysis of fMRI Data. *Human Brain Mapping*, 24, 109-129.

[13] Welvaert, M., Durnez, J., Moerkerke, B., Verdoolaege, G., Rosseel, Y., 2011. neuRosim: An R Package for Generating fMRI Data. *Journal of Statistical Software*, 40(10).

[14] Yuan, M., Lin, Y., 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society B*, 68, 4967.

Appendix A

Simulation of Preprocessed fMRI Data

```
seed1 <- NULL

seed0 <- NULL


n1 <-

n2 <-

n <- n1 + n2

SNR <-


#Evaluate and store results for ith observation

for (i in 1:n1){

seedi<-sample(1:10000,1)

seed1[[i]] <- seedi

}


for (i in 1:n2){

seedi<-sample(1:10000,1)

seed0[[i]] <- seedi

}

library(neuRosim)

#Parameters

dimx <- 20

dimy <- 20
```

```
dimz <- 10

nscan <- 100
TR <- 2
total <- TR*nscan
os <- seq(1, total, 20)
dur <- 7

regions <- simprepSpatial(regions = 2, coord = list(c(2, 2, 2),
 c(-2,-3,1)), radius = c(3,1), form = "sphere")

onset <- list(os,os)

duration <- list(dur, dur)

effect1 <- list(7, 15)
effect2 <- list(12, 8)

design1 <- simprepTemporal (regions = 2, onsets = list(os,os),
durations = duration, TR = TR, hrf = "double-gamma", effectsize =
effect1, totaltime=total)

design2 <- simprepTemporal (regions = 2, onsets = list(os,os),
durations = duration, TR = TR, hrf = "double-gamma", effectsize
effect2, totaltime=total)

#Subjects 1-25
```

```
data <- NULL
a <- NULL

for(i in 1:n1){

set.seed(seed1[i])

a <- simVOLfmri(design = design1, image = regions, base = 100,
dim= c(dimx,dimy,dimz),SNR = SNR, noise = "mixture", type =
"rician", rho.temp = c(0.142,0.108,0.084), rho.spat = 0.4, w=
c(0.05,0.1,0.01,0.09,0.05,0.7))
data[[i]] <- a
}

#Subjects 26-50
for(i in 1:n2){

set.seed(seed0[i])

a <- simVOLfmri(design = design2, image = regions, base = 100,
dim= c(dimx,dimy,dimz),SNR = SNR, noise = "mixture", type =
"rician", rho.temp = c(0.142,0.108,0.084), rho.spat = 0.4, w=
c(0.05,0.1,0.01,0.09,0.05,0.7))
data[[i+n1]] <- a
}
```

Appendix B

Simulation Analysis Code for R

```
#Sort data all subjects

vts<- NULL

vtsoutijk <- NULL

a <- NULL


for(m in 1:n){

for (i in 1:dimx){

for (j in 1:dimy){

for (k in 1:dimz){

vtsoutijk<- data[[m]][i,j,k,]

vtsoutijk <- t(vtsoutijk)

a <- cbind(a, vtsoutijk)

}

}

}

vts[[m]] <- a

a <- NULL

}


vtsALL <- NULL


for(i in 1:n){
```

```
vtsALL <- rbind(vtsALL,vts[[i]])
}

vtsV <- NULL
a <- NULL

for(i in 1:(dimx*dimy*dimz)){
a <-matrix(vtsALL[,(100*i-99):(100*i)], n, 100)
vtsV[[i]] <- a
}

vtsV <- t(vtsV)

#begin basis expansion
library(fda)

knots <- seq(0,200,10)
norder <- 4
nbasis <- length(knots) + norder - 2
T <- seq(2,200,2)
Trange = c(0,200)

lambda=1e6

#Create b-spline basis
bbasisV <- create.bspline.basis(Trange,nbasis,norder,knots)
```

```
curv.Lfd <- int2Lfd(2)

curv.fdParV <- fdPar(bbasisV, curv.Lfd,lambda)


#choose smoothing parameter

lambdas = 10^seq(-4,4,by=0.5)

mean.gcv = rep(0,length(lambdas))


#Initialize dataframes

SmoothV <- NULL

SmoothVfd <-NULL

A <- NULL


for(j in 1:(dimx*dimy*dimz)){

for(ilam in 1:length(lambdas)){

  # Set lambda

   curv.fdParVi <- curv.fdParV

   curv.fdParVi$lambda <- lambdas[ilam]


   # Smooth

   Smoothi <- smooth.basis(T,t(vtsV[[j]]),curv.fdParVi)


   # Record average gcv

   mean.gcv[ilam] <- mean(Smoothi$gcv)

 }


#plot(lambdas,mean.gcv,type='b',log='x')

best = which.min(mean.gcv)
lambdabest = lambdas[best]

curv.fdParV$lambda = lambdabest

bj = smooth.basis(T,t(vtsV[[j]]),curv.fdParV)

SmoothV[[j]] <-bj



#Create A matrices

c <- SmoothV[[j]]$fd

SmoothVfd[[j]] <- c

d <- SmoothVfd[[j]]$coefs

d <- matrix(d,n,nbasis)

A[[j]] <- d

}



#Create PSI matrix

Psi1 <- inprod(bbasisV,bbasisV)

e<- NULL

APsi <- NULL


#multiply Am by Psim

for(i in 1:(dimx*dimy*dimz)){

e <- A[[i]]%*%Psi1

APsi[[i]] <- e

}


#Perform PCA in APsi matrices
```

```
APsiPCA <- NULL

for(i in 1:(dimx*dimy*dimz)){

h <- princomp(APsi[[i]], cor = TRUE, scores = TRUE)

APsiPCA[[i]] <- h

}


c<-NULL

d <- NULL

e<-NULL

#Calculate proportion of variance and choose number of PCs

PropVar <- NULL

for ( j in 1:(dimx*dimy*dimz)){

for ( i in 1:nbasis){

sdev <- APsiPCA[[j]]$sdev

c[i] <- (sdev[i])^2

c <- t(c)

}

for( i in 1:nbasis){

d[i] <- c[i]/sum(c)

}

e <- cumsum(d)

PropVar[[j]] <- e

}

NPCV <- NULL

NPC <- NULL

for(j in 1:(dimx*dimy*dimz)){

d <- NULL
for(i in 1:nbasis){

if(PropVar[[j]][i] < .91){

d[i] <- 1

}

else{ d[i] <- 0}

}

NPCV[[j]]<-d

NPC[j] <- sum(NPCV[[j]])

}

NPCV <- NULL

NPC <- NULL

for(j in 1:(dimx*dimy*dimz)){

d <- NULL

for(i in 1:nbasis){

if(PropVar[[j]][i] < .95){

d[i] <- 1

}

else{ d[i] <- 0}

}

NPCV[[j]]<-d

NPC[j] <- sum(NPCV[[j]])

}


#Select # PCs

min(NPC)

max(NPC)
```

```
numPC <-


#extract pCs
PC <- NULL
for(i in 1:(dimx*dimy*dimz)){
d <- APsiPCA[[i]]$scores[,1:numPC]
PC[[i]]<-d
}



#Create Y vector
Y <- c(rep(1,n1),rep(0,n2))
Y <- t(Y)
Y <- t(Y)


#Select training and validation sets
train <- sample(1:n,n1)
valid <- NULL
for(i in 1:n){
if (i %in% train){}
else{
valid <- rbind(valid,i)
}
}
PCtrain <- NULL
PCvalid <- NULL
x<- NULL
```

```
a <- NULL


#Training Design
for(j in 1:(dimx*dimy*dimz)){
a <- NULL
x <- NULL
for(i in 1:n1){


a <- PC[[j]][train[i],]
x<-rbind(x,a)
}
PCtrain[[j]]<-x
}
#Validation Design
for(j in 1:(dimx*dimy*dimz)){
a <- NULL
x <- NULL
for(i in 1:n1){


a <- PC[[j]][valid[i],]
x<-rbind(x,a)
}
PCvalid[[j]]<-x
}


#Training Y
Ytrain <- NULL
```

```
for(i in 1:n1){
Ytrain[i] <- Y[train[i]]
}

#Valid Y
Yvalid <- NULL
for(i in 1:n1){
Yvalid[i] <- Y[valid[i]]
}
#grplasso
library(grplasso)
library(calibrate)
Int <- ones(n1,1)

#Validaton Set as Validation
lasX<-NULL
lasX<-cbind(lasX,Int)

for(i in 1:(dimx*dimy*dimz)){
lasX <- cbind(lasX,PCtrain[[i]])
}

index <- NULL
index <- c(index,NA)
for(i in 1:(dimx*dimy*dimz)){
b <-rep(i,numPC)
index <- c(index,b)
```

```
}

lambdalas <- lambdamax(lasX, y=Ytrain, index=index, penscale = sqrt, model = LogReg()) *
lasfit <- grplasso(lasX, y=Ytrain, index= index, lambda = lambdalas, model= LogReg(), pe

Coefs <- lasfit$coefficients
dim(Coefs)

CoSUM <- NULL
for(i in 1:6){
CoSUM[i]<-sum(Coefs[2:(dimx*dimy*dimz*numPC + 1),i])
}

CoCo <- NULL
VarNUM <- NULL
for(j in 1:6){
g <- NULL
for(i in 1:(numPC*dimx*dimy*dimz + 1)){
if(Coefs[i,j] ==0){
g[i] <- 0}
else{g[i] <- 1}
}
CoCo[[j]]<-g

}
SumCoCo <- NULL
for(i in 1:6){
```

```
SumCoCo[i] <- sum(CoCo[[i]])

}

#View number of functional variables selected by grplasso

SumCoCo


#Find Lihat and Pihat and classify yhat

ModelCoef <- Coefs[,2]

Alpha <- ModelCoef[1]

Betastar <- ModelCoef[2:(numPC*dimx*dimy*dimz +1)]

Betastar <- t(Betastar)

Betastar <- t(Betastar)


M<-NULL

for(i in 1:(dimx*dimy*dimz)){

b <- PCvalid[[i]]%*%Betastar[(numPC*(i-1)+1):(numPC*i)]

M[[i]] <- b

}


APsiB <- 0*ones(n1,1)

for(i in 1:(dimx*dimy*dimz)){

APsiB <- APsiB + M[[i]]

}


#Find Lhat

Lhatvalid <- NULL

Lhatvalid = Alpha*ones(n1,1) + APsiB
```

```
 #Pistar1
 #find Pi(i)'s
Pstar <- NULL
YstarV <- NULL

for (i in 1:n1){
Pstar[[i]] <- (exp(Lhatvalid[i]))/(exp(Lhatvalid[i])+1)
if(Pstar[i]>0.5){
YstarV[[i]] <- 1
}
else{YstarV[[i]]<- 0}
}

#Training Set as Validation
lasX<-NULL

lasX<-cbind(lasX,Int)

for(i in 1:(dimx*dimy*dimz)){
lasX <- cbind(lasX,PCvalid[[i]])
}

index <- NULL
index <- c(index,NA)
for(i in 1:(dimx*dimy*dimz)){
b <-rep(i,numPC)
index <- c(index,b)
```

```
}


#Group Lasso

lambdalas <- lambdamax(lasX, y=Yvalid, index=index, penscale = sqrt, model = LogReg()) *

lasfit <- grplasso(lasX, y=Yvalid, index= index, lambda = lambdalas, model= LogReg(), pe


CoefsV <- lasfit$coefficients

dim(CoefsV)


CoSUM <- NULL

for(i in 1:6){

CoSUM[i]<-sum(CoefsV[2:(dimx*dimy*dimz*numPC + 1),i])

}


VCoCo <- NULL

VarNUM <- NULL

for(j in 1:6){

g <- NULL

for(i in 1:(numPC*dimx*dimy*dimz + 1)){

if(CoefsV[i,j] ==0){

g[i] <- 0}

else{g[i] <- 1}

}

VCoCo[[j]]<-g


}
```

```
SumVCoCo <- NULL
for(i in 1:6){
SumVCoCo[i] <- sum(VCoCo[[i]])
}

#View number of functional variables selected by grplasso
SumVCoCo

#Find Lihat and Pihat and classify yhat
ModelCoefV <- CoefsV[,2]
AlphaV <- ModelCoefV[1]
BetastarV <- ModelCoefV[2:(numPC*dimx*dimy*dimz +1)]
BetastarV <- t(BetastarV)
BetastarV <- t(BetastarV)

M<-NULL
for(i in 1:(dimx*dimy*dimz)){
b <- PCtrain[[i]]%*%BetastarV[(numPC*(i-1)+1):(numPC*i)]
M[[i]] <- b
}

APsiBV <- 0*ones(n1,1)
for(i in 1:(dimx*dimy*dimz)){
APsiBV <- APsiBV + M[[i]]
}

#Find Lhat
```

```
Lhattrain <- NULL

Lhattrain = AlphaV*ones(n1,1) + APsiBV


#Find Pi(i)'s

PstarT <- NULL

YstarT <- NULL


for (i in 1:n1){

PstarT[[i]] <- (exp(Lhattrain[i]))/(exp(Lhattrain[i])+1)

if(PstarT[i]>0.5){

YstarT[[i]] <- 1

}

else{YstarT[[i]]<- 0}

}
```