Subgroup-Specific Regression Models

by

Marjan Yaghoubi

B.Sc., University of Science and Culture, 2010
M.Sc., Iran University of Science and Technology, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Mathematics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

May 2021

© Marjan Yaghoubi, 2021

The following individuals certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis/dissertation entitled:

Subgroup-Specific Regression Models

submitted by Marjan Yaghoubi in partial fulfilment of the requirements of the degree of Master of Science

Dr. Javad Tavakoli, Irving K. Barber Faculty of Science
Supervisor

Dr. Paramjit Gill, Irving K. Barber Faculty of Science
Supervisory Committee Member

Dr. Jeffrey Andrews, Irving K. Barber Faculty of Science
Supervisory Committee Member

Dr. Robert Lalonde, Irving K. Barber Faculty of Science
University Examiner

Abstract

Data sets are becoming massive with ever increasing advances in data collection technologies and are altering the nature of biomedical research. With many techniques, these huge data sets can be challenging, or even impossible, to accurately analyse. In biomedical settings, data sets are frequently heterogeneous, with samples representing various subtypes of diseases that are thought to have variations with respect to underlying biology. A motivating example is the study of progressive diseases such as Alzheimer's disease (AD). While there is a significant increase in the number of studies that concentrate on regression modeling of the disease progression, they ignore the fact that the pattern change are profoundly different for patients with various initial profiles. Estimating separate models for each subgroup is extremely difficult due to small sample sizes in the high-dimensional setting, but may obtain results that are more accurate and reliable. Moreover, recognizing homogeneous subgroups of predictors can be cumbersome in high-dimensional regression analysis over subgroups of samples. This thesis attempts to improve upon an established method of regularized regression for group-structured datasets by using a linear combination of two penalty functions to select predictive clusters of correlated variables, and to allow for subgroup-specific parameter estimates. In order to showcase the performance of the suggested methodology, we conducted a series of experiments on Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset including three groups of Cognitively Normal Controls (CN), Late Mild Cognitive Impairment (LMCI), and Alzheimer's disease (AD) subjects to estimate Mini-Mental State Examination (MMSE) scores in multiple future time points. Results reveal the effectiveness of the suggested method in terms of Root Mean Square Error (RMSE) over several available well-known statistical methods in two subgroups, AD and LMCI. However, in CN group, our proposed method performed better than other methods at two time points. We also investigated the prediction performance of our proposed method with multiple multi-task learning regression methods.

Lay Summary

The great progress in data acquisition has resulted in major changes in scientific fields. Biomedical areas are especially fascinating to look at, as data sets in this field are becoming increasingly heterogenous, with samples spanning multiple subgroups that may be expected to have variations in patterns of association between observed measurements and a response of interest. As a motivating example, we consider a study of Alzheimer's disease (AD). Regression is a method of analysis that is ubiquitous in biomedical studies. With regression methods being so prevalent, it is crucial that it is being done efficiently and with enhanced accuracy. Along with the rise in the number of group-structured datasets, it is necessary to develop efficient regression methods to attain subgroup-specific parameters and structure. We begin our work by reviewing several previously presented supervised regression frameworks. Then, we modify a current regression method, which consider high-dimensional regression in the group-structured setting. Finally, we compare the performance of the model with multiple available well-known models in terms of root mean square error and correlation coefficients for Alzheimer's disease dataset.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my sincere gratefulness to my supervisor Dr. Javad Tavakoli for his expert advice, guidance, and support throughout my studies. I will be forever grateful for the opportunity of being his student and work under his supervision. I would also express my gratitude to Dr. Paramjit Gill, Dr. Jeffrey Andrews, and Dr. Robert G. Lalonde for taking time to evaluate my work.

Dedication

I would like to thank my husband and family for their support, patience, and motivation.

Chapter 1

Introduction

In this chapter, an overview of two main concepts in mathematics and statistics is given. The first one is the concept of mathematical optimization, as many recent applied statistics problems can be presented in the context of convex optimization. The second one is the field of machine learning, which is closely related to the optimization problem.

1.1 Optimization

Optimization is so prevalent in real-world application as many of the strategies adopted by humans seek to minimize a certain cost, or maximize a certain profit for a specific task. An optimization problem involves searching for the best element of a certain space ($\mathbb{X}$) with respect to some constraints. The standard form of an optimization problem defined as:

$$\underset{\mathtt{x}\in\mathbb{X}}{\text{minimize}}\{f(\mathtt{x})\} \quad \text{subject to} \quad \mathtt{x} \in C, \tag{1.1}$$

where $\mathtt{x}$ are the optimization variables, the objective function $f : \mathbb{X} \to \mathbb{R}\cup\{\infty\}$ denotes the cost of selecting $\mathtt{x} \in \mathbb{X}$, and $C \subset \mathbb{X}$ is the feasible region, where every acceptable solution must lie in C. The solution of Problem 1.1 is the element $\mathtt{x}_{op} \in C$ such that for every $\mathtt{x} \in C$, $f(\mathtt{x}_{op}) \leq f(\mathtt{x})$ is satisfied.

Problem 1.1 is a general formulation of a constrained problem, that is the solution must satisfy a specific criterion. However, it can be relaxed to the unconstrained problem

$$\underset{\mathtt{x}\in\mathbb{X}}{\text{minimize}}\{f(\mathtt{x})\}. \tag{1.2}$$

In general, every constrained problem can be converted into an unconstrained problem by making proper adjustments to its objective function [Bec17].

In optimization setting, two major concerns are: 1) the problem should be properly defined in terms of the variables, the objective function, and the feasible region. 2) the optimization problem should be manageable. This

actively demonstrates that, a balance between the problem's complexity and the cost of solving it needs to be found.

The space over which the optimization problems discussed in this thesis is a Euclidean space $\mathbb{E}$ with inner product, denoted by $\langle ., . \rangle$ and norm $\|.\|$. $\|.\|_p$ denotes the ℓ_p norm of its argument. In addition, the objective functions are restricted to functions on $\mathbb{E}$ which can take any real value and ∞ (extended real-valued functions).

1.1.1 Basic Concepts

In this section some general concepts associated with convex optimization are presented. Most of these concepts are included in [Bec17, BV04].

Definition 1.1. (Distance Function). The distance function $d_C : \mathbb{E} \longrightarrow [0, \infty)$ to a nonempty set $C \subset \mathbb{E}$ is defined by:

$$d_C(\mathtt{x}) = \inf_{c \in C} \|\mathtt{x} - c\| . \tag{1.3}$$

Definition 1.2. (Projection). The projection of $\mathtt{x} \in \mathbb{E}$ onto a closed, nonempty set $C \subset \mathbb{E}$ is the point $z = P_C(\mathtt{x}) \in C$ such that $d_C(\mathtt{x}) = \|\mathtt{x} - z\|$.

Definition 1.3. (Indicator Function). The indicator function of a closed set $C \subset \mathbb{E}$ is defined as:

$$\delta_C(\mathtt{x}) = \begin{cases} 0 & \text{if } \mathtt{x} \in C, \\ \infty & \text{if } \mathtt{x} \notin C. \end{cases} \tag{1.4}$$

As stated above, every constained optimization problem can also be expressed as an unconstrained optimization problem using the indicator function:

$$\min_{\mathtt{x} \in \mathbb{E}} \{f(\mathtt{x})\} \quad \text{s.t.} \quad \mathtt{x} \in C \equiv \quad \min_{\mathtt{x} \in \mathbb{E}} \{f(\mathtt{x}) + \delta_C(\mathtt{x})\}. \tag{1.5}$$

Definition 1.4. (Lower Semi-Continuous Function). $f : \mathbb{E} \longrightarrow \mathbb{R} \cup \{\infty\}$ is lower semi-contininuous or lsc at $\mathtt{x} \in \mathbb{E}$ if the following inequality is hold for every sequence $\{\mathtt{x}_n\} \longrightarrow \mathtt{x}$:

$$f(\mathtt{x}) \leq \lim_{n \to \infty} \inf f(\mathtt{x}_n). \tag{1.6}$$

Definition 1.5. (Convex Function). A function $f : \mathbb{E} \longrightarrow \mathbb{R} \cup \{\infty\}$ is convex if for all $\lambda \in [0, 1]$, the following inequality is satisfied:

$$(\forall x, y \in \mathrm{dom} f) \quad f((1 - \lambda)x + \lambda y) \leq (1 - \lambda) f(x) + \lambda f(y). \tag{1.7}$$

The objective functions of the minimization problems tackled in this thesis are all convex, since a local minimum is also a global minimum in this type of functions.

Definition 1.6. (Strictly Convex Function). A function $f : \mathbb{E} \longrightarrow \mathbb{R} \cup \{\infty\}$ is strictly convex if for all $\lambda \in (0, 1)$, the following inequality is satisfied:

$$(\forall x, y \in \mathrm{dom} f) \quad f((1-\lambda)x + \lambda y) < (1-\lambda)f(x) + \lambda f(y). \tag{1.8}$$

1.1.2 Gradient-based Optimization

In many optimization problems, the assumption is that the objective functions are convex and smooth (continuous and differentiable). This allows to solve problems in the form of (1.2) by methods which are based on the gradient. In this case, a necessary and sufficient optimality condition is

$$\nabla f(\mathtt{x}_{op}) = 0. \tag{1.9}$$

Thus, a solution of Equation 1.9 is equivalent to solving the minimization problem in (1.2). In many cases, Equation 1.9 has to be solved numerically by using iterative algorithms, such as Gradient Descent Methods (GDMs). According to [BV04], the outline of a GDM is given in Algorithm 1. This algorithm alternates between two steps: finding a descent direction, and the choice of a step size. A natural choice for the descent direction is the negative gradient $d = -\nabla f(\mathtt{x})$. The GDM algorithm runs until a stopping condition, usually of the form $\left\|\nabla f(\mathtt{x})\right\|_2 \leq \nu$, where $\nu > 0$, is satisfied.

Algorithm 1: Gradient Descent Method

Result: $\mathtt{x}_{op} \simeq \arg\min_{\mathtt{x}\in\mathbb{X}}\{f(\mathtt{x})\}$
Input: a convex and smooth function f ;
Initialization: a starting point $\mathtt{x}^{[0]} \in \mathbb{X}$;
for $i = 0, 1, \cdots$ **do**
  set a descent direction $d^{[i]} \in \mathbb{X}$;
  choose a step size $t^{[i]} > 0$;
  update $\mathtt{x}^{[i+1]} := \mathtt{x}^{[i]} + t^{[i]} d^{[i]}$;
end

1.2 Subdifferential Calculus

Many optimization problems of interest are non-differentiable. In such cases, classical gradient-based techniques cannot be implemented and other

methods such as Proximal Gradient Methods (PGMs) [Bec17] will need to be utilized. In particular, PGMs generalize the concept of gradient by those of subgradient.

Definition 1.7. (Subgradient). If $f : \mathbb{E} \to (-\infty, \infty]$ is a proper (dom $f \neq \emptyset$) function, then a subgradient of f at $\mathtt{x} \in \mathbb{E}$ is a vector $g \in \mathbb{E}$ such that

$$(\forall y \in \mathbb{E}) \quad f(\mathtt{x}) + \langle g, y - \mathtt{x} \rangle \leq f(y). \tag{1.10}$$

Definition 1.8. (Subdifferential). If $f : \mathbb{E} \to (-\infty, \infty]$ is a proper function, then the subdifferntial is defined as:

$$\partial f(\mathtt{x}) = \{g \in \mathbb{E} | \ (\forall y \in \mathbb{E}) \quad f(\mathtt{x}) + \langle g, y - \mathtt{x} \rangle \leq f(y)\}. \tag{1.11}$$

According to [Roc96], several properties of the subdifferential are:

1. $\partial f(\mathtt{x})$ is either empty or a closed convex set, for $\mathtt{x} \in \mathbb{E}$.
2. for $\mathtt{x} \in \mathbb{E}$, f is differentiable at $\mathtt{x} \Leftrightarrow \partial f(\mathtt{x}) = \{\nabla f(\mathtt{x})\}$.
3. $\partial(\lambda f)(\mathtt{x}) = \lambda \partial f(\mathtt{x}), \ \ \forall \lambda > 0$.

Theorem 1.9. *[Roc96, Fermat's Rule] The point $\mathtt{x} \in \mathbb{E}$ is a global minimizer of a proper convex function $f : \mathbb{E} \to (-\infty, \infty]$ if and only if $0 \in \partial f(\mathtt{x})$.*

Proof. Indeed,

$$\begin{aligned} 0 \in \partial f(\mathtt{x}) &\Leftrightarrow (\forall y \in \mathbb{E}) \quad f(\mathtt{x}) + \langle 0, y - \mathtt{x} \rangle \leq f(y) \\ &\Leftrightarrow (\forall y \in \mathbb{E}) \quad f(\mathtt{x}) \leq f(y) \\ &\Leftrightarrow \mathtt{x} \text{ is a global minimizer of } f. \end{aligned}$$

□

1.3 Proximal Gradient Methods

In the following, the concept of the prox(imity) operator [Mor65] is described, which is the fundamental concept in the PGM [BT09].

Definition 1.10. (Prox Operator). The proximal mapping or prox operator of a function $f : \mathbb{E} \to (-\infty, \infty]$ at point $\mathtt{x} \in \mathbb{E}$ is given by

$$prox_f(\mathtt{x}) = \arg\min_{u \in \mathbb{E}} \left\{ \frac{1}{2} \|u - \mathtt{x}\|^2 + f(u) \right\}. \tag{1.12}$$

An altenative definition of the prox operator is

$$prox_f(\mathtt{x}) = (\mathrm{Id} + \partial f)^{-1}(\mathtt{x}). \tag{1.13}$$

It is worth noting that, the zeros of the subdifferntial of a proper convex function are the fixed points of its prox operators. Therefore, by Fermat's rule, we have:

$$\mathtt{x} = \arg\min_{u\in\mathbb{E}} f(u) \Leftrightarrow 0 \in \partial f(\mathtt{x}) \Leftrightarrow \mathtt{x} = prox_f(\mathtt{x}). \tag{1.14}$$

In the following the PGMs are introduced. Consider the following general composite problem

$$\min_{\mathtt{x}\in\mathbb{E}} \big\{F(\mathtt{x}) := f(\mathtt{x}) + g(\mathtt{x})\big\}, \tag{1.15}$$

where $f : \mathbb{E} \to (-\infty, \infty]$ is a proper closed convex and L-smooth function, and $g : \mathbb{E} \to (-\infty, \infty]$ is just proper closed and convex function. PGM is described in Algorithm 2.

Algorithm 2: Proximal Gradient Method

Result: $\mathtt{x}^{[i+1]} \simeq \arg\min_{\mathtt{x}\in\mathbb{E}}\{F(\mathtt{x})\}$
initialization: pick a starting point $\mathtt{x}^{[0]} \in \mathbb{E}$;
for $i = 0, 1, \cdots$ **do**
  choose a step size $t = \frac{1}{L}$;
  update $\mathtt{x}^{[i+1]} := prox_{\frac{1}{L}g}(\mathtt{x}^{[i]} - t\nabla f(\mathtt{x}^{[i]}))$;
end

PGM consists of a gradient step followed by a prox operator, and the update step can be compactly written as

$$\mathtt{x}^{[i+1]} = T_L^{f,g}(\mathtt{x}^{[i]}),$$

where T is a prox-grad operator defined by

$$T_L^{f,g}(\mathtt{x}) = prox_{\frac{1}{L}g}(\mathtt{x} - \frac{1}{L}\nabla f(\mathtt{x})).$$

1.3.1 FISTA

Fast Iterative Shrinkage- Thresholding Algorithm (FISTA) [BT09] is also used to minimize the sum of a smooth and a non-smooth functions. In particular, the objective function is $F = f_1 + f_2$, where the smooth term f_1 is a convex extended real-valued function on $\mathbb{E}$, with Lipschitz gradient ∇f_1 and Lipschitz constant L. The non -smooth term f_2 is a convex, proper and lsc function. Therefore, the optimization problem is:

$$\min_{\mathtt{x}\in\mathbb{E}} \big\{F(\mathtt{x})\big\} = \min_{\mathtt{x}\in\mathbb{E}} \big\{f_1(\mathtt{x}) + f_2(\mathtt{x})\big\}. \tag{1.16}$$

By Fermat's rule, $\mathtt{x}^{\text{opt}}$ is a minimizer of F if and only if

$$\begin{aligned}
0 \in \partial F(\mathtt{x}^{\text{opt}}) &= \partial\left(f_1 + f_2\right)(\mathtt{x}^{\text{opt}}) \\
&= \nabla f_1(\mathtt{x}^{\text{opt}}) + \partial f_2(\mathtt{x}^{\text{opt}}) \\
&\Leftrightarrow \mathtt{x}^{\text{opt}} \in \mathtt{x}^{\text{opt}} + \delta\nabla f_1(\mathtt{x}^{\text{opt}}) + \delta\partial f_2(\mathtt{x}^{\text{opt}}), \forall\delta > 0 \\
&\Leftrightarrow \mathtt{x}^{\text{opt}} - \delta\nabla f_1(\mathtt{x}^{\text{opt}}) \in x^{\text{opt}} + \delta\partial f_2(\mathtt{x}^{\text{opt}}) \\
&\Leftrightarrow \mathtt{x}^{\text{opt}} \in (Id + \delta\partial f_2)^{-1}(\mathtt{x}^{\text{opt}} - \delta\nabla f_1(\mathtt{x}^{\text{opt}})) \\
&\Leftrightarrow \mathtt{x}^{\text{opt}} = \text{prox}_{\delta f_2}(\mathtt{x}^{\text{opt}} - \delta\nabla f_1(\mathtt{x}^{\text{opt}})).
\end{aligned}$$

FISTA summarized in Algorithm 3.

Algorithm 3: Fast Iterative Shrinkage- Thresholding Algorithm

Result: $\mathtt{x}^{[i+1]} \simeq \arg\min_{\mathtt{x}\in\mathbb{E}}\{F(\mathtt{x})\}$
Input: f_1 with ∇f_1 and a Lipschitz constant L , f_2;
Initialization: $\mathtt{x}^{[0]} \in \mathbb{E}$;
$y^{[0]} = x^{[0]}$;
$t^{[0]} = 1$;
for $i = 0, 1, \cdots$ **do**
  $\mathtt{x}^{[i+1]} := \text{prox}_{\frac{1}{L}f_2}(y^{[i]} - \frac{1}{L}\nabla f_1(y^{[i]}))$;
  $t^{[i+1]} = \frac{1+\sqrt{1+4(t^{[i]})^2}}{2}$;
  $y^{[i+1]} = \mathtt{x}^{[i+1]} + \frac{t^{[i]}-1}{t^{[i+1]}}(\mathtt{x}^{[i+1]} - \mathtt{x}^{[i]})$;
end

Beck and Teboulle in [BT09] showed that the sequence $\mathtt{x}^{[i+1]}$ converges to the minimum of Problem 1.16.

1.4 Machine Learning

In statistical and mathematical communities, machine learning (ML) concerns the development and study of systems that can learn from data. According to [DHS01], the design of a ML framework involves several tasks. Figure 1.1 represents the general ML lifecycle, which includes the following steps:

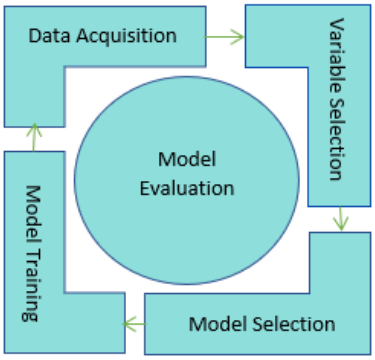

Figure 1.1: Machine Learning Cycle.

Data Acquisition. The data is collected, cleaned, and preprocessed in this step. An instance of this stage is feature normalization.

Variable Selection. The selection of significant features for the problem at hand is included in this step. In particular, this step is intended for high-dimensional data with a relatively limited number of samples.

Model Selection. Model selection is an important step when creating any sort of statistical model after data collection and feature selection, as the complexity of the model should be sufficient to learn the undelying data structure. If the model is too basic, then the relationship between the input and the output can not be learned. On the other hand, if the model is too complex, then the model will also learn noisy observation, and for new and unknown observations it will generalize very poorly. These effects are referred to under-fitting and over-fititing, respectively.

Model training. The model that learns from the data is developed through a process called training. This step is usually formulated as an optimization problem.

Model Evaluation. In this step, the quality of the model is evaluated based on some metrics.

In particular, the focus of this thesis is on supervised learning (where a response variable is measured), in which a function/process has to be determined from a set of input-output pairs, also known as a training set, that map a specific input data to the output . Afterwards, it can be applied to estimate the output for unobserved data. The building blocks of supervised learning are model, parameters, and objective function.

Model. The model refers to the mathematical framework in which the output (prediction) is made from the input. The linear regression model,

where the prediction is given as a linear combination of weighted input features, is a common example.

Parameters. The parameters are those that should be learnt from the data. For instance, in linear regression, the parameters are regression coefficients.

Objective Function. In order to train the model, we need to define the objective function to measure how well the model fits to the data. The objective functions are usually separated into two terms, a loss function (L), and a penalty (regularization) term (R):

$$Obj(.) = L(.) + \gamma R(.),$$

L measures how predictive our model is with respect to the training data, such as Mean Squared Error (MSE) for regression problems. R measures and controls the complexity of the model, include additional information about the ground truth of the problem. γ is a regularization parameter and the performance of the model depends on this value. It is responsible for the balance between the complexity and the accuracy of the model. Therefore, a good choice of γ is important and usually determined by cross validation procedure.

1.4.1 Bias and Variance

Bias and variance are the fundamental concepts in the field of ML. High variance related to over-fitting, and large bias related to under-fitting. These terms, bias and variance or "bias-variance tradeoff" [GBD92], are used to describe the performance of a model. The inability for an ML method to capture the true structure that exists in the training data is called bias, and the difference in fits between datasets is called variance. In practice, the ideal algorithm has low bias and low variability. This is done by finding a balance between a simple model and a complex model. Three widely used methods for achieving this are regularization, boosting and bagging (they are further detailed in Chapter 2).

The remainder of the thesis is organized as follows:

- Chapter 2 provides a literature review for linear regession models, where regularized learning models play a pivotal role, indicating also how to solve them under a common approach based on PGMs. Furthermore, there is a focus on ensemble learning models.

- Chapter 3 presents details on the structured linear regression models. Moreover, a new regularized approach, called the generalized joint lasso, is

proposed. Also, multiple multi-task learning methods present in this chapter, which later in chapter 4, we show how these models are successfully applied to the real-world, health-related problem, where they can give accurate predictions.

- Chapter 4 provides results from the proposed method to Alzheimer's Disease Neuroimaging Initiative (ADNI) data set.

- Chapter 5 provides a summary and ideas for future work.

Chapter 2

Literature Review

In statistical and machine learning communities, linear regression, also known as least squares, is commonly used to predict a numerical response/output variable from a set of predictors/features. The output is estimated as a weighted linear combination of the predictors. A line is fit to the data using least squares, or in other words, a line that results in the minimum sum of squared residuals is found.

Notation

Suppose a sample of n cases, each represented by a p-dimensional input vector and a 1-dimensional output vector. Let $X = (\mathtt{x}_1, \cdots, \mathtt{x}_p) \in \mathbb{R}^{n \times p}$ represent the input matrix (matrix of predictors), and let $Y \in \mathbb{R}^{n \times 1}$ denote the output vector (response vector). The vector of coefficients of a linear model is denoted by $\beta \in \mathbb{R}^{p \times 1}$. For the sake of brevity, all the models discussed in the following section are considered without intercept term. More precisely, the predictors are standardized via removal of the mean and scaling to unit variance and the respose variable is centered such that $\sum_{i=1}^{n} y_i = 0$, $\sum_{i=1}^{n} \mathtt{x}_{ij} = 0$, and $\sum_{i=1}^{n} \mathtt{x}_{ij}^2 = 1$ for $j = 1, \cdots, p$.

2.1 Linear regression and least squares

The least squares approach is commonly used in linear regression problem in order to fit a line to the data. In linear regression problem, we seek to find a coefficient vector $\beta^* \in \mathbb{R}^{p \times 1}$ such that

$$\beta^* \in \arg\min_{\beta} \left\{ \frac{1}{2} \|Y - X\beta\|_2^2 \right\}. \tag{2.1}$$

To solve this minimization problem we can take the derivative of (2.1) with respect to β and set it equal to zero to obtain the normal equations

$$X^T X \beta^* = X^T Y. \tag{2.2}$$

Solving Equation 2.2 for β^* is equivalent to solving the minimization problem in (2.1). Many methods, both direct and iterative, exist to solve Equation 2.2 [Bjo96].

Since least squares approach has no control on model complexity, its performance will depend on the relation between the dimensionality of the data (the number of features p) and the sample size n. If the sample size is insufficient with respect to number of features to estimate these coefficients, then the model cannot be fit naively.

In many fields, such as biomedical studies, the problem is often of large dimension ($p > n$), so direct solvers will not suffice and iterative methods with regularization will need to be utilized. Regularization is when additional information is included about the desired solution. One option for regularization is to add a penalty term to the function being minimized where the penalty term has a specific structure in order to enforce a desired outcome. For example, the ℓ_1 norm can be used if sparse solutions are desired. If a penalty was added to (2.1), we would obtain

$$\beta^* = \arg\min_{\beta} \left\{ \frac{1}{2} \|Y - X\beta\|_2^2 + \lambda R(\beta) \right\}, \tag{2.3}$$

for some penalty term $R(\beta)$. A common method for solving Equation 2.3 is the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA).

2.2 Regularized Least Squares

Regularizations are the set of techniques that attempts to reduce the variance of the model by fitting a function appropriately on the given training set [Fri89]. Common regularized regression techniques include Lasso [Tib96], Ridge [HK70], and Elastic Net [ZH05].

2.2.1 Least absolute shrinkage and selection operator

Least absolute shrinkage and selection operator (Lasso) is a popular type of regularized regression analysis method, which uses ℓ_1 penalty function in order to alleviate the overfitting problem of least squares and impose sparsity structure on the vector of regression coefficients. Lasso model enhances the prediction accuracy of the resulting statistical model by shrinking the regression coefficients toward zero and set the regression coefficients of irrelevant variables to zero [HTF09]. Lasso, however, has a tendency to select only one variable from a group of highly correlated variables, so it is unable

to discover grouping information [ZH05]. The Lasso model is defined as the solution of the optimization problem

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \frac{1}{2} \|Y - X\beta\|_2^2 \right\}, \quad s.t. \ \|\beta\|_1 \leq t, \tag{2.4}$$

where $t > 0$ is a tuning parameter that controls the amount of shrinkage, and $\|.\|_1$ denotes the ℓ_1 norm of the coefficients (i.e., $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$). The smaller value of t will make more of the coefficients to be exactly zero and will shrink coefficients toward the origin. Equation 2.4 can be written in the Lagrangian form as

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \frac{1}{2} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}, \tag{2.5}$$

where $\lambda > 0$ is a tuning parameter which controls the strength of the constraint (sparsity level). The greater amount of shrinkage, more coefficients with zero value, can be obtained by setting λ to a large value, and therefore a simpler model can be achieved. On the other hand, when λ is small, Lasso would produce similar parameter estimates as least squares approach. The value for λ is determined using cross-validation procedure.

For Equation 2.5, the objective function is non-differentiable, hence it fits nicely in the framework of PGMs; in particular FISTA. In order to apply FISTA, Equation 2.5 is divided into a smooth termf_1 and a non-smooth term f_2:

$$\min_{\beta} \left\{ \underbrace{\frac{1}{2} \|Y - X\beta\|_2^2}_{f_1(\beta)} + \underbrace{\lambda \|\beta\|_1}_{f_2(\beta)} \right\}. \tag{2.6}$$

The gradient of $f_1(\beta)$ is $\nabla f_1(\beta) = (X^T y - X^T X\beta)$. The Lipschitz constant L is given by the largest eigenvalue of $X^T X$. Thus, FISTA can be applied once the prox operator of f_2 is obtained. Prox operator of the ℓ_1 norm is soft- thresholding (see Appendix A) applied element-wise:

$$\text{soft}_\lambda(\beta) = \begin{pmatrix} sgn(\beta_1)[|\beta_1| - \lambda]_+ \\ sgn(\beta_2)[|\beta_2| - \lambda]_+ \\ \vdots \\ sgn(\beta_p)[|\beta_p| - \lambda]_+ \end{pmatrix}.$$

The operator $[y]_+ := \max(y, 0)$ represents the nonnegative part of y.

2.2.2 Ridge Regression

Ridge regression model is similar to Lasso where the ℓ_1 penalty function ($\sum_{j=1}^{p} |\beta_j|$) is replaced by the ℓ_2 penalty ($\sum_{j=1}^{p} \beta_j^2$). Ridge regression tends to shrink the correlated predictors toward each other. The corresponding optimization problem is:

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\}, \tag{2.7}$$

where $\|.\|_2^2$ denotes the Euclidean (ℓ_2) norm of the coefficients (i.e., $\|\beta\|_2^2 = \sum_{j=1}^{p} \beta_j^2$), and $\lambda > 0$ is a regularization parameter.

This problem has a closed-form solution as

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T Y, \tag{2.8}$$

where $I \in \mathbb{R}^{p \times p}$ is the identity matrix. As $\lambda \to 0$, the solution of Equation 2.8 tends to the least squares one . Practically, the ℓ_2 norm shrinks the coefficients towards zero. In particular, when $X^T X = I$, we have

$$\hat{\beta}_{ridge} = \frac{1}{1+\lambda} \beta^*.$$

Both Lasso and Ridge regression are ideally suited for high-dimensional data [ZH08] wherein some kind of penalties have been applied to restrict the number of predictors entering the model. The principal difference between these two is the geometric shape of the constraint, ℓ_1 norm is square and ℓ_2 norm is circular [Fu98].

2.2.3 Elastic Net

Elastic net (ElNet) combines ℓ_1 and ℓ_2 penalty functions for group predictor selection, based on the fact that ridge regression tends to shrink the correlated predictors toward each other. The Elastic net estimate of $\hat{\beta} = (\hat{\beta}_1, \cdots, \hat{\beta}_p)$ is equivalent to the following constrained optimization problem

$$\hat{\beta}_{Enet} = \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 \right\}, \quad s.t. \ \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \leq t. \tag{2.9}$$

The function $\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2$ is called the elastic net penalty, which is a convex combination of the Lasso and Ridge penalty. t and $\alpha \in [0, 1]$ are tuning parameters.

Equation 2.9 can be written in the Lagrangian form

$$\hat{\beta}_{Enet} = \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 + \lambda \left(\alpha \|\beta\|_1 + \lambda(1-\alpha)\|\beta\|_2^2 \right) \right\}. \tag{2.10}$$

This optimization problem is also non-differentiable due to the ℓ_1 norm. Since ℓ_2 norm is differentiable, it can be included ino the smooth term f_1, and thus we have

$$\min_{\beta} \left\{ \underbrace{\|Y - X\beta\|_2^2 + \lambda\alpha\|\beta\|_2^2}_{f_1(\beta)} + \underbrace{\lambda(1-\alpha)\|\beta\|_1}_{f_2(\beta)} \right\}. \tag{2.11}$$

Subsequently, the gradient of $f_1(\beta)$ can be found easily, and the prox operator of $f_2(\beta)$ can be found following the same philosophy discussed in the Lasso problem. Therefore, FISTA can be applied.

2.3 Ensemble Learning

Ensemble learning is an ML paradigm in which an optimal predictive model is generated by combining the predictions of multiple weak learners (or base models), resulting in predictions that are more reliable than any single model. There are two types of Ensemble learning, Boosting and Bootstrap Aggregation (or Bagging).

2.3.1 Decision Tree

Decision trees (DTs) [Bel59] are very popular weak learners for ensemble methods. Every individual tree has several branches, nodes, and leaves. It begins at the very top with the root node that represents entire sample. This node then splits into a left and right nodes (sub-nodes or decision nodes or internal nodes), and how far this splitting goes is known as the tree depth. The last node in each branch (sub-tree) is called the leaf or terminal node. At the end of the leaf node, the average of the value of the dependent variable is the final prediction.

2.3.2 Random Forest

Random Forests (RF) [Bre01] are built from DTs that combine the simplicity of decision trees with flexibility which results in substantial improvement in prediction accuracy. RF uses straightforward algorithm as follows:

1. A bootstrapped dataset of the same size as the training set is created from the training dataset.
2. A decision tree is formed using the bootstapped dataset. However, a random subset of features is used when building the tree.
3. Repeat step 1 and 2, say 100 times. At this step a wide variety of trees are created, which makes RF more effective and robust than individual decision trees.
4. Estimate the accuracy of the Random Forest.

2.3.3 Gradient Boosting Algorithm

One of the most effective and powerful boosting techniques for building predictive models is Gradient Boosting (GB) [Fri00]. GB for regression proceeds in the following fashion:

1. Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable Loss Function $L(y_i, F(x))$
2. Initialize model with a constant value:
$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$
3. for $m = 1$ to M:
   (A) Compute the pseudo- residuals:
$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \cdots, n.$$
   (B) Fit a regression tree to the pseudo-residuals and create terminal nodes R_{jm}, for $j = 1, \cdots, J_m$
   (C) For $j = 1, \cdots, J_m$ compute:
$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$
   (D) Update the model:
$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_m I(x \in R_{jm}).$$

4. Output $F_M(x)$.

The x_i's refer to each row of measurements that is used to predict the response variable and the y_i's refer to the response variable measured for each subject in the dataset. n is the number of subjects in the dataset. Loss function $L(y_i, F(x))$ evaluates how well the response variable is predicted, where $F(x)$ is a function that gives the predicted values. In step 3 of GB algorithm, M trees are built, then the derivative of the loss function with respect to the predicted value is calculated. In r_{im}, i is the sample number and m is the index for the tree that is built. In part (B) of step 3, terminal nodes $R_{j,m}$ are refered to the leaves in our regression tree, where j is the index for each leaf in the tree. In part (C) of step 3, the output values γ_{jm} for each leaf is determined by solving the minimization problem where the previous prediction is taken into account. $x_i \in R_{ij}$ means all elements in that particular leaf node. In practice, the output values are always the average of the residuals that end up in the same leaf. In part (D), a new predictions for each sample is made. These new predictions are based on the previous predictions, the learning rate $\nu \in [0, 1]$, and the output values from the newest tree. A small ν reduces the effect each tree has on the final prediction, and this improves accuracy in the long run.

Chapter 3

Structured Linear Regression

This chapter introduces the Fused Lasso model under the same paradigm of regularized learning techniques. Furthermore, a modification of a current regularized regression model, the so-called joint lasso, is developed. This model is based on the Generalized Fused Lasso but for heterogenous datasets. The corresponding solver algorithm for this model is also included. Finally, this chapter provides details on multi-task learning method.

3.1 Fused Lasso and its generalization

Tibshirani et al. [TSR+05] introduced Fused Lasso (FL) that imposes sparsity in the coefficients, and also accounts for spatial association of predictors. In order to reflect this structure in the model, in addition to the ℓ_1 norm, another regularizer is added to the final model, which encourages similarity between coefficients corresponding to adjacent features. FL yields estimates using the following optimization problem

$$\hat{\beta}_{FL} = \arg\min_{\beta} \frac{1}{2}\|Y - X\beta\|_2^2, \quad s.t. \ \sum_{j=1}^{p} |\beta_j| \leq t \ \text{ and } \ \sum_{j=1}^{p-1} |\beta_{j+1} - \beta_j| \leq s, \tag{3.1}$$

where t and s are tuning parameters. The second constraint (fusion penalty) encourages sparsity in the differences of coefficients. The fusion penalty can be written as follows:

$$\|H\beta\|_1 = \sum_{j=1}^{p-1} |\beta_{j+1} - \beta_j|, \tag{3.2}$$

where $H \in \mathbb{R}^{(p-1)\times p}$ denotes the differencing matrix defined as:

$$h_{ij} = \begin{cases} -1 & \text{if } i = j \\ 1 & \text{if } j = i + 1 \\ 0 & \text{otherwise.} \end{cases}$$

A natural extension of the FL is to remove the ordering restriction on features. She in [She10] presented a generalization of the FL, called the Clustered Lasso (Classo), to reflect this behaviour. Classo solves the following optimization problem

$$\hat{\beta}_{classo} = \arg\min_{\beta} \frac{1}{2}\|Y - X\beta\|_2^2, \quad s.t. \ \sum_{j=1}^{p} |\beta_j| \leq t \ \text{ and } \ \sum_{j<k}^{p} |\beta_j - \beta_k| \leq s. \tag{3.3}$$

FL and Classo methods can be unified in the framework of the generalized lasso (genlasso) [TT11]:

$$\hat{\beta}_{genlasso} = \arg\min_{\beta} \frac{1}{2}\|Y - X\beta\|_2^2, \quad s.t. \ \|H\beta\|_1 \leq t, \tag{3.4}$$

where $H \in \mathbb{R}^{m \times p}$ is a specified penalty matrix.

Authors in [JLL+15] present a variant of the genlasso, called Hexagonal Operator for Regression with Shrinkage and Equality Selection (HORSES), that selects positively correlated predictors in high dimensional setting. In fact, HORSES finds a homogeneous subgroup structure within the feature space. In contrast to the Elnet that puts negatively correlated predictors into the same group, the regularization term formulation in HORSES encourages grouping of positively correlated variables. The shape of the constraint region for HORSES is hexagonal, which encourages similarity between coefficients only in $y = x$ direction. The corresponding optimization problem is:

$$\hat{\beta}_{HORSES} = \arg\min_{\beta} \frac{1}{2}\|Y - X\beta\|_2^2, \quad s.t. \ \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j<k} |\beta_j - \beta_k| \leq t, \tag{3.5}$$

where $p^{-1/2} \leq \alpha \leq 1$. Equation 3.5 can be written in the Lagrangian form as follows:

$$\hat{\beta}_{HORSES} = \arg\min_{\beta} \left\{ \frac{1}{2}\|Y - X\beta\|_2^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j<k} |\beta_j - \beta_k| \right) \right\}. \tag{3.6}$$

HORSES is superior to other current methods in a variety of ways:

1. It selects groups of predictors, rather than randomly select one to represent the entire collinear group as the Lasso does.

2. Positively correlated variables are grouped together rather than both positively and negatively correlated variables.

3.2 Joint Lasso

In many applied fields, samples are divided into subgroups that may not be similar in terms of the underlying regression models. Such concerns can be seen in biomedical fields, where observations represent disease subtypes in terms of underlying biology, and hence do not have similar patterns of association between predictors and a response variable. If such a structure exists, the model should reflect it. Therefore, Dondelinger et al. in [DM18] proposed joint lasso (JL), which is used to jointly estimate subgroup-specific regression coefficients whilst producing a sparse solution and encouraging similarity between subgroup-specific regression coefficients. Joint lasso fits nicely in heterogenous datasets (group-structured datastes), as it provides subgroup-specific sparsity patterns.

Each subgroup $k \in \{1, \cdots, K\}$ considers the same set of p features, with different subgroup sample size n_k. Joint lasso yields estimates using

$$\hat{B} = \underset{B=[\beta_1 \cdots \beta_K]}{\arg\min} \sum_{k=1}^{K} \left\{ \frac{1}{2n_k} \|y_k - X_k \beta_k\|_2^2 + \lambda \|\beta_k\|_1 + \gamma \sum_{k'>k} \tau_{k,k'} \|\beta_k - \beta_{k'}\|_1 \right\}, \tag{3.7}$$

and a variant with an ℓ_2 penalty in the last term. λ, γ and τ as tuning parameters and the role of τ is to control the extent of fusion across specific subgroups. By default, all τ's are set to unity.

The sum over k in joint lasso is necessary to account for different sample sizes in different groups. Total sample size is $n = \sum_{k=1}^{K} n_k$. In the above formulation, $X_k \in \mathbb{R}^{n_k \times p}$ is the feature matrix for subgroup k and the corresponding vector of responses, $y_k \in \mathbb{R}^{n_k \times 1}$. $B = [\beta_1 \cdots \beta_K]$ is a $p \times K$ matrix that collects all coefficients together. $\beta_k \in \mathbb{R}^p$ denote subgroup-specific coefficients.

3.3 Generalized Joint Lasso (Our Model)

Grouping positively correlated variables during variable selection procedure when analysing a system is beneficial. The particular reason for this is that the variables within each category behave similarly, implying that they determine features that affect the system through the same pathways [JLL+15]. In this thesis, we therefore propose "generalized joint lasso" (GJL) as a modified version of the method outlined in the previous section. In order to simultaneously identify homogeneous subgroups of variables and

jointly estimate the regression coeficients in group-structured data, we introduce generalized joint lasso as

$$\hat{B} = \underset{B=[\beta_1 \cdots \beta_K]}{\arg\min} \sum_{k=1}^{K} \left\{ \frac{1}{2n_k} \|y_k - X_k \beta_k\|_2^2 + \lambda \left(\alpha \|\beta_k\|_1 + (1-\alpha) \sum_{k'>k} \tau_{k,k'} \|\beta_k - \beta_{k'}\|_1 \right) \right\}, \tag{3.8}$$

where $p^{-1/2} \leq \alpha \leq 1$. Problem 3.8 is non-differentiable due to having a non-smooth penalty function. [CKL$^+$10] described a proximal relaxation of this kind of problems in multi-task regression setting that introduces additional smoothing to transform the non-smooth objective function into a continuously differentiable function. Motivated by their work in [DM18, CKL$^+$10], it is straightforward to adapt their optimization method for the GJL model.

First, an undirected graph $G = (V, E)$ with vertex set $V = \{1, ..., K\}$ corresponding to the subgroup and edges between all vertices is created. The Problem 3.8 can then be formulated as

$$f_L(B) = \sum_k \left\{ \frac{1}{2n_k} \|y_k - X_k \beta_k\|_2^2 \right\} + \|BC\|_1 , \tag{3.9}$$

where $C = (\lambda_1 I, \lambda_2 H)$ is a $K \times (K + |E|)$ matrix, with $\lambda_1 = \lambda\alpha$, $\lambda_2 = \lambda(1-\alpha)$, and $I \in \mathbb{R}^{K \times K}$ the identity matrix . The last term in (3.9) includes both ℓ_1 and fusion penalties. The matrix H defined as

$$H_{k,e} = \begin{cases} \tau_{m,l} & \text{if } e = (m,l) \text{ and } k = m \\ -\tau_{m,l} & \text{if } e = (m,l) \text{ and } k = l \\ 0 & \text{otherwise.} \end{cases}$$

Because of duality between ℓ_1 and ℓ_∞, the overall penalty in Equation 3.9 can be written as $\|BC\|_1 = \max_{\|A\|_\infty \leq 1} \langle A, BC \rangle$, where $A \in \mathcal{Q} = \{A \mid \|A\|_\infty \leq 1, A \in \mathbb{R}^{p \times (K+|E|)}\}$ is an auxiliary matrix, and $\|.\|_\infty$ is the matrix ℓ_∞ norm, defined as the maximum absolute value of all entries. Following Chen et al. in [CKL$^+$10], a smooth approximation of $\|BC\|_1$ can be written as

$$f_\mu(B) = \max_{\|A\|_\infty \leq 1} \langle A, BC \rangle - \mu d(A), \tag{3.10}$$

where $\mu > 0$, and $d(A) \equiv \frac{1}{2}\|A\|_F^2$, with $\|.\|_F$ the Frobenius norm. Theorem 1 in [CKL$^+$10] shows that $f_\mu(B)$ is smooth with gradient $\nabla f_\mu(B) = A^* C^T$, where A^* is the optimal solution of Problem 3.10, and the closed-form expressions of it are introduced in the following lemma.

Lemma 3.1. *Let A^* be the optimal solution of Equation 3.10:*

$$A^* = S(BC/\mu),$$

where

$$S(x) = \begin{cases} x & if\ -1 < x < 1 \\ 1 & if\ x \geq 1 \\ -1 & if\ x \leq -1. \end{cases}$$

Substituting $\|BC\|_1$ with $f_\mu(B)$ in Equation 3.9, we have

$$\tilde{f}_L(B) = \sum_k \left\{ \frac{1}{2n_k} \|y_k - X_k\beta_k\|_2^2 \right\} + f_\mu(B), \tag{3.11}$$

which is smooth with

$$\nabla \tilde{f}_L(B) = \sum_k \left\{ \frac{1}{n_k} X_k^T (X_k\beta_k - y_k) \right\} + A^* C^T. \tag{3.12}$$

Moreover, $\nabla \tilde{f}_L(B)$ is Lipschitz continuous with an upper bound of the Lipschitz constant

$$L_U = \lambda_{\max}(X_k^T X_k) + \frac{\lambda_1^2 + 2 * \lambda_2^2 \max_{k \in V} d_k}{\mu}, \tag{3.13}$$

where $\lambda_{\max}(X_k^T X_k)$ is the largest eigenvalue of $(X_k^T X_k)$ and $d_k = \sum_{k'}^K \tau_{k,k'}$.

The result of the aforementioned procedure is that a smooth lower bound of $f_L(B)$, namely $\tilde{f}_L(B)$, is derived. Finally, the Nesterov's method [Nes05] can be applied for optimizing (3.11) as shown in Algorithm 4. For more details on this optimization approach see [CKL$^+$10].

Algorithm 4: Proximal gradient method for the generalized joint lasso

Result: $\hat{B} = B^{[i]}$
Input:$X, Y, \lambda_1, \lambda_2, G, \mu$;
Initialization: buid $C = (\lambda_1 I, \lambda_2 H)$; find L_U according to (3.13);
$W^{[0]} = 0 \in \mathbb{R}^{p \times K}$;
for $i = 0, 1, \cdots$ *until covergence of* $B^{[i]}$ **do**
  Compute $\nabla \tilde{f}(W^{[i]})$ according to Equation 3.12;
  Compute the gradient descent step: $B^{[i]} = W^{[i]} - \frac{1}{L_U} \nabla \tilde{f}(W^{[i]})$;
  Set $Z^{[i]} = -\frac{1}{L_U} \sum_{m=0}^{i} \frac{m+1}{2} \nabla \tilde{f}(W^{[i]})$;
  Set $W^{[i+1]} = \frac{i+1}{i+3} B^{[i]} + \frac{2}{i+3} Z^{[i]}$;
end

3.4 Multi-Task Learning

A standard methodology in ML is learning one task at a time (single-task analysis). However, multi-task learning (MTL) [Car98] has gained growing degree of attention to solve multiple learning tasks simultaneously. The goal of MTL is to learn the shared information among related tasks in order to enhance prediction accuracy [LJY09]. This section will discuss multiple multi-task learning techniques for predicting multiple related outcomes from a common set of predictors.

Consider a multi-task learning setting with t response variables (tasks). Suppose that p is the number of predictors, which is shared across all the tasks, and n is the number of samples. Let $X \in \mathbb{R}^{n\times p}$ indicate the matrix of predictors, $Y \in \mathbb{R}^{n\times t}$ represent a matrix of t responses over the same set of observations. $B \in \mathbb{R}^{p\times t}$ denote the parameter matrix, with column $\beta_{.i} \in \mathbb{R}^p$ corresponds to task i, $i = 1, \cdots, t$, row $\beta_{j.} \in \mathbb{R}^t$ corresponds to feature $j = 1, \cdots, p$, and y_k denotes the k-th column of Y.

The representative multitask learning methods used for comparison in Chapter 4 include the following models:

1. Multi-task feature learning with $\ell_{2,1}$ norm: [LJY09] introduced an MTL framework using a regularization based on the $\ell_{2,1}$ norm

$$\|B\|_{2,1} = \sum_{j=1}^{p} \left\|\beta_{j.}\right\|_2,$$

   where ℓ_2 encourages grouping (refers to the grouping effect of ℓ_2 norm) of the weights corresponding to the j-th feature across several tasks. Thus, the complete MTL model via the $\ell_{2,1}$-norm regularization is:

$$\min_B \{\frac{1}{2}\|Y - XB\|_F^2 + \lambda\|B\|_{2,1}\}. \tag{3.14}$$

   The formulation of the $\ell_{2,1}$ norm encourages multiple input features from different tasks to share similar sparsity patterns for parameters, and the selection of predictors is based on the joint strength of all tasks jointly.

2. Multi-task ElasticNet [PVG+11]: The objective function for this model is

$$\min_B \{\frac{1}{2}\|Y - XB\|_F^2 + \alpha\rho\|B\|_{2,1} + \alpha(1-\rho)/2\|B\|_F^2\}, \tag{3.15}$$

   where $0 < \rho \leq 1$. The penalty function of Multi-task ElasticNet is a combination of $\ell_{2,1}$ and ℓ_2.

3. Trace-Norm Regularized Multi-Task Learning: [JY09] proposed a formulation of MTL based on trace norm in order to discover the low-rank common subspace among different tasks. Hence, the optimization problem becomes

$$\min_B \{\frac{1}{2}\|Y - XB\|_F^2 + \lambda\|B\|_*\}, \tag{3.16}$$

where $\|.\|_*$ = denotes the trace norm defined as the sum of all the singular values of B. The effect of trace norm regularization is to force B to have a low rank [PTJY09]. The low-rank regularization is a useful technique for obtaining the common low-dimensional subspace for several tasks [HSH16].

4. Multi-task learning with network incorporation: [EMP05] introduced the regularization function

$$J(\beta) = \frac{1}{2}\sum_{s,t=1}^{T} A_{st}\|\beta_s - \beta_k\|^2,$$

where T is the number of tasks and A is the graph adjacency matrix to capture the task similarities. Thus, the optimization problem becomes

$$\min_B \frac{1}{2}\{\|Y - XB\|_F^2 + \lambda J(\beta)\}. \tag{3.17}$$

5. Graph-Guided Fused Lasso (GFLasso) [KSX09]: One potential structure on the response variables that would be useful to capture is some correlation structure that relates correlated responses. The behaviour of the coefficients should reflect this structure, in the sense that the coefficients corresponding to highly correlated responses should be similar. The large limitation of techniques like Lasso is that they must be repeated over every single response variable. In other words, Lasso for multiple correlated responses is equivalent to solving multiple independent regressions for each response. Thus, the information across multiple responses is not integerated into the model such that the estimates reflect the possible relatedeness in the regression coefficients for those correlated responses [KSX09]. In the GFLasso, the correlation structure of tasks is represented as a graph, with each task as a node, and the relationship between two tasks as an edge. This method links several response variables in a single regularized linear regression framework, and jointly analyze them.

The correlation structure over the set of t tasks is presented by an undirected graph $G = (V, E)$, where V represents the set of nodes, each representing one of the t tasks, and E refers to the set of edges. Each edge $e_{k,k'} \in E$ corresponds to an edge from the k-th task to the k'-th task, and $|r_{k,k'}|$ encodes the strength of the correlation between these two nodes. The pairwise Pearson correlation coefficients for each pair of tasks is computed, and then two nodes are linked with an edge only if their correlation coefficient is above a given threshold τ. The authors introduced two variants of the GFLasso: G_cFLasso and G_wFLasso. The first version uses unweighted graph and the second one takes into account the edge weights information in G.

If two tasks are connected with an edge in the graph G, their variation across observations might be explained by the same set of features with similar strength. In G_cFLasso, two regression coefficients $\beta_{.k}$ and $\beta_{.k'}$ for each feature m are fused if tasks k and k' are connected in G. The G_cFLasso solves

$$\hat{B} = \arg\min_{B} \left\{ \frac{1}{2}\|Y - XB\|_F^2 + \lambda_1 \|B\|_1 + \lambda_2 \sum_{(k,k')\in E} \left\|\beta_{.k} - sign(r_{k,k'})\beta_{.k'}\right\|_1 \right\}, \tag{3.18}$$

where λ_1 and λ_2 are regularization parameters, and the last term (the summation term) is called a fusion penalty. A natural extension of G_cFLasso is G_wFLasso wherein the edge weights is taken into account. Also, in G_wFLasso, the amount of correlation controls the amount of fusion. Thus, the complete optimization problem for G_wFLasso is:

$$\begin{aligned} \hat{B} = \arg\min_{B} \Bigg\{ & \frac{1}{2}\|Y - XB\|_F^2 + \lambda_1 \|B\|_1 \\ & + \lambda_2 \sum_{(k,k')\in E} f(r_{kk'}) \left\|\beta_{.k} - sign(r_{k,k'})\beta_{.k'}\right\|_1 \Bigg\}. \end{aligned} \tag{3.19}$$

Possible choices for $f(r_{kk'})$ could be $|r_{k,k'}|$ and $r_{k,k'}^2$. For comparison results in Chapter 4, we used $|r_{k,k'}|$.

For implementation of Multi-task ElasticNet model, we use Scikit-learn [PVG+11]. For other aformentioned multi-task methods, we make use of the RMTL package [CZS18] available for use in the R programming language.

3.5 Chapter Summary

This chapter has introduced Generalized Joint Lasso (GJL) for high-dimensional regression in the group-structured setting that simultaneously selects positively correlated variables and provides subgroup-specific coefficient estimates. The corresponding optimization procedure for solving this model was also introduced. Finally, this chapter has reviewed several multi-task models, which are later used to show the usefulness of multi-task learnign methods in biomedical problems.

Chapter 4

Data Analysis Results

This chapter will discuss a real-world application of the regularized linear regression models, ensemble models, and multi-task regression models in the field of Alzheimer's disease research. It is shown how these previously presented models can be used to predict the progression of the disease. In this setting, these models fit nicely, since 1) the dimensionality of the feature space can be very large relative to the number of observations; and 2) the sparse models might be preferred, since they are more interpretable than other more complicated models. For implementation of these models, we make use of the R programming language [Tea20] and Scikit-learn machine learning library in Python [PVG+11]. The source code for all models is provided in Appendix B.

4.1 Current Statistics of Alzheimer's Disease

The most prevalent type of dementia in adults aged 65 and above is Alzheimer's disease (AD), which predominantly affects memory and cognitive functions. In the United States, it is ranked the fifth leading cause of death for those age 65 and older [Ass17]. AD has brought a major burden to the health care system and economy. According to Alzheimer's Disease International (ADI) Report [Int19], the global prevalence of AD is expected to increase to 152 million by 2050. Moreover, dementia including Alzheimer's disease, is expected to cost more than 1.1 trillion by 2050 for the United States alone [Ass17]. These statistics may seem surprising, but as the population of seniors is increasingly growing, they are projected to increase as time goes on.

AD was first discovered by a German clinical psychiatrist and neuroanatomist, Dr. Alois Alzheimer, in the early 20^{th} century [HN03]. AD is characterized by gradual loss of cognitive functioning, namely thinking, remembering, and reasoning, which impeded the ability of an individual to perform activities of daily living [oA20]. AD is known to progress over time by destroying memory cells in the brain, resulting in loss of memory. Today more than ever it is crucial to understand the stages of the AD in order to

improve the lives of those affected by the disease and establish a clear plan on how its prevelance can be predicted and detected.

4.1.1 Stages of AD

Human brains start shrinking and losing weight as they begin to age. The brain not only shrinks in Alzheimer's patients, but also starts to form knots. These knots are chemical-releasing twisted protein fibers that destroy the nerve cells in the brain. Such changes affect the ability of a person to learn and recall [Naz11]. The disease advances from mild to moderate to severe cognitive impairment and the speed of symptoms vary from individual to individual [Ass17]. While several factors such age, genetics, and diet have been hypothetically linked to the onset of AD, so far, there are no reliable biomarkers used to diagnose the early stages of AD [Arm13]. Scientists systematically seek to validate these hypotheses by testing disease progression, with the goal of improving cognitive function of those affected by the disease and developing successful medications and treatments, at least, to delay the progression of the disease [Ass17, Int19].

Conducting brain scans, such as magnetic resonance imaging (MRI) and positron emission tomography (PET) have already been demonstrated to provide support in the direct observation and inspection of brain abnormalities such as cerebral atrophy [CRS10]. A convenient clinical diagnostic technique is to perform neuropsychological examinations, such as the Mini-Mental State Examination (MMSE), and the Alzheimer's Disease Assessment Scale cognitive total score (ADAS) that can be used to recognize disease-related abnormalities. The reliable link between these clinical scores and AD prognosis has been shown in several studies [DCG$^+$09]. Predicting cognitive performance of patients from neuroimaging feautures is important focuse of the study of AD and is recieving a growing degree of attention recently.

4.1.2 Data

Alzheimer's Disease Neuroimaging Initiative (ADNI)[1] database is a longitudinal study designed to collect demographic, imaging, clinical assessment, and genetic data from participants who are tracked and reassessed over time to follow the pathology of the progression of the disease. Several researchers carried out studies [LCW$^+$19, LGC$^+$17] using the ADNI database to identify regions of interest (ROI) in the brain that should be

[1] http://adni.loni.usc.edu/

revelant to the detection of AD. The first phase of ADNI (ADNI1) started in 2004 with a follow-up period of five years, ADNI-GO phase began in 2009 and followed participants for two years, in 2011 ADNI2 started. Recently, ADNI3, the fourth phase, began in 2016 and includes scientists at 59 research clinics in the United States and Canada [ADN20]. Progression and prediction of AD at multiple time points are more important than ever before. Techniques to describe AD progression can allow clinicians to create new treatments and track their effectiveness. In ADNI, all participants received 1.5-T structural MRI. This dataset encourages scientists to design new strategies for reliably predicting patients' future status. At baseline, patients were classified as either cognitively normal (CN), early mild cognitive impairment (EMCI), late mild cognitive impairment (LMCI), and severe cognitive impairment (dementia or AD). The MMSE score range of $[0, 30]$ is commonly used as a disease progression indicator, where lower scores indicate greater cognitive disability [FFM75]. During the MMSE, a medical expert asks a patient a series of questions and tests designed to assess a variety of daily mental abilities, such as time and place orientation, focus and calculation, immediate and delayed recall of words, and language functions [LCW$^+$19]. The MMSE scores of patients are measured repeatedly at multiple time points. A MMSE score of 20 to 24 suggests EMCI, 13 to 20 suggests LMCI, and less than 12 indicates AD [Ass17]. Table 4.1 summarizes the demographic characteristics of all subjects, including age, gender, and education at baseline.

In this thesis we use the baseline MRI features (see Appendix C for detailed list of predictors) processed by a team from UCSF using FreeSurfer image analysis suite (https://surfer.nmr.mgh.harvard.edu/) and baseline MMSE as inputs and MMSE scores at multiple time points as output variables. The dataset that we use in this work includes CN, LMCI and AD subgroups. For CN and LMCI groups, data is availabe for all time points, but for AD group, no data is available for M36.

Table 4.1: Summary of subjects demographics considered for this study

Subjects	CN	LMCI	AD
Number (F/M)	110/119	141/257	91/100
Age(y, mean ± sd)	75.87 ±5.01	74.73 ± 7.38	75.26 ± 7.46
Education (y, mean ± sd)	16.06 ± 2.85	15.63 ± 3.03	14.69 ± 3.15

4.2 Problem Description

The main goal of this thesis is to accurately predict the progression of Alzheimer's disease at the different stages of the disease. Inferring the trajectories of AD progression over time is important than ever before and is the basis for the implementation of appropriate and prompt therapeutic strategies that may be subject-specific. However, the majority of studies on AD have focused on the prediction of disease progression at a single point in time. These studies ignore the dependency structure that exists between subsequent time points [TAS+18]. Since the patterns of AD progression rely heavily on the time of diagnosis, the demographic information of the patients, and several other features, a single regression model imposed on all of the patients with wide ranges of baseline data may be mis-specified. Fig 4.1 illustrates the change patterns of MMSE scores for several patients in three different study groups (CN, LMCI , and AD) over the three-year time period. As one can see from the figure below, it is obvious that the progression of the disease does not follow a steady pattern, making it difficult to develop an effective model.

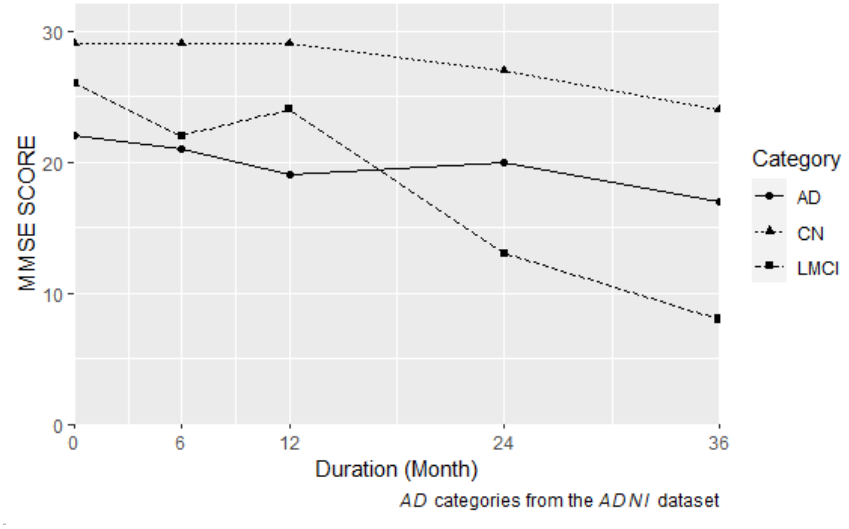

Figure 4.1: The change patterns of MMSE score for different study groups of AD.

Additionally, to handle missing data, we calculate the Euclidean distance

between the observations with missing values and all other observations and then select the first five closest subjects to those with missing values and replace the missing value with the average measurements of those subjects.

4.3 Results and Discussion

In this work, we investigate whether baseline MRI features can be used to predict AD progression measured by MMSE scores. Specifically, we use the baseline MRI features and baseline MMSE score to predict MMSE scores in the next 3 years. We further performed the following preprocessing steps:

- exclude features with more than 15 percent missing entries;
- remove patients without baseline MRI data;
- exclude patients with missing value of MMSE scores;
- complete the missing entries using the technique we discussed in Section 4.2.

Since neurodegeneration in AD occurs years before symptoms appear and clinical intervention is more successful in the early stages of the disease, it is necessary to accurately predict cognitive scores, such as MMSE [ZYLY11]. So, our objective is to predict the MMSE scores at different time points with our method, and compare the relative performance of various approaches. The speed of the progression of AD have a strong dependence on the initial stages of a patient. If so, establishing a single regression model on all of the subjects with different initial patterns may be mis-specified. To address this issue, we adopted a framework which models each subgroup of the AD separately. We evaluate the performance of our proposed method with other regression techniques, including Lasso, Ridge, Elastic net, RF, GB, and JL. The results are presented in Tables 4.2 to 4.7. We indicate each time point in the tables by the duration starting from the baseline. For example, M06 denotes 6 months after the baseline, M12 indicates 12 months after the baseline, and so on. We use MRI at baseline to predict MMSE scores at four time points: M06, M12, M24, M36. 10-fold cross validation is used to evaluate the performance of the models. The data were z-scored before applying the regression methods. For the quantitative performance evaluation, we employed the metric of Root Mean Squared Error (RMSE) and Correlation Coefficient (CC) between the predicted MMSE scores and the target MMSE scores for each regression task. Smaller values of RMSE

and larger values of CC indicate better regression performance. From the experimental results in Tables 4.2, 4.3, and 4.4, we observe the following:

1. For AD group, GJL and JL both demonstrated an improved performance over the other methods, such as Lasso, Ridge, ElNet, RF, GB, in terms of RMSE , while GJL performed the best among all competing methods for M12 and M24. JL performed better than GJL for M06. Overall, The GJL offers substantial gains compared with other methods for this subgroup. The biggest gain with GJL method is for the AD subgroup at M12 and M24.

2. For LMCI group, GJL method outperformed JL methods in all time points. Moreover, GJL demonstrated an improved performance over Lasso, Ridge, ElNet, RF, and GB methods in M12, M24, and M36 time points; however, Lasso and ElNet performed better for M06.

3. It can be seen that a notable difference between JL and GJL is in the LMCI and AD subgroups, or in other words, the largest improvement in prediction performance is in AD and LMCI subgroups.

4. For CN group, GJL performed better than JL in M24 and M36. It can be seen that Lasso, Ridge, ElNet, RF, and GB were best suited to predict CN status than the JL and GJL models for M06, M12.

5. Overall, the GJL approach seems to perform slightly better than the JL approach for M24 and M36 in CN group.

6. There is a decrease in performance compared to the Lasso, Ridge, ElNet, RF, and GB methods for JL and GJL methods in CN subgroup.

Table 4.2: Performance comparison of various methods in terms of RMSE for AD class

Method	M06	M12	M24
Lasso	3.38	4.09	4.68
Ridge	3.68	4.29	4.96
ElNet	3.38	4.11	4.67
RF	3.71	4.42	5.46
GB	3.63	4.32	5.28
JL	2.42	3.01	4.87
GJL	2.59	2.84	3.20

Table 4.3: Performance comparison of various methods in terms of RMSE for CN class

Method	M06	M12	M24	M36
Lasso	1.00	0.91	1.12	1.30
Ridge	1.00	0.90	1.16	1.29
ElNet	0.99	0.90	1.11	1.29
RF	1.00	0.90	1.15	1.31
GB	1.00	0.90	1.14	1.32
JL	1.04	1.29	1.11	1.55
GJL	1.08	1.30	1.09	1.31

Table 4.4: Performance comparison of various methods in terms of RMSE for LMCI class

Method	M06	M12	M24	M36
Lasso	2.10	2.52	3.39	3.82
Ridge	2.21	2.57	3.58	4.01
ElNet	2.10	2.45	3.39	3.87
RF	2.26	2.58	3.62	4.02
GB	2.14	2.48	3.53	3.64
JL	2.41	2.48	3.36	3.47
GJL	2.21	2.36	2.80	3.10

The correlation coefficients between the actual outcomes and the predictions are shown in Tables 4.5, 4.6, and 4.7. For AD class, both JL and GJL demonstrated an improved performance over the other competing methods for M12, while ElNet performed the best among all competing methods for M24. For LMCI group, all other methods performed better than JL and GJL methods for M12, M24, and M36. For M06, both JL and GJL have better CC. For CN group, GJL demonstrated an improved performance over the other competing methods for M06 and M12, while ElNet and Ridge performed better than other methods for M24 and M36, respectively.

Table 4.5: Performance comparison of various methods in terms of CC for AD class

Method	M06	M12	M24
Lasso	0.58	0.55	0.67
Ridge	0.43	0.48	0.65
ElNet	0.57	0.52	0.69
RF	0.44	0.47	0.58
GB	0.44	0.49	0.56
JL	0.52	0.62	0.51
GJL	0.54	0.61	0.60

Table 4.6: Performance comparison of various methods in terms of CC for CN class

Method	M06	M12	M24	M36
Lasso	0.14	0.06	0.21	0.12
Ridge	0.09	0.06	0.02	0.15
ElNet	0.14	0.058	0.26	0.14
RF	0.08	0.12	0.09	0.04
GB	0.09	0.05	0.16	0.00
JL	0.09	0.01	0.26	0.04
GJL	0.16	0.11	0.19	0.03

Table 4.7: Performance comparison of various methods in terms of CC for LMCI class

Method	M06	M12	M24	M36
Lasso	0.52	0.56	0.63	0.64
Ridge	0.44	0.54	0.57	0.62
ElNet	0.52	0.58	0.63	0.63
RF	0.40	0.54	0.55	0.62
GB	0.50	0.55	0.59	0.70
JL	0.59	0.42	0.39	0.44
GJL	0.52	0.50	0.47	0.42

Tables 4.8, 4.9, and 4.10 show the Root Mean Squared Error (RMSE) between the predicted MMSE scores and the actual RMSE scores for each multi-task regression models discussed in chapter 3. We consider the pre-

diction of the MMSE score at each time point as a task. We emphasize that the purpose of the following is to illustrate the usefulness of multi-task learning methods for data analysis and prediction in AD.

Table 4.8: Performance comparison of various methods in terms of RMSE for AD class

Method	M06	M12	M24
MultiElNet	2.99	4.04	4.57
L_{21}	3.64	4.46	5.50
low rank	3.71	4.64	5.47
network structure	4.00	4.85	6.14
GFLasso	1.23	1.12	0.94

Table 4.9: Performance comparison of various methods in terms of RMSE for CN class

Method	M06	M12	M24	M36
MultiElNet	1.03	1.18	1.17	1.26
L_{21}	0.80	1.05	0.96	1.52
low rank	0.79	1.07	0.99	1.52
network structure	0.78	1.06	0.96	1.52
GFLasso	1.29	1.36	1.29	1.51

Table 4.10: Performance comparison of various methods in terms of RMSE for LMCI class

Method	M06	M12	M24	M36
MultiElNet	2.40	2.79	3.77	4.38
L_{21}	2.32	2.31	3.09	3.86
low rank	2.28	2.32	3.11	3.76
network structure	2.75	2.77	3.87	4.46
GFLasso	1.09	1.03	1.04	0.98

From the exprimental results in Tables 4.2, 4.3, 4.4, 4.8, 4.9, and 4.10, we observe the following:

1. In AD subgroup, the GFLasso largely outperformed all other competing multi-task learning analyses. Moreover, GFLasso method im-

proved predictive performance over the independent regression methods (Lasso, Ridge, ElNet, RF, GB, JL, and GJL) in this subgroup.

2. Our method (GJL) outperformed MultiElNet, $L_{2,1}$, low rank, and network structure models in AD class for all time points.

3. There is a decrease in performance compared to all other mutli-task learning methods for GFLasso in CN subgroup.

4. GJL outperformed GFLasso in CN group at all time points, however, it seems that MultiElNet, $L_{2,1}$, low rank, and network structure models performed better than GJL in CN group.

5. In LMCI subgroup, the GFLasso outperformed all other competing multi-task learning analyses. Furthermore, GFLasso method improved predictive performance over the all independent regression methods in this subgroup.

6. GJL method performed better than MultiElNet, $L_{2,1}$, low rank, and network structure models at M06, M24, and M36 time points in LMCI subgroup.

Tables 4.11, 4.12, and 4.13 show the corrleation coefficient (CC) between the predicted MMSE scores and the actual RMSE scores.

Table 4.11: Performance comparison of various methods in terms of CC for AD class

Method	M06	M12	M24
MultiElNet	0.49	0.52	0.60
L_{21}	0.33	0.31	0.50
low rank	0.30	0.28	0.51
network structure	-0.28	-0.36	0.43
GFLasso	0.18	0.14	0.50

Table 4.12: Performance comparison of various methods in terms of CC for CN class

Method	M06	M12	M24	M36
MultiElNet	0.04	-0.03	0.02	0.08
L_{21}	-0.02	0.13	0.14	0.03
low rank	0.02	-0.01	0.05	0.07
network structure	0.16	0.02	-0.00	-0.04
GFLasso	0.03	0.01	0.09	0.07

Table 4.13: Performance comparison of various methods in terms of CC for LMCI class

Method	M06	M12	M24	M36
MultiElNet	0.54	0.53	0.61	0.56
L_{21}	0.55	0.52	0.60	0.55
low rank	0.56	0.52	0.60	0.60
network structure	-0.36	-0.28	0.11	0.47
GFLasso	0.46	0.48	0.49	0.52

It can be seen that MultiElNet method performed the best among all competing multi-task methods in terms of CC in AD subgroup for all time points. For LMCI group, MultiElNet method performed better than all competing methods at M12 and M24. From Tables 4.11 and 4.5 , we observe that all single-task learning methods such as Lasso, Ridge, ElNet, RF, GB, JL, and GJL, peformed better than all multi-task learning methods in terms of CC in AD class .

4.4 Chapter Summary

In this chapter, subjects who underwent MRI, and neuropsychological tests such as MMSE have been selected from ADNI dataset. The MMSE score is the target variable to be predicted with our method. In order to show the impact of GJL, we compared the predictive performance of our proposed model with multiple state-of-the-art regression models. We also include several available well-known multi-task learning methods to show the usefulness of these learning techniques in health-related problem such as predicting the cognitive scores of patients with Alzheimer's disease at multiple future time points. Overall, the results show that there is not one

single method outperforming the others, but instead a group of three top methods including JL, GJL, and GFlasso that would all be a suitable choice.

Chapter 5

Conclusions

Large datasets can be broken into smaller datasets, that may have simillarities but not necessarily identically distributed. This thesis proposed and investigated a high-dimensional regression learning framework over subgroups of observations to support predicting MMSE cognitive scores for subjects with Alzheimer's disease at multiple future time points. For each Alzheimer's disease stages, separate sets of regression models are trained to increase the prediction accuracy. More specifically, We modified several calculations from a previously published approach called "joint lasso" by Dondelinger and Mukherjee [DM18]. We provided a "generalized" version of joint lasso which groups positively correlated variables together and produces a sparse solution for group-structured datasets, and also provides subgroup-specific regression coefficient estimates.

We studied the prediction performance of our proposed method with other linear and nonlinear single-task learning methods such as Ridge, Lasso, Elastic net, Random Forest, Gradient boosting, and joint lasso regression techniques. We also investigated the prediction performance of our proposed method with several multi-task learning models.

It is further noteworthy that the joint lasso and generalzed joint lasso have the advantage of allowing for subgroup-specific sparsity patterns and parameter estimates, which can be of scientific interest. We further note that if groups in dataset only slightly different, establishing one single regression model may be more effective, or in other words, the choice of pooling or not pooling analysis wil be data-dependent.

This work is based on linear models, but methods with nonlinear kernel function can be used to model the output scores as nonlinear functions of input measurements, which can provide additional insights to interpret data. Moreover, future work should provide details about why GJL appears to perform better than the other single-task learning methods in later time points.

Bibliography

[ADN20] ADNI. Alzheimer's disease neuroimaging initiative (http://adni.loni.usc.edu/) [online]. 2020. → pages 28

[Arm13] Richard A. Armstrong. What causes alzheimer's disease? *Folia Neuropathol*, 51(3):169–188, 2013. → pages 27

[Ass17] Alzheimer's Association. Alzheimer's disease facts and figures. *Alzheimer's & Dementia*, 13(4):325–373, 2017. → pages 26, 27, 28

[Bec17] Amir Beck. *First-order methods in optimization*. Society for Industrial and Applied Mathematics, 2017. → pages 1, 2, 4

[Bel59] William A. Belson. Matching and prediction on the principle of biological classification. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 8(2):65–75, 1959. → pages 14

[Bjo96] Ake Bjorck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, 1996. → pages 11

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. → pages 14

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. → pages 4, 5, 6

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. → pages 2, 3

[Car98] Rich Caruana. Multitask learning. *In Learning to Learn. Springer*, pages 95–133, 1998. → pages 22

[CKL+10] Xi Chen, Seyoung Kim, Qihang Lin, Jaime G. Carbonell, and Eric p. Xing. Graph-structured multi-task regression and an

efficient optimization method for general fused lasso. *arXiv*, 1005.3579, 2010. → pages 20, 21

[CRS10] Rudy J. Castellani, Raj K. Rolston, and Mark A. Smith. Alzheimer's disease? *Disease-a-Month*, 56(9):484–546, 2010. → pages 27

[CZS18] Han Cao, Jiayu Zhou, and Emanuel Schwarz. Rmtl: An r library for multi-task learning. *Bioinformatics*, 35(10):1797–1798, 2018. → pages 24

[DCG+09] Simon Duchesne, Anna Caroli, Cristina Geroldi, D. Louis Collins, and Giovanni. B. Frisoni. Relating one-year cognitive change in mild cognitive impairment to baseline mri features. *Neuroimage*, 47(4):1363–1370, 2009. → pages 27

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2001. → pages 6

[DM18] Frank Dondelinger and Sach Mukherjee. The joint lasso: high-dimensional regression for group structured data. *Biostatistics*, 21(2):219–235, 2018. → pages 19, 20, 38

[EMP05] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(1):615–637, 2005. → pages 23

[FFM75] M. F. Folstein, S. E. Folstein, and P.R. McHugh. "mini-mental state" a practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric Research*, 12(3):189–198, 1975. → pages 28

[Fri89] Jerome H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989. → pages 11

[Fri00] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000. → pages 15

[Fu98] Wenjiang J. Fu. Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998. → pages 13

[GBD92] Stuart Geman, Elie Bienenstock, and Rene Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992. → pages 8

[HK70] Arthur E. Hoerl and Robert W. Kannard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. → pages 11

[HN03] Hanns Hippius and Gabriele Neundörfer. The discovery of alzheimer's disease. *Dialogues in Clinical Neuroscience*, 5(1):101–108, 2003. → pages 26

[HSH16] Zhouyuan Huo, Dinggang Shen, and Heng Huang. New multi-task learning model to predict alzheimer's disease cognitive assessment. *Medical Image Computing and Computer-Assisted Intervention*, pages 317–325, 2016. → pages 23

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning- data mining, inference and prediction.* Springer Series in Statistics. Springer, New York, NY, USA, 2009. → pages 11

[Int19] Alzheimer's Disease International. World alzheimer report 2019: Attitudes to dementia. *Alzheimer's Disease International (ADI)*, 2019. → pages 26, 27

[JLL+15] Woncheol Jang, Johan Lim, Nicole A. Lazar, Ji Meng Loh, and Donghyeon Yu. Some properties of generalized fused lasso and its applications to high dimensional data. *Journal of the Korean Statistical Society*, 44(3):352–365, 2015. → pages 18, 19

[JY09] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. *In Proceedings of the 26th annual international conference on machine learning. ACM*, pages 457–464, 2009. → pages 23

[KSX09] Seyoung Kim, Kyung-Ah Sohn, and Eric P. Xing. A multivariate regression approach to association analysis of a quantitative trait network. *Bioinformatics*, 25(12):204–212, 2009. → pages 23

[LCW+19] Xiaoli Liu, Peng Cao, Jianzhong Wang, Jun Kong, and Dazhe Zhao. Fused group lasso regularized multi-task feature learning and its application to the cognitive performance prediction of

alzheimer's disease. *Neuroinformatics*, 17(4):271–294, 2019. → pages 27, 28, 60

[LGC+17] Xiaoli Liu, Andre R. Goncalves, Peng Cao, Dazhe Zhao, and Arindam Banerjee. Modeling alzheimer's disease cognitive scores using multi-task sparse group lasso. *Computerized Medical Imaging and ics*, 66:100–114, 2017. → pages 27

[LJY09] Jun Liu, Shiwang Ji, and Jieping Ye. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. *In Proceedings of the 25th conference on uncertainty in artificial intelligence*, pages 339–348, 2009. → pages 22

[Mor65] J.J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la S. M. F.*, 93(2):273–299, 1965. → pages 4

[Naz11] Linda Nazarko. Understanding dementia: diagnosis and development. *British Journal of Healthcare Assistants*, 5(5):216–220, 2011. → pages 27

[Nes05] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005. → pages 21

[oA20] National Institute of Aging. Alzheimer's disease fact sheet. *Retrieved from Alzheimer's Disease Education and Referral Center (ADEAR) website*, 2020. → pages 26

[PTJY09] Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2009. → pages 23

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. → pages 22, 24, 26

[Roc96] R. Tyrrell Rockafellar. *Convex analysis*. Princeton University Press (Princeton Landmarks in Mathematics and Physics), 1996. → pages 4

[She10] Yiyuan She. Sparse regression with exact clustering. *Electronic Journal of Statistics*, 4:1055–1096, 2010. → pages 18

[TAS+18] Solale Tabarestani, Maryamossadat Aghili, Mehdi Shojaie, Christian Freytes, and Malek Adjouadi. Profile-specific regression model for progression prediction of alzheimer's disease using longitudinal data. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1353–1357, 2018. → pages 29

[Tea20] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. → pages 26

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. → pages 11

[TSR+05] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 67(1):91–108, 2005. → pages 17

[TT11] Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *Annals of Statistics*, 39(3):1335–1371, 2011. → pages 18

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005. → pages 11, 12

[ZH08] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594, 2008. → pages 13

[ZYLY11] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping Ye. A multi-task learning formulation for predicting disease progression. *in Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 814–822, 2011. → pages 30

Appendix

Appendix A

Proximity Operator

Prox operator of the ℓ_1 norm has a closed-form expression.
If $f(x) = \lambda|x|$ where $\lambda > 0$ on $\mathbb{R}$, then

$$\partial f(\mathtt{x}) = \begin{cases} -\lambda & \text{if } \mathtt{x} < 0 \\ [-\lambda, \lambda] & \text{if } \mathtt{x} = 0 \\ \lambda & \text{if } \mathtt{x} > 0 \end{cases}$$

$$prox_f(y) = \operatorname{sgn}(y) \max\{|y| - \lambda, 0\} = \begin{cases} y - \lambda & \text{if } y > \lambda, \\ 0 & \text{if } |y| \leq \lambda, \\ y + \lambda & \text{if } y < -\lambda. \end{cases}$$

This function is called the soft thresholding.
From defintion, we know

$$prox_f(y) = \underset{\mathtt{x} \in \mathbb{E}}{\arg\min} \left\{ \frac{1}{2} \|\mathtt{x} - y\|^2 + f(\mathtt{x}) \right\},$$

and

$$\mathtt{x} = prox_f(y) \leftrightarrow y - \mathtt{x} \in \partial f(\mathtt{x}) \quad (*).$$

There are three cases:
Case 1: $|y| \leq \lambda \longrightarrow y - 0 \in [-\lambda, \lambda] = \partial f(0)$ and so $(*)$ holds.
Case 2: $y > \lambda$, then $y - (y - \lambda) = \lambda$, therefore $(*)$ holds with $x = y - \lambda$.
Case 3: $y < -\lambda$, similar to case2.

Appendix B

R Source Code

The following is the code used for filling NA values.

```
EUC = function( y , X ){

  D = c()

  for(w in 1:dim(X)[1]){

    D[w] = sqrt( sum( ( y - X[w,] )^2 ) )

  }
  return( D )
}

for(j in 1:length(Y)){

  Ys = Y[,j]
  Na = which( is.na( Ys ) )

  for(i in 1:length(Na)){

    D_euc = EUC( X[Na[i],] , X[-Na[i],] )

    Y[Na[i],j] = round( mean( Ys[ order( D_euc ) ][1:3] , na.rm = TRUE ), 2 )

    print( c(j,i) )
  }

}

sum( is.na( Y ))
```

The following code is used for implementing Ridge regression.

```
#Ridge Regression

ridge <- train(MMSE_m06 ~ .,
               train,
               method= "glmnet",
               tuneGrid=expand.grid(alpha=0,
                                    lambda=10^seq(-3, 2, by = .1)),
               trControl=custom)
#Prediction
p1 <- predict(ridge,train)
sqrt(mean((train$MMSE_m06-p1)^2))

p2 <- predict(ridge,test)
sqrt(mean((test$MMSE_m06-p2)^2))



ridge$bestTune$lambda
best_ridge <- which(ridge$results$lambda == ridge$bestTune$lambda)
ridge$results$RMSE[best_ridge]
ridge$results$RMSESD[best_ridge]
```

The following code is used for implementing Lasso regression.

```
#Lasso Regression


lasso <- train(MMSE_m06 ~ .,
               train,
               method= "glmnet",
               tuneGrid=expand.grid(alpha=1,
                                    lambda=10^seq(-3, 2, by = .1)),
               trControl=custom)


#Prediction
p3 <- predict(lasso,train)
sqrt(mean((train$MMSE_m06-p3)^2))

p4 <- predict(lasso,test)
sqrt(mean((test$MMSE_m06-p4)^2))



lasso$bestTune$lambda
best_lasso <- which(lasso$results$lambda == lasso$bestTune$lambda)
lasso$results$RMSE[best_lasso]
lasso$results$RMSESD[best_lasso]
```

The following code is used for implementing Elastic Net regression.

```
#ElNetRegression


ElNet <- train(MMSE_m06 ~ .,
               train,
               method= "glmnet",
               tuneGrid=expand.grid(alpha=seq(0,1,length=10),
                                    lambda=10^seq(-3, 2, by = .1)),
               trControl=custom)


#Prediction
p5 <- predict(ElNet1,train)
sqrt(mean((train$MMSE_m06 -p5)^2))

p6 <- predict(ElNet1,test)
sqrt(mean((test$MMSE_m06 -p6)^2))



ElNet1$bestTune$lambda
best_ElNet1 <- which(ElNet1$results$lambda == ElNet1$bestTune$lambda)
ElNet1$results$RMSE[best_ElNet1]
ElNet1$results$RMSESD[best_ElNet1]
```

The following code is used for implementing Random Forest for regression.

```
#Grid search
control <- trainControl(method = "cv",
                        number = 10,
                        search = "grid")


tunegrid <- expand.grid(.mtry=c(1:15))
rf_grid <- train(data.m06.MMSCORE ~ .,
             train,
             method= "rf",
             tuneGrid=tunegrid,
             trControl=control)

print(rf_grid)
rf_grid$results

summary(rf_grid)
plot(rf_grid)

#Prediction
p1 <- predict(rf_grid,train)
sqrt(mean((train$data.m06.MMSCORE -p1)^2))

p2 <- predict(rf_grid,test)
sqrt(mean((test$data.m06.MMSCORE -p2)^2))
cor(p2,test$data.m06.MMSCORE)
```

The following code is used for implementing Gradient Boost for regression.

```
#Custom Control Parameters
control <- trainControl(method = "cv",
                        number = 10)


tunegrid <- expand.grid(interaction.depth=c(1, 3, 5), n.trees = c(500,1000,1500,2000),
                        shrinkage=c(0.001,0.01,0.1,1),
                        n.minobsinnode=10)
gb_grid <- train(data.m12.MMSCORE ~ .,
                 train,
                 method= "gbm",
                 tuneGrid=tunegrid,
                 trControl=control)

print(gb_grid)
gb_grid$results
gb_grid$besttune

#Prediction
library(Metrics)
p1 <- predict(gb_grid,train)
sqrt(mean((train$data.m12.MMSCORE -p1)^2))
RMSE(train$data.m12.MMSCORE,p1)



p2 <- predict(gb_grid,test)
sqrt(mean((test$data.m12.MMSCORE -p2)^2))
RMSE(test$data.m12.MMSCORE,p2)
cor(p2,test$data.m12.MMSCORE)
```

The following code is used for implementing JL and GJL models.

```
RMSE_TEST = function( X.scale.Tr , Y.scale.Tr , Y.non.scale.Tr , DX.BL.Tr , DATA_TEST , Y.TEST ,
lambda.vals , gamma.vals ){
 NOT_NA = which( is.na( Y.scale.Tr ) == FALSE )
 Xs = as.matrix( X.scale.Tr[ NOT_NA  , ] )
 Ys =   Y.scale.Tr[ NOT_NA ]
 groups = as.factor( DX.BL.Tr[ NOT_NA ] )
 p = dim(Xs)[2]
 k =length( unique( DX.BL.Tr ) )
 RESULT = fusedLassoProximal(
  Xs, Ys, groups = groups  , lambda = lambda.vals , gamma = gamma.vals  ,
  tol = 1e-06 ,  G = matrix( 1 , k ,k ) , intercept = FALSE , scaling = TRUE ,
  num.it = 1000, c.flag= FALSE ,   conserve.memory =p >= 10000 )
 MEAN_Y_non.scale = tapply( Y.non.scale.Tr[NOT_NA] , groups , mean )
 SD_Y_non.scale = tapply( Y.non.scale.Tr[NOT_NA] , groups , sd )
 BETA_ESTIMATE = c()
 for( i in 1:length( DATA_TEST$DX_bl ) ){
  BETA_ESTIMATE = rbind( BETA_ESTIMATE ,  RESULT[, DATA_TEST$DX_bl[i] + 1 ] )
 }
 PRED_MATRIX = cbind( DATA_TEST[,3:322] , BETA_ESTIMATE ,
                SD_Y_non.scale[DATA_TEST$DX_bl + 1] , MEAN_Y_non.scale[DATA_TEST$DX_bl + 1])
```

```
 PREDICT = rowSums( PRED_MATRIX[ , 1:320] * PRED_MATRIX[ , 321:640] ) *

  PRED_MATRIX[ , 641] + PRED_MATRIX[ , 642]

 RMSE = sqrt( mean( ( PREDICT -  Y.TEST )^2 , na.rm = TRUE ) )

 RMSE_g0 = sqrt( mean( ( PREDICT[DATA_TEST$DX_bl==0] -  Y.TEST[DATA_TEST$DX_bl==0] )^2 , na.rm =
TRUE ) )

 RMSE_g1 = sqrt( mean( ( PREDICT[DATA_TEST$DX_bl==1] -  Y.TEST[DATA_TEST$DX_bl==1] )^2 , na.rm =
TRUE ) )

 RMSE_g2 = sqrt( mean( ( PREDICT[DATA_TEST$DX_bl==2] -  Y.TEST[DATA_TEST$DX_bl==2] )^2 , na.rm =
TRUE ) )

 COR = CORR( PREDICT , Y.TEST  )

 COR_g0 = CORR( PREDICT[DATA_TEST$DX_bl==0] , Y.TEST[DATA_TEST$DX_bl==0] )

 COR_g1 = CORR( PREDICT[DATA_TEST$DX_bl==1] , Y.TEST[DATA_TEST$DX_bl==1] )

 COR_g2 = CORR( PREDICT[DATA_TEST$DX_bl==2] , Y.TEST[DATA_TEST$DX_bl==2] )

 return(  list( RMSE = c( RMSE = RMSE , RMSE_group0 = RMSE_g0, RMSE_group1 = RMSE_g1,
RMSE_group2 = RMSE_g2 ) ,

          Correlation = c( Correlation = COR , Cor_group0 = COR_g0 , Cor_group1 = COR_g1 , Cor_group2
= COR_g2 ),

          y = Y.TEST  ,  yh = PREDICT )  )


}

CORR = function( x , y ){

 COV = sum ( ( x - mean(x,na.rm=TRUE) ) * ( y - mean(y ,na.rm=TRUE) ) , na.rm = TRUE )

 VARS = sqrt( sum ( ( x - mean(x,na.rm=TRUE) )^2  , na.rm = TRUE  ) * sum ( ( y - mean(y,na.rm=TRUE)
)^2 , na.rm = TRUE ) )

 if( COV == 0 ){ return( 0 ) }else{ return( COV/VARS ) }

}

DATA_GROUP_SCALE = DATA

G0 = scale( DATA[ DATA$DX_bl == 0 , 3:326 ] )

G1 = scale( DATA[ DATA$DX_bl  == 1 , 3:326 ] )

G2 = scale( DATA[ DATA$DX_bl == 2 , 3:326 ] )
```

```
DATA_GROUP_SCALE[  DATA$DX_bl == 0 , 3:326 ] = G0
DATA_GROUP_SCALE[  DATA$DX_bl == 1 , 3:326 ] = G1
DATA_GROUP_SCALE[  DATA$DX_bl == 2 , 3:326 ] = G2


 TEST_ID = sample( dim(DATA_GROUP_SCALE)[1] , .3*dim(DATA_GROUP_SCALE)[1]  )
 DATA_TRAIN_SCALE = DATA_GROUP_SCALE[-TEST_ID , ]
 DATA_TEST_SCALE =  DATA_GROUP_SCALE[TEST_ID , ]
 DATA_TRAIN_NON_SCALE = DATA[ -TEST_ID , ]
 DATA_TEST_NON_SCALE = DATA[ TEST_ID , ]
 VRIABLE = i
 RESULT_RMSE_TEST =
  RMSE_TEST( X.scale.Tr = DATA_TRAIN_SCALE[,3:322] ,
         Y.scale.Tr = DATA_TRAIN_SCALE[,322+VRIABLE] ,
         Y.non.scale.Tr = DATA_TRAIN_NON_SCALE[,322+VRIABLE] ,
         DX.BL.Tr = DATA_TRAIN_SCALE$DX_bl ,
         DATA_TEST = DATA_TEST_SCALE  ,
         Y.TEST =  DATA_TEST_NON_SCALE[,322+VRIABLE],
         lambda.vals = ... , gamma.vals = ....)
```

The following code is used for implementing multi-task elastic net regression.

```
from sklearn import linear_model

import pandas as pd

import numpy as np

import random

DATA_LMCI = pd.read_csv("C:\\Users\\marjan02\\Desktop\\Marjan\\ADNI\\Thesis\\Final Final ")


type( DATA_LMCI )

# scale

X_scale = ( DATA_LMCI.iloc[0:282,0:320] - DATA_LMCI.iloc[0:282,0:320].mean() ) /
DATA_LMCI.iloc[0:282,0:320].std()

Y =  DATA_LMCI.iloc[0:282,320:324]

from sklearn.model_selection import train_test_split

X_scale_Train, X_scale_Test , Y_Train , Y_Test = train_test_split( X_scale, Y, test_size=0.30,
random_state=88)

RATIO = np.arange(.01,.99,.05)

ALPHA = np.arange(.5,5,.1)

MODEL_LMCI_CV= linear_model.MultiTaskElasticNetCV(cv=10 ,fit_intercept=True, normalize=False ,
random_state= 123,l1_ratio=RATIO, eps=0.001, n_alphas=100, alphas=ALPHA )

MODEL_LMCI_CV.fit(X_scale_Train,Y_Train)

# best l1_ratio obtained by cross-validation.

MODEL_LMCI_CV.l1_ratio_

BEST_RATIO = MODEL_LMCI_CV.l1_ratio_

BEST_ALPHA = MODEL_LMCI_CV.alpha_

MODEL_LMCI = linear_model.MultiTaskElasticNet( fit_intercept=True, normalize=False , random_state=
123 ,l1_ratio=BEST_RATIO , alpha=BEST_ALPHA )

MODEL_LMCI.fit(X_scale_Train,Y_Train)

MODEL_LMCI.score(X_scale_Train,Y_Train)

PRED = MODEL_LMCI.predict(X_scale_Test)

MSE = ( ( PRED - Y_Test )**2 ).mean(axis=0)

RMSE = MSE**.5
```

The following code is used for implementing Graph-Guided Fused Lasso with the gflasso R package.

```
cv2 <- cv_gflasso(X= x.train,Y= y.train,R=corr,
                  additionalOpts = list(delta_conv=1e-5,iter_max=1e5),
                  k=10 )

cv2$optimal
cv2$mean
cv2$SE
cv_plot_gflasso(cv2)



gfmod1 <- gflasso(X= x.train,
                  Y= y.train,
                  R= corr,
                  opts = list(lambda=cv2$optimal$lambda,gamma=cv2$optimal$gamma,delta_conv=1e-5,iter_max=1e5))

predy1 <- predict_gflasso(gfmod1,x.train)
predy2 <- predict_gflasso(gfmod1,x.test)

library(caret)
RMSE(y.train[1],predy1[1])
RMSE(y.train[2],predy1[2])
RMSE(y.train[3],predy1[3])

RMSE(y.test[1],predy2[1])
RMSE(y.test[2],predy2[2])
RMSE(y.test[3],predy2[3])


colnames(gfmod1$B) <- colnames(y.train)
pheatmap(gfmod1$B, annotation_row = data.frame("MRI" = colnames(x.train),
                                               row.names = rownames(gfmod1$B)), show_rownames = F)
```

The following code is used for implementing multi-task learning with $\ell_{2,1}$ norm with the RMTL R package.

```
#perform the cross validation
library(RMTL)
cvfitr <- cvMTL(X,Y, type="Regression", Regularization="L21",
          Lam1_seq=10^seq(2,-3, -0.1),
          Lam2=0,
          opts=list(init=0,  tol=10^-6, maxIter=1500),
          nfolds=10 ,stratify=FALSE, parallel=FALSE)
#the output lam1 value with minimum CV error
print (paste0("estimated lam1: ", cvfitr$Lam1.min))
#plot CV errors across lam1 sequence in the log space
plot(cvfitr)
#train a MTL model
model<-MTL(X, Y, type="Regression", Regularization="L21",
       Lam1=cvfitr$Lam1.min,Lam2 = 0,
       opts=list(init=0,  tol=10^-6,maxIter=1500),
       Lam1_seq=cvfitr$Lam1_seq)
# predict
str(predict(model, X1)) # for regression
PRED <- predict(model, X1)
sqrt(mean((as.vector(PRED[[1]])- as.vector(Y1[[1]]))^2))
sqrt(mean((as.vector(PRED[[2]])- as.vector(Y1[[2]]))^2))
sqrt(mean((as.vector(PRED[[3]])- as.vector(Y1[[3]]))^2))
sqrt(mean((as.vector(PRED[[4]])- as.vector(Y1[[4]]))^2
cor(PRED[[1]],Y1[[1]])  #
cor(as.vector(PRED[[1]]),as.vector(Y1[[1]]))
cor(PRED[[2]],Y1[[2]])  #
cor(as.vector(PRED[[2]]),as.vector(Y1[[2]]))
cor(PRED[[3]],Y1[[3]])  #
cor(as.vector(PRED[[3]]),as.vector(Y1[[3]]))
```

The following code is used for implementing low rank multi-task learning with the RMTL R package.

```
# MTL with low-rank structure
cvfit3<-cvMTL(X, Y, type="Regression", Regularization="Trace",
        Lam1_seq=10^seq(2,-3, -0.1),
        Lam2=0,
        opts=list(init=0,  tol=10^-6, maxIter=1500),
        nfolds=10 ,stratify=FALSE, parallel=FALSE)
#Train
m3=MTL(X, Y, type="Regression", Regularization="Trace",
    Lam1=cvfit3$Lam1.min,
    Lam1_seq=cvfit3$Lam1_seq,Lam2= 0,
    opts=list(init=0,  tol=10^-6,maxIter=1500))
# predict
str(predict(m3, X1)) # for regression
PRED3 <- predict(m3, X1)
sqrt(mean((as.vector(PRED3[[1]])- as.vector(Y1[[1]]))^2))
sqrt(mean((as.vector(PRED3[[2]])- as.vector(Y1[[2]]))^2))
sqrt(mean((as.vector(PRED3[[3]])- as.vector(Y1[[3]]))^2))
sqrt(mean((as.vector(PRED3[[4]])- as.vector(Y1[[4]]))^2))
cor(PRED3[[1]],Y1[[1]])
cor(as.vector(PRED3[[1]]),as.vector(Y1[[1]]))
cor(PRED3[[2]],Y1[[2]])
cor(as.vector(PRED3[[2]]),as.vector(Y1[[2]]))
cor(PRED3[[3]],Y1[[3]])
cor(as.vector(PRED3[[3]]),as.vector(Y1[[3]]))
cor(PRED3[[4]],Y1[[4]]) #
cor(as.vector(PRED3[[4]]),as.vector(Y1[[4]]))
```

The following code is used for implementing multi-task learning with network structure with the RMTL R package.

```
#MTL with network structure
library(corrplot)
corrplot(cor(x.train[,1:20]))
cr <- cor(y.train)
cor.test(CN_RMTL$m06.MMSCORE,CN_RMTL$m12.MMSCORE)
corrplot(cr)
corrplot(cr,method = "pie")
corrplot(cr,method = "color")
corrplot(cr,method = "number")
corrplot(cr,type = "lower")
cvfit4<-cvMTL(X, Y, type="Regression", Regularization="Graph", G=cr,
        Lam1_seq=10^seq(2,-3, -0.1),
        Lam2=0,
        opts=list(init=0,  tol=10^-6, maxIter=1500),
        nfolds=10 ,stratify=FALSE, parallel=FALSE)
#Train
m4=MTL(X, Y, type="Regression", Regularization="Graph",
    Lam1=cvfit4$Lam1.min, Lam1_seq=cvfit4$Lam1_seq, G=cr,
    Lam2= 0,
    opts=list(init=0,  tol=10^-6,maxIter=1500))
# predict
str(predict(m4, X1)) # for regression
PRED4 <- predict(m4, X1)
sqrt(mean((as.vector(PRED4[[1]])- as.vector(Y1[[1]]))^2))
sqrt(mean((as.vector(PRED4[[2]])- as.vector(Y1[[2]]))^2))
sqrt(mean((as.vector(PRED4[[3]])- as.vector(Y1[[3]]))^2))
sqrt(mean((as.vector(PRED4[[4]])- as.vector(Y1[[4]]))^2))
cor(PRED4[[1]],Y1[[1]])  #
cor(as.vector(PRED4[[1]]),as.vector(Y1[[1]]))
```

Appendix C

List of Variables

The names of predictors are listed in Tables C.1, C.2, and C.3. TA, TS, SA, and CV stand for thickness average, standard deviation of thickness, surface area, and volume of the cortical and subcortical regions of the brain, respectively. Laterality denotes the various types of features measured for L (left hemisphere), R (right hemisphere), and Bilateral (whole hemisphere) [LCW$^+$19]. For more details on how the FreeSurfer is being used to extarct features of MRI scans see [LCW$^+$19]. The MMSE score at baseline, the cortical thickness average (TA) of left Isthmus Cingulate, and the volume of right Inferior Parietal play important roles in predicting MMSE at all time points.

Table C.1: List of variables

Name	Type	Laterality
Banks superior temporal sulcus	CV, SA, TA, TS	L, R
Caudal anterior cingulate cortex	CV, SA, TA, TS	L, R
Caudal middle frontal gyrus	CV, SA, TA, TS	L, R
Cuneus cortex	CV, SA, TA, TS	L, R
Entorhinal cortex	CV, SA, TA, TS	L, R
Frontal pole	CV, SA, TA, TS	L, R
Fusiform gyrus	CV, SA, TA, TS	L, R
Inferior parietal cortex	CV, SA, TA, TS	L, R
Inferior temporal gyrus	CV, SA, TA, TS	L, R
Insula	CV, SA, TA, TS	L, R
IsthmusCingulate	CV, SA, TA, TS	L, R
Lateral occipital cortex	CV, SA, TA, TS	L, R
Lateral orbital frontal cortex	CV, SA, TA, TS	L, R
Lingual gyrus	CV, SA, TA, TS	L, R
Medial orbital frontal cortex	CV, SA, TA, TS	L, R
Middle temporal gyru	CV, SA, TA, TS	L, R
Paracentral lobule	CV, SA, TA, TS	L, R

Table C.2: List of variables

Name	Type	Laterality
Parahippocampal gyrus	CV, SA, TA, TS	L, R
Pars opercularis	CV, SA, TA, TS	L, R
Pars orbitalis	CV, SA, TA, TS	L, R
Pars triangularis	CV, SA, TA, TS	L, R
Pericalcarine cortex	CV, SA, TA, TS	L, R
Postcentral gyrus	CV, SA, TA, TS	L, R
Posterior cingulate cortex	CV, SA, TA, TS	L, R
Precentral gyrus	CV, SA, TA, TS	L, R
Precuneus cortex	CV, SA, TA, TS	L, R
Rostral anterior cingulate cortex	CV, SA, TA, TS	L, R
Rostral middle frontal gyrus	CV, SA, TA, TS	L, R
Superior frontal gyrus	CV, SA, TA, TS	L, R
Superior parietal cortex	CV, SA, TA, TS	L, R
Superior temporal gyrus	CV, SA, TA, TS	L, R
Supramarginal gyrus	CV, SA, TA, TS	L, R
Temporal pole	CV, SA, TA, TS	L, R
Transverse temporal cortex	CV, SA, TA, TS	L, R
Hemisphere	SA	L, R
Total intracranial volume	CV	Bilateral
Accumbens area	SV	L, R
Amygdala	SV	L, R
Caudate	SV	L, R
Cerebellum cortex	SV	L, R
Cerebellum white matter	SV	L, R
Cerebral cortex	SV	L, R
Cerebral white matter	SV	L, R
Choroid plexus	SV	L, R
Hippocampus	SV	L, R
Inferior lateral ventricle	SV	L, R
Lateral ventricle	SV	L, R
Pallidum	SV	L, R
Putamen	SV	L, R
Thalamus	SV	L, R
Ventricle diencephalon	SV	L, R
Vessel	SV	L, R

Table C.3: List of variables

Name	Type	Laterality
Brain stem	SV	Bilateral
Corpus callosum anterior	SV	Bilateral
Corpus callosum central	SV	Bilateral
Corpus callosum middle anterior	SV	Bilateral
Corpus callosum middle posterior	SV	Bilateral
Corpus callosum posterior	SV	Bilateral
Cerebrospinal fluid	SV	Bilateral
Fourth ventricle	SV	Bilateral
Non white matter hypointensities	SV	Bilateral
Optic chiasm	SV	Bilateral
Third ventricle	SV	Bilateral
White matter hypointensities	SV	Bilateral

.

Appendix D

Degrees of freedom

Tables D.1, D.2, and D.3 present the number of variables retained for each sparse method in LMCI, AD, and CN subgroups.

Table D.1: Degrees of freedom for LMCI class

Method	Lasso	ElNet	JL	GJL
df M06	7	19	24	167
df M12	11	11	22	168
df M24	9	11	44	44
df M36	19	21	39	43

Table D.2: Degrees of freedom for AD class

Method	Lasso	ElNet	JL	GJL
df M06	16	117	26	166
df M12	16	111	29	168
df M24	18	150	44	50

Table D.3: Degrees of freedom for CN class

Method	Lasso	ElNet	JL	GJL
df M06	1	149	13	167
df M12	2	1	21	168
df M24	11	8	44	54
df M36	17	34	39	69