TOPICS IN HIGH-DIMENSIONAL STATISTICAL LEARNING

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR
COMPUTATIONAL AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Xiaotong Suo
June 2018

Abstract

Our research explores three topics in high-dimensional statistical learning. First, we consider a regression scenario where it is natural to impose an order constraint on the coefficients. We propose an order-constrained version of ℓ_1-regularized regression for this problem, and show how to solve it efficiently using the well-known Pool Adjacent Violators Algorithm as its proximal operator. We illustrate this idea on real and simulated data. We then consider regression scenarios where it is natural to allow coefficients to vary as smooth functions of other variables. We propose two constrained versions for this problem and show how to solve them efficiently. Last, we study canonical correlation analysis (CCA) in high-dimensional settings and propose a sparse CCA framework, and provide two efficient algorithms. We discuss links between CCA and linear discriminant analysis (LDA). We demonstrate its use on real and simulated data.

Acknowledgments

This work would not have been possible without help of many people. I would like to thank

- Robert Tibshirani and Michael Saunders for being my wonderful advisors and role models. I am very appreciative for their countless support, encouragement, and guidance.
- Jonathan Taylor for always inspiring me to view problems in different ways. I am also appreciative of him being a reader for my dissertation.
- Balasubramanian Narasimhan for being very helpful on the package development and serving on my committee.
- Margot Gerritsen for generously supporting me financially during my last year at Stanford and being an amazing director for ICME.
- Victor Minden and Bradley Nelson for being wonderful collaborators.
- Friends and colleagues for being supportive and making my years at Stanford fun and enjoyable.
- My parents for their unconditional love, their dedication and their support.
- My fiancé, Brad, for being not only a incredible supporter but also a true believer in me. I am grateful for his encouragement, companionship, and making me the luckiest person that I know.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the past few decades, massive datasets have become available and the term ‘big data’ appears in almost every field. Classical statistics used to face only few observations and features, and often cannot be applied directly in high-dimensional settings, where the number of features often exceeds the number of observations. Many recent advances in statistics have been developed in high-dimensional settings. We briefly review some relevant techniques in the literature and introduce our contributions.

1.1 Supervised learning in high dimensions

Suppose that we observed data (x_i, y_i) for $i = 1, 2, \dots, n$, where n is the number of observations, $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ is a vector containing p real-valued feature measurements, and y_i is a real-valued response value. We consider the usual linear regression framework

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i \tag{1.1}$$

with $\mathrm{E}(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = \sigma^2$. Ordinary least squares regression finds the solution to (1.1) by minimizing the sum of squared errors:

$$\underset{\beta_0, \beta}{\text{minimize}} \quad \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2. \tag{1.2}$$

To simplify notation, we rewrite (1.2) in matrix notation as

$$\underset{\beta\in\mathbb{R}^p}{\text{minimize}} \quad \|y - X\beta\|_2^2, \tag{1.3}$$

where the first column of X is a vector of 1s. Solutions to (1.3) satisfy $X^T X\hat{\beta} = X^T y$. In high-dimensional settings, there are two challenges for this model: *a*) if $p > n$, $X^T X$ is singular and $\hat{\beta}$ cannot be calculated. *b*) $\hat{\beta}$ is not sparse, and when p is large, the fitted model cannot be easily interpreted. Many methods have been developed to solve these two challenges. One well studied approach for extending (1.3) to high-dimensional settings is to add a penalty term involving the parameters β. For example, ridge regression or ℓ_2-regularized regression [29] chooses β to solve

$$\underset{\beta}{\text{minimize}} \quad \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \tag{1.4}$$

for some $\lambda > 0$. The unique solution to (1.4) satisfies $(X^T X + \lambda I)\hat{\beta} = X^T y$. The penalty on the elements of β regularizes the sample covariance matrix $X^T X$. However, this approach does not guarantee the solution $\hat{\beta}$ to be sparse. The lasso or ℓ_1-regularized regression [51] chooses the parameters β to solve

$$\underset{\beta}{\text{minimize}} \quad \|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \tag{1.5}$$

where $\lambda > 0$ is a given tuning parameter. This problem yields sparse solutions $\hat{\beta}$ for sufficiently large values of λ. Therefore, lasso automatically performs feature selection in high-dimensional settings. Furthermore, the problem is convex and many efficient solvers have been developed; for instance, `glmnet` in R [22]. Many other penalties on the elements of β have been considered in the literature, and to list a few, see Jacob et al. [35], Simon et al. [46], Tibshirani et al. [52].

In this thesis, we consider two extensions to linear regression model. We call the resulting procedures **ordered lasso** and **time varying lasso**. In the ordered lasso chapter, we consider regression scenarios where it is natural to impose an order constraint on the coefficients. We propose an order-constrained version of ℓ_1-regularized regression for this problem, and show how to solve it efficiently using the well known Pool Adjacent Violators Algorithm as its proximal operator. The main application of this idea is to time-lagged regression, where we predict an outcome at time t from features at the previous K time points. In this setting it is natural to assume that the coefficients decay as we move farther away from t, and hence the order constraint is reasonable. Potential application areas include financial time series and prediction of dynamic patient outcomes based on clinical measurements. We illustrate this idea on real and simulated data.

We then introduce a more relaxed model in chapter 3 and consider regression scenarios where it is natural to allow coefficients to vary as smooth functions of other variables. We propose two constrained versions for this problem and show how to solve them efficiently.

1.2 Unsupervised learning in high dimensions

Unsupervised learning is commonly viewed as learning without ground truth or a response. Typically, one has a set of n observations $X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}$, and the goal is to learn important properties of X without any response y. In this thesis, we consider having two sets of n observations $X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}$ and $Y = \begin{pmatrix} y_1^T \\ \vdots \\ y_n^T \end{pmatrix}$ and study canonical correlation between these two matrices. Canonical correlation analysis (CCA) was proposed by Hotelling [31] to measure linear relationships between two multidimensional variables. In high-dimensional settings, the classical canonical correlation analysis breaks down. We propose a sparse canonical correlation analysis by adding ℓ_1 constraints on the canonical vectors and show how to solve the resulting optimization problem efficiently using the linearized alternating direction method of multipliers (ADMM) and using TFOCS as a black box. We demonstrate its use on real and simulated data. While CCA is often used as an unsupervised tool, we also discuss links between CCA with Fisher's linear discriminant analysis (LDA) [21], a supervised learning tool.

1.3 Numerical optimization algorithms

Efficient optimization algorithms play important roles in the era of big data. We briefly review some numerical optimization algorithms used in this dissertation.

1.3.1 Proximal gradient descent

Suppose that we want to solve

$$\underset{x}{\text{minimize}} \quad f(x) = g(x) + h(x), \tag{1.6}$$

where g is convex and differentiable, and h is convex but not differentiable. Gradient descent method cannot be directly applied, but proximal gradient methods are often used to address this issue. The proximal mapping is defined as

$$\mathbf{prox}_t(x) = \underset{z}{\text{argmin}} \quad \frac{1}{2t}\|x - z\|_2^2 + h(z),$$

and the proximal gradient descent algorithm is as follows:

- Initialize $x^{(0)}$.
- Repeat for $k = 1, 2, \ldots$:

  $$x^{(k)} = \mathbf{prox}_{t_k}(x^{(k-1)}) - t_k \nabla g(x^{(k-1)}), \tag{1.7}$$

  where $t_k > 0$ is a step-size.

Note that the update for $x^{(k)}$ in (1.7) can be rewritten as

$$x^{(k)} = x^{(k-1)} - t_k G_{t_k}(x^{(k-1)}),$$

where G_t is called the generalized gradient of f:

$$G_t(x) = \frac{x - \mathbf{prox}_t(x - t\nabla g(x))}{t}.$$

If g and f satisfy the following assumptions:

- g is convex, differentiable, $\mathrm{dom}(g) = \mathbb{R}^n$ and ∇g is Lipschitz continuous with constant $L > 0$,
- h is convex, $\mathbf{prox}_t(x) = \mathrm{argmin}_z \frac{1}{2t}\|x - z\|_2^2 + h(z)$ can be evaluated,

proximal gradient descent with fixed step size $t \leq \frac{1}{L}$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}.$$

Therefore, proximal gradient descent can obtain a convergence rate of $O(\frac{1}{k})$. Similar to gradient descent, we can perform proximal gradient descent with backtracking line search. At each iteration, we fix a parameter $0 < \beta < 1$, and start with $t = 1$. If $g(x - tG_t(x)) \geq g(x) - t\nabla g(x)^T G_t(x) + \frac{t}{2}\|G_t(x)\|_2^2$, we shrink t to βt; otherwise, we perform the normal proximal gradient update.

1.3.2 Accelerated proximal gradient method

The optimal convergence rate $O(\frac{1}{k^2})$ can be achieved using acceleration methods, see [6], [9], [55]. There are many versions of acceleration. We present one of the choices used in this dissertation:

- Initialize $x^{(0)}$ and let $x^{(-1)} = x^{(0)}$.
- Repeat for $k = 1, 2, \ldots$

$$\begin{aligned} v &= x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}), \\ x^{(k)} &= \mathbf{prox}_{t_k}(v - t_k \nabla g(v)). \end{aligned}$$

With the assumptions above, the accelerated proximal gradient method with fixed step $t \leq \frac{1}{L}$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{2\|x^{(0)} - x^*\|_2^2}{t(k+1)^2}.$$

1.3.3 Alternating direction method of multipliers

The alternating direction method of multipliers (ADMM) [23] solves problems of the form

$$\begin{array}{ll} \underset{x,z}{\text{minimize}} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array} \tag{1.8}$$

where $x \in \mathbb{R}^n, z \in \mathbb{R}^n$, $A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}$, $c \in \mathbb{R}^p$, $f : \mathbb{R}^n \to \mathbb{R} \cup \{\mathbb{R}, +\infty\}$, and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$. In order to guarantee convergence, the usual assumptions for f and g are closed, proper and convex [10]. The augmented Lagrangian is

$$L_\rho(x, z, u) = f(x) + g(z) + u^T(Ax + Bz - c) + \rho/2\|Ax + Bz - c\|_2^2.$$

The generic ADMM algorithm is

- Repeat for $k = 1, 2, 3, \dots,$

$$\begin{aligned} x^{(k)} &= \underset{x}{\text{argmin}}\, L_\rho(x, z^{(k-1)}, u^{(k-1)}), \\ z^{(k)} &= \underset{z}{\text{argmin}}\, L_\rho(x^{(k)}, z, u^{(k-1)}), \\ u^{(k)} &= u^{(k)} + \rho(Ax^{(k)} + Bz^{(k)} - c). \end{aligned}$$

If $B = -I$ and $c = 0$, (1.8) becomes

$$\begin{array}{ll} \underset{x,z}{\text{minimize}} & f(x) + g(z) \\ \text{subject to} & Ax = z, \end{array} \tag{1.9}$$

and the augmented Lagrangian is

$$L_\rho(x, z, u) = f(x) + g(z) + u^T(Ax - z) + \frac{\rho}{2}\|Ax - z\|_2^2.$$

In linearized ADMM, the last term of the augmented Lagrangian

$$\begin{aligned}\frac{\rho}{2}\|Ax - z\|_2^2 &= \frac{\rho}{2}(A^T Ax)^T x - \rho(Az)^T x + \frac{\rho}{2} z^T z \\ &= \frac{\rho}{2}(A^T Ax - 2Az)^T x + \frac{\rho}{2} z^T z\end{aligned}$$

is replaced by $\rho(A^T Ax^{(k)} - Az^{(k)})^T x + \mu/2\|x - x^{(k)}\|_2^2$, where $0 < \mu \leq \lambda/\|A\|_2^2$. It can be viewed as linearizing the quadratic term $x^T A^T Ax$ and adding a new quadratic constraint to update x. The generic algorithm for linearized ADMM is:

- Repeat for $k = 1, 2, \ldots$

$$\begin{aligned}x^{(k)} &= \mathbf{prox}_{\mu f}(x^{(k-1)} - \frac{\mu}{\lambda}A^T(Ax^{(k-1)} - z^{(k-1)} + u^{(k-1)})), \\ z^{(k)} &= \mathbf{prox}_{\lambda g}(Ax^k + u^{k-1}), \\ u^{(k)} &= u^{k-1} + Ax^{(k+1)} - z^{(k+1)}.\end{aligned}$$

Under certain assumptions, the ADMM iterates satisfy residual convergence, objective convergence and dual variable convergence. Details can be seen in [10]. The convergence rate is not currently known but it roughly converges at a linear rate and behaves like first-order methods.

1.3.4 Biconvexity

We briefly review some concepts in biconvex optimization problems. A detailed survey can been seen in [25]. The set $B \subset X \times Y$ is a biconvex set on $X \times Y$ if B_x is convex for every $x \in X$ and B_y is convex for every $y \in Y$, where $B_x = \{y \in Y : (x, y) \in B\}$ and $B_y = \{x \in X : (x, y) \in B\}$. A function $f : B \to R$ on a biconvex set $B \in X \times Y$ is called a biconvex function on B, if $f_x(\bullet) := f(x, \bullet) : B_x \to \mathbb{R}$ is a convex function on B_x for every fixed $x \in X$, and $f_y(\bullet) := f(\bullet, y) : B_y \to \mathbb{R}$ is a convex function on B_y for every fixed $y \in Y$. We call an optimization of the form

$$\underset{x,y}{\text{minimize}} \quad f(x, y) : (x, y) \in B \tag{1.10}$$

biconvex if the set B is biconvex on $X \times Y$ and the objective function f is biconvex on B. One approach to solving biconvex problems is to use an alternating minimization method:

- Repeat until convergence:
    1. Fix x and solve for y.
    2. Fix y and solver for x.

Each subproblem is a convex problem and thus a global solution of each subproblem can be achieved. However, since the biconvex problem is nonconvex, a global solution of the problem cannot be guaranteed in general.

Chapter 2

An ordered lasso

In this chapter we add an additional order constraint on the coefficients, and we call the resulting problem the *ordered lasso*. We derive an efficient algorithm for solving it. The main application of this idea is to time-lagged regression, where we predict an outcome at time t from features at the previous K time points. In this case, it is natural to assume that the coefficients decay as we move farther away from t so that the order (monotonicity) constraint is reasonable. For example, in a model for estimating fertility rate at t as a function of personal exemption, a reasonable assumption is that personal exemption at $t, t-1, \ldots, t-K$ all have some effect on fertility rate at t. It is also reasonable to assume that personal exemption at t has greater impact on fertility rate at t than personal exemption at previous time points [61]. Moreover, directly from the monotonicity constraint, a key feature of our procedure is that it *automatically* determines the most suitable value of K for each predictor.

The chapter is organized as follows. Section 2.1 contains motivations and algorithms for solving the ordered lasso and the *strongly ordered lasso* (which enforces monotonicity in absolute value), as well as results comparing the ordered and the standard lasso on simulated data. Section 2.2 contains detailed algorithms for applying the ordered lasso and the strongly ordered lasso to time-lagged regression. We demonstrate the use of such algorithms on real and simulated data in Sections 2.2.4 and 2.2.5. We also apply this framework to auto-regressive (AR) time series and compare its performance with the traditional method for fitting the AR model using least squares with the Akaike information criterion and Bayesian information criterion, and with the lasso procedure for fitting the AR model [39]. Section

2.4 generalizes the ordered lasso and the strongly ordered lasso to the logistic regression model. Section 2.5 contains some discussion and directions for future work.

2.1 Lasso with an order constraint

2.1.1 The ordered lasso

We consider the lasso problem with an additional monotonicity constraint:

$$\begin{aligned} &\underset{\beta_0,\beta}{\text{minimize}} && \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \\ &\text{subject to} && |\beta_1| \geq |\beta_2| \geq \ldots \geq |\beta_p|. \end{aligned} \tag{2.1}$$

This setup makes sense in problems where some natural order exists among the features. However, the problem is not convex. Hence we modify the approach by writing each $\beta_j = \beta_j^+ - \beta_j^-$ with $\beta_j^+, \beta_j^- \geq 0$ and pose the problem

$$\begin{aligned} &\underset{\beta_0,\beta^+,\beta^-}{\text{minimize}} && \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}(\beta_j^+ - \beta_j^-))^2 + \lambda \sum_{j=1}^{p} (\beta_j^+ + \beta_j^-) \\ &\text{subject to} && \beta_1^+ \geq \beta_2^+ \geq \ldots \geq \beta_p^+ \geq 0, \quad \beta_1^- \geq \beta_2^- \geq \ldots \geq \beta_p^- \geq 0. \end{aligned} \tag{2.2}$$

The penalty term encourages sparsity in β_j^+ and β_j^-. The use of positive and negative components (rather than absolute values) makes this a convex problem. Its solution typically has one or both of each pair $(\hat{\beta}_j^+, \hat{\beta}_j^-)$ equal to zero, in which case $|\hat{\beta}_j| = \hat{\beta}_j^+ + \hat{\beta}_j^-$ and the solutions $|\hat{\beta}_j|$ are monotone non-increasing in j. However, this need not be the case, as it is possible for both $\hat{\beta}_j^+$ and $\hat{\beta}_j^-$ to be positive and the $|\hat{\beta}_j|$ to have some non-monotonicity. In other words, the constraints strongly encourage, but don't require, that the solutions be monotone in absolute value. A similar approach was used in the interaction models of Bien et al. [8]. This problem can be solved by a standard quadratic programming algorithm, and this works well for small problems. For larger problems there is an efficient first-order generalized gradient algorithm that uses the Pool Adjacent Violators Algorithm (PAVA) for isotonic regression as its proximal operator (for example, see de Leeuw et al. [18]). We describe details of the algorithm in the next subsection.

2.1.2 Algorithmic details of the ordered lasso

We assume that the predictors and outcome are centered so that the intercept has the solution $\hat{\beta}_0 = 0$. For illustrative purposes, we write our data in matrix form. Let $\mathbf{X}$ be the $N \times p$ data matrix and $\mathbf{y}$ be the vector of length N containing the response value for each observation. We first consider the problem

$$\begin{aligned} &\underset{\beta}{\text{minimize}} \quad \frac{1}{2}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda \sum_{j=1}^{p} \beta_j, \\ &\text{subject to} \quad \beta_1 \geq \beta_2 \geq \ldots \geq \beta_p \geq 0. \end{aligned} \tag{2.3}$$

We let $h(\beta) = \lambda \sum_{j=1}^{p} \beta_j + \mathbb{I}_C(\beta)$, where $\mathbb{I}$ is an indicator function and C is the convex set given by $\{\beta \in \mathbb{R}^p \mid \beta_1 \geq \beta_2 \geq \ldots \geq \beta_p \geq 0\}$. $\mathbb{I}_C(\beta)$ is equal to 0 if β is in the convex set C and infinity otherwise. We want to calculate the proximal mapping of $h(\beta)$:

$$\mathbf{prox}_h(\beta) = \underset{\mathbf{u}}{\text{argmin}} \quad \lambda \sum_{j=1}^{p} u_j + \mathbb{I}_C(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \beta\|^2. \tag{2.4}$$

There is an elegant way to obtain this proximal mapping. We first consider solving the problem

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{2}\sum_{j=1}^{n}(y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n} \theta_i \\ &\text{subject to} \quad \theta_1 \geq \theta_2 \geq \ldots \geq \theta_n \geq 0. \end{aligned} \tag{2.5}$$

The solution can be obtained from an isotonic regression using the well-known Pool Adjacent Violators Algorithm [4]. In particular, if $\{\hat{\theta}_i\} = \{\hat{y}_i^\lambda\}$ is the solution to the isotonic regression of $\{y_i - \lambda\}$, i.e.,

$$\begin{aligned} \{\hat{\theta}_i\} = \quad &\underset{\boldsymbol{\theta}}{\text{argmin}} \quad \frac{1}{2}\sum_{i=1}^{n}(y_i - \lambda - \theta_i)^2 \\ &\text{subject to} \quad \theta_1 \geq \theta_2 \geq \ldots \geq \theta_n, \end{aligned} \tag{2.6}$$

then $\{(\hat{y}_i^\lambda)_+\}$ solves problem (2.5), where $x_+ = \max\{x, 0\}$. Hence the solution to (2.4) is

$$\mathbf{prox}_h(\beta) = (\hat{\beta}^\lambda)_+. \tag{2.7}$$

Using this in the proximal gradient algorithm gives the first-order generalized gradient update step of β for solving (2.3):

$$\beta \leftarrow \mathbf{prox}_{\gamma h}(\beta - \gamma \mathbf{X}^T(\mathbf{X}\beta - \mathbf{y})). \tag{2.8}$$

The value $\gamma > 0$ is a step size that is adjusted by backtracking to ensure that the objective function is decreased at each step. To solve (2.2) we augment each predictor x_{ij} with $x^*_{ij} = -x_{ij}$ and write $x_{ij}\beta_j = x_{ij}\beta_j^+ + x^*_{ij}\beta_j^-$. We denote the expanded parameters by (β^+, β^-) and apply the proximal operator (2.8) alternately to $\mathbf{X}$ and $\mathbf{X}^*$ to obtain the minimizers $(\hat{\beta}^+, \hat{\beta}^-)$. Details for solving (2.2) can be seen in Algorithm 1. Isotonic regression can be computed in $O(N)$ operations [26] and hence (2.8) can be computed in $O(pN)$ operations (the number of entries in the data matrix $\mathbf{X}$). Therefore, the ordered lasso algorithm can be applied to large datasets.

The ordered lasso can be easily adapted to the elastic net [64] and the adaptive lasso [63] by some simple modifications to the proximal operator in (2.7).

Algorithm 1 Ordered Lasso

```
function ORDERED LASSO($\mathbf{X} \in \mathbb{R}^{n \times p}, \mathbf{y} \in \mathbb{R}^n,\ \mathbf{X}^* = -\mathbf{X}$)
    Initialize $\hat{\boldsymbol{\beta}}^+, \hat{\boldsymbol{\beta}}^- = 0 \in \mathbb{R}^p, \lambda$
    while (not converged) do
        Fix $\hat{\boldsymbol{\beta}}^{-(k)}$,
        $\hat{\boldsymbol{\beta}}^{+(k+1)} \leftarrow \mathbf{prox}_{t_k\lambda}(\hat{\boldsymbol{\beta}}^+ - \hat{\boldsymbol{\beta}}^- - t_k\mathbf{X}^T(\mathbf{X}\hat{\boldsymbol{\beta}}^+ + \mathbf{X}^*\hat{\boldsymbol{\beta}}^- - \mathbf{y}))$;
        Fix $\hat{\boldsymbol{\beta}}^{+(k+1)}$,
        $\hat{\boldsymbol{\beta}}^{-(k+1)} \leftarrow \mathbf{prox}_{\tilde{t}_k\lambda}(\hat{\boldsymbol{\beta}}^{+(k+1)} - \hat{\boldsymbol{\beta}}^- - \tilde{t}_k\mathbf{X}^T(\mathbf{X}\hat{\boldsymbol{\beta}}^{+(k+1)} + \mathbf{X}^*\hat{\boldsymbol{\beta}}^- - \mathbf{y}))$;
    end while
end function
```

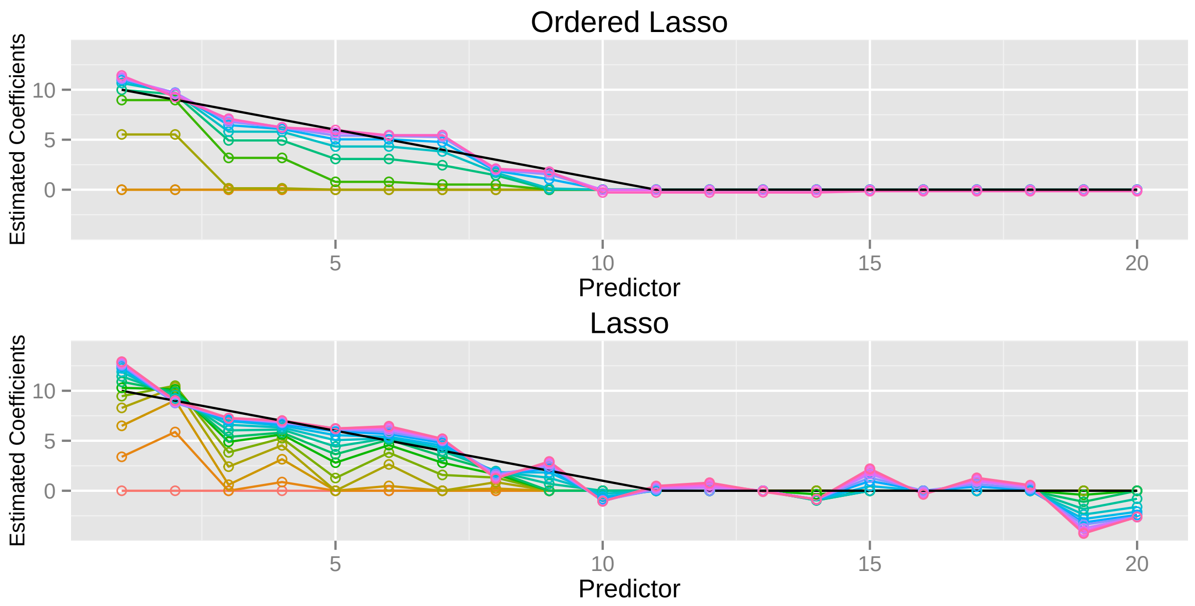

Figure 2.1. Example of the ordered lasso compared to the standard lasso: the data was generated from a true monotone sequence of coefficients plus Gaussian noise: $y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \sigma \cdot Z_i$, with $x_{ij} \sim N(0,1)$, $\beta = (10, 9, \ldots, 2, 1, 0, 0, \ldots 0)$, $\sigma = 7$. There were 20 predictors and 30 observations. The black profiles show the true coefficients and the colored profiles are the estimated coefficients for different values of λ from the largest (at bottom) to the smallest (at top).

2.1.3 Comparison between the ordered lasso and the lasso

Figure 2.1 shows a comparison between the ordered lasso and the standard lasso. The data was generated from a true monotone sequence plus Gaussian noise. The black profiles show the true coefficients, while the colored profiles are the estimated coefficients for different values of λ, from largest (at bottom) to smallest (at top). The corresponding plot for the lasso is shown in the bottom panel. The ordered lasso—exploiting the monotonicity—does a much better job of recovering the true coefficients than the lasso, as seen by the fluctuations of the estimated coefficients in the tails of the lasso plot.

2.1.4 The strongly ordered lasso

Previously in the ordered lasso, we wrote $\beta_j = \beta_j^+ - \beta_j^-$ and solved for β_j^+ and β_j^-, for $j = 1, 2, \ldots, p$. Though the resulting estimates $\hat{\beta}_j^+$ and $\hat{\beta}_j^-$ are monotone non-increasing, the resulting solutions $\hat{\beta}_j$ might not be monotone non-increasing in absolute value. To obtain solutions guaranteed to be monotone in absolute value, we extend our procedure: we first compute the minimizers to the ordered lasso problem

$$\begin{aligned} &\underset{\beta_0,\beta^+,\beta^-}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}(\beta_j^+ - \beta_j^-))^2 + \lambda \sum_{j=1}^{p}(\beta_j^+ + \beta_j^-) \\ &\text{subject to} \quad \beta_1^+ \geq \beta_2^+ \geq \ldots \geq \beta_p^+ \geq 0, \quad \beta_1^- \geq \beta_2^- \geq \ldots \geq \beta_p^- \geq 0. \end{aligned} \tag{2.9}$$

Denote solutions to (2.9) by $(\hat{\beta}_0, \hat{\beta}) = (\hat{\beta}_0, \hat{\beta}^+ - \hat{\beta}^-)$. Then we let $s(\hat{\beta}_j)$ be the sign of $\hat{\beta}_j$ for $j = 1, 2, \ldots, p$ and solve

$$\begin{aligned} &\underset{\theta_0,\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{N}(y_i - \theta_0 - \sum_{j=1}^{p} x_{ij}\theta_j)^2 + \lambda \sum_{j=1}^{p} s(\hat{\beta}_j)\theta_j \\ &\text{subject to} \quad \theta_1 s(\hat{\beta}_1) \geq \theta_2 s(\hat{\beta}_2) \geq \cdots \geq \theta_p s(\hat{\beta}_p) \geq 0. \end{aligned} \tag{2.10}$$

This produces a monotone solution (in absolute value) that is not necessarily the global minimum of the non-convex problem, but is guaranteed to be a stationary point.

Theorem 1. *The solution $(\hat{\theta}_0, \hat{\boldsymbol{\theta}})$ to (2.10) exists, and is a stationary point for (2.1).*

Proof Since (2.10) is a convex problem, the solution to (2.10) exists [11]. We rewrite the second step optimization problem (2.10) in matrix form:

$$\begin{aligned} &\underset{\theta_0,\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{y} - \theta_0 - \mathbf{X}\boldsymbol{\theta}\|^2 + \lambda \mathbf{s}^T\boldsymbol{\theta} \\ &\text{subject to} \quad \begin{pmatrix}\mathbf{A}\\ \mathbf{C}\end{pmatrix}\boldsymbol{\theta} \leq 0, \end{aligned} \tag{2.11}$$

where

$$\mathbf{s} = \begin{pmatrix} s(\hat{\beta}_1) \\ \vdots \\ \mathrm{s}(\hat{\beta}_p) \end{pmatrix}, \quad \mathbf{C}_{p\times p} = \begin{pmatrix} -s(\hat{\beta}_1) & & \\ & \ddots & \\ & & -s(\hat{\beta}_p) \end{pmatrix}, \quad \mathbf{A}_{(p-1)\times p} = \begin{pmatrix} -s(\hat{\beta}_1) & \mathrm{s}(\hat{\beta}_2) & & \\ & \ddots & \ddots & \\ & & -s(\hat{\beta}_{p-1}) & \mathrm{s}(\hat{\beta}_p) \end{pmatrix}.$$

The KKT condition for (2.11) is

$$-\mathbf{X}^T(\mathbf{y} - \theta_0 - \mathbf{X}\boldsymbol{\theta}) + \lambda\mathbf{s} + \begin{pmatrix}\mathbf{A}^T & \mathbf{C}^T\end{pmatrix}\begin{pmatrix}\boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2\end{pmatrix} = 0, \tag{2.12}$$

where $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$ are Lagrange multipliers and $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2 \geq 0$. We denote solutions to (2.12) by $(\hat{\theta}_0, \hat{\boldsymbol{\theta}})$ and $(\boldsymbol{\xi}_1^*, \boldsymbol{\xi}_2^*)$. We now write optimization problem (2.1) in matrix form and examine its KKT condition:

$$\begin{aligned} &\underset{\beta_0,\beta}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{y} - \beta_0 - \mathbf{X}\beta\|^2 + \lambda\sum_{i=1}^{p}|\beta_i| \\ &\text{subject to} \quad \begin{pmatrix}\mathbf{U} \\ \mathbf{I}\end{pmatrix}|\beta| \leq 0, \end{aligned} \tag{2.13}$$

where

$$|\beta| = \begin{pmatrix}|\beta_1| \\ \vdots \\ |\beta_p|\end{pmatrix}, \quad \mathbf{U}_{p-1\times p} = \begin{pmatrix}-1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1\end{pmatrix}, \quad \mathbf{I}_{p\times p} = \begin{pmatrix}-1 & & \\ & \ddots & \\ & & -1\end{pmatrix}.$$

This problem is not convex and thus we write the KKT condition in terms of sub-gradients. From [45], the sub-gradient of $f(x) = |x|$ is

$$\partial f = s(x) = \begin{cases}-1 & x < 0 \\ [-1, 1] & x = 0 \\ 1 & x > 0\end{cases}.$$

The KKT condition for (2.13) is thus

$$0 \in -\mathbf{X}^T(\mathbf{y} - \beta_0 - \mathbf{X}\beta) + \lambda\mathbf{s} + \mathbf{J}^T\begin{pmatrix}\mathbf{U}^T & \mathbf{I}^T\end{pmatrix}\begin{pmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{pmatrix}, \tag{2.14}$$

where $\mathbf{J}_{p\times p}$ is the Jacobian matrix $\mathbf{J}_\beta(|\beta|) = \begin{pmatrix} s(\beta_1) & & \\ & \ddots & \\ & & s(\beta_p) \end{pmatrix}$ and $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ are Lagrange multipliers with $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \geq 0$. It is not difficult to show that $\mathbf{J}^T\mathbf{U}^T = \mathbf{A}^T$ and $\mathbf{J}^T\mathbf{I}^T = \mathbf{C}^T$. Since $s(\hat{\beta}_j) \in [-1, 1], j = 1, \ldots, p$, and if we plug $(s(\hat{\beta}), (\hat{\theta}_0, \hat{\boldsymbol{\theta}}), (\boldsymbol{\xi}_1^*, \boldsymbol{\xi}_2^*))$ into (2.14), we know 0 is contained trivially in the right-hand side of (2.14). Therefore, $(\hat{\theta}_0, \hat{\boldsymbol{\theta}})$ is a stationary point for problem (2.13). □

Compared to the ordered lasso procedure described in Section 2.1.2, the strongly ordered lasso guarantees that the estimated coefficients $\hat{\beta}_j$ are monotone non-increasing in absolute value.

2.1.5 A different relaxation

We consider a similar model:

$$\begin{aligned} &\underset{\beta_0,\beta^+,\beta^-}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}(\beta_j^+ - \beta_j^-))^2 + \lambda\sum_{j=1}^{p}(\beta_j^+ + \beta_j^-) \\ &\text{subject to} \quad \beta_1^+ + \beta_1^- \geq \beta_2^+ + \beta_2^- \geq \cdots \geq \beta_p^+ + \beta_p^- \geq 0, \quad \beta_j^+, \beta_j^- \geq 0. \end{aligned} \tag{2.15}$$

The constraints in the ordered lasso (2.2) imply the ordered constraints in (2.15) but not vice versa. It is also a convex problem and the penalty term encourages sparsity in $\beta_j^+ + \beta_j^-$ and thus in β_j^+, β_j^-. Experiments showed that it performs slightly worse than the ordered lasso, so we do not consider this formulation further.

2.1.6 Relaxation of the monotonicity requirement

As a different generalization of our approach, we can relax problem (2.2) as follows:

$$\begin{aligned}
&\underset{\beta_0,\beta^+,\beta^-}{\text{minimize}} && \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}(\beta_j^+ - \beta_j^-))^2 \\
&&& + \lambda\sum_{j=1}^{p}(\beta_j^+ + \beta_j^-) + \theta_1\sum_{j=1}^{p-1}(\beta_j^+ - \beta_{j+1}^+)_+ + \theta_2\sum_{j=1}^{p-1}(\beta_j^- - \beta_{j+1}^-)_+ \\
&\text{subject to} && \beta_j^+, \beta_j^- \geq 0, \quad j = 1, 2 \ldots, p.
\end{aligned} \tag{2.16}$$

As $\theta_1, \theta_2 \to \infty$, the last two penalty terms force monotonicity in β_j^+ and β_j^- and this is equivalent to (2.2). However, for intermediate positive values of θ_1, θ_2, these penalties encourage near-monotonicity. This idea was proposed by Tibshirani et al. [54] for data sequences, generalizing the isotonic regression problem. The authors derive an efficient algorithm `NearIso`, which is a generalization of the well-known PAVA procedure mentioned above. Operationally, this creates no extra complication in our framework: we simply use `NearIso` in place of PAVA in the generalized gradient algorithm described in Section 2.1.2.

2.2 Sparse time-lagged regression

In this section we apply the ordered lasso and the strongly ordered lasso to the time-lagged regression problem. There are two problems we consider. The first is the *static outcome* problem, where we observe outcome at a fixed time t and predictors at a series of time points, and the outcome at time t is predicted from the predictors at previous time points. We also consider the *rolling prediction* problem, where we observe both outcome and predictors at a series of time points and the outcome is predicted at each time point from the predictors at previous time points.

2.2.1 Static prediction from time-lagged features

Here we consider the problem of predicting an outcome at a fixed time point from a set of time-lagged predictors. We assume that our data has the form

$$\{y_i, x_{i11}, \dots x_{iK1}, x_{i12}, \dots x_{iK2}, \dots x_{i1p}, \dots, x_{iKp}\},$$

for $i = 1, 2, \dots, N$ with N being the number of observations. The value x_{ikj} is the measurement of predictor j of observation i at time-lag k from the current time t. In other words, we predict the outcome at time t from p predictors, each measured at K time points preceding the current time t. Our model has the form

$$y_i = \beta_0 + \sum_{j=1}^{p} \sum_{k=1}^{K} x_{ikj}\beta_{kj} + \epsilon_i,$$

with $\mathbb{E}(\epsilon_i) = 0$ and $\text{Var}(\epsilon_i) = \sigma^2$. We denote $\hat{y}_i = \beta_0 + \sum_{j=1}^{p} \sum_{k=1}^{K} x_{ikj}\beta_{kj}$, write each $\beta_{kj} = \beta^+_{kj} - \beta^-_{kj}$, and solve

$$\begin{aligned} &\underset{\beta_0, \beta^+, \beta^-}{\text{minimize}} && \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p}\sum_{k=1}^{K}(\beta^+_{kj} + \beta^-_{kj}) \\ &\text{subject to} && \beta^+_{1j} \geq \beta^+_{2j} \geq \cdots \geq \beta^+_{Kj} \geq 0, \quad \beta^-_{1j} \geq \beta^-_{2j} \geq \cdots \geq \beta^-_{Kj} \geq 0, \quad \forall j. \end{aligned} \tag{2.17}$$

This model makes the plausible assumption that each predictor has an effect up to K time units away from the current time t, and this effect is monotone non-increasing as we move farther back in time.

In order to solve (2.17), we first write each $\beta^{\pm}_{kj}$ in the form

$$\Big\{ \underbrace{\beta^{\pm}_{11}, \beta^{\pm}_{21}, \dots, \beta^{\pm}_{K1}}_{\text{block 1}} \mid \underbrace{\beta^{\pm}_{12}, \beta^{\pm}_{22}, \dots, \beta^{\pm}_{K2}}_{\text{block 2}} \mid \dots \mid \underbrace{\beta^{\pm}_{1p}, \beta^{\pm}_{2p}, \dots, \beta^{\pm}_{Kp}}_{\text{block p}} \Big\}.$$

This leads to a blockwise coordinate descent procedure, with one block for each predictor. For example, at step j, we compute the update for block j while holding the rest of the blocks constant. With a sufficiently large time-lag K, the procedure automatically chooses an appropriate number of non-zero coefficients for each predictor, and zeros out the rest in each block because of the order constraint on each predictor. Details can be seen in Algorithm 2.

Algorithm 2 Ordered Lasso for Static Prediction

```
function ORDERED LASSO($\mathbf{X} \in \mathbb{R}^{n\times(Kp)}, \mathbf{y} \in \mathbb{R}^n, \mathbf{X}^* = -\mathbf{X}$)
    Initialize $\hat{\boldsymbol{\beta}}^+, \hat{\boldsymbol{\beta}}^- \in \mathbb{R}^{Kp}$, $\hat{\beta}^+_{kj} = 0$, $\hat{\beta}^-_{kj} = 0, \lambda$
    while not converged do
        for $j = 1, \dots, p$ do
            For each $i$, $r_i = y_i - \sum_{\ell \neq j} \sum_{k=1}^K (x_{ik\ell}\hat{\beta}^+_{k\ell} + x^*_{ik\ell}\hat{\beta}^-_{k\ell})$
            Apply the ordered lasso (Algorithm 1) to data
             $\{r_i, (x_{i1j}, \dots, x_{iKj}), (x^*_{i1j}, \dots, x^*_{iKj}), i = 1, 2, \dots, n\}$
             to obtain new estimates $\{\hat{\beta}_{kj}, k = 1, 2, \dots K\}$;
        end for
    end while
end function
```

2.2.2 Rolling prediction from time-lagged features

Here we assume that our data has the form $\{y_t, x_{t1}, \dots, x_{tp}\}$ for $t = 1, 2, \dots, N$. In detail, we have a time series for which we observe the outcome and the values of each predictor at N different time points. We consider a time-lagged regression model with a maximum lag of K time points:

$$y_t = \beta_0 + \sum_{j=1}^{p} \sum_{k=1}^{K} x_{t-k,j}\beta_{kj} + \epsilon_t,$$

with $\mathrm{E}(\epsilon_t) = 0$ and $\mathrm{Var}(\epsilon_t) = \sigma^2$. We write each $\beta_{kj} = \beta^+_{kj} - \beta^-_{kj}$ and propose the problem

$$\begin{aligned} &\underset{\beta_0, \beta^+, \beta^-}{\text{minimize}} \quad \frac{1}{2}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2 + \lambda \sum_{j=1}^{p}\sum_{k=1}^{K}(\beta^+_{kj} + \beta^-_{kj}) \\ &\text{subject to} \quad \beta^+_{1j} \geq \beta^+_{2j} \geq \ldots \geq \beta^+_{Kj} \geq 0, \quad \beta^-_{1j} \geq \beta^-_{2j} \geq \ldots \geq \beta^-_{Kj} \geq 0, \quad \forall j. \end{aligned} \tag{2.18}$$

To solve this problem, we convert it into the form of Section 2.2.1. We build a larger feature matrix $\mathbf{Z}$ of size $N \times (Kp)$, with K columns for each predictor. In detail, each row has the form $\left\{x_{t-1,1}, x_{t-2,1}, \dots, x_{t-K,1} \mid x_{t-1,2}, x_{t-2,2}, \dots, x_{t-K,2} \mid \cdots \mid x_{t-1,p}, x_{t-2,p}, \dots, x_{t-K,p}\right\}$.

Each block corresponds to a predictor lagged for $1, 2, \ldots, K$ time units. The matrix $\mathbf{Z}$ has N such rows, corresponding to time points $t-1, t-2, \ldots, t-K$. Again, we augment each predictor $x_{t-k,j}$ with $x^*_{t-k,j} = -x_{t-k,j}$, choose a sufficiently large time-lag K, and let the procedure zero out extra coefficients for each predictor. We can solve (2.18) using block coordinate descent as in the previous section. Details are shown in Algorithm 3.

Algorithm 3 Ordered Lasso for Rolling Prediction

```
function TIME LAGGED ORDERED LASSO($\mathbf{X} \in \mathbb{R}^{n\times(Kp)}, \mathbf{y} \in \mathbb{R}^n, \mathbf{X}^* = -\mathbf{X}$)
    Initialize $\hat{\boldsymbol{\beta}}^+, \hat{\boldsymbol{\beta}}^- \in \mathbb{R}^{Kp}$, $\hat{\beta}^+_{kj} = 0$, $\hat{\beta}^-_{kj} = 0, \lambda$
    while not converged do
        for $j = 1, \ldots, p$ do
            For each $t$, $r_t = y_t - \sum_{\ell \neq j} \sum_{k=1}^K (x_{t-k,\ell}\beta^+_{k\ell} + x^*_{t-k,\ell}\beta^-_{k\ell})$
            Apply the ordered lasso (Algorithm 1) to data
              $\{r_t, (x_{t-1,j}, \ldots, x_{t-K,j}), (x^*_{t-1,j}, \ldots, x^*_{t-K,j}), t = 1, 2, \ldots, n\}$
              to obtain new estimates $\{\hat{\beta}_{kj}, k = 1, 2, \ldots, K\}$;
        end for
    end while
end function
```

2.2.3 The strongly ordered lasso applied to time-lagged features

The strongly ordered lasso can be adapted to time-lagged regression by the following two-step procedure:

1. Apply Algorithm 2 or Algorithm 3 to obtain signs of the estimated coefficients for each predictor.

2. If there exists a predictor with non-monotone coefficients, apply the strongly ordered lasso procedure to each predictor using blockwise coordinate descent, as described in Section 2.2.1.

2.2.4 Simulated examples

Figure 2.2 shows an example of the ordered lasso procedure applied to a rolling time-lagged regression. The simulated data consists of four predictors with a maximum lag of

5 time points and 111 observations. The true coefficients for each of the four predictors were $(7,5,4,2,0)$, $(5,3,0,0,0)$, $(3,0,0,0,0)$ and $(0,0,0,0,0)$. The features were generated as i.i.d. $N(0,1)$ with Gaussian noise of standard deviation 7. The figure shows the true coefficients (black), and estimated coefficients of the ordered lasso (blue) and the standard lasso (orange) from 20 simulations. For each method, the coefficient estimates with the smallest mean squared error (MSE) in each realization are plotted. We see that the ordered lasso does a better job of recovering the true coefficients. The average mean squared errors for the ordered lasso and the lasso were $4.08(.41)$ and $6.11(.54)$, respectively.

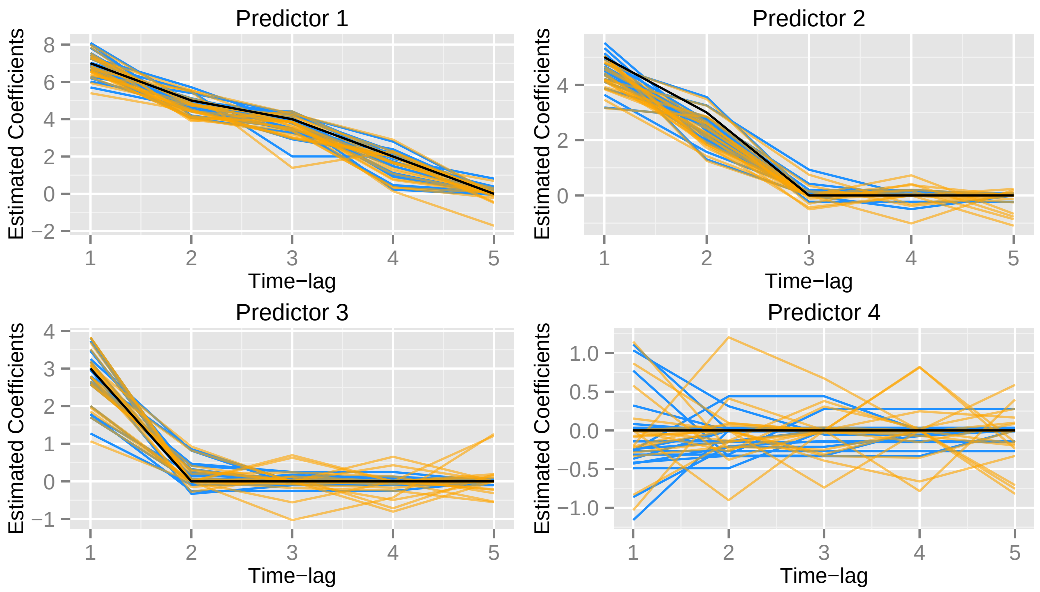

Figure 2.2. True coefficients (black), coefficient estimates for the ordered lasso (blue) and the standard lasso (orange) from 20 simulations.

Figure 2.3 shows a larger example with a maximum lag of 20 time points. The features were generated as i.i.d. $N(0,1)$ with Gaussian noise of standard deviation 7. Let $f(a,b,L)$ denote the equally spaced sequence from a to b of length L. The true coefficients for each of the four predictors were $\{f(5,1,20)\}$, $\{f(5,1,10), f(0,0,10)\}$, $\{f(5,1,5), f(0,0,15)\}$ and $\{f(0,0,20)\}$. The left panel of the figure shows the mean squared error of the standard lasso and the ordered lasso over 30 simulations. The value of λ giving the minimum MSE was chosen in each realization. In the right panel we have randomly permuted the true predictor coefficients for each realization, thereby causing the monotonicity to be violated (on average), but keeping the same signal-to-noise ratio. Not surprisingly, the ordered lasso

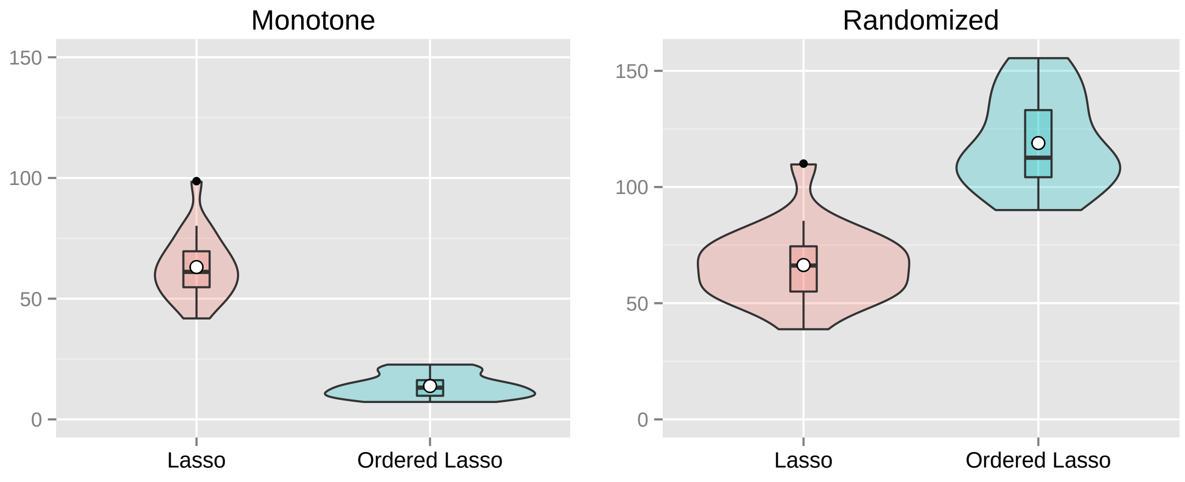

Figure 2.3. The lasso and the ordered lasso, applied to time-lagged features. Shown is the mean squared error over 30 simulations using the minimizing value of λ for each realization. In the left panel, the true coefficients are monotone; in the right, they have been scrambled so that monotonicity does not hold.

does better when the true coefficients are monotone, while the reverse is true for the lasso. However, we also see that in an absolute sense one can achieve a much lower MSE in the monotone setting of the left panel.

2.2.5 Performance on the Los Angeles ozone data

These data are available at `http://statweb.stanford.edu/~tibs/ElemStatLearn/data.html`. They represent the level of atmospheric ozone concentration from eight daily meteorological measurements made in the Los Angeles basin for 330 days in 1976. The response variable is the log of the daily maximum of the hourly-averaged ozone concentrations in Upland, California. We divided the data into training and validation sets of approximately the same size, and considered models with a maximum time-lag of 20 days.

Figure 2.4 shows the prediction error curves over the validation set, for the "cross-sectional" lasso (predicting from measurements on the same day), the lasso (predicting from measurements on the same day and the previous 19 days), and the ordered lasso, which adds the monotonicity constraint to the lasso procedure. We see that the ordered

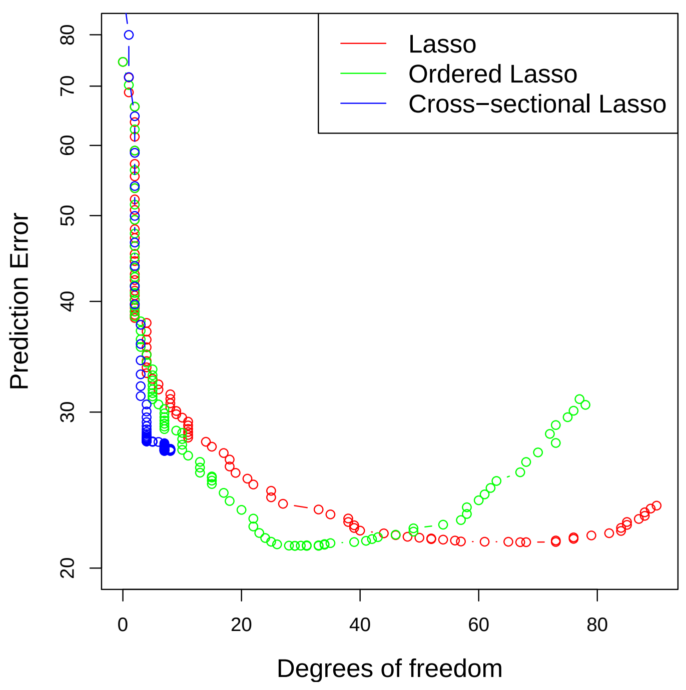

Figure 2.4. Los Angeles ozone data: prediction error curves. The cross-sectional lasso (blue) predicts from measurements on the same day, the lasso (red) predicts from measurements on the same day and previous 19 days, and the ordered lasso (green) adds the monotonicity constraint to the lasso. The ordered lasso achieves the lowest prediction with the fewest degrees of freedom.

lasso and the lasso applied to time-lagged features achieve lower errors than the "cross-sectional" lasso. In addition, the ordered lasso achieves the minimum with fewer degrees of freedom (defined in Section 2.3).

Figure 2.5 shows the estimated coefficients from the ordered lasso (top), the strongly ordered lasso (middle), and the lasso (bottom). The ordered lasso and the strongly ordered lasso yield simpler and more interpretable solutions. For each predictor, the ordered lasso and the strongly ordered lasso also determine the most suitable estimate of the time-lag interval, beyond which the estimated coefficients are zero. For example, in the ordered lasso plot, the estimated coefficients of the predictor "wind" are zero beyond a time-lag of 14 days from the current time t, whereas the estimated coefficients for "humidity" are zero beyond a time-lag of 7 days from the current time t. It is also worth pointing out that even though the ordered lasso yields more interpretable solutions, the estimated coefficients are not monotone non-increasing in absolute value, as marked by blue circles in the top panel of Figure 2.5. On the other hand, the strongly ordered lasso not only produces a similar plot, but also guarantees the estimated coefficients are monotone non-increasing in absolute value.

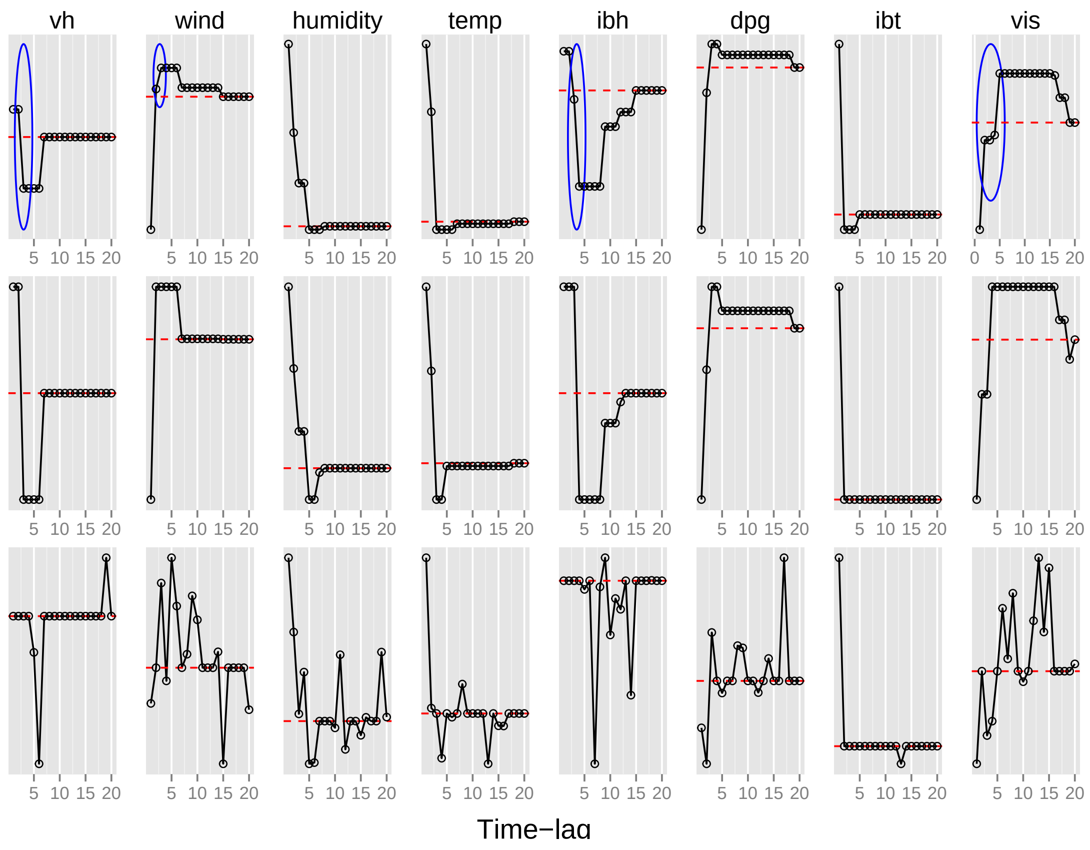

Figure 2.5. Ozone data: estimated coefficients from the ordered lasso (top), the strongly ordered lasso (middle), and the lasso (bottom) versus time-lag. The blue circles mark regions where the estimated coefficients are not monotone non-increasing in absolute value. For reference, a dashed red horizontal line is drawn at zero.

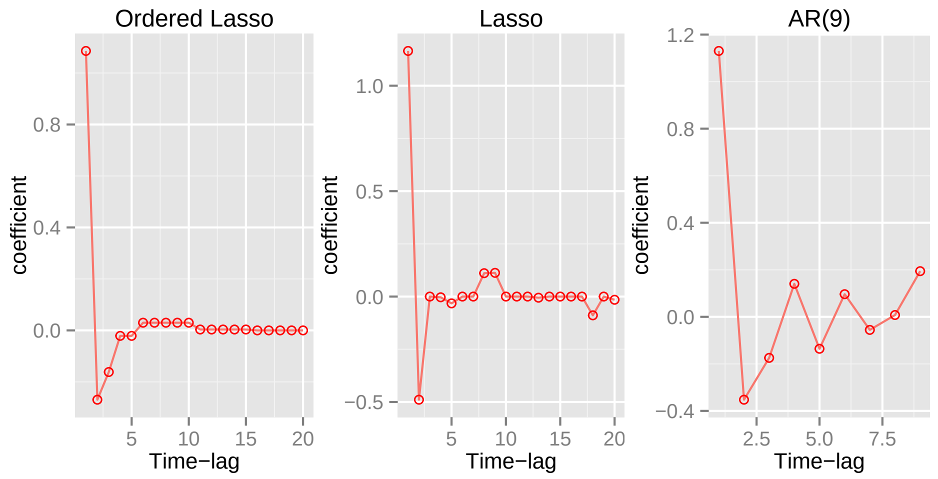

Figure 2.6. Sunspot data: estimated coefficients of the ordered lasso, the lasso and the standard AR fit.

2.2.6 Auto-regressive time series applied to sunspot data and simulated data

In an auto-regressive (AR) time series model, one predicts each value y_t from the values $y_{t-1}, y_{t-2}, \ldots, y_{t-k}$ for some maximum lag, or "order" k. This fits into the time-lagged regression framework, where the regressors are simply the time series itself at previous time points. Our proposal for monotone constraints in the AR model seems to be novel. Nardi and Rinaldo [39] studied the application of the standard lasso to the AR model and derived its asymptotic properties.

Schmidt and Makalic [44] suggested a Bayesian approach to the lasso based on the partial autocorrelation representation of AR models. In the following example, we compare coefficient estimates and order estimates among the ordered lasso, the strongly ordered lasso, the lasso, and the standard AR fit.

Performance on the sunspot data

The data for this example is available in the R package as `sunspot.year`. The data contains 289 measurements representing yearly numbers of sunspots from 1700 to 1988. Figure 2.6 shows the results of the auto-regressive model fit to the yearly sunspot data. We separated the series into training and validation series of about equal size. The standard AR fit (right

Method \ Estimated lag	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
AR/AIC	0	0	67	14	7	3	4	3	2	0
AR/BIC	0	0	98	2	0	0	0	0	0	0
Ordered Lasso	0	0	62	16	4	6	4	1	2	5
Strongly Ordered Lasso	0	0	66	14	3	5	3	3	3	3

Table 2.1. Estimates of AR lag from the ordered lasso, the strongly ordered lasso, and AR model using least squares with AIC and BIC from 100 simulations. The data was generated as $y_i = \sum_{k=1}^{3} y_{i-k}\beta_k + \sigma \cdot Z_i$ where $\sigma = 4$, $y_i, Z_i \sim N(0,1)$, and $\beta = \{0.35, 0.25, 0.25\}$. Each entry represents the number of times that a specific lag was estimated in 100 simulations. The results show that the ordered lasso and the strongly ordered lasso have similar performance to AR/AIC for this task.

panel) chose an order of 9 using least squares and the AIC. The ordered lasso (with λ chosen by two-fold cross validation) suggests an order of 10 (out of a maximum of 20) and gives a well-behaved sequence of coefficients. The regular lasso (middle panel)—with no monotonicity constraints—gives a less clear picture. All four estimates had about the same error on the validation set.

Performance on simulated data

Table 2.1 shows the results of an experiment comparing the ordered lasso and the strongly ordered lasso to the standard AR fitting using least squares with AIC and BIC from 100 simulations. The goal was to estimate the lag of the time series (number of non-zero coefficients) , as in the previous figure. The true series was of length 1000 with an actual lag of 3, and the maximum lag considered was 10. The data was divided into training and validation series of approximately the same size. The ordered lasso and the strongly ordered lasso used the second half of the series to estimate the best value of λ and estimate the order of the series. The results show that the ordered lasso and the strongly ordered lasso have similar performance to AR/AIC for this task.

2.3 Degrees of freedom

Given a fit vector $\hat{\mathbf{y}}$ for estimation from a vector $\mathbf{y} \sim N(\boldsymbol{\mu}, \mathbf{I} \cdot \sigma^2)$, the degrees of freedom of the fit can be defined as

$$\mathrm{df}(\hat{\mathbf{y}}) = \frac{1}{\sigma^2} \sum_{i=1}^{N} \mathrm{Cov}(y_i, \hat{y}_i) \tag{2.19}$$

[19]. This applies even if $\hat{\mathbf{y}}$ is an adaptively chosen estimate. Zou et al. [65] show that for the lasso, the number of non-zero coefficients in the solution is an unbiased estimate of the degrees of freedom. Tibshirani and Taylor [53] give analogous estimates for generalized penalties. For near-isotonic regression described in Section 2.1.6, letting $\hat{k}$ denote the number of nonzero "plateaus" in the solution, Tibshirani et al. [54] show that

$$\mathbb{E}(\hat{k}) = \mathrm{df}(\hat{\mathbf{y}}). \tag{2.20}$$

For the ordered lasso, this can be applied directly in the orthogonal design case to yield (2.20). For general $\mathbf{X}$, we conjecture that the same result holds, and can be established by studying the properties of projection onto the convex constraint set (as detailed in Tibshirani and Taylor).

Heuristically, one can also perform a significance test to check the monotonicity of the coefficients. We consider testing

$$\begin{aligned} &H_0 : \beta_1^+ \geq \beta_2^+ \geq \ldots \beta_p^+ \geq 0; \beta_1^- \geq \beta_2^- \geq \ldots \beta_p^- \geq 0 \\ &H_1 : \beta^+ \text{ and } \beta^- \text{not ordered.} \end{aligned}$$

We propose the following test statistic and conjecture that it has χ^2 distribution under the null hypothesis:

$$T = \frac{(RSS(\hat{\beta}_{\text{lasso}}) - RSS(\hat{\beta}_{\text{ordered lasso}}))}{\hat{\sigma}^2} \sim \chi^2_{\text{df(lasso)}-\text{df(orderedlasso)}},$$

where df(orderedlasso) and df(lasso) are defined above.

2.4 Logistic regression model

Here we show how to generalize the ordered lasso to logistic regression. Assume that we observe $(\mathbf{x}_i, y_i), i = 1, 2, \ldots, N$ with $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ and $y_i = 0$ or 1. The log-likelihood function is

$$l(\beta_0, \beta) = \sum_{i=1}^{N} (y_i(\beta_0 + \mathbf{x}_i^T \beta) - \log(1 + e^{\beta_0 + \mathbf{x}_i^T \beta})).$$

With the ordered lasso, we write each $\beta_j = \beta_j^+ - \beta_j^-$ with $\beta_j^+, \beta_j^- \geq 0$ for $j = 1, 2, \ldots, p$ and solve

$$\begin{aligned} &\underset{\beta_0, \beta^+, \beta^-}{\text{maximize}} && l(\beta_0, \beta^+ - \beta^-) - \lambda(\sum_{j=1}^{p} (\beta_j^+ + \beta_j^-) \\ &\text{subject to} && \beta_1^+ \geq \cdots \geq \beta_p^+ \geq 0, \quad \beta_1^- \geq \cdots \geq \beta_p^- \geq 0 \end{aligned} \tag{2.21}$$

We write our data in matrix form and use the iteratively reweighted least squares method (IRLS) to solve (2.21) [27], i.e., at each iteration we solve

$$\begin{aligned} &\underset{\beta_0, \beta^+, \beta^-}{\text{minimize}} && \frac{1}{2}(\mathbf{z} - \beta_0 \mathbf{1} - \mathbf{X}(\beta^+ - \beta^-))^T \mathbf{W}(\mathbf{z} - \beta_0 \mathbf{1} - \mathbf{X}(\beta^+ - \beta^-)) + \lambda \sum_{i=1}^{p} (\beta_i^+ + \beta_i^-)) \\ &\text{subject to} && \beta_1^+ \geq \cdots \geq \beta_p^+ \geq 0, \quad \beta_1^- \geq \cdots \geq \beta_p^- \geq 0, \end{aligned}$$

where $\mathbf{z} = \beta_0^{\text{old}} \mathbf{1} + \mathbf{X}(\boldsymbol{\beta}_{\text{old}}^+ - \boldsymbol{\beta}_{\text{old}}^-) + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$, $\mathbf{p}$ is a vector with

$$\mathbf{p}_i = \frac{\exp(\beta_0^{\text{old}} + \mathbf{x}_i^T (\boldsymbol{\beta}_{\text{old}}^+ - \boldsymbol{\beta}_{\text{old}}^-))}{1 + \exp(\beta_0^{\text{old}} + \mathbf{x}_i^T (\boldsymbol{\beta}_{\text{old}}^+ - \boldsymbol{\beta}_{\text{old}}^-))},$$

and $\mathbf{W}$ is a diagonal matrix with $\mathbf{W}_{ii} = \mathbf{p}_i(1 - \mathbf{p}_i)$. We apply the ordered lasso (Algorithm 1) to solve (2.22) with modified updates:

$$\begin{aligned} \beta_0 &\leftarrow \beta_0 - \gamma \mathbf{1}^T \mathbf{W}(\beta_0 \mathbf{1} + \mathbf{X}\beta - \mathbf{z}), \\ \beta &\leftarrow \mathbf{prox}_{\gamma\lambda}(\beta - \gamma \mathbf{X}^T \mathbf{W}(\beta_0 \mathbf{1} + \mathbf{X}\beta - \mathbf{z})). \end{aligned}$$

One can also apply the strongly ordered lasso to logistic regression similarly if the estimated coefficients from the ordered lasso are not monotone non-increasing in absolute value.

Applying the ordered lasso to the logistic regression model with time-lagged features, we approximate the log-likelihood function as in (2.22) and use Algorithm 2 or Algorithm 3 to solve the weighted least squares minimization subproblem. Similar extensions can be made to other generalized linear models.

2.5 Discussion

We have proposed an order-constrained version of the lasso and provided an efficient solution to the resulting problem. This procedure has natural applications to the static and rolling prediction problems, based on time-lagged variables. It can be applied to any dynamic prediction problem, including financial time series and prediction of dynamic patient outcomes based on clinical measurements. For future work, we could generalize our framework to higher-dimensional notions of monotonicity, which could be useful for spatial data. The R package `orderedLasso` that implements the algorithms is available on the CRAN website.

Chapter 3

Time varying regularized regression

3.1 Introduction

Suppose that we observe data $(\mathbf{x}_i, y_i)$ for $i = 1, 2, \dots, n$, where n is the number of observations, $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ is a vector of p feature measurements, and y_i is a response value. We consider the usual linear framework,

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i$$

with $\mathbb{E}(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = \sigma^2$. When $p \gg n$, Tibshirani [51] regularized the problem by adding the ℓ_1-norm of the coefficients as a penalty:

$$\underset{\beta_0, \boldsymbol{\beta}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|,$$

where $\lambda > 0$ is a given tuning parameter.

Though the properties of linear models have been well established, classical linear models are often unrealistic in applications. For example, as discussed by Fan and Zhang [20], if we were to estimate the cross-country growth, the standard growth analyses assume linear model assumptions. However, these assumptions are often violated as it is highly likely that a country growth rate depends on the development of the country and thus linear models fail to consider the dynamic pattern of such relations. Therefore, varying coefficient models are natural extensions to classical linear models. Not only do these enjoy nice interpretability of the data but they also can capture the dynamic patterns of the impacts of the predictors.

There has been some research done on this topic. They were first introduced by Chambers [14] in the context of local regression models. The smoothing spline method for estimating the varying coefficient models was studied by Hastie and Tibshirani [28] and Chiang et al. [16]. Huang and Shen [32] studied the polynomial spline method. Park et al. [41] provided a nice review of different kernel-local polynomial smoothing methods.

In this chapter, we consider a model where we assume that the coefficients are functions of one common variable, such as time:

$$y(t) = u + \beta_0(t) + \sum_{j=1}^{p} x_j(t)\beta_j(t) + \epsilon$$

with $\mathbb{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2$, and u is the constant effect over the time. We propose a constrained version of this model and call the resulting procedure *time varying ℓ_1 regularized regression* (TV-ℓ_1). We present an efficient algorithm for solving the resulting problem.

This chapter is organized as follows. Section 3.2 contains motivations and an algorithm for solving TV-ℓ_1. We show a close connection with the lasso problem in Section 3.3.1. We present a robust version of TV-ℓ_1 in Section 3.3.2. We also extend our model to a different framework, call the resulting procedure *time varying group lasso regression*, and provide a detailed algorithm for solving it. We apply TV-ℓ_1 to different simulated examples in Section 3.4 and a dynamical system in Section 3.5. Section 3.6 contains some discussions and directions for future work.

3.2 Time varying ℓ_1 regularized regression (TV-ℓ_1)

Consider the time varying coefficient model

$$y_i = u + \beta_{i0} + \sum_{j=1}^{p} x_{ij}\beta_{ij}(t_i) + \epsilon \tag{3.1}$$

for $i = 1, 2, \dots, n$, where n is the total number of observations. As before, we assume that $\mathbb{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = 0$. In this situation, u is a constant effect across all the observations, and β_{ij} is a function of t_i for $j = 1, \dots p$. This model makes sense in problems where the regression coefficients are not constant for all observations. For example, the coefficients are a function of time. In this situation, linear models could not fully capture the dynamic patterns of the coefficients and thus could lead to a large bias.

We consider solving problem

$$\underset{\beta_0, \beta}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}(y_i - u - \beta_{i0} - \sum_{j=1}^{p} x_{ij}\beta_{ij})^2 + \tag{3.2}$$
$$\lambda \sum_{i=1}^{n}\sum_{j=0}^{p}|\beta_{ij}| + \frac{\lambda_2}{2}\sum_{i=1}^{n-1}\sum_{j=1}^{p}\frac{(\beta_{i+1,j} - \beta_{i,j})^2}{t_{i+1} - t_i},$$

where β_{ij} could be a function of t_i for $j = 1, \dots, p$. The ℓ_1 penalty term encourages sparsity in the coefficients and the ℓ_2 penalty term makes the coefficients vary smoothly along the index variable t. Together, both penalties encourage sparsity and smoothness in the coefficients.

In order to solve (3.2), we rewrite this model in matrix notation. Let A_0 be the identity matrix $A_0 = I_{n\times n}$. Let A_j be a diagonal matrix containing observations for the j-th features, $A_j = \begin{pmatrix} x_{1j} & & \\ & \ddots & \\ & & x_{nj} \end{pmatrix}$ for $j = 1, 2, \dots, p$, and $\boldsymbol{\theta}_j$ be a vector $\boldsymbol{\theta}_j = (\beta_{1j}, \dots \beta_{nj})$ for $j = 0, \dots, p$. Let D be an $(n-1) \times n$ matrix with i-th diagonal element $\frac{1}{\sqrt{t_{i+1} - t_i}}$ and i-th upper diagonal element $-\frac{1}{\sqrt{t_{i+1} - t_i}}$, for $i = 1, \dots, n-1$. Thus, our model becomes

$$\underset{u, \boldsymbol{\theta}_j, j=0,\dots,p}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{y} - u\mathbf{1} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j\|_2^2 + \lambda\sum_{j=0}^{p}\|\boldsymbol{\theta}_j\|_1 + \frac{\lambda_2}{2}\sum_{j=0}^{p}\boldsymbol{\theta}_j^T D^T D\boldsymbol{\theta}_j, \tag{3.3}$$

where $\mathbf{1}$ is a column vector of 1 with length n. The minimization problem can be solved efficiently using blockwise minimization updates. Let $\boldsymbol{\theta} = \begin{pmatrix}\boldsymbol{\theta}_0 & \boldsymbol{\theta}_1 & \cdots & \boldsymbol{\theta}_p\end{pmatrix}$. With

$\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{j-1}, \boldsymbol{\theta}_{j+1}, \dots, \boldsymbol{\theta}_p$ fixed, the update rule for $\boldsymbol{\theta}_j$ can be simplified to solving

$$(A_j^T A_j + \lambda_2 D^T D)\hat{\boldsymbol{\theta}}_j = S(A_j^T(y - \sum_{k \neq j} A_k \theta_k), \lambda),$$

where the soft-threshold operator $S(x, \lambda)$ is defined as

$$S(x, \lambda) = \begin{cases} x - \lambda & x > \lambda \\ x + \lambda & x < -\lambda \\ 0 & x \in [-\lambda, \lambda] \end{cases} . \tag{3.4}$$

The matrix $A_j^T A_j + \lambda_2 D^T D$ is symmetric and tri-diagonal, so that the linear system $R\theta_j = b$ can be solved in $O(n)$ using the function `solve.tridiag` function of the package `limSolve` in R. A detailed algorithm for solving (3.3) can be seen in Algorithm 4.

Algorithm 4 Time varying ℓ_1 regularized regression (TV-ℓ_1)

```
function TIME VARYING($\mathbf{y} \in \mathbb{R}^n$ ; $A_0, A_1, \dots, A_p; \lambda$)
    Initialize $\theta_0, \theta_1, \dots, \theta_p$
    while not converged do
        $u = \frac{1}{n}\sum_{i=1}^n (y_i - \sum_{j=0}^p A_{ij}\theta_{ij})$
        Solve $(A_j^T A_j + \lambda_2 D^T D)\hat{\boldsymbol{\theta}}_j = S(A_j^T(y - \sum_{k \neq j} A_k \theta_k), \lambda), \quad j = 0, \dots, p$
    end while
end function
```

3.2.1 Prediction

The estimated coefficients of TV-ℓ_1 capture the dynamic pattern of the coefficients. When it comes to predicting a new response y with a new observation x, it also allows flexible prediction rules. For example, $\hat{y}$ can be estimated from the median of the predicted values, the mean of the predicted values, or the most recent of the predicted values:

$$\hat{y} = \text{median}(u + \boldsymbol{\theta}_0^T + \mathbf{x}^T \hat{\boldsymbol{\theta}}), \tag{3.5}$$

$$\hat{y} = \text{mean}(u + \boldsymbol{\theta}_0^T + \mathbf{x}^T \hat{\boldsymbol{\theta}}), \tag{3.6}$$

$$\hat{y} = u + \boldsymbol{\theta}_0(n) + \mathbf{x}^T \hat{\boldsymbol{\theta}}_n. \tag{3.7}$$

Generally speaking, if the coefficients are believed to have a large noise, the median value of the predicted values in (3.5) yields the smallest prediction error. If the coefficients are believed to be functions of other variables, such as time, the prediction using the most recent estimated coefficients for prediction (3.7) may have the best performance. If the data is seasonal, we can plug in the seasonal average for a better prediction, as seen in Section 3.4. When the coefficients are believed to be constant, the mean value of the predicted values gives the best result.

3.3 Extensions of TV-ℓ_1

In this section, we discuss a few extensions of our model, and we see that TV-ℓ_1 is closely related to many existing models in the literature.

3.3.1 Connection with the lasso problem

We show that the estimates of TV-ℓ_1 can be obtained from the solution of the lasso problem on an augmented data set. We augment $\mathbf{y}$, $u\mathbf{1}$ with $(n-1)$ zeros: $\tilde{\mathbf{y}} = \begin{pmatrix}\mathbf{y}\\0\end{pmatrix}$ and $\tilde{u} = \begin{pmatrix}u\mathbf{1}\\0\end{pmatrix}$. We also augment $\mathbf{X}$ as

$$\tilde{\mathbf{X}} = \begin{pmatrix} A_0 & A_1 & \dots & A_p \\ \sqrt{\lambda_2}D & \sqrt{\lambda_2}D & \dots & \sqrt{\lambda_2}D \end{pmatrix},$$

a $(2n-1)\times(p+1)n$ matrix. We then write $\tilde{\boldsymbol{\theta}} = \begin{pmatrix}\boldsymbol{\theta}_0 & \boldsymbol{\theta}_1 & \dots & \boldsymbol{\theta}_p\end{pmatrix}^T$ and solve

$$\underset{\tilde{\boldsymbol{\theta}},\tilde{\mathbf{u}}}{\text{minimize}} \quad \frac{1}{2}\|\tilde{\mathbf{y}} - \tilde{\mathbf{u}}\mathbf{1} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\theta}}\|^2 + \lambda_1\|\tilde{\boldsymbol{\theta}}\|_1.$$

This transformation enables one to use the R package `glmnet` [22] to solve (3.3). However, since we have to augment our data and double our memory requirement, this approach is not as efficient as our algorithm.

3.3.2 Robust time varying ℓ_1 regularized regression

When our data is contaminated with outliers and influential points, robust regression [33] is a common alternative to least squares. We consider a robust version of the time varying ℓ_1 regularized regression

$$\text{(3.8)} \qquad \underset{\boldsymbol{\theta}_0,\boldsymbol{\theta},u}{\text{minimize}} \qquad \underbrace{\rho\Big(\frac{\mathbf{y} - u\mathbf{1} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j}{s}\Big)}_{(*)} + \lambda \sum_{j=0}^{p} \|\boldsymbol{\theta}_j\|_1 + \frac{\lambda_2}{2}\sum_{j=0}^{p} \boldsymbol{\theta}_j^T D^T D\boldsymbol{\theta}_j.$$

We can use iteratively reweighted least squares [30] to solve problem (3.8), i.e, in each iteration, we approximate $(*)$ by

$$\Big(\frac{\mathbf{y} - u\mathbf{1} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j)}{s}\Big)^T W\Big(\frac{\mathbf{y} - u\mathbf{1} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j}{s}\Big) = \frac{1}{s^2}\|\mathbf{y} - u\mathbf{1} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j\|_W^2$$

and then solve

$$\text{(3.9)} \qquad \underset{\boldsymbol{\theta}_0,\boldsymbol{\theta}}{\text{minimize}} \qquad \frac{1}{s^2}\|y - u\mathbf{1} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j\|_W^2 + \lambda \sum_{j=0}^{p} \|\boldsymbol{\theta}_j\|_1 + \frac{\lambda_2}{2}\sum_{j=0}^{p} \boldsymbol{\theta}_j^T D^T D\boldsymbol{\theta}_j.$$

In each iteration, the solution to problem (3.9) is

$$\text{(3.10)} \qquad \boldsymbol{\theta}_j = \Big(\frac{A_j^T W A_j}{s} + \lambda_2 D^T D\Big)^{-1} S\Big(A_j^T W\big(\frac{y - \sum_{k\neq j} A_k\theta_k}{s}\big), \lambda\Big), \quad j = 0, \ldots, p,$$

where S is the soft-threshold operator defined in (3.4). There are different choices of the function ρ. For example, we consider the Huber function [34]

$$\rho(e) = \begin{cases} \frac{1}{2}e^2 & \text{for } |e| \le k \\ k|e| - \frac{1}{2}k^2 & \text{for } |e| > k \end{cases}.$$

The weight function for the Huber function is

$$\text{(3.11)} \qquad w(e) = \begin{cases} 1 & \text{if } |e| \le k \\ k/|e| & \text{if } |e| > k \end{cases}.$$

The detailed algorithm can be seen in Algorithm 5.

Algorithm 5 Robust time varying ℓ_1 regularized regression

```
function ROBUST TIME VARYING($\mathbf{y} \in \mathbb{R}^n; A_0, A_1, \dots, A_p; \lambda$)
    Initialize $\theta_0, \theta_1, \dots, \theta_p$
    while (not converged) do
        while (not converged) do
            update $W$ matrix based on ((3.11))
            $\hat{u} = \frac{\sum_{i=1}^n W_{ii} y_i}{\sum_{i=1}^n W_{ii}} - \frac{\sum_{i=1}^n \sum_{j=1}^p W_{ii} A_j^T \boldsymbol{\theta}_j}{\sum_{i=1}^n W_{ii}}$
            $\hat{\boldsymbol{\theta}}_j = (\frac{A_j^T W A_j}{s} + \lambda + 2D^T D)^{-1} S\big(A_j^T W\big(\frac{y - \sum_{k \neq j} A_k \theta_k}{s}\big), \lambda\big), j = 0, \dots, p$
        end while
    end while
end function
```

3.3.3 Time varying ℓ_1 groupwise regression

We can also apply the time varying ℓ_1 regularized regression to a longitudinal sample. A longitudinal sample for m subjects is represented by

$$\{Y_{ij}, X_i(t_{ij}), t_{ij};\ i = 1, \dots, n,\ j = 1, \dots, m\}$$

If we believe that a common varying coefficient set can explain all the subjects, we can solve the problem

$$\text{(3.12)} \quad \underset{u,\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{g} \sum_{s=1}^{g} \frac{1}{2} \|\mathbf{y}_s - u\mathbf{1} - \sum_{j=0}^{p} A_{sj} \boldsymbol{\theta}_j\|^2 + \lambda_1 \sum_{j=0}^{p} \|\boldsymbol{\theta}_j\|_1 + \frac{\lambda_2}{2} \sum_{j=0}^{p} \boldsymbol{\theta}_j^T D^T D \boldsymbol{\theta}_j,$$

where g is the total number of groups. We apply the same procedure of solving the time varying ℓ_1 regularized group lasso to solve (3.12). The detailed algorithm for solving (3.12) can be seen in Algorithm 6.

3.3.4 Extensions to time varying group lasso regression

In (3.1), an ℓ_1 penalty is used to encourage the sparsity of coefficients for each observation. Though the ℓ_1 penalty encourages sparsity, it generally does not zero out the whole sequence

Algorithm 6 Time varying ℓ_1 regularized groupwise regression

```
function GROUPWISE REGRESSION($\mathbf{y} \in \mathbb{R}^{gk}$ ; $A_{s0}, A_{s1}, \dots, A_{sp}, s = 1, \dots, g; \lambda$)
    Initialize $\theta_0, \theta_1, \dots, \theta_p$;
    while (not converged) do
        $u = \frac{1}{g} \sum_{s=1}^{g} (\bar{\mathbf{y}}_s - \overline{\sum_{j=0}^{p} A_{sj}\boldsymbol{\theta}_j})$
        $\hat{\theta}_j = (\sum_{s=1}^{g} A_{sj}^T A_{sj} + \lambda_2 D^T D)^{-1} S(\sum_{s=1}^{g} A_{sj}^T (y_s - \sum_{k \neq j} A_{sk}\theta_k), \lambda), j = 0, \dots, p$;
    end while
end function
```

of a particular group. Motivated by the group lasso of Yuan and Lin [62], we propose to add an ℓ_2 norm to zero out some coefficient sequences. This idea is also used by Simon et al. [46]. The resulting problem is

$$\text{(3.13)} \qquad \underset{\boldsymbol{\theta}_0, \boldsymbol{\theta}, u}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{y} - u\mathbf{1} - \sum_{j=0}^{p} A_j \boldsymbol{\theta}_j\|^2 + \lambda \sum_{j=0}^{p} \|\boldsymbol{\theta}_j\|_2 + \frac{\lambda_2}{2} \sum_{j=0}^{p} \boldsymbol{\theta}_j^T D^T D \boldsymbol{\theta}_j.$$

We now derive the necessary steps to solve (3.13). The objective in (3.13) is a sum of convex functions and therefore convex; and the optimal solution is characterized by subgradient conditions. We denote $\mathbf{r}_{(-j)} = \mathbf{y} - u\mathbf{1} - \sum_{k \neq j} A_k \boldsymbol{\theta}_k$. Holding $\boldsymbol{\theta}_0, \dots \boldsymbol{\theta}_{j-1}, \boldsymbol{\theta}_{j+1}, \dots, \boldsymbol{\theta}_p$ fixed, we optimize $\boldsymbol{\theta}_j$ using

$$\text{(3.14)} \qquad \underset{\boldsymbol{\theta}_j}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{r}_{(-j)} - A_j \boldsymbol{\theta}_j\|_2^2 + \lambda \|\boldsymbol{\theta}_j\|_2 + \frac{\lambda_2}{2} \boldsymbol{\theta}_j^T D^T D \boldsymbol{\theta}_j.$$

Therefore, $\hat{\boldsymbol{\theta}}_j$ must satisfy

$$A_j^T (\mathbf{y} - \sum_{j=0}^{p} A_j \boldsymbol{\theta}_j) = \lambda \gamma \ + \lambda_2 D^T D \boldsymbol{\theta}_j,$$

where γ is the sub-gradient $\|\boldsymbol{\theta}_j\|_2$:

$$\text{(3.15)} \qquad \gamma = \begin{cases} \frac{\boldsymbol{\theta}_j}{\|\boldsymbol{\theta}_j\|_2} & \text{if } \boldsymbol{\theta}_j \neq 0_{n \times 1} \\ \in \{\gamma : \|\gamma\|_2 \leq 1\} & \text{if } \boldsymbol{\theta}_j = 0_{n \times 1} \end{cases}.$$

If $\boldsymbol{\theta}_j \neq 0$, we have

$$A_j^T(\mathbf{y} - \sum_{j=0}^{p} A_j\boldsymbol{\theta}_j) = \lambda\gamma \ + \lambda_2 D^T D\boldsymbol{\theta}_j.$$

We now derive the algorithm for solving (3.13) using blockwise descent. This derivation follows closely the derivation of the sparse group lasso algorithm by Simon et al. [46]. Since the penalty is separable between different coefficient sequences, blockwise descent is guaranteed to converge. Let

$$l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_j) = \frac{1}{2}||\mathbf{r}_{(-j)} - \mathbf{X}^{(j)}\boldsymbol{\theta}||_2^2 + \frac{\lambda_2}{2}\boldsymbol{\theta}^T D^T D\boldsymbol{\theta}.$$

Note that this is a loss function without the l_2 norm penalty. We are using $\boldsymbol{\theta}$ here to denote $\boldsymbol{\theta}_j$. Now we approximate l near $\boldsymbol{\theta}_0$ and get

$$M(\boldsymbol{\theta}) = l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \nabla l(\mathbf{r}_{(-k)}, \boldsymbol{\theta}_0) + \frac{1}{2t}||\boldsymbol{\theta} - \boldsymbol{\theta}_0||_2^2 + \lambda||\boldsymbol{\theta}||_2.$$

Minimizing $M(\boldsymbol{\theta})$ is equivalent to minimizing $\tilde{M}(\boldsymbol{\theta})$:

$$\tilde{M}(\boldsymbol{\theta}) = \frac{1}{2t}||\boldsymbol{\theta} - (\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0))||_2^2 + \lambda||\boldsymbol{\theta}||_2.$$

The derivative with respect to $\boldsymbol{\theta}$ gives the following subgradient conditions:

$$(\boldsymbol{\theta} - (\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0))) + t\lambda\gamma = 0,$$

where γ is defined in (3.15). We have two scenarios according to the subgradient conditions:

1. $\hat{\boldsymbol{\theta}} = 0$ if $||\boldsymbol{\theta}_0 - t\nabla l(-\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0)||_2 \leq t\lambda$.

2. Otherwise, $\hat{\boldsymbol{\theta}}$ satisfies

$$(1 + \frac{t\lambda}{||\hat{\boldsymbol{\theta}}||_2})\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0).$$

Taking the norm of the both sides, we get the norm of $\hat{\boldsymbol{\theta}}$:

$$||\hat{\boldsymbol{\theta}}||_2 = ||\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0)||_2 - t\lambda.$$

Substituting this into the previous equation gives the estimate for $\hat{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{\theta}} = (1 - \frac{t\lambda}{||\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0)||_2})(\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0).$$

The update for $\boldsymbol{\theta}$ as $U(\boldsymbol{\theta}_0, t)$ is now
(3.16)

$$U(\boldsymbol{\theta}_0, t) = \begin{cases} 0 & \text{if } ||\boldsymbol{\theta}_0 - t\nabla l(-\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0)||_2 \le t\lambda \\ (1 - \frac{t\lambda}{||\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0)||_2})(\boldsymbol{\theta}_0 - t\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0) & \text{otherwise} \end{cases},$$

and note that

$$\nabla l(\mathbf{r}_{(-j)}, \boldsymbol{\theta}_0) = -A^{(j)^T}\mathbf{r}_{(-j)} + (\frac{A^{(j)^T}A^{(j)}}{n} + \lambda_2 D^T D)\boldsymbol{\theta}_0.$$

While holding $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{j-1}, \boldsymbol{\theta}_{j+1} \ldots, \boldsymbol{\theta}_p$ fixed, we iterate (3.16) by updating $\boldsymbol{\theta}_j$ each iteration until $\boldsymbol{\theta}_j$ converges to the optimal solution. Applying this to each block in turn will lead to convergence. The detailed algorithm can be seen in Algorithm 7.

3.3.5 Connection with the group lasso

Problem (3.13) is closely related to the group lasso [62]. We augment $\tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}$, $\tilde{\mathbf{X}} = \begin{pmatrix} A_0 & A_1 & \ldots & A_p \\ \sqrt{\lambda_2}D & \sqrt{\lambda_2}D & \ldots & \sqrt{\lambda_2}D \end{pmatrix}$, $\tilde{\mathbf{u}} = \begin{pmatrix} u\mathbf{1} \\ 0 \end{pmatrix}$, and $\tilde{\boldsymbol{\theta}} = \begin{pmatrix} \boldsymbol{\theta}_0 \\ \vdots \\ \boldsymbol{\theta}_p \end{pmatrix}$. We then solve

$$\underset{\tilde{\mathbf{u}}, \tilde{\boldsymbol{\theta}}}{\text{minimize}} \quad \frac{1}{2}||\tilde{\mathbf{y}} - \tilde{\mathbf{u}} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\theta}}||^2 + \lambda_1||\tilde{\boldsymbol{\theta}}||_2.$$

Algorithm 7 Time varying group lasso regression

```
function GROUP LASSO REGRESSION($\mathbf{y} \in \mathbb{R}^n \ A_0, A_1, \dots, A_p$)
    Fix $\lambda$; Initialize $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_p, \beta = 0.8, l = 1$
    while (not converged) do
        $\boldsymbol{\theta}_0 = (A_0^T A_0 + \lambda_2 D^T D)^{-1}(A_0^T(\mathbf{y} - \sum_{j=1}^p A_j \boldsymbol{\theta}_j))$
        for $k = 1, \dots, p$ do
            $\mathbf{r}_{-k} = \mathbf{y} - \sum_{j=0, j\neq k} A_j \boldsymbol{\theta}_j$;
            if $\| - A_k^T \mathbf{r}_{-k}\|_2 \leq \lambda_1$ then
                $\hat{\boldsymbol{\theta}}_k = 0$
            end if
            while (not converged) do
                Let $t = 1, l = 1, \boldsymbol{\xi}_{(l,k)} = \boldsymbol{\theta}_{(k,l)} = \boldsymbol{\theta}_k$
                while (not converged) do
                    update gradient $g = \nabla l(\mathbf{r}_{(-k)}, \boldsymbol{\theta}_{(k,l)}$
                    repeat
                        $t = t \times \beta$
                    until $l(U(\theta^{(k,l)}, t) \leq l(\theta^{(k,l)}) + g^T \Delta_{(l,t)} + \frac{1}{2t}\|\Delta_{(l,t)}\|_2^2$
                    $\boldsymbol{\xi}_{(k,l)} \leftarrow U(\theta^{(k,l)}, t)$
                    Update the center via a Nesterov step:
                        $$\boldsymbol{\theta}_{(k,l+1)} \leftarrow \boldsymbol{\theta}_{(k,l)} + \frac{l}{l+3}(\boldsymbol{\xi}_{(k,l)} - \boldsymbol{\theta}_{(k,l)})$$
                    $l = l + 1$
                end while
            end while
        end for
    end while
end function
```

3.4 Simulated examples

In this section, we consider simulated examples in different scenarios and compare the performance of TV-ℓ_1 with least squares and the lasso.

3.4.1 Time varying coefficients

We first consider the following setup:

$$\begin{aligned}
\beta_0(t) &= -2\sin(\frac{t\pi}{120}) + \frac{t}{50} + 6, \\
\beta_1(t) &= \cos(\frac{(t-30)\pi}{100}) - \frac{t}{45} + 6, \\
\beta_2(t), \beta_3(t), &\ldots, \beta_5(t) = 0,
\end{aligned} \tag{3.17}$$

where t is an index that varies from 0 to 199 and is incremented by 1. The true coefficients can be seen as a function of time. We now examine the effect of changing λ_2 on the coefficients while holding λ fixed. We generate the data as follows:

$$x_t \sim \mathsf{N}(0,1), \quad y_t = \beta_0 + \sum_{j=1}^{5} \beta_j(t)x_t + \sigma\mathsf{N}(0,1),$$

where $\sigma = 2$ and $t = 1, \ldots, 200$, and $\beta_j(t), j = 1, \ldots, p$ are described in (3.17). The results can be seen in Figure 3.1–Figure 3.3. The estimated coefficient curves get smoother as we increase λ_2. For $\hat{\beta}_3(t), \hat{\beta}_4(t), \hat{\beta}_5(t)$, some fluctuations are seen around 0. Our procedure recovers the general trend of the true coefficients, whereas least squares does not display the dynamic pattern of the coefficients.

3.4.2 Contamination among coefficients

We now consider a scenario where the true coefficients are contaminated by a random noise. This situation may happen when the coefficient effects are corrupted during a certain time

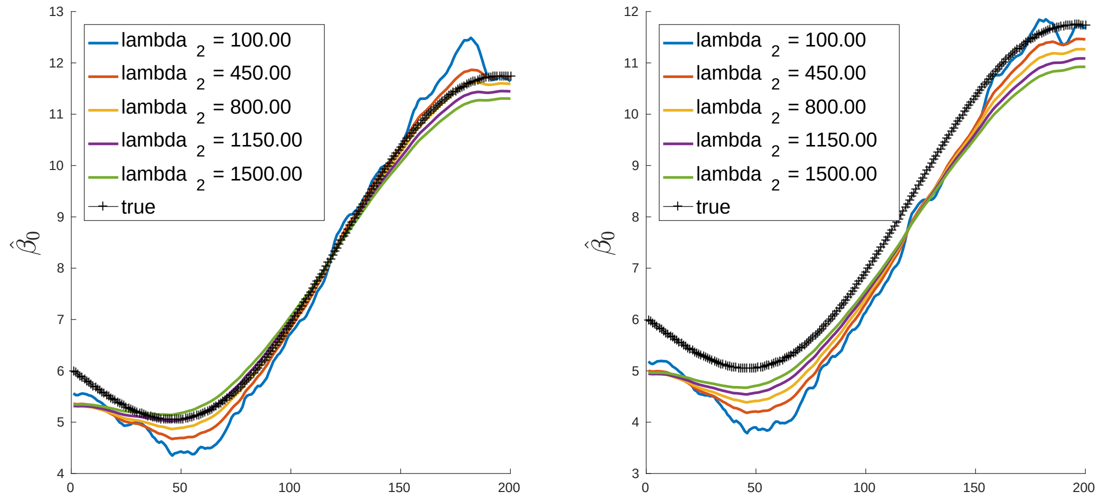

Figure 3.1. The estimated coefficients $\hat{\beta}_0$ of TV-ℓ_1 (colored) and the true coefficients β_0 (black). $\lambda = 0.1$ on the left panel and $\lambda = 1$ on the right panel. As λ_2 increases, the estimated coefficients become smoother. As λ increases, the estimated coefficients become smaller.

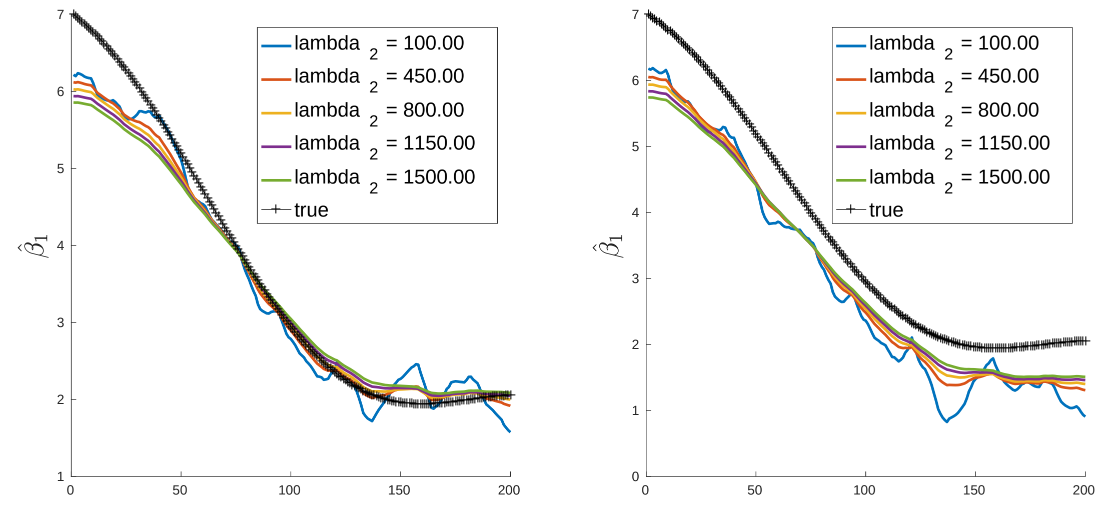

Figure 3.2. The estimated coefficients $\hat{\beta}_1$ of TV-ℓ_1 (colored) and the true coefficients β_1 (black). $\lambda = 0.1$ on the left panel and $\lambda = 1$ on the right panel. As λ_2 increases, the estimated coefficients become smoother. As λ increases, the estimated coefficients become smaller.

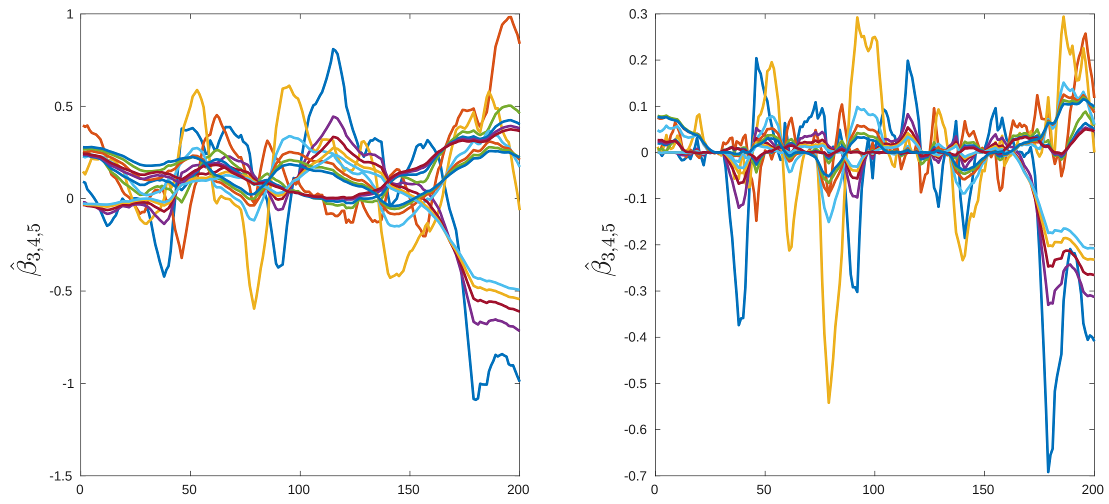

Figure 3.3. The estimated coefficients of TV-ℓ_1 $\hat{\beta}_{3,4,5}$(colored) and the true coefficients $\beta_{3,4,5}$ are all 0. $\lambda = 0.1$ on the left panel and $\lambda = 1$ on the right panel. The fluctuations among estimated coefficients decrease as we increase λ.

period. The coefficients are generated as follows:

$$\begin{aligned}
&\xi \sim \mathsf{Uni}(0,1),\\
&\beta_{ij} = \begin{cases} 2 & \text{for } i = 1,\ldots,19;\ 51,\ldots,60;\ j = 1,\ldots,5,\\ 2+0.5\xi & \text{for } i = 20,\ldots,50;\ j = 1,\ldots,5, \\ 0 & \text{for } i = 1,\ldots,60;\ j = 6,\ldots,30 \end{cases},\\
&x_{ij} \sim \mathsf{N}(0,1), \quad y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_{ij} + \sigma\epsilon, \quad \sigma = 1, \quad \epsilon \sim \mathsf{N}(0,1).
\end{aligned} \tag{3.18}$$

We choose λ and λ_2 by 5-fold cross-validation on the training data set, and we generate our test data as follows:

$$\begin{aligned}
&\beta_{i0} = 1 \text{ for } i = 1,\ldots,20\\
&\beta_{i,j} = 2 \text{ for } i = 1,\ldots,20,\ j = 1,\ldots,10\\
&x_{i,j} \sim \mathsf{N}(0,1), \quad y_i = \beta_{0,i} + \sum_{j=1}^{30} \beta_{i,j}x_{i,j} + \mathsf{N}(0,1) \text{ for } i = 1,\ldots,20.
\end{aligned} \tag{3.19}$$

Table 3.1: Performance comparison between the lasso and TV-ℓ_1.

Methods	Train error	Test error
Lasso	0.87(0.27)	1.54(0.54)
TV-ℓ_1 (mean)	0.86(0.24)	1.49 (0.47)
TV-ℓ_1 (median)	0.86(0.239)	1.49(0.46)

In this situation, we expect the lasso should perform well because the true test data are generated from uncontaminated coefficients. We run the simulation process 20 times according to (3.19) and average the training error and test error. For each simulation, we use (3.5) and (3.6) to obtain estimated response. The results in Table 3.1 show that TV-ℓ_1 performs slightly better than the lasso.

3.5 Application to dynamical systems

In this section, we consider extracting physical laws using a data-driven approach. Researchers in the domain of dynamical systems have recently started using sparsity-promoting techniques to discover governing physical equations from measurement data, under the assumption that the structure of the model is dependent on a few important terms, for example, see Brunton et al. [12]. We consider a dynamical system and show our algorithm can successfully recover the dynamical structure of the model.

Consider $\dot{x}(t) = f(x(t))$, where $x(t) = [x_1(t), \ldots, x_n(t)]^T \in \mathbb{R}^n$ represents the state of the system at time t and the nonlinear function $f(x(t))$ represents the dynamic constraints that define the equations of motion of the system.

Lorenz system Consider a time varying Lorenz system [37]:

$$
\begin{aligned}
\dot{x} &= \sigma(t)(y - x); \quad \sigma(t) = 10 - \sin(t/3), \\
\dot{y} &= x(\rho - z) - y, \\
\dot{z} &= xy - \beta(t)z; \quad \beta(t) = 2 + 0.25\sin(t).
\end{aligned}
\tag{3.20}
$$

The dynamics of the model can be seen in Figure 3.4. The goal is to reconstruct the dynamics of x, y, z without full knowledge of the right-hand side of (3.20). Table 3.2 illustrates how we can construct the data matrix X and y to recover the underlying dynamical system. In

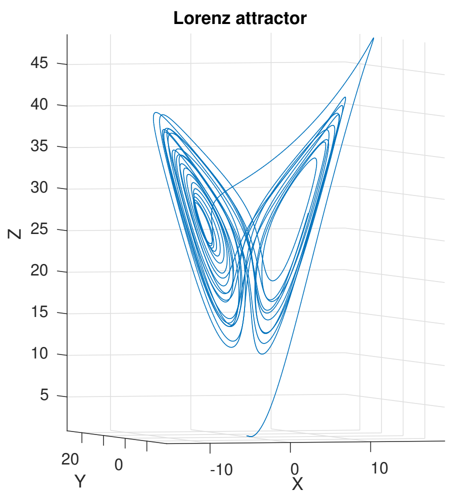

Figure 3.4: Lorenz system

Table 3.2. An example feature matrix and response vector in a dynamical system. The left table represents our feature matrix and the right table represents the response vector.

x	y	z	xy	yz	xz	y^2	z^3	z^4	z^5
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×

$\dot{y}$
×
×
×

order to improve computational efficiency, we can first use the lasso to identify the nonzero coefficients as a heuristic and then use TV-ℓ_1 to refine the selected estimated coefficients. As shown in the left panel of Figure 3.5 – Figure 3.7, we can see that

- $\dot{x}$ could be a function of x, y;
- $\dot{y}$ could be a function of x, y, and xz;
- $\dot{z}$ could be a function of xy, z.

We now apply TV-ℓ_1 to the refined matrices and results can be seen in the right panel of those figures. Different colors represent different values of λ_2 and black lines plot the true coefficients. We see that we can recover the general trend of the estimated coefficients for the dynamical system, whereas linear squares in this situation cannot capture the whole picture. There are many other dynamical systems where the coefficients are function of time. For example, in modeling of population, the common equation for the number of

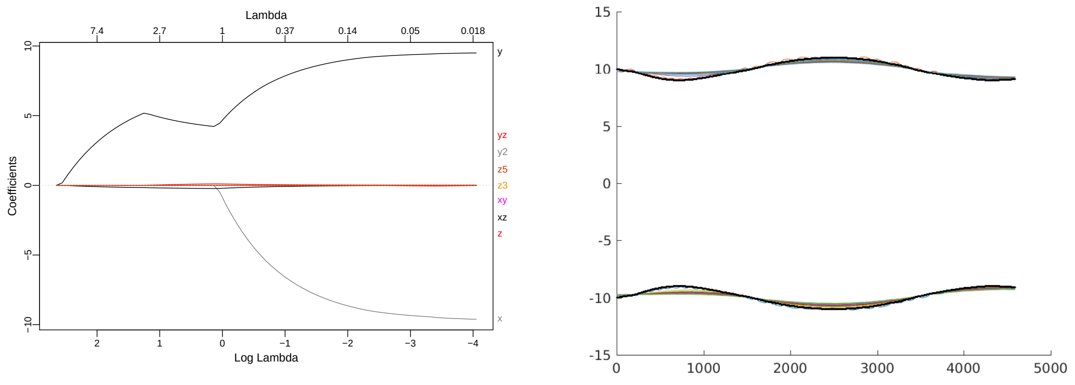

Figure 3.5. The lasso path when regressing $\dot{x}$ onto the data matrix described in the Table 3.2. The estimated coefficients from TV-ℓ_1 are in the right panel.

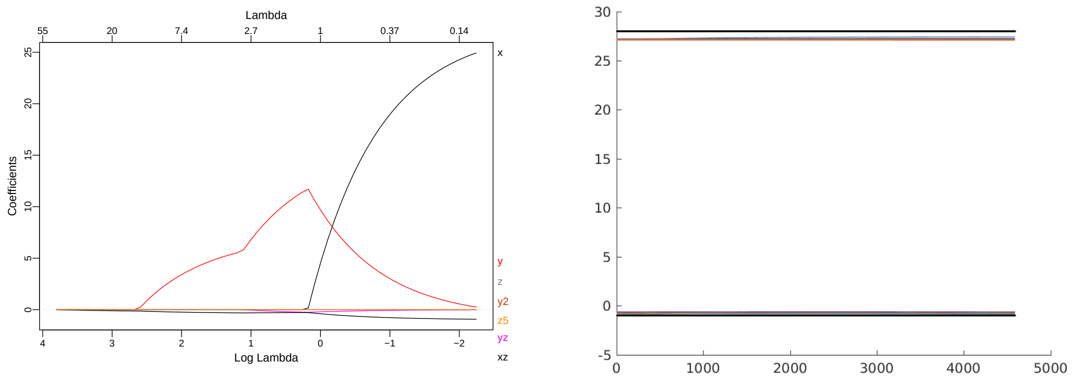

Figure 3.6. The lasso path when regressing $\dot{y}$ onto the data matrix described in the Table 3.2. The estimated coefficients from TV-ℓ_1 are in the right panel.

populations can be written as $dp/dt = rp(1-p/k)$, where p denotes the population, r denotes the population growth rate, and k denotes the carrying capacity (maximum sustainable population). Normally r and k are constant, but recently researchers have pointed out that constant effects may not capture the whole picture of the change of population because r and k change with time. One can apply TV-ℓ_1 to obtain estimates for r and k.

3.6 Discussion

We have proposed ℓ_1 and ℓ_2 regularized versions of the time varying regressions and provided efficient algorithms for the resulting problem. These algorithms have natural

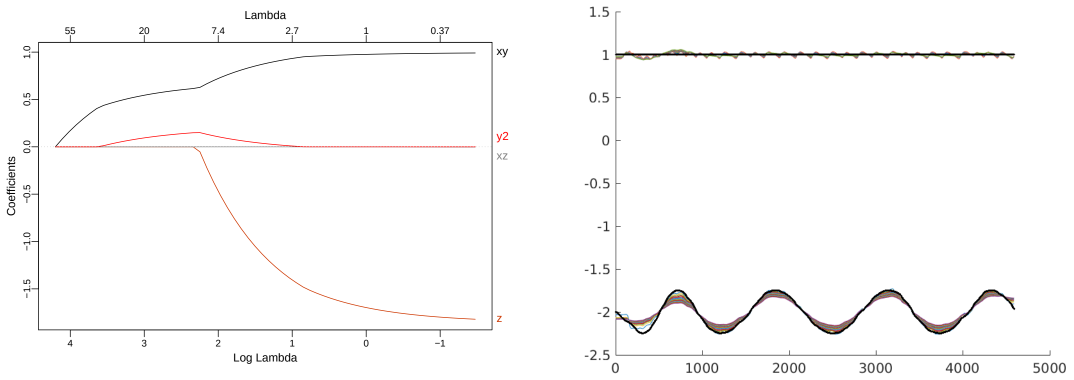

Figure 3.7. The lasso path when regressing $\dot{z}$ onto the data matrix described in the Table 3.2. The estimated coefficients from TV-ℓ_1 are in the right panel.

application to problems where the dynamic patterns are of significance in predictions and linear models cannot capture such information. For future work, we could extend our framework to other generalized linear models.

Chapter 4

Sparse canonical correlation analysis

4.1 Introduction

Correlation measures dependence among two or more random variables. The most popular measure is Pearson's correlation coefficient. For random variables $x, y \in \mathbb{R}$, the population correlation coefficient is defined as $\rho_{x,y} = \mathrm{Cov}(x, y)/(\sqrt{\mathrm{Var}(x)}\sqrt{\mathrm{Var}(y)})$. It is important to remove the variance in random variables x and y by dividing by their standard deviation. We could not emphasize more the importance of this standardization, and we present two toy examples in Table 4.1. Clearly, x and y are more correlated in the left table than the right table even though the covariance between x and y is seemingly much smaller in the left table than the right. This example demonstrates that a large covariance doesn't mean large correlation and a small covariance doesn't small correlation. Moreover, correlation is a unit-less quantity and thus doesn't depend on the units of variables, whereas covariance does. Therefore, correlation is a better notion of dependence than covariance. Canonical correlation studies the correlation between two multidimensional random variables. Let $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$ be random variables with covariances Σ_x, Σ_y and joint covariance Σ_{xy}.

Table 4.1. Covariance Matrix: the correlation between x and y in the left table is $\mathrm{Corr}(x, y) = 0.9$ whereas on the right it is $\mathrm{Corr}(x, y) = \frac{1}{3}$. It is important to remove the variance in random variables x and y before considering the correlation.

Covariance	x	y
x	0.1	0.09
y	0.09	0.1

and

Covariance	x	y
x	0.9	0.3
y	0.3	0.9

In words, we seek linear combinations of x and y such that the resulting values are most correlated. The mathematical definition is

$$\frac{u^T \Sigma_{xy} v}{\sqrt{u^T \Sigma_x u}\sqrt{v^T \Sigma_y v}}. \tag{4.1}$$

Solving (4.1) is easy in low-dimensional settings, i.e., $n \gg p$, because with change of variables $\Sigma_x^{1/2} u = a$ and $\Sigma_y^{1/2} v = b$, (4.1) becomes

$$\underset{a \in \mathbb{R}^p\ b \in \mathbb{R}^q}{\text{maximize}} \quad \frac{a^T \Sigma_x^{-1/2} \Sigma_{xy} \Sigma_y^{-1/2} b}{\sqrt{a^T a}\sqrt{b^T b}}, \tag{4.2}$$

which can be solved using the singular value decomposition of the new matrix $\Sigma_x^{-1/2} \Sigma_{xy} \Sigma_y^{-1/2}$. However, when $p \gg n$, this method is not feasible because Σ_x^{-1} and Σ_y^{-1} cannot be estimated accurately. Moreover, we might want to seek a sparse representation of features in x and y to obtain interpretability of the data.

Let $X \in \mathbb{R}^{n \times p}, Y \in \mathbb{R}^{n \times q}$ be the data matrices. We consider a regularized version of the problem:

$$\begin{array}{ll} \underset{u,v}{\text{minimize}} & -\operatorname{Cov}(Xu, Yv) + \tau_1 |u|_1 + \tau_2 |v|_1 \\ \text{subject to} & \operatorname{Var}(Xu) = 1; \quad \operatorname{Var}(Yv) = 1, \end{array}$$

and since the constraints of the minimization problem are not convex, we further relax it to

$$\begin{array}{ll} \underset{u,v}{\text{minimize}} & -\operatorname{Cov}(Xu, Yv) + \tau_1 |u|_1 + \tau_2 |v|_1 \\ \text{subject to} & \operatorname{Var}(Xu) \le 1; \quad \operatorname{Var}(Yv) \le 1. \end{array} \tag{4.3}$$

The resulting problem is still nonconvex but it is biconvex. This formulation has two advantages. First, it does not require calculation of Σ_X^{-1} and Σ_Y^{-1}. Second, it encourages the sparsity of canonical vectors u and v.

Related Work Though some research has been done on canonical correlation analysis in high-dimensional settings, there are issues we would like to point out:

1. Computationally efficient algorithms. We have not found an existing algorithm to solve (4.3) that scales efficiently.

2. Correct relaxations. An efficient algorithm to find sparse canonical vectors was proposed by Witten et al. [60] and implemented in the R package `PMA`. Their algorithm relaxes the constraints $\mathrm{Var}(Xu) = 1$ and $\mathrm{Var}(Yv) = 1$ to $\|u\|_2 \leq 1$ and $\|v\|_2 \leq 1$, with the assumptions of $\Sigma_x = I$ and $\Sigma_y = I$. The relaxations of the covariance matrices are not realistic in high-dimensional settings, and the resulting correlation is no longer unit-less. Our algorithm relaxes $\mathrm{Var}(Xu) = 1$ to $\mathrm{Var}(Xu) \leq 1$, and $\mathrm{Var}(Yv) = 1$ to $\mathrm{Var}(Yv) \leq 1$. Though we cannot guarantee the constraints are active at a solution, it is often the case.

3. Simulated examples. We consider a variety of simulated examples, including some that are frequently considered in the literature. We also present some examples that are not considered in the literature and are more challenging but whose structure is closer to that of real data sets.

The chapter is organized as follows. Section 4.2 contains motivations and algorithms. Section 4.2.3 contains an algorithm to find rth canonical vectors, though we only focus on estimating the first pair of canonical vectors. We show that finding sparse canonical vectors is equivalent to finding sparse eigenvectors in a special case in Section 4.3. We demonstrate the use of such algorithms on simulated data in Section 4.5 and show detailed comparisons among sparse CCA methods proposed by Gao et al. [24], Witten et al. [60], and Tan et al. [50]. Section 4.8 contains some discussion and directions for future research.

4.2 Sparse canonical correlation analysis

4.2.1 The basic idea

Problem (4.3) is equivalent to solving

$$\underset{u,v}{\text{minimize}} \quad -\mathrm{Cov}(Xu, Yv) + \tau_1\|u\|_1 + \tau_2\|v\|_1 + \mathbb{1}\{u : \mathrm{Var}(Xu) \leq 1\} + \mathbb{1}\{v : \mathrm{Var}(Yv) \leq 1\}, \tag{4.4}$$

where $\mathbb{1}\{\bullet\} = 0$ if $\bullet \leq 1$. Problem (4.4) is biconvex, i.e., if we fix u, the resulting minimization is convex with respect to v, and if we fix v, the minimization is convex with respect to u. Our algorithm seeks a local minimum with the following procedure.

- Repeat until converged:

    1. Fix v, solve for u:

    $$\text{(4.5)} \qquad \underset{u}{\text{minimize}} \quad -\operatorname{Cov}(Xu, Yv) + \tau_1\|u\|_1 + \mathbb{1}\{u : \operatorname{Var}(Xu) \leq 1\}$$

    2. Fix u, solve for v:

    $$\text{(4.6)} \qquad \underset{v}{\text{minimize}} \quad -\operatorname{Cov}(Xu, Yv) + \tau_2\|v\|_1 + \mathbb{1}\{v : \operatorname{Var}(Yv) \leq 1\}$$

In Section 4.2.2, we describe how to solve subproblems (4.5) and (4.6).

The method proposed by Witten et al. [60] is based on the formulation

$$\begin{array}{ll} \underset{u,v}{\text{minimize}} & -\operatorname{Cov}(Xu, Yv) + \tau_1\|u\|_1 + \tau_2\|v\|_1 \\ \text{subject to} & \|u\|_2 \leq 1; \quad \|v\|_2 \leq 1, \end{array}$$

in which the covariance matrices X^TX and Y^TY have been replaced by identity matrices. They also used an alternating minimization approach, and with one variable fixed, the other variable has a closed form solution. As a result, this formulation can be solved very efficiently. However, a simple example shows that the solution can be inaccurate and non-sparse.

Example 1: We generate our data as follows:

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim \mathsf{N}(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_Y \end{pmatrix}),$$

where

$$(\Sigma_x)_{ij} = (\Sigma_y)_{ij} = 0.9^{|i-j|}, \qquad \Sigma_{XY} = \Sigma_X(u_1\rho v_1^T)\Sigma_Y,$$

u_1 and v_1 are specified sparse canonical vectors with the numbers of nonzero elements chosen to be 5, 5, respectively. The locations of nonzero elements are chosen randomly and normalized with respect to the true covariance of X and Y, i.e., $u_1^T \Sigma_X u_1 = 1$ and $v_1^T \Sigma_Y v_1 = 1$.

We first present a proposition from Chen et al. [15].

Proposition 1. *Consider the problem*

$$\begin{array}{ll} \underset{a,b}{\textit{maximize}} & a^T \Sigma_{xy} b \\ \textit{subject to} & a^T \Sigma_x a = 1; \quad b^T \Sigma_y b = 1. \end{array} \tag{4.7}$$

When Σ_{xy} has rank 1, the solution (up to sign jointly) of (4.7) *is (θ, η) if and only if the covariance structure between X and Y can be written as $\Sigma_{xy} = \lambda \Sigma_x \theta \eta^T \Sigma_y$, where $0 \le \lambda \le 1$, $\theta^T \Sigma_x \theta = 1$, and $\eta^T \Sigma_y \eta = 1$. In other words, the correlation between $a^T X$ and $b^T Y$ is maximized by* $\mathrm{Corr}(X\theta, Y\eta)$*, and λ is the canonical correlation between X and Y. More generally, if Σ_{xy} has rank r, the solution of* (4.7) *is (θ_1, η_1) if and only if the covariance structure between X and Y can be written $\Sigma_{xy} = \Sigma_x (\sum_{i=1}^r \lambda_i \theta_i \eta_i^T) \Sigma_y$.*

The sample size is $n = 500$ and $p_u = p_v = 600$, $\rho = 0.9$. We denote the solution of Witten et al. [60] by $\hat{u}_w, \hat{v}_w$, and our solution by $\hat{u}_1, \hat{v}_1$. We have two main goals when we solve for canonical vectors: maximizing the correlation while maintaining sparsity in the canonical vectors. A common way to measure the performance is to use the Pareto curve, seen in Figure 4.1 and Figure 4.2. The left panel traces

$$x : \frac{-\hat{u}^T X^T Y \hat{v}}{\sqrt{\hat{u}^T X^T X \hat{u}} \sqrt{\hat{v} Y^T Y \hat{v}}} \text{ vs. } y : \|\hat{u}\|_1 + \|\hat{v}\|_1,$$

and the right panel traces

$$x : \frac{-\hat{u}^T \Sigma_{XY} \hat{v}}{\sqrt{\hat{u}^T \Sigma_X \hat{u}} \sqrt{\hat{v}^T \Sigma_Y \hat{v}}} \text{ vs. } y : \|\hat{u}\|_1 + \|\hat{v}\|_1.$$

We prefer a point that is close to the left corner of the Pareto curve, because it represents a solution consisting of sparse canonical vectors and also achieves the maximum correlation.

The left panel of Figure 4.1 is a plot of the estimated correlation $\hat{u}^T X^T Y \hat{v}$ versus the sum of $\|\hat{u}\|_1$ and $\|\hat{v}\|_1$, averaged over 100 simulations. The right panel plots estimated

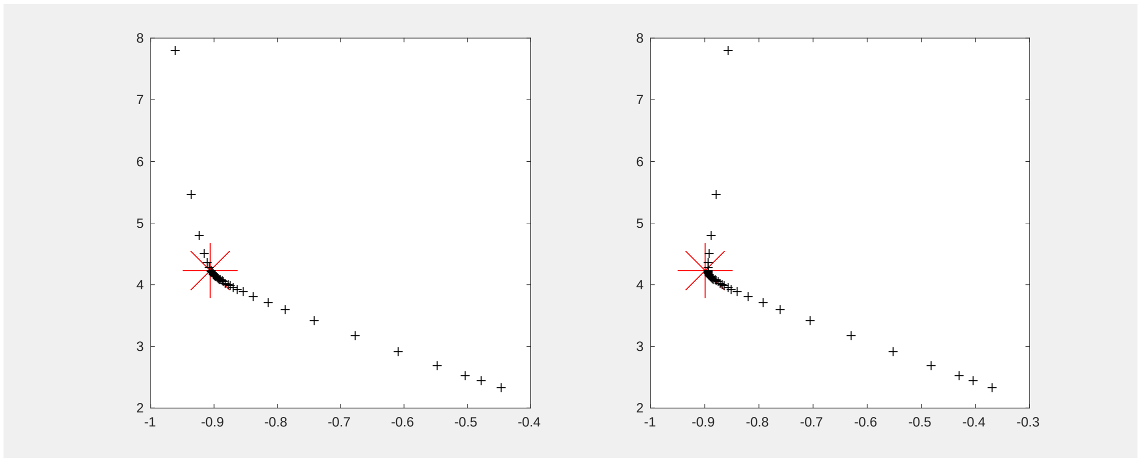

Figure 4.1. Pareto curves of our estimators. The left panel plots the estimated correlation $\hat{u}^T X^T Y \hat{v}$ versus the sum of $\|\hat{u}\|_1$ and $\|\hat{v}\|_1$, averaged over 100 simulations. The red dot corresponds to $(u^T X^T Y v,\ \|u\|_1 + \|v\|_1)$. The right panel plots the estimated correlation $\hat{u}^T \Sigma_{XY} \hat{v}$ versus the sum of $\|\hat{u}\|_1$ and $\|\hat{v}\|_1$, averaged over 100 simulations. The red dot corresponds to $(u^T \Sigma_{XY} v,\ \|u\|_1 + \|v\|_1)$. Note that the red dot (true solution) is on the Pareto curve in both cases, which means that our algorithm can achieve this optimal value with the right choice of regularizers.

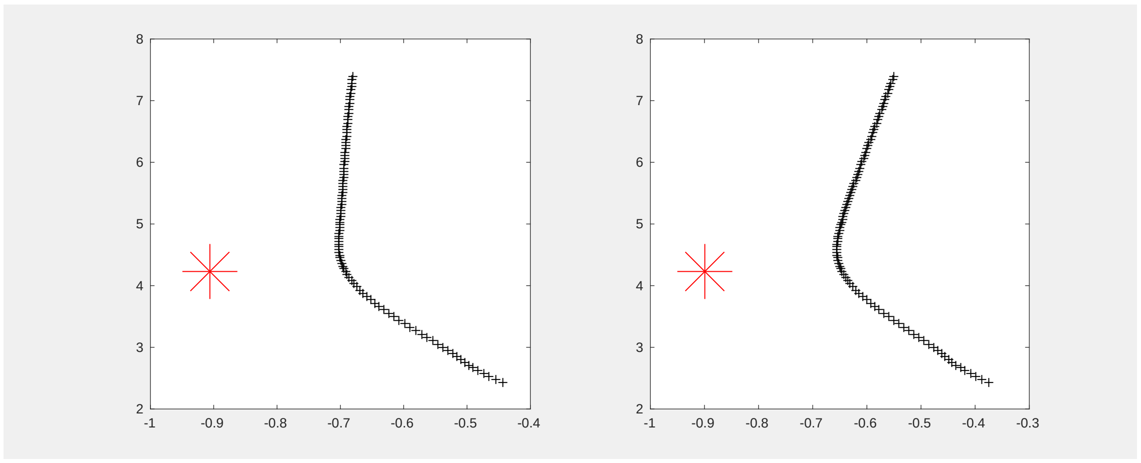

Figure 4.2. Pareto curves of Witten et al. [60]. The left panel plots the estimated correlation $\hat{u}^T X^T Y \hat{v}$ versus the sum of $\|\hat{u}\|_1$ and $\|\hat{v}\|_1$, averaged over 100 simulations. The red dot corresponds to $(u^T X^T Y v,\ \|u\|_1 + \|v\|_1)$. The Right panel plots the estimated correlation $\hat{u}^T \Sigma_{XY} \hat{v}$ versus the sum of $\|\hat{u}\|_1$ and $\|\hat{v}\|_1$, averaged over 100 simulations. The red dot corresponds to $(u^T \Sigma_{XY} v,\ \|u\|_1 + \|v\|_1)$. Note that the red dot is not on the Pareto curve in both cases, which means that the algorithm cannot achieve this optimal value with any choice of regularizers.

correlation $\hat{u}^T\Sigma_{XY}\hat{v}$ versus the sum of $||\hat{u}||_1$ and $||\hat{v}||_1$, averaged over 100 simulations. Note that we replace the sample covariance with the true covariance. From both panels, with the right choice of regularizers, we see that our algorithm can achieve the optimal values. However, as shown in Figure 4.2, the solutions of Witten et al. [60] are very far from the true solution. The red dots are not on their solutions' path, meaning that their results do not achieve the optimal value with any choices of regularizer.

4.2.2 Algorithmic details

Linearized alternating direction minimization method

We assume that the data matrices X and Y are centered and divided by the square root of the number of samples. We write (4.5) as

$$\underset{u}{\text{minimize}} - u^T X^T Y v + \tau_1 ||u||_1 + \mathbb{1}\{u : ||Xu||_2 \leq 1\}, \tag{4.8}$$

where $\hat{\Sigma}_x = X^T X, \hat{\Sigma}_y = Y^T Y, \hat{\Sigma}_{xy} = X^T Y$. Letting $z = Xu$, we have

$$\begin{aligned} &\underset{u,z}{\text{minimize}} \quad \underbrace{-u^T X^T Y v + \tau_1 ||u||_1}_{f(u)} + \underbrace{\mathbb{1}\{||z||_2 \leq 1\}}_{g(z)} \\ &\text{subject to} \quad Xu = z. \end{aligned} \tag{4.9}$$

We can use the Linearized Alternating Direction Method of Multipliers [40] (ADMM) to solve this problem. ADMM minimizes the augmented Lagrangian by solving for each variable and dual variable one by one until convergence. A detailed derivation is in the Appendix, and the complete algorithm can been seen in Algorithm 8.

TFOCS

Another approach to solving problem (4.5) is to use TFOCS [7]. Since v is fixed, letting $c = v^T Y^T X$ and minimizing the objective function is equivalent to minimizing

$$- cu/\tau_1 + ||u||_1 + \mathbb{1}\{u : ||Xu||_2 \leq 1\}. \tag{4.10}$$

Instead of optimizing (4.10), we minimize

$$\|u\|_1 + \frac{1}{2}\mu\|u - (u_{old} + \frac{1}{\tau_1\mu}c)\|_2^2 + \mathbb{1}\{u : \|Xu\|_2 \le 1\}$$

for $\mu > 0$. Intuitively, we solve (4.10) without going too far from the current approximation. This formulation can be solved using `tfocs_SCD`.

Algorithm 8 ADMM for Sparse CCA

```
1:  function CCA($X, Y$)
2:      Initialize $u^0$, $v^0$, $z^0$, $\xi^0$
3:      while not converged do
4:          Fix $v^k$
5:          while not converged do
                $u^{k+1} \leftarrow \mathbf{prox}_{\mu f}(u^k - \frac{\mu}{\lambda}X^T(Xu^k - z^k + \xi^k))$
                $z^{k+1} \leftarrow \mathbf{prox}_{\lambda g}(Xu^{k+1} + \xi^k)$
                $\xi^{k+1} \leftarrow \xi^k + Xu^{k+1} - z^{k+1}$
6:          end while
7:          Fix $u^{k+1}$
8:          while not converged do
                $v^{k+1} \leftarrow \mathbf{prox}_{\mu f}(v^k - \frac{\mu}{\lambda}Y^T(Yv^k - z^k + \xi^k))$
                $z^{k+1} \leftarrow \mathbf{prox}_{\lambda g}(Yv^{k+1} + \xi^k)$
                $\xi^{k+1} \leftarrow \xi^k + Yv^{k+1} - z^{k+1}$
9:          end while
10:     end while
11: end function
```

4.2.3 The remaining canonical vectors

Given the first $r-1$ canonical vectors $U = \begin{pmatrix} u_1 & \dots & u_{r-1} \end{pmatrix}$ and $V = \begin{pmatrix} v_1 & \dots & v_{r-1} \end{pmatrix}$, we consider solving for the r-th canonical vectors using the problem

$$\begin{array}{ll} \underset{u,v}{\text{minimize}} & -u^T X^T Y v + \tau_1 \|u\|_1 + \tau_2 \|v\|_1 + \mathbb{1}\{u : \|Xu\|_2 \le 1\} + \mathbb{1}\{v : \|Yv\|_2 \le 1\} \\ \text{subject to} & U^T X^T X u = 0; \quad V^T Y^T Y v = 0. \end{array}$$

We again use the alternating minimization approach. Fixing v, we get $\hat{u}$ by solving

$$\begin{array}{ll} \underset{u,z}{\text{minimize}} & -u^T X^T Y v + \tau_1 |u|_1 + \mathbb{1}\{z : \|z\|_2 \le 1\} \\ \text{subject to} & Xu = z; \quad U^T X^T X u = 0_{r-1}, \end{array}$$

and fixing $\hat{u}$, we get $\hat{v}$ by solving

$$\begin{array}{ll} \underset{v,z}{\text{minimize}} & -u^T X^T Y v + \tau_1 |v|_1 + \mathbb{1}\{z : \|z\|_2 \le 1\} \\ \text{subject to} & Yv = z; \quad V^T Y^T Y v = 0_{r-1}. \end{array}$$

The constraints can be combined as

$$\begin{pmatrix} X \\ U^T X^T X \end{pmatrix} u - \begin{pmatrix} I \\ 0 \end{pmatrix} z = 0; \quad \begin{pmatrix} Y \\ V^T Y^T Y \end{pmatrix} v - \begin{pmatrix} I \\ 0 \end{pmatrix} z = 0. \tag{4.11}$$

Letting $\tilde{X} = \begin{pmatrix} X \\ U^T X^T X \end{pmatrix}$ and $\tilde{Y} = \begin{pmatrix} Y \\ V^T Y^T Y \end{pmatrix}$, we see that

$$u^T \tilde{X}^T \tilde{Y} v = u^T X^T Y v + u^T X^T X U V^T Y^T Y v = u^T X^T Y v,$$

where the last equality is due to the fact that the constraints (4.11) are satisfied. To get the r-th canonical vectors, we can use Algorithm 8 with the new matrices $\tilde{X}$ and $\tilde{Y}$ (CCA($\tilde{X}$,

$\tilde{Y}$)) with a modification of

$$\mathbf{prox}_{\lambda g}(x) = \begin{cases} \mathbf{prox}_{\lambda g}(x[1:n]) & \text{for } x[1:n] \\ 0 & \text{for } x[n+1:n+r] \end{cases},$$

where $x[1:n]$ denotes elements from 1 to n in x, in order to enforce the constraints $U^T X^T Xu = 0$ and $V^T Y^T Yv = 0$.

4.2.4 A bridge for the covariance matrix

As mentioned, Witten et al. [60] proposed to replace the covariance matrix with an identity matrix. Since their formulation can be solved efficiently, it is of interest to investigate the relation between our method and theirs. Therefore, we consider a covariance matrix of the form

$$\alpha_x X^T X + (1-\alpha_x)\mathbf{I}_{p_u,p_u}$$

and a similar one for Y. The term $||Xu||_2^2$ gives

$$u^T(\alpha_x X^T X + (1-\alpha_x)\mathbf{I})u = \alpha_x ||Xu||_2^2 + (1-\alpha_x)||u||_2^2 = ||\underbrace{\begin{pmatrix} \sqrt{\alpha_x} X \\ \sqrt{1-\alpha_x}\mathbf{I}_{p_u,p_u} \end{pmatrix}}_{(*)} u||_2^2.$$

This form can be minimized using Algorithm 8 by changing the linear operator to the matrix $(*)$ above. To see how solutions change from Witten et al. [60] to our solution, we can use the above to see the path using different choices of α_x, α_y.

4.2.5 A predictive view

We consider a scenario where one of the matrices X and Y is low-dimensional and we show how to use CCA as a prediction tool under a normal distribution assumption. Without loss of generality, we may assume that $X \in \mathbb{R}^{n\times p}, n > p$ and $Y \in \mathbb{R}^{n\times q}, n < q$. Assume the data matrices X and Y are generated from

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathsf{N}(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_Y \end{pmatrix}).$$

The conditional distribution of $X|Y$ is given by

$$X|Y \sim \mathsf{N}(\mu_1 + \Sigma_{XY}\Sigma_Y^{-1}(Y - \mu_2), \Sigma_X - \Sigma_{XY}\Sigma_Y^{-1}\Sigma_{YX}).$$

From Proposition 1, $\hat{\Sigma}_{XY} = \hat{\Sigma}_X \hat{\rho}\hat{u}\hat{v}^T\hat{\Sigma}_Y$, which is equivalent to $\hat{\Sigma}_{XY}\hat{\Sigma}_Y^{-1} = \hat{\Sigma}_X\hat{\rho}\hat{u}\hat{v}^T$, where $\hat{\Sigma}_x$ can be estimated from sample data X; see for example, [43], [13]. If we are given a new observation y, we can estimate x as follows:

$$\mathbb{E}[x|y] = \mu_1 + \hat{\Sigma}_x\hat{\rho}\hat{u}\hat{v}^T(y - \mu_2).$$

4.2.6 Semidefinite programming approach

We now show that a variation of problem (4.3) can be solved using a semidefinite programming approach. This idea is borrowed from d'Aspremont et al. [17]'s approach to sparse principal components analysis with some modifications. The problem

$$\begin{array}{ll} \underset{u,v}{\text{minimize}} & -u^T X^T Y v \\ \text{subject to} & u^T X^T X u = 1; \quad v^T Y^T Y v = 1 \\ & \text{card}(v) \le k_v; \quad \text{card}(u) \le k_u \end{array} \tag{4.12}$$

where card(u) denotes the number of non-zero elements of a vector u, can be relaxed to

$$\begin{array}{ll} \underset{H}{\text{minimize}} & \textbf{trace}(YX^T H_{12}) \\ \text{subject to} & H \succeq 0, \quad \textbf{trace}(X^T X H_{11}) = 1, \quad \textbf{trace}(Y^T Y H_{22}) = 1; \\ & 1^T|H_{11}|1 \le k_1 \quad 1^T|H_{22}|1 \le k_2, \\ & 1^T|H_{12}|1 \le \sqrt{k_1 k_2}, \end{array}$$

where $H = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$, $H_{11} = uu^T, H_{12} = uv^T, H_{21} = vu^T, H_{22} = vv^T$, and 1 is a vector of 1s. We then relax it to

$$\begin{array}{ll} \underset{H}{\text{minimize}} & \mathbf{trace}(YX^T H_{12}) - \rho 1^T |H| 1 \\ \text{subject to} & H \succeq 0, \quad \mathbf{trace}(X^T X H_{11}) = 1, \quad \mathbf{trace}(Y^T Y H_{22}) = 1, \end{array}$$

where $\rho > 0$ is a regularizer constant.

There are many other relaxations of problem (4.12). However, semidefinite programming problems can be very computationally expensive using existing methods, especially when p is much greater than n. Therefore, we do not compute sparse canonical vectors using this formulation. Finding an efficient algorithm for this problem would be an interesting direction for future research.

4.3 A special case

In this section, we consider a special case in which the covariance matrices of x and y are the identity. With $\Sigma_x = \Sigma_y = I$, suppose the covariance matrix has singular value decomposition $\Sigma_{xy} = U\Lambda V^T$, $U \in \mathbb{R}^{p\times k}, V \in \mathbb{R}^{q\times k}$, and $\Lambda \in \mathbb{R}^{k\times k}$ is diagonal so that Σ_{xy} has rank k. We show that sparse CCA is similar to a sparse principal component analysis. Note that $U^T U = I_k$ and $V^T V = I_k$. The joint distribution of x and y is

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathsf{N}(\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} I & U\Lambda V^T \\ V\Lambda U^T & I \end{pmatrix}).$$

Proposition 2. *Estimation of u and v can be obtained using spectral decomposition and thus we can use sparse principal components software to obtain u and v.*

Proof Let $\Sigma = A + I = \begin{pmatrix} 0 & U\Lambda V^T \\ V\Lambda U^T & 0 \end{pmatrix} + \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$, where

$$A = \begin{pmatrix} 0 & U\Lambda V^T \\ V\Lambda U^T & 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} U \\ V \end{pmatrix}\Lambda \begin{pmatrix} U^T & V^T \end{pmatrix} - \frac{1}{2}\begin{pmatrix} U \\ -V \end{pmatrix}\Lambda \begin{pmatrix} U^T & -V^T \end{pmatrix}.$$

Let U_i, V_i denote the ith columns of U and V respectively, and denote

$$W_i = \frac{1}{\sqrt{2}}\begin{pmatrix} U_i \\ V_i \end{pmatrix}, \qquad W_{k+i} = \frac{1}{\sqrt{2}}\begin{pmatrix} U_i \\ -V_i \end{pmatrix}$$

for $i = 1, \ldots, k$. Note that $W_i^T W_j = \mathbb{I}(i = j)$, for $i, j = 1, \ldots, 2k$. Let $\{W_i\}_{i=2k+1}^{p+q}$ be an orthonormal set of vectors orthogonal to $\{W_i\}_{i=1}^{2k}$. Then the matrix $\Sigma = A + I$ has the spectral decomposition

$$\Sigma = \sum_{i=1}^{k}(1 + \Lambda_{i,i})W_i W_i^T + \sum_{i=k+1}^{2k}(1 - \Lambda_{i-k,i-k})W_i W_i^T + \sum_{i=2k+1}^{p_u+p_v} W_i W_i^T.$$

Therefore, Σ can be thought of as a spiked covariance matrix, where the signal to noise ratio (SNR) can be interpreted as $1 + \min_i \Lambda_{i,i}$. We know that in the high-dimensional regime, if SNR $\geq \sqrt{p/n}$ we can recover u and v even if u and v are not sparse. However, if SNR $< \sqrt{p/n}$, we need to enforce sparsity in u and v; see Baik et al. [2] and Paul [42] for details. □

We can see from Proposition 2 that if the covariance matrices of x and y are the identity, or act more or less like identity matrices, finding canonical vectors can be roughly viewed as finding sparse principal components. Therefore, in this case, estimating canonical vectors is roughly as hard as computing sparse eigenvectors.

4.4 Equivalence to Fisher's linear discriminant analysis

Fisher's linear discriminant analysis (LDA) [21] is a classification tool and was designed to find projections of the data in order to maximize the between-group variance relative to within-group variance. Let X be the data matrix with n rows x_i^T that belong to K classes, and w_c be the class c, for $c = 1, \ldots, K$. Mathematically, it solves

$$\begin{aligned} &\underset{a}{\text{maximize}} && a^T S_B a \\ &\text{subject to} && a^T S_W a = 1, \end{aligned} \tag{4.13}$$

Table 4.2. Error comparison between our method and `msda`. All values are the median of 100 simulations. Our method achieves the same classification error as `msda`. The numbers of non-zeros are 33 and 33 in the estimated discriminant vectors and 26 and 36 in the estimated canonical vectors.

Test error	`msda`	our method
Error %	8.3	8.3
Std Dev%	0.039	0.04
Non-zeros	(33,33)	(26,36)

where $S_B = \sum_{c=1}^{K} n_c(\bar{x}^{(c)} - \bar{x})(\bar{x}^{(c)} - \bar{x})^T$ amd $S_W = \sum_{c=1}^{K} \sum_{x \in w_c}(x - \bar{x}^{(c)})(x - \bar{x}^{(c)})^T$, $\bar{x}^{(c)} = \frac{1}{n_c} \sum_{x_i \in w_c, i=1,\ldots,n} x_i$ and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Many researchers have noticed that CCA is equivalent to LDA when one of the matrices X or Y is written in certain forms and LDA is thus viewed as a special case of CCA. The discovery is traced back to Bartlett [5]. Sun and Chen [49] give discussions of why LDA is equivalent to CCA when Y represents the class labels of the corresponding observations in X, and Barker and Rayens [3] provide detailed proofs for some cases. One representation is as follows: let Y_i be the ith row of matrix Y,

$$Y_i = \begin{pmatrix} Y_{i1} & \ldots & Y_{iK} \end{pmatrix}; \quad Y_{ij} = \begin{cases} 1 & \text{if } i \in w_j \\ 0 & \text{otherwise} \end{cases}, \text{ for } j = 1, \ldots, K.$$

We examine the performance of our method on the GDS1615 dataset with the method proposed by Mai et al. [38] and implemented as the R package `msda`. `msda` performs high-dimensional multi-class linear discriminant analysis and has the best performance on this dataset. The dataset contains 22283 gene expression levels of 127 people, either normal or with Crohn's disease, or with ulcerative colitis. It is publicly available from Gene Expression Omnibus with number GDS1615. The dataset was pre-processed and the details can be seen in the paper [38]. The final dataset contains 127 people with 127 gene expression levels. We divide the data into a training set and test set, with a ratio of roughly 2 to 1. 10-fold cross-validation was performed on the training set for both methods to choose the best parameter. The result on the test set can be seen in Table 4.2. Our sparse CCA algorithm does as well as `msda` on this data. Figure 4.3 plots the result from applying our method, where $\text{xvar}_1 = X\hat{u}_1$ and $\text{xvar}_2 = X\hat{u}_2$, and $\hat{u}_1, \hat{u}_2$ are the first and second sparse canonical vectors of X. The numbers of nonzero elements for $\hat{u}_1, \hat{u}_2$ are 26, 39. Most observations are well separated, as seen in the figure.

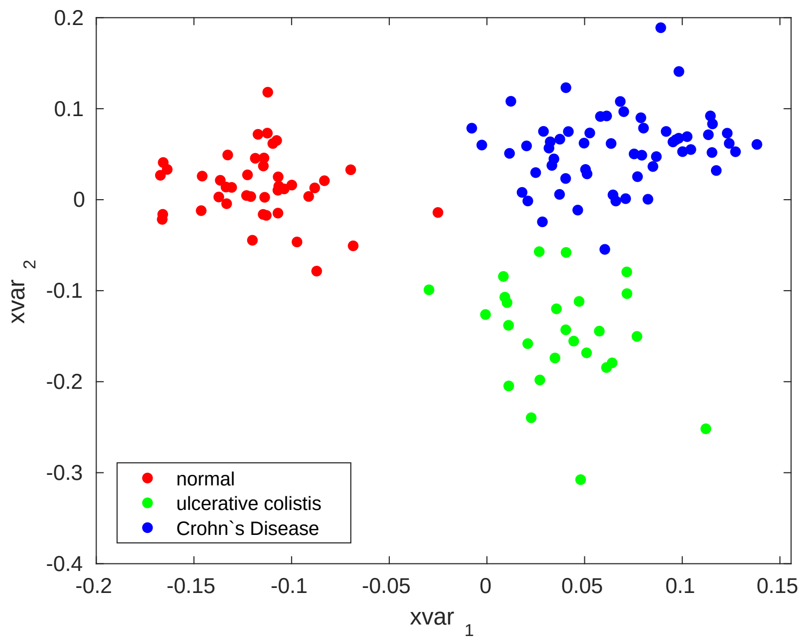

Figure 4.3. Sparse CCA applied to GDS1615 dataset. $\text{xvar}_1 = X\hat{u}_1$ and $\text{xvar}_2 = X\hat{u}_2$ where $\hat{u}_1, \hat{u}_2$ are the first and second sparse canonical vectors of X. The numbers of nonzero elements of $\hat{u}_1$ and $\hat{u}_2$ are 26, 39. Most observations are well separated by our method.

4.5 Simulated data

In this section, we carefully analyze different cases of covariance structure of x and y and compare the performance of our method with other methods. We first explain how we generate the data.

Let $X \in \mathbb{R}^{n\times p}$ and $Y \in \mathbb{R}^{n\times q}$ be data generated from the model

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim \mathsf{N}(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{pmatrix}),$$

where $\Sigma_{xy} = \rho\Sigma_x uv^T\Sigma_y$, where u and v are the true canonical vectors, and ρ is the true canonical correlation. We would like to estimate u and v from the data matrices X and Y. We compare our method with other methods available on different choices of triplets: (n, p, q), where n is the number of samples, p is the number of features in X, and q is the number of features in Y. In order to measure the discrepancy of the estimated $\hat{u}$, $\hat{v}$ from the true u and v, we use the sine of the angle between $\hat{u}$ and u, $\hat{v}$ and v from Johnstone and Lu [36]:

$$\textbf{Loss}(\hat{v}, v) = \min(\|\hat{v} - v\|_2^2, \|\hat{v} + v\|_2^2), \tag{4.14}$$

where $\|\hat{v}\|_2 = \|v\|_2 = 1$.

4.5.1 Identity-like covariance models

In the sparse canonical correlation analysis literature, structured covariances of x and y are thoroughly investigated. For example, the covariance of x may be the identity or Toeplitz, or have a sparse inverse covariance. From the plot of the covariances matrix in Figure 4.4 and Figure 4.5, we can see that Toeplitz and sparse inverse covariances act more or less like identity matrices. Since the covariances of x and y act more or less like identity matrices, as discussed previously, computing u and v is roughly as hard as computing sparse eigenvectors. In other words, the covariance of x and y does not change the signal in u and v much and as a result, the signal in Σ_{xy} is very sparse. In this case, an initial guess is very important. We propose the following procedure:

1. Denoise the matrix X^TY by soft thresholding the matrix elementwise. Call the

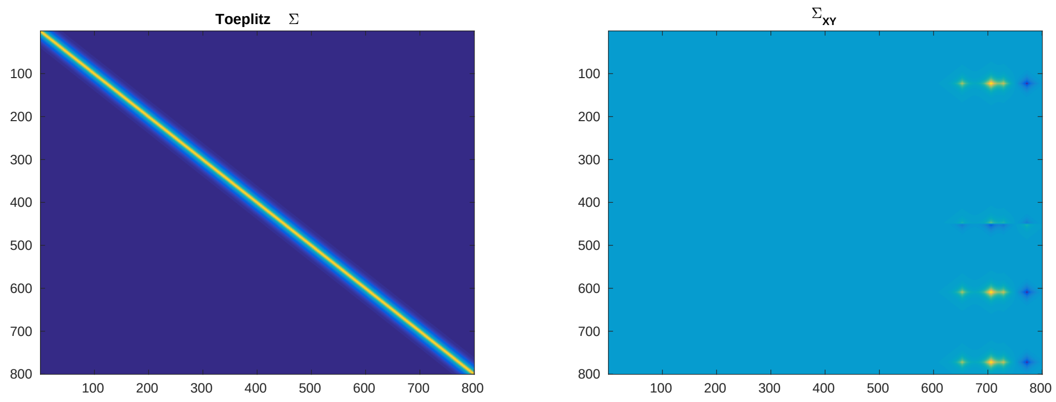

Figure 4.4. Toeplitz matrices with $\sigma_{ij} = 0.9^{|i-j|}$. We see that even though it is not exactly an identity matrix, the general structure does look like one.

resulting matrix S_{xy}.

2. Obtain an initial guess as follows:

   (a) Take the singular value decomposition $S_{xy} = \hat{U}\hat{S}\hat{V}^T$.

   (b) Normalize each column u_i, v_i in $\hat{U}$ and $\hat{V}$ by $u_i \leftarrow \frac{u_i}{\sqrt{u_i^T(X^TX)u_i}}$ and $v_i \leftarrow \frac{v_i}{\sqrt{v_i^TY^TYv_i}}$. Denote the resulting matrices by $\tilde{U}$ and $\tilde{V}$.

   (c) Calculate $\tilde{D} = \tilde{U}^TX^TY\tilde{V}^T$. Choose the index k where the maximum diagonal element of $\tilde{D}$ is obtained. Obtain the initial guess u_k and v_k.

3. Use the initial guess u_k and v_k to start the alternating minimization algorithm.

We consider three types of covariance matrix in this category: Toeplitz, identity, and sparse inverse matrices.

1. $\Sigma = I_p$.

2. $\Sigma = (\sigma_{ij})$, where $\sigma_{ij} = 0.9^{|i-j|}$ for all $i, j \in p, q$. Here Σ are Toeplitz matrices. See the plot of the Toeplitz matrix and its corresponding $\Sigma_{xy} = \Sigma_x\rho uv^T\Sigma_y$. We see that though it is not an identity matrix, it behaves more or less like one. Note that the smaller the Toeplitz constant is, the more it looks like an identity matrix.

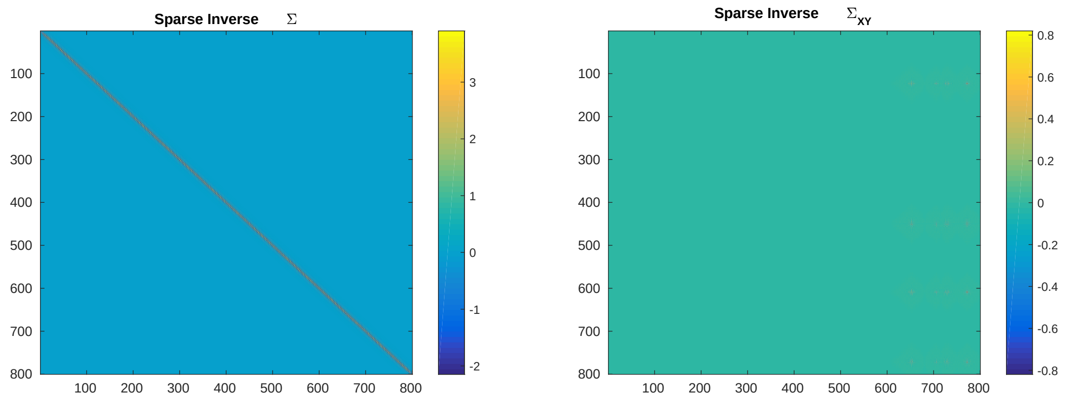

Figure 4.5. Sparse inverse matrix. $\Sigma = (\frac{\sigma^0_{ij}}{\sqrt{\sigma^0_{ii}\sigma^0_{jj}}})$. $\Sigma^0 = (\sigma^0_{ij}) = \Omega^{-1}$ where $\Omega = (\omega_{ij})$ with $\omega_{ij} = \mathbb{1}_{\{i=j\}} + 0.5 \times \mathbb{1}_{\{|i-j|=1\}} + 0.4 \times \mathbb{1}_{\{|i-j|=2\}}, i, j \in [p]$. The resulting Ω is very sparse.

3. $\Sigma = (\sigma^0_{ij})$, where $(\sigma^0_{ij}) = \Omega^{-1}$ and $\Omega = (\omega_{ij})$ with

$$\omega_{ij} = \mathbb{1}_{\{i=j\}} + 0.5 \times \mathbb{1}_{\{|i-j|=1\}} + 0.4 \times \mathbb{1}_{\{|i-j|=2\}}, i, j \in [p].$$

The resulting Ω is very sparse and thus Σ_x and Σ_y have sparse inverse matrices.

In each example, we simulate 100 data sets X and Y in order to average our performance. We set the number of nonzeros in u and v to be 5. The indices of nonzeros are randomly chosen. We vary the number of nonzeros in the next comparison. For each simulation, we have a sequence of regularizers τ_u and τ_v to choose from. For simplicity, we choose the best τ_u and τ_v such that the estimated $\hat{u}$ and $\hat{v}$ minimize the **Loss** defined above in every method.

We present our results in Table 4.3–4.5. Some notation is presented in the tables and we now explain it here. $\hat{\rho}$ is the estimated canonical correlation between data X and Y; $e_u = \textbf{Loss}(\hat{u}, u)$; and $e_v = \textbf{Loss}(\hat{v}, v)$. We compare our results with the methods proposed by Witten et al. [60] and Gao et al. [24]. Since we are not able to run the code from Tan et al. [50] very efficiently, we compare our method with their approach in the next section. In order to compare them in the same units, we calculate the estimates of each method and then normalize them by $X\hat{u}$ and $Y\hat{v}$ respectively. We then normalize the estimates to have norm 1. We report the estimated canonical correlation, loss of u, and loss of v in the format (ρ, e_u, e_v) for each method in all tables. We see that the SCCA method proposed by Gao

Table 4.3. Error comparison for identity matrices. We use a format of $(\hat{\rho}, \mathbf{Loss}(\hat{u}), \mathbf{Loss}(\hat{v}))$ to represent each method's result. Our method performed similarly to SCCA.

(n, p, q)	Our method	SCCA	PMA
(400, 800,800)	(0.90,0.056,0.062)	(0.90,0.060,0.066)	(0.76, 0.26,0.27)
(500, 600, 600)	(0.90,0.05,0.056)	(0.90, 0.053, 0.057)	(0.61,0.15,0.16)
(700, 1200,1200)	(0.90,0.045, 0.043)	(0.90,0.045, 0.043)	(0.6, 0.143,0.144)

Table 4.4. Error comparison for Toeplitz matrices. We use a format of $(\hat{\rho}, \mathbf{Loss}(\hat{u}), \mathbf{Loss}(\hat{v}))$ to represent each method's result.

(n, p, q)	Our method	SCCA	PMA
(400, 800, 800)	(0.91, 0.173 ,0.218)	(0.91, 0.213, 0.296)	(0.52,1.038,1.067)
(500, 600, 600)	(0.90, 0.136, 0.098)	(0.90, 0.145, 0.109)	(0.55, 1.11, 0.94)
(700, 1200, 1200)	(0.90, 0.109, 0.086)	(0.90, 0.110, 0.088)	(0.60, 1.098,0.89)

et al. [24] performs similarly to ours. However, their two-step procedure is computationally expensive compared to ours and it is hard to choose regularizers. For the identity case in Table 4.3, the estimates by Witten et al. [60] do the worst among three methods, even though their formulation assumes covariance matrices to be identity, and worse for the Toeplitz and sparse inverse matrices, as seen in Table 4.4–4.5. Their method fails to provide accurate approximations because of low sample size we considered.

4.5.2 Spiked covariance models

In this section, we consider spiked covariance matrices of $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$:

$$\mathrm{Cov}(x) = \sum_{i=1}^{k_x} \lambda_i w_i w_i^T + I_p, \quad \mathrm{Cov}(y) = \sum_{i=1}^{k_y} \lambda_i w_i w_i^T + I_q.$$

Table 4.5. Error comparison for sparse inverse matrices. We use a format of $(\hat{\rho}, \mathbf{Loss}(\hat{u}), \mathbf{Loss}(\hat{v}))$ to represent each method's result.

(n, p, q)	Our method	SCCA	PMA
(400, 800, 800)	(0.92,0.092,0.149)	(0.92,0.129, 0.190)	(0.6,0.68,0.79)
(500, 600, 600)	(0.90, 0.068, 0.059)	(0.90, 0.069, 0.0623)	(0.67, 0.64, 0.41)
(700, 1200, 1200)	(0.90, 0.050 ,0.044)	(0.90, 0.051, 0.047)	(0.55,0.61,0.33)

In Example 2 we see that even if we have more observations than the number of features, the traditional singular value decomposition (SVD) can return poor results.

Example 2: We generate Σ_x and Σ_y as follows:

$$\Sigma_x = \sum_{i=1}^{k} \lambda_{x,i} w_{x,i} w_{x,i}^T + I, \quad \Sigma_y = \sum_{i=1}^{k} \lambda_{y,i} w_{y,i} w_{y,i}^T + I,$$

where $w_{x,1}, \ldots, w_{x,k}$, $w_{y,1}, \ldots, w_{y,k}$ are independent orthonormal vectors in $\mathbb{R}^p$, $\mathbb{R}^q$ and $\lambda_{x,i} = \lambda_{y,i} = 250$ and $k = 20$. The covariance Σ_{xy} is generated as $\Sigma_{xy} = \Sigma_x \rho u v^T \Sigma_y$, where u and v are the true canonical vectors and have 10 nonzero elements with indices randomly chosen. We generate the data matrices $X \in \mathbb{R}^{n \times p}$ and $Y^{n \times q}$ from the distribution

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy}^T & \Sigma_y \end{pmatrix}\right).$$

Therefore, when $n = 1000, p = 800, q = 800$, we should be able to estimate u and v using the SVD of the matrix

$$\hat{\Sigma}_x^{-1/2} \hat{\Sigma}_{xy} \hat{\Sigma}_y^{-1/2} = (X^T X)^{-1/2} (X^T Y)(Y^T Y)^{-1/2}.$$

However, the estimated $\hat{u}$ and $\hat{v}$ can be seen in Figure 4.6. The results are wrong and not sparse. This is an indication that we need more samples to estimate the canonical vectors. When we increase the sample size to $n = 3000$, the estimates of u and v are more accurate but not very sparse, as seen in Figure 4.7. For our method with $n = 400$, the estimated $\hat{u}$ and $\hat{v}$ can be seen in Figure 4.8. Our method returns sparse and better estimates for u and v.

4.5.3 A detailed comparison

To further illustrate the accuracy of our method, we compare it with the methods proposed by Tan et al. [50] using the plot of scaled sample size versus estimation error. We choose the same set up as theirs because their method performed better in comparison with PMA.

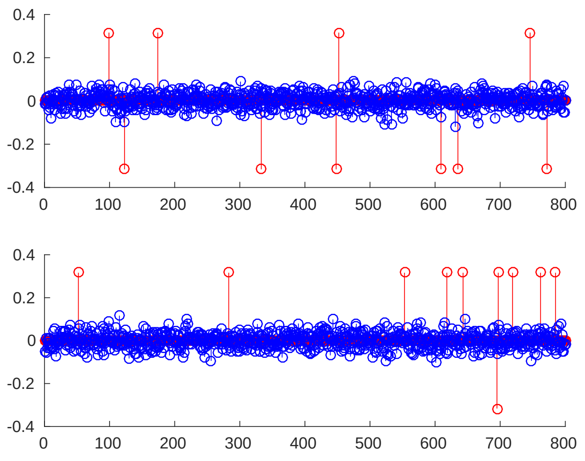

Figure 4.6. *Plot of estimated $\hat{u}$, $\hat{v}$ from SVD (blue) and true u, v (red)*, The number of observations $n = 1000$, with $p = 800, q = 800$. The estimated u and v using SVD of the transformed estimated covariance matrix are not good estimates of the true u and v. The results are wrong and not sparse. This is an indication that we need more samples to estimate the canonical vectors.

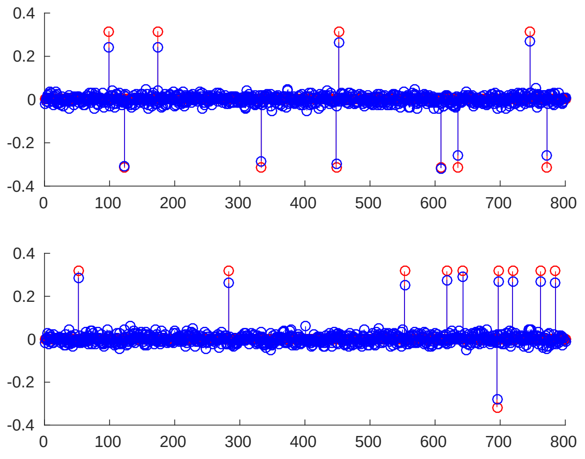

Figure 4.7. Plot of Estimated u, v from SVD (blue) and true u, v (red), The number of observations $n = 3000$, with $p = 800$, $q = 800$. Though the correct support is recovered but solution is not as sparse as our method.

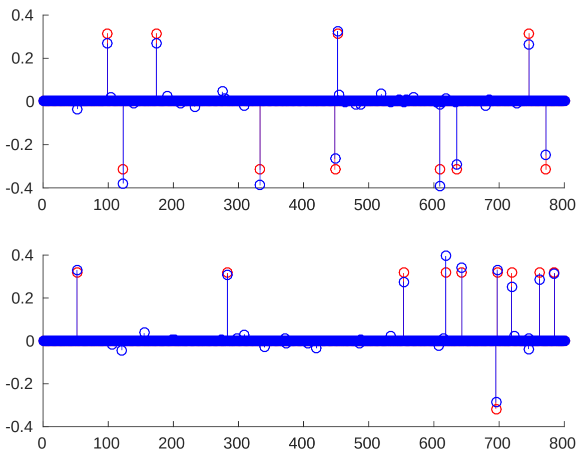

Figure 4.8. Plot of estimates u (top panel), and v (bottom panel) from our method (blue) and true u, v (red), The number of observations is $n = 400$, with $p = 800, q = 800$. Note that we use less samples than the results of Figure 4.6. We can successfully recover the correct support and return a sparse solution.

The data was simulated as

$$\rho = 0.9, \quad u_j = \frac{1}{\sqrt{5}}, \quad v_j = \frac{1}{\sqrt{5}} \text{ for } j = 1, 6, 11, 16, 21,$$

and Σ_x and Σ_y are block-diagonal matrices with five blocks, each of dimension $d/5 \times d/5$, where element (j, j') of each block takes the value $0.7^{|j-j'|}$. The set-up of our simulation is exactly the same as for Tan et al. [50]. In order to make comparisons between two methods, we normalize both $\hat{u}$ and $\hat{v}$ to have norm 1 for both methods and average the mean square errors of 100 simulations. As shown in Figure 4.9, our method outperforms theirs in general, and in the low sample regime, our method does significantly better.

4.6 Application to Human Connectome Project

The human connectome project (HCP) [57] acquires high-quality imaging and behavioral data from a large sample of healthy young adult subjects, in an effort to map macroscopic human brain circuits and their association with behavior. It contains imaging data, which measures functional and structural brain connectivity, and non-image data, which contains demographics, psychometrics, and other behavioral measures. A large part of the datasets

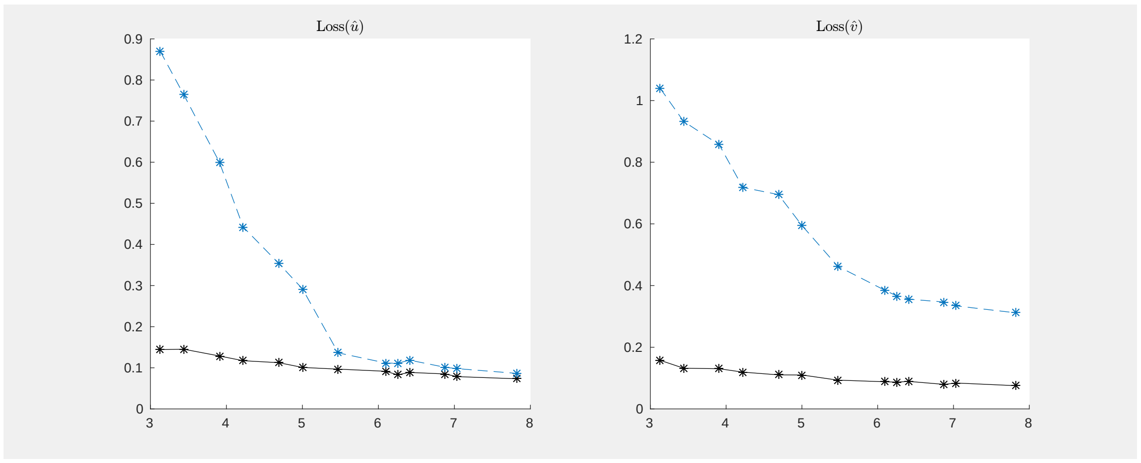

Figure 4.9. A comparison between performance of our method and the method proposed by Tan et al. [50]. The left panel is the **Loss**($\hat{u}$) versus rescaled sample size $n/s\log(d)$, and the right panel is the **Loss**($\hat{v}$) versus rescaled sample size $n/s\log d$. The Blue line is the result of Tan et al. [50] and the black line is the result of our method.

are publicly available at `https://db.humanconnectome.org/`. This project gives researchers a tool to potentially answer open questions such as what makes humans unique, and how brain networks integrate information through complex neural connections. One of those questions is whether any specific parts of brain are associated with specific sets of behavior variables when performing a certain task.

A natural approach of relating image to non-imaging data is to use canonical correlation analysis, and many researchers in neuroscience community have used this approach. While there has been an intent to use CCA on this data, the methods in the literature so far have made one of the following modifications in response to the high dimensionality of the imaging data:

- Principal component analysis is first conducted to reduce the dimensions of the imaging data and then canonical correlation analysis is performed [47], but the solutions are generally dense.

- The variance of the imaging data is assumed to be the identity [1], which leads to a sparse singular value decomposition.

These modifications limit the ability of algorithms to find true canonical correlations in the data, and the second one even violates the mathematical definition of CCA. We apply our method to behavioral variables and functional statistical maps. In this dissertation, we study the 'n-back' task designed to measure working memory. In this particular task, all subjects were presented with items one at a time and then asked to identify each item that repeats relative to the item that occurred n times before. The data contains 495 subjects and to be more specific, the data matrices X and Y are constructed as follows:

- We obtain unrestricted behavioral data from HCP. We first delete the variables with 80% missing data and replace NA values with the mean value of the corresponding column. This process is also performed by Asteris et al. [1]. We then choose the variables that intersect with the paper by Smith et al. [47], which include scores from psychological tests, physiological measurements, and self-reported behavior questionnaires. The dimension of the resulting Y matrix is 495×61.

- We use the pre-computed 2back-0back statistical contrast maps provided by the HCP. It was preprocessed using standard preprocessing such as motion correction. Details of the preprocessing can be found in [57] and on the website `https://db.humanconnectome.org`. The `nilearn` python package was then used to resample the Voxels to different scales. The higher the scale, the coarser the imaging data is, which potentially can eliminate the effect of correlations among Voxels nearby. The dimensions of the resulting matrix X are 495×109350 (corresponding to scaling factor 2), 495×32400 (corresponding to scaling factor 3), 495×13068 (corresponding to scaling factor 4), and 495×6804 (corresponding to scaling factor 5).

We present the results of applying Sparse CCA to four resulting datasets, where each set corresponds to one scaling factor. The non-zero elements in estimated canonical vector $\hat{v}_1$ corresponding to behavior variables can be found in Figure 4.10. The font size reflects the canonical vector weights in absolute value. Even though we run four different datasets, we get similar behavior variables with very minor differences and thus we present one set of the results. We see that the variables that play more important roles are PMAT24_A_CR, CardSort_Unadj, PicVocab_AgeAdj, ListSort_AgeAdj, PercStree_Unadj, AngeHostil_Unadj, etc. PMAT24_A_CR measures the number of correct responses to a test where participants are presented patterns made up from $2 \times 2, 1 \times 5$ etc. squares with one of the squares missing and must pick one of five response choices that best fits the missing square. PercStress_Unadj measures how unpredictable, uncontrollable, overloaded respondents find their lives. It is interesting to find that variables such as PMAT24_A_CR, PicVocal_AgeAdj play opposite effects compared with variables such as AngHostil_Unadj, PerStress_Unadj for this particular task. We can see that positive factors are the features that are related to intelligence; however, the negative effects are dominated by variables such as emotion and stress. The negative factors were not discovered by Asteris et al. [1]. Detailed descriptions for each feature in the figure are given at `https://wiki.humanconnectome.org/display/PublicData/HCP+Data+Dictionary+Public-+500+Subject+Release`. Figures 4.11—4.14 show the results of estimated canonical vectors $\hat{u}_1$ of the imaging data mapped back into the brain. When the scaling factor increases, the resolution of images decreases. We identified activation regions (in red) known to be involved in executive function and working memory and deactivation regions (in blue) known to be associated with engagement of difficult cognitive functions but with some differences relative to the result of Asteris et al. [1]. The study of brain functions is still at an early stage and we hope that sparse CCA can be used as a tool to advance this field.

4.7 Cross-language document retrieval

In cross-language document retrieval, among a collection of multilingual documents the goal is to find the most relevant documents in the target languages. CCA has been shown to be an efficient tool by Sriperumbudur et al. [48] and Vinokourov et al. [59]. In this setting, CCA tries to find a low-dimensional representation or a basis in languages, in order to maximize correlations among documents. In this dissertation, we apply sparse CCA to multilingual databases and seek a low-dimensional model that is only dependent on a small subsets of words in documents with a maximum correlation. Sparse CCA allows us to find such a representation and the resulting words can be translations of one another [48].

The data sets are downloaded from `http://optima.jrc.it/Acquis/index_2.2.html`. They contain regulations of the European Union translated into different languages. We used the documents in English, French, and Spanish. We removed non-words, ID numbers, diatrics, unicode punctuation, and isolated characters, and represented words in terms of TFIDF [56] (term frequency-inverse document frequency) to reflect how important

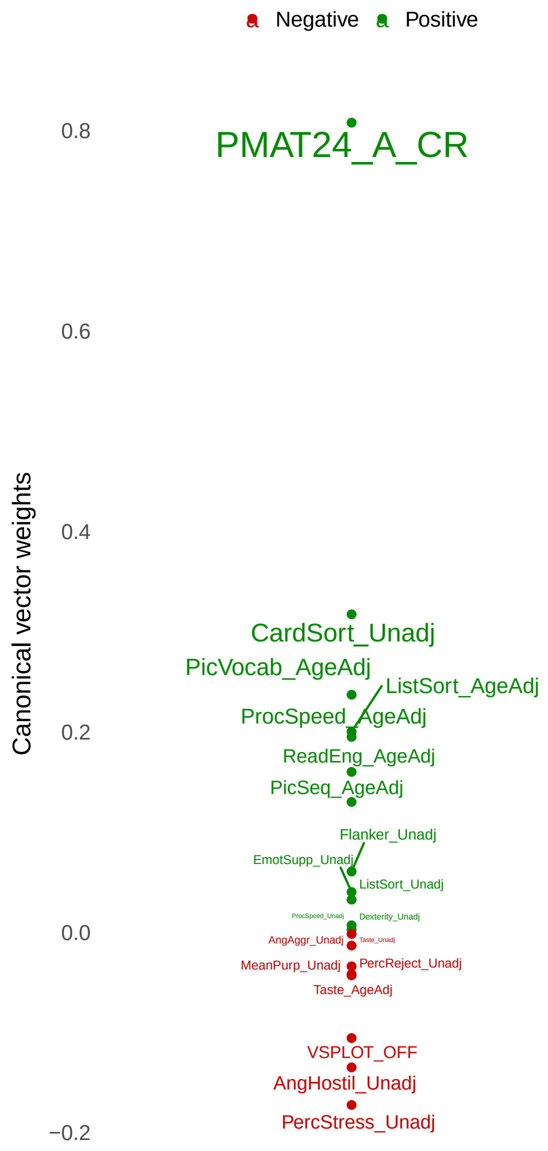

Figure 4.10. Behavior variables corresponds to the non-zero elements of estimated canonical vector $\hat{v}_1$. The font size reflects the canonical vector weights in absolute value.

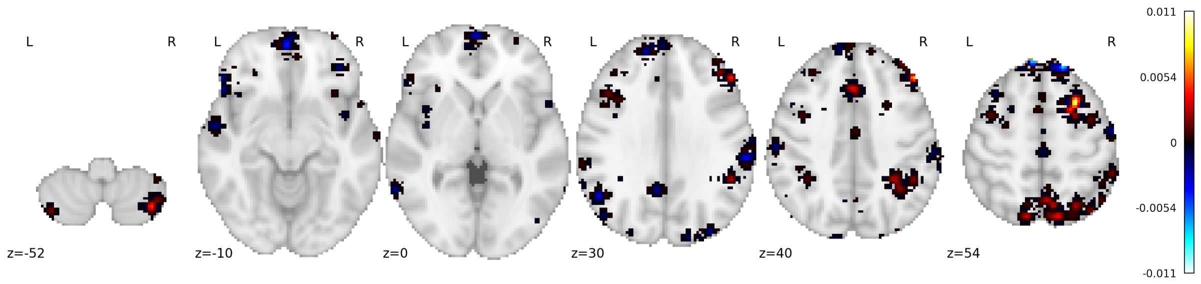

Figure 4.11. Result of applying sparse CCA to matrix Y and matrix X with scaling factor 2. The estimated canonical vectors are mapped back to the brain with positive values corresponding to activation regions (in red) and negative values corresponding to deactivation regions (in blue).

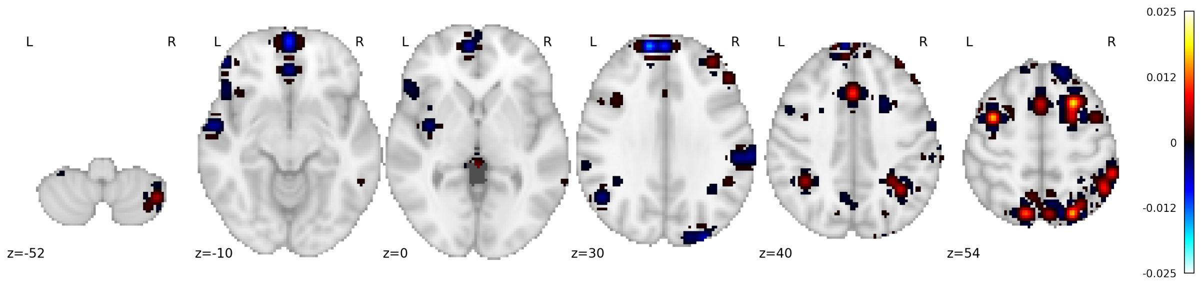

Figure 4.12. Result of applying sparse CCA to matrix Y and matrix X with scaling factor 3. The estimated canonical vectors are mapped back to the brain with positive values corresponding to activation regions (in red) and negative values corresponding to deactivation regions (in blue).

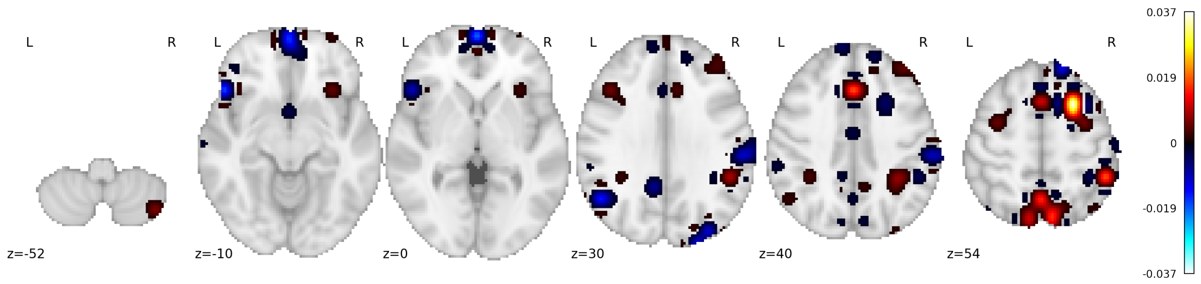

Figure 4.13. Result of applying sparse CCA to matrix Y and matrix X with scaling factor 4. The estimated canonical vectors are mapped back to the brain with positive values corresponding to activation regions (in red) and negative values corresponding to deactivation regions (in blue).

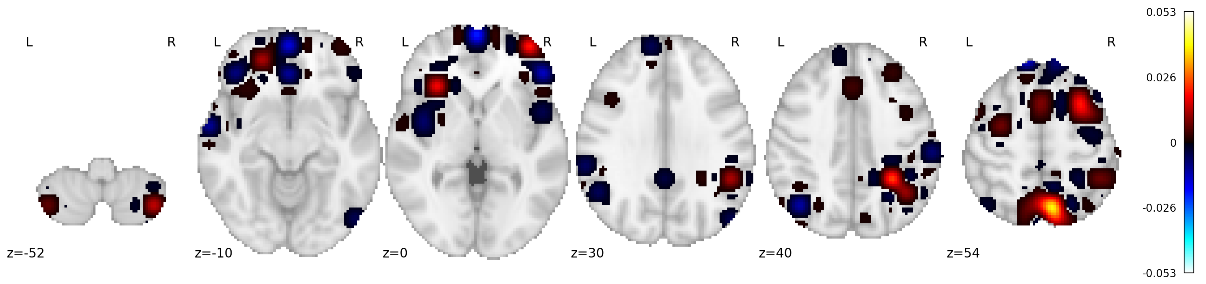

Figure 4.14. Result of applying sparse CCA to matrix Y and matrix X with scaling factor 5. The estimated canonical vectors are mapped back to the brain with positive values corresponding to activation regions (in red) and negative values corresponding to deactivation regions (in blue).

a word is to a corpus.

There are a total of 7749 documents in common among the three languages, with 26149 unique words in English, 29340 unique words in French, 33638 unique words in Spanish. We analyzed them in pairs. The resulting matrices are large (e.g., $7,749 \times 26,149$ for English) but very sparse. In order to speed up computations and minimize memory usage, we store the centered matrices in a sparse plus rank-one form, and use a randomized SVD to initialize our algorithm.

We visualize the results using word clouds in Figures 4.15–4.16. The font size reflects the estimated canonical correlation vectors $\hat{u}$ and $\hat{v}$ in absolute values. In order to achieve high canonical correlation, the resulting canonical vectors put more weight on certain key words in different languages. For example, reglement and reglamento mean regulation in French and Spanish. One can easily pick out literal translations from the estimated canonical vectors.

4.8 Discussion

We have proposed a sparse canonical correlation framework and shown how to solve it efficiently using ADMM and TFOCS. We have extensively studied different simulation scenarios and shown our estimates are sparse, efficient and accurate. Though our formulation is non-convex, global solutions are often obtained, as seen among simulated examples. We have presented the results of our methods on real datasets and demonstrated its use in

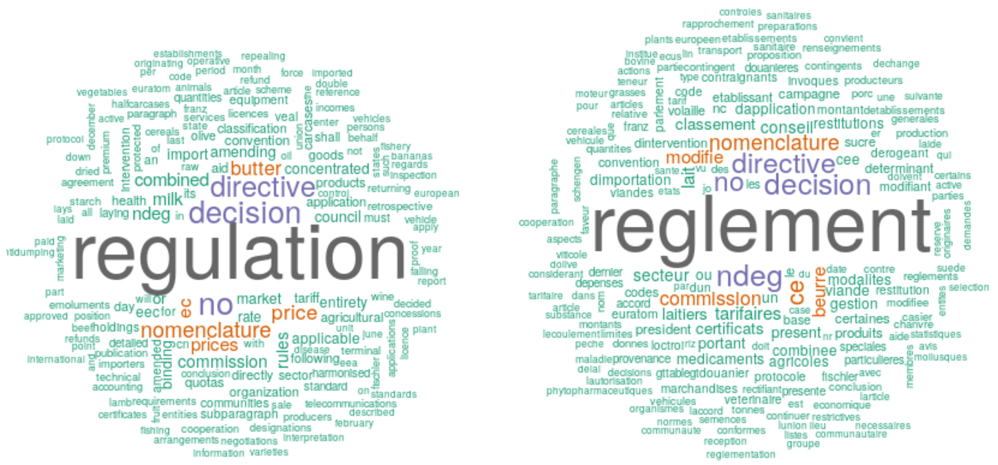

Figure 4.15. Sparse canonical correlation applied to documents in English and French. Estimated canonical vectors in $\hat{u}$ and $\hat{v}$ put more weight on words with the same meaning in different languages. Reglement and beurre in French mean regulation and butter, respectively.

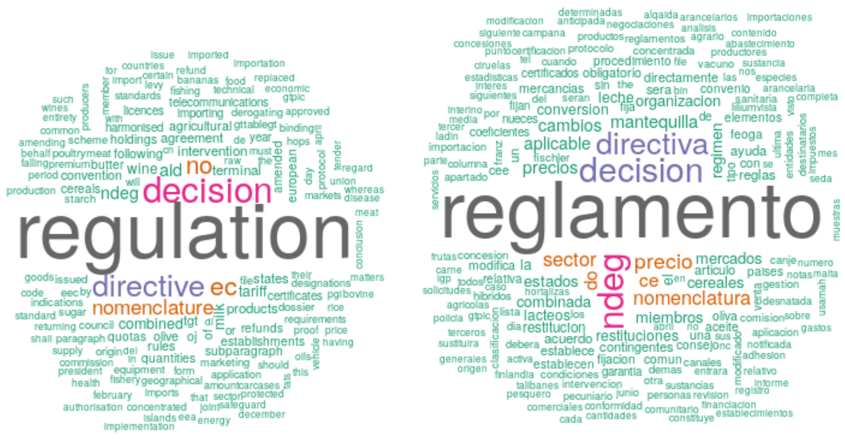

Figure 4.16. Sparse canonical correlation applied to documents in English and Spanish. Estimated canonical vectors in $\hat{u}$ and $\hat{v}$ put more weight on words with the same meaning in different languages. Reglamento and precio in Spanish mean regulation and price, respectively.

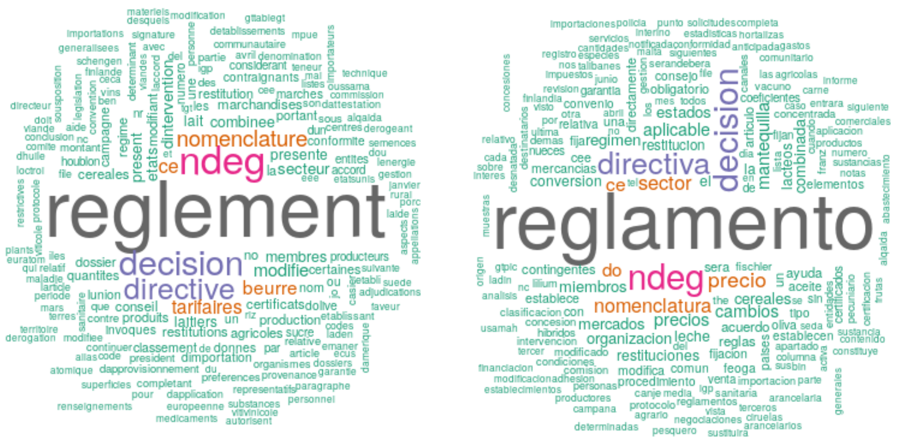

Figure 4.17. Sparse canonical correlation applied to documents in French and Spanish. Estimated canonical vectors in $\hat{u}$ and $\hat{v}$ put more weight on words with the same meaning in different languages.

real-world applications. There are many directions for further research and we list a few here.

- LDA is a special case of CCA when one of the matrices in CCA is written to contain class information of the data. As discussed by Sun and Chen [49], the indicator matrix or class label can be viewed as a hard label for each data point. Instead of using an indicator matrix, we can use a soft class label encoding for each sample. They showed some improvements on some datasets in low-dimensional settings and it would be interesting to explore whether the advantages remain in high-dimensional settings.

- In the HCP project, the X matrices analyzed were from original images, and in order to reduce the correlation, we used different scaling factors. Another approach would be to project the images into some known basis such as the MSDL atlas [58] and then perform sparse CCA on the resulting matrices.

Chapter 5

Appendix

Detailed derivations for linearized ADMM

The augmented Lagrangian form of (4.9) is

$$L(u, z, \phi) = -u^T X^T Y v + \tau_1 \|u\|_1 + \mathbb{1}\{\|z\|_2 \leq 1\} + \phi^T (Xu - z) + \frac{\rho}{2}\|Xu - z\|_2^2.$$

Thus, the updates of variables are obtained from

$$\begin{aligned}
u^{k+1} &= \underset{u}{\operatorname{argmin}}\{-u^T X^T Y v + \tau_1 \|u\|_1 + {\phi^k}^T (Xu - z^k) + \frac{\rho}{2}\|Xu - z^k\|_2^2\}, \\
z^{k+1} &= \underset{z}{\operatorname{argmin}}\{\mathbb{1}\{\|z\|_2 \leq 1\} + \phi^T (Xu^{k+1} - z) + \frac{\rho}{2}\|Xu^{k+1} - z\|_2^2\}, \\
\phi^{k+1} &= \phi^k + \rho(Xu^{k+1} - z^{k+1}).
\end{aligned}$$

Letting $\xi^k = \frac{\phi^k}{\rho}$ and adding some constants, we get

$$\begin{aligned}
u^{k+1} &= \underset{u}{\operatorname{argmin}}\{-u^T X^T Y v + \tau_1 \|u\|_1 + \rho {\xi^k}^T (Xu - z^k) + \frac{\rho}{2}\|Xu - z^k\|_2^2 + \frac{\rho}{2}\|\xi\|_2^2\}, \\
z^{k+1} &= \underset{z}{\operatorname{argmin}}\{\mathbb{1}\{\|z\|_2 \leq 1\} + \rho \xi^T (Xu^{k+1} - z) + \frac{\rho}{2}\|Xu^{k+1} - z\|_2^2 + \frac{\rho}{2}\|\xi\|_2^2\}, \\
\xi^{k+1} &= \xi^k + (Xu^{k+1} - z^{k+1}).
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
u &\leftarrow \underset{u}{\operatorname{argmin}}\{-u^T X^T Y v + \tau_1 \|u\|_1 + \frac{\rho}{2}\|Xu - z + \xi\|_2^2\},\\
z &\leftarrow \underset{z}{\operatorname{argmin}}\{\mathbb{1}\{\|z\|_2 \leq 1\} + \frac{\rho}{2}\|Xu - z + \xi\|_2^2\},\\
\xi &\leftarrow \xi + (Xu - z).
\end{aligned}$$

The Linearized ADMM replaces the quadratic term $\frac{\rho}{2}\|Xu - z + \xi\|_2^2$ by a linear term in order to speed it up:

$$\begin{aligned}
u &\leftarrow \underset{u}{\operatorname{argmin}}\{-u^T X^T Y v + \tau_1 \|u\|_1 + \rho(X^T X u^k - X^T z^k)^T u + \frac{\mu}{2}\|u - u^k\|_2^2\},\\
z &\leftarrow \underset{z}{\operatorname{argmin}}\{\mathbb{1}\{\|z\|_2 \leq 1\} + \frac{\rho}{2}\|z - Xu^{k+1} + \xi^k\|_2^2\},\\
\xi &\leftarrow \xi + (Xu^{k+1} - z^{k+1}).
\end{aligned}$$

Letting $\rho = 1/\lambda$ and $\tilde{\mu} = \frac{1}{\mu}$, we get

$$\begin{aligned}
u &\leftarrow \underset{u}{\operatorname{argmin}}\{-u^T X^T Y v + \tau_1 \|u\|_1 + \frac{1}{\lambda}(X^T X u^k - X^T z^k)^T u + \frac{1}{2\tilde{\mu}}\|u - u^k\|_2^2\},\\
z &\leftarrow \underset{z}{\operatorname{argmin}}\{\mathbb{1}\{\|z\|_2 \leq 1\} + \frac{\rho}{2}\|z - Xu^{k+1} + \xi^k\|_2^2\},\\
\xi &\leftarrow \xi + (Xu^{k+1} - z^{k+1}).
\end{aligned}$$

Letting $\mu = \tilde{\mu}$, for the first minimization problem, after some simple algebra we get

$$u^{k+1} = \underset{u}{\operatorname{argmin}}\{-u^T X^T Y v + \tau_1 \|u\|_1 + \frac{1}{2\mu}\|u - (u^k - \frac{\mu}{\lambda}(X^T(Xu^k - z^k + \xi^k))\|_2^2\}.$$

Therefore, our detailed updates are

$$\begin{aligned}
u^{k+1} &\leftarrow \mathbf{prox}_{\mu f}(u^k - \frac{\mu}{\lambda} X^T (Xu^k - z^k + \xi^k)),\\
z^{k+1} &\leftarrow \mathbf{prox}_{\lambda g}(Xu^{k+1} + \xi^k),\\
\xi^{k+1} &\leftarrow \xi^k + Xu^{k+1} - z^{k+1}.
\end{aligned}$$

The analytic proximal mapping of f and g can be easily derived: $f(x)$ involves a soft threshold and $g(x)$ is a projection onto the convex set (a norm ball):

$$\mathbf{prox}_{\mu f}(x) = \begin{cases} x + \mu c - \mu\tau & \text{if } x + \mu c > \mu\tau \\ x + \mu c + \mu\tau & \text{if } x + \mu c < -\mu\tau \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{prox}_{\lambda g}(x) = \begin{cases} x & \text{if } \|x\|_2 \le 1 \\ \frac{x}{\|x\|_2} & \text{otherwise} \end{cases}, \tag{5.1}$$

where $c = X^T Y v$ (the gradient of the objective function with respect to one canonical vector with the other canonical vector fixed).

Equivalence between CCA and LDA

Let X and Y be the original data matrices. Let Y be the label matrix and without loss of generality, we can assume that Y has the form $Y = \begin{pmatrix} 1_{n_1} & & & \\ & 1_{n_2} & & \\ & & \ddots & \\ & & & 1_{n_c} \end{pmatrix}_{n \times n_C}$, where C is the number of class labels. The sample covariance matrices can be written

$$\begin{aligned} S_x &= \frac{1}{n-1}(X - \frac{1}{n}11^T X)^T (X - \frac{1}{n}11^T X) = \frac{1}{n-1} X^T (I - \frac{1}{n}11^T) X, \\ S_y &= \frac{1}{n-1}(Y - \frac{1}{n}11^T Y)^T (Y - \frac{1}{n}11^T Y) = \frac{1}{n-1} Y^T (I - \frac{1}{n}11^T) Y, \\ S_{xy} &= \frac{1}{n-1} X^T (I - \frac{1}{n}11^T) Y. \end{aligned}$$

We know that the CCA problem solves

$$\begin{aligned} \underset{u,v}{\text{maximize}} \quad & u^T S_{xy} v \\ \text{subject to} \quad & u^T S_x u = 1, \quad v^T S_y v = 1. \end{aligned}$$

From the KKT conditions, if u^*, v^*, λ_u^* and λ_v^* are a maximizer of the CCA problem, we have

$$S_{xy}S_y^{-1}S_{yx}u^* = \lambda_u^* S_x u^*,$$
$$S_{yx}S_x^{-1}S_{xy}v^* = \lambda_v^* S_y v^*,$$

We now derive the inverse of S_y using the Sherman-Morrison formula:

$$\begin{aligned} S_y^{-1} &= (n-1)(Y^TY - \frac{1}{n}Y^T 1 1^T Y)^{-1} \\ &= (n-1)(Y^TY - \frac{1}{n}\begin{pmatrix} n_1 \\ \vdots \\ n_q \end{pmatrix}\begin{pmatrix} n_1 & \dots & n_q \end{pmatrix})^{-1} \\ &= (n-1)[(Y^TY)^{-1} - \frac{1}{n}\frac{(Y^TY)^{-1}uv^T(Y^TY)^{-1}}{1+u^T(Y^TY)^{-1}v}], \end{aligned} \tag{5.2}$$

where $u = v = \begin{pmatrix} n_1 \\ \vdots \\ n_q \end{pmatrix}$, $(Y^TY)^{-1} = \begin{pmatrix} \frac{1}{n_1} & & & \\ & \frac{1}{n_2} & & \\ & & \ddots & \\ & & & \frac{1}{n_c} \end{pmatrix}$. (5.2) becomes

$$(n-1)[(Y^TY)^{-1} - \frac{1}{n(n+1)}11^T].$$

To simplify $S_{xy}S_y^{-1}S_{yx}$, S_{xy} can be rewritten as

$$\begin{aligned} S_{xy} &= \frac{1}{n-1}X^T(I - \frac{1}{n}11^T)Y \\ &= \frac{1}{n-1}\begin{pmatrix} \sum_{i=1}^{n_1} x_i^{(1)} & \sum_{i=1}^{n_2} x_i^{(2)} & \dots & \sum_{i=1}^{n_q} x_i^{(q)} \end{pmatrix} - \frac{1}{n-1}\sum x_i \begin{pmatrix} n_1 & \dots & n_q \end{pmatrix} \\ &= \frac{1}{n-1}\begin{pmatrix} \sum_{i=1}^{n_1} x_i^{(1)} - \bar{x}n_1 & \sum_{i=1}^{n_2} x_i^{(2)} - \bar{x}n_2 & \dots & \sum_{i=1}^{n_q} x_i^{(q)} - \bar{x}n_q \end{pmatrix}. \end{aligned}$$

Therefore, $S_{xy}S_y^{-1}S_{yx}$ is simplified to

$$S_{xy}S_y^{-1}S_{yx} = \frac{1}{n-1}\begin{pmatrix}\bar{x}_{n_1}^{(1)} - \bar{x} & \bar{x}_{n_2}^{(2)} - \bar{x} & \dots & \bar{x}_{n_q}^{(q)} - \bar{x}\end{pmatrix}\begin{pmatrix}\sum_{i=1}^{n_1} x_i^{(1)} - n_1\bar{x} \\ \sum_{i=1}^{n_2} x_i^{(2)} - n_2\bar{x} \\ \vdots \\ \sum_{i=1}^{n_q} x_i^{(q)} - n_q\bar{x}\end{pmatrix}$$
$$= \frac{1}{n-1}\sum_{i=1}^{n_q} n_i(\bar{x}_i^{(i)} - \bar{x})(\bar{x}_i^{(i)} - \bar{x})^T.$$

Thus we have

$$\begin{aligned} S_B u^* &= \lambda_u^* S_x u^* \\ S_B u^* &= \lambda_u^*(S_B + S_W)u^* \\ (1-\lambda_u^*)S_B u^* &= \lambda_u^* S_W u^* \\ S_B u^* &= \frac{\lambda_u^*}{1-\lambda_u^*} S_W u^*. \end{aligned} \tag{5.3}$$

Recall that in linear discriminant analysis (LDA), we solve

$$\begin{aligned} \underset{a}{\text{minimize}} \quad & a^T S_B a \\ \text{subject to} \quad & a^T S_W a = 1, \end{aligned}$$

By the KKT conditions, if a^*, λ_a^* is a maximizer of the LDA problem, we have $S_B a^* = \lambda_a^* S_W a^*$. Letting $\lambda_a^* = \frac{\lambda_u^*}{1-\lambda_u^*}$, it is exactly equivalent to CCA by (5.3).

Bibliography

[1] Megasthenis Asteris, Anastasios Kyrillidis, Oluwasanmi Koyejo, and Russell Poldrack. A simple and provable algorithm for sparse diagonal CCA. *arXiv preprint arXiv:1605.08961*, 2016.

[2] Jinho Baik, Gérard Ben Arous, Sandrine Péché, et al. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *The Annals of Probability*, 33(5):1643–1697, 2005.

[3] Matthew Barker and William Rayens. Partial least squares for discrimination. *Journal of Chemometrics*, 17(3):166–173, 2003.

[4] R. E. Barlow, D.J. Bartholomew, J. M. Bremner, and H. D. Brunk. *Statistical Inference Under Order Restrictions; the Theory and Application of Isotonic Regression*. Wiley, New York, 1972.

[5] Maurice S Bartlett. Further aspects of the theory of multiple regression. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 34, pages 33–40. Cambridge University Press, 1938.

[6] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

[7] Stephen R. Becker, Emmanuel J. Candès, and Michael C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165, 2011.

[8] J. Bien, J. Taylor, and R. Tibshirani. A lasso for hierarchical interactions. *Annals of Statistics*, 42(3):1111–1141, 2013.

[9] Jérôme Bobin, Stephen Becker, and Emmanuel Candes. A fast and accurate first-order method for sparse recovery. *SIAM J Imaging Sciences*, 2011.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–124, 2011.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[13] Tony Cai and Weidong Liu. Adaptive thresholding for sparse covariance matrix estimation. *Journal of the American Statistical Association*, 106(494):672–684, 2011.

[14] John M. Chambers. *Statistical Models in S*. CRC Press, Inc., Boca Raton, FL, USA, 1991. ISBN 0412052911.

[15] M. Chen, C. Gao, Z. Ren, and H. H. Zhou. Sparse CCA via precision adjusted iterative thresholding. *ArXiv e-prints*, November 2013.

[16] Chin-Tsang Chiang, John A Rice, and Colin O Wu. Smoothing spline estimation for varying coefficient models with repeatedly measured dependent variables. *Journal of the American Statistical Association*, 96(454):605–619, 2001.

[17] Alexandre d'Aspremont, Laurent El Ghaoui, Michael I. Jordan, and Gert R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007. doi: 10.1137/050645506.

[18] Jan de Leeuw, Kurt Hornik, and Patrick Mair. Isotone optimization in R: Pool-adjacent-violators (PAVA) and active set methods. *Journal of Statistical Software*, 32 (5):1–24, 2009.

[19] B. Efron. How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, 81:461–70, 1986.

[20] Jianqing Fan and Wenyang Zhang. Statistical methods with varying coefficient models. *Statistics and its Interface*, 1(1):179–195, 2008.

[21] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Eugen.*, 7: 179–188, 1936.

[22] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, 1(4), 2009.

[23] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[24] C. Gao, Z. Ma, and H. H. Zhou. Sparse CCA: Adaptive estimation and computational barriers. *ArXiv e-prints*, September 2014.

[25] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

[26] S. J. Grotzinger and C. Witzgall. Projections onto simplices. *Applied Mathematics and Optimization*, 12(1):247–270, 1984.

[27] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York, 2001.

[28] Trevor Hastie and Robert Tibshirani. Varying-coefficient models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(4):pp. 757–796, 1993. ISSN 00359246.

[29] A. E. Hoerl and R.W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[30] Paul W. Holland and Roy E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.

[31] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, 1936. doi: 10.1093/biomet/28.3-4.321.

[32] Jianhua Z. Huang and Haipeng Shen. Functional coefficient regression models for non-linear time series: A polynomial spline approach. *Scandinavian Journal of Statistics*, 31(4):515–534, 2004. ISSN 1467-9469.

[33] P. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 53:73–101, 1964.

[34] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. doi: 10.2307/2238020.

[35] L. Jacob, G. Obozinski, and J-P. Vert. Group lasso with overlap and graph lasso. *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[36] Iain M. Johnstone and Arthur Yu Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693, 2009. doi: 10.1198/jasa.2009.0121. PMID: 20617121.

[37] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[38] Qing Mai, Yi Yang, and Hui Zou. Multiclass sparse discriminant analysis. *arXiv preprint arXiv:1504.05845*, 2015.

[39] Y. Nardi and A. Rinaldo. Autoregressive process modeling via the lasso procedure. *Journal of Multivariate Analysis*, 102(3):528 – 549, 2011. ISSN 0047-259X. doi: http://dx.doi.org/10.1016/j.jmva.2010.10.012.

[40] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3): 127–239, January 2014. ISSN 2167-3888. doi: 10.1561/2400000003.

[41] Byeong U. Park, Enno Mammen, Young K. Lee, and Eun Ryung Lee. Varying coefficient regression models: A review and new developments. *International Statistical Review*, 83(1):36–64, 2015. ISSN 1751-5823. doi: 10.1111/insr.12029.

[42] Debashis Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, pages 1617–1642, 2007.

[43] Mohsen Pourahmadi. *High-dimensional Covariance Estimation: with High-dimensional Data*, volume 882. John Wiley & Sons, 2013.

[44] Daniel F. Schmidt and Enes Makalic. Estimation of stationary autoregressive models with the Bayesian lasso. *Journal of Time Series Analysis*, 2013. ISSN 1467-9892. doi: 10.1111/jtsa12027.1467-9892.2013.12027.

[45] N. Z. Shor, Krzysztof C. Kiwiel, and Andrzej Ruszcayǹski. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag New York, Inc., 1985.

[46] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013. doi: 10.1080/10618600.2012.681250.

[47] Stephen M Smith, Thomas E Nichols, Diego Vidaurre, Anderson M Winkler, Timothy EJ Behrens, Matthew F Glasser, Kamil Ugurbil, Deanna M Barch, David C Van Essen, and Karla L Miller. A positive-negative mode of population covariation links brain connectivity, demographics and behavior. *Nature Neuroscience*, 18(11): 1565, 2015.

[48] Bharath K Sriperumbudur, David A Torres, and Gert RG Lanckriet. A majorization-minimization approach to the sparse generalized eigenvalue problem. *Machine Learning*, 85(1-2):3–39, 2011.

[49] Tingkai Sun and Songcan Chen. Class label versus sample label-based CCA. *Applied Mathematics and Csomputation*, 185(1):272–283, 2007.

[50] K. M. Tan, Z. Wang, H. Liu, and T. Zhang. Sparse generalized eigenvalue problem: Optimal Statistical Rates via Truncated Rayleigh Flow. *ArXiv e-prints*, April 2016.

[51] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[52] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistics Society Series B*, 67(1):91–108, 2005.

[53] Ryan Tibshirani and Jonathan Taylor. On the degrees of freedom of the lasso. *Annals of Statistics*, (40):1198–1232, 2011.

[54] Ryan Tibshirani, Holger Hoefling, and Robert Tibshirani. Nearly-isotonic regression. *Technometrics*, 53(1):54–61, 2011.

[55] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. 01 2008.

[56] Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[57] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, Wu-Minn HCP Consortium, et al. The WU-Minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.

[58] Gaël Varoquaux, Alexandre Gramfort, Fabian Pedregosa, Vincent Michel, and Bertrand Thirion. Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 562–573. Springer, 2011.

[59] Alexei Vinokourov, Nello Cristianini, and John Shawe-Taylor. Inferring a semantic representation of text via cross-language correlation analysis. In *Advances in Neural Information Processing Systems*, pages 1497–1504, 2003.

[60] Daniela M. Witten, Robert Tibshirani, and Trevor Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009. doi: 10.1093/biostatistics/kxp008.

[61] J. M. Wooldridge. *Introductory Econometrics: A Modern Approach*. South-Western College Publishing, 2009.

[62] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2007.

[63] H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

[64] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67(2):301–320, 2005.

[65] Hui Zou, Trevor Hastie, and Robert Tibshirani. On the “degrees of freedom" of the lasso. *Annals of Statistics*, 35(5):2173–2192, 2007.