LEARNING WITH SPARSITY: STRUCTURES, OPTIMIZATION AND APPLICATIONS

Xi Chen

July 2013
CMU-ML-13-105

School of Computer Science
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee
Jaime Carbonell, Chair
Tom Mitchell
Larry Wasserman
Robert Tibshirani (Stanford)

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

This research was sponsored by the Air Force Research Laboratory under grant number FA87500720137 and by a fellowship from IBM Corporation.

Dedicated to my grandparents Jingrong Wang and Kai Chen, and my wife Yingze Wang.

ABSTRACT

The development of modern information technology has enabled collecting data of unprecedented size and complexity. Examples include web text data, microarray & proteomics, and data from scientific domains (e.g., meteorology). To learn from these high dimensional and complex data, traditional machine learning techniques often suffer from the curse of dimensionality and unaffordable computational cost. However, learning from large-scale high-dimensional data promises big payoffs in text mining, gene analysis, and numerous other consequential tasks.

Recently developed sparse learning techniques provide us a suite of tools for understanding and exploring high dimensional data from many areas in science and engineering. By exploring sparsity, we can always learn a parsimonious and compact model which is more interpretable and computationally tractable at application time. When it is known that the underlying model is indeed sparse, sparse learning methods can provide us a more consistent model and much improved prediction performance. However, the existing methods are still insufficient for modeling complex or dynamic structures of the data, such as those evidenced in pathways of genomic data, gene regulatory network, and synonyms in text data.

This thesis develops structured sparse learning methods along with scalable optimization algorithms to explore and predict high dimensional data with complex structures. In particular, we address three aspects of structured sparse learning:

1. Efficient and scalable optimization methods with fast convergence guarantees for a wide spectrum of high-dimensional learning tasks, including single or multi-task structured regression, canonical correlation analysis as well as online sparse learning.

2. Learning dynamic structures of different types of undirected graphical models, e.g., conditional Gaussian or conditional forest graphical models.

3. Demonstrating the usefulness of the proposed methods in various applications, e.g., computational genomics and spatial-temporal climatological data. In addition, we also design specialized sparse learning methods for text mining applications, including ranking and latent semantic analysis.

In the last part of the thesis, we also present the future direction of the high-dimensional structured sparse learning from both computational and statistical aspects.

KEYWORDS

Machine Learning, Sparse Learning, Optimization, Structure, Regression, Multi-task Regression, Canonical Correlation Analysis, Undirected Graphical Models, First-order Method, Stochastic Optimization, Text Mining, Ranking, Latent Semantic Analysis, Spatial-temporal Data, Computational Genomics.

ACKNOWLEDGMENTS

First, I would like to thank my advisor Jaime Carbonell for his great advice and support through the entire Ph.D. journey. I am particularly grateful to him for not only directing me towards exciting problems and applications, but also allowing me the freedom to pursue the topics I am passionate about. His vision on new research problems, his leadership as the department head, as well as his extremely nice and sincere personality all deeply influence me and will continue to shape me in the future.

I also deeply appreciate my other thesis committee members, Tom Mitchell, Larry Wasserman and Robert Tibshirani. They put a lot of effort on this thesis and provided many insightful suggestions. In fact, their help on me is far beyond the scope of this thesis. Our department head, Tom Mitchell, is the most amazing department head that I could ever dream to have. He established the first machine learning department in the world and I am so lucky to be a part of the department. He helped me so much from the very first day that I joined the department, from choosing advisor and selecting courses, to career preparation and job search. Larry Wasserman (and John Lafferty)'s great course on 10702 open the door of the field of statistical machine learning to me, which is such a great field that I would love to devote my entire career to explore. In addition to the research, I worked with Tom Mitchell and Larry Wasserman as teaching assistants in Machine Learning (10701) and Statistical Machine Learning (10702) classes. I learned so much from them on how to be a good teacher. Moreover, I feel so fortunate to have Robert Tibshirani from Stanford University as my external committee member, who is the founding father of the sparse learning field.

I am very grateful to my research advisor, Manuel Blum, during my Master at ACO program. He guided me at the beginning of my graduate study as father figures. I learned from him not only how to approach to a research problem, but also how to be a good and helpful person. Even after I started my Ph.D. study, we had meetings and discussions on a weekly or bi-weekly basis. Through my six years' graduate study, Prof. Manuel Blum cares not only about my progress on research but also my future career and life.

I also thank many other faculty members in Machine Learning and Statistics Department: Christos Faloutsos, Stephen Fienberg, Geoffrey Gordon, Steve Hanneke, Robert Kass, Seyoung Kim, Zico Kolter, John Lafferty, Jing Lei, Jeff Schneider, Aarti Singh, Eric Xing, Yiming Yang, as well as in Tepper School of Business: Egon Balas, Gerard Cornuejols and Javier Peña. I was so fortunate to have many great courses from them and have very fruitful collaborations with many of them, including Seyoung, Kim, John Lafferty, Eric Xing and Javier Peña.

During my Ph.D., I did three great internships at Microsoft Research, IBM and NEC Research Lab. I sincerely thank my mentors,

managers and friends in these labs for their great help and support: Dengyong Zhou, Lin Xiao, John Platt, Chris J.C. Burges, Paul Bennett, Kevyn-Collins Thompson, Susan Dumais, Eric Horvitz, Jingrui He, Yan Liu, Hanghang Tong, Rick Lawrence, Yanjun Qi, Bing Bai and Hans Peter Graf. I thank IBM for generously providing the Ph.D. fellowship to support the research in this thesis.

I am also indebted to many other colleagues and friends at CMU, who played critical role during my Ph.D. through collaborations, discussions and suggestions. Some of them include: Sivaraman Balakrishnan, Byron Boots, Duen Horng (Polo) Chau, Tinglong Dai, Bin Fan, Rob Hall, Qirong Ho, Tzu-Kuo Huang, Haijie (Jay) Gu, Mladen Kolar, Akshay Krishnamurthy, Hai-Son Le, Song Le, Yucheng Low, Qihang Lin, Han Liu, Yingda Lu, Yifei Ma, Ankur Parikh, Andrea Qualizza, Aaditya Ramdas, James Sharpnack, Mingyu Tang, Liang Xiong, Guang Xiang, Min Xu, Yang Xu, Chong Wang, Xiaolin Yang, Junming Yin, Yisong Yue, Linxue Zhang, Yi Zhang, Bin Zhao, Yuan Zhou, Jun Zhu. In particular, I would like to thank Han Liu and Qihang Lin. Han Liu taught me a lot on modern statistics. Qihang Lin, as one of my most important collaborators, not only taught me knowledge about the frontier of modern optimization, but also proposed many interesting problems to me. We had so many enjoyable discussions and I was often amazed by his depth and breath in mathematics.

I also owe special thanks to Diane Stidle and Sharon Cavlovich, who had done a perfect administration job and provided crucial assistance in making sure that my graduate experience is enjoyable.

Most importantly, I want to thank my grandparents, Jingrong Wang and Kai Chen. As elder Chinese scholars in mathematics and physics, at the age of 80s, they are still studying modern analysis and quantum field theory with incredible enthusiasm. Their enthusiasm about new knowledge and attitude towards science and life deeply influence me over more than 20 years and will continue to influence my entire life. This thesis is dedicated to them. I love you and miss you so much, though I never say that. I also thank my parents for their companionship, love and support. Finally, I would like to thank my wife, Yingze Wang for her unconditional love and sacrifice during the past five years — you are the best !

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

Part I

THESIS OVERVIEW

1 THESIS OVERVIEW

1.1 MOTIVATION AND STATEMENT

Modern data acquisition techniques produce massive amounts of high-dimensional data with complex structures from various domains, such as microarray and proteomics data, web text data, climatological data, and image data, etc. The task of understanding and extracting useful knowledge from these massive data presents significant challenges for machine learning and statistics. For example, in tumor classification problems, we need to select most predictive genes from thousands of genes to find promoting factors of a disease. In such a case, we face the challenge that the data is high-dimensional but the number of available samples is very limited. In addition, it is essential to incorporate the structural prior information among genes extracted from the biological domain into the learning procedure. Consider text mining task (e.g. document ranking and classification) as another example. To deal with text data, which is not only high-dimensional but also astronomical in size, the learning algorithms demand a much better scalability. Due to the high-dimensionality and complex structures of these data, traditional machine learning techniques cannot be easily applied. To address the attendant challenges, recently developed sparse learning methods provide us a suite of powerful tools. However, many of existing sparse learning methods either fail to incorporate rich structural information, or are computationally very expensive.

The main goal of this thesis is to develop both flexible and computationally efficient sparse learning methods to better understand and explore the high dimensional and complex real datasets.

1.2 THESIS OVERVIEW

The central hypothesis of the thesis is that a suite of learning and optimization techniques based on exhibiting structures and sparseness can and will enable more accurate reliable solutions to larger and richer predictive tasks across a wide variety of domains. In particular, this thesis focuses addressing both computational and statistical aspects of learning from high-dimensional large-scale structured data in the following three different aspects:

- **Optimization for learning from large-scale and high-dimensional data:** To exploit high-dimensional data, it is important to utilize the structural information in the features. We proposed a smoothing proximal gradient method as a unified framework to address computational challenges of learning from high-dimensional structured data [29, 31]. We further extended the proposed opti-

mization algorithm to *stochastic settings* for data-intensive or online applications. The developed *stochastic optimization algorithm* [30] achieves the optimal convergence rate and can be easily parallelized to handle web-scale data.

- **Statistical methods for learning complex structures:** In addition to the prediction of response values, we also proposed to predict the *structure* of responses [92]. In particular, we developed learning methods to infer *network structures* of high-dimensional responses *evolving with* the inputs (e.g., time, locations or tasks).

- **Applications:** We have applied the proposed methods to a broad class of real-world problems. Examples include text mining (e.g., learning to rank [26], structured topic modeling [28]), spatio-temporal environmental data analysis [92], and bioinformatics [29, 25].

1.3 MAIN RESULTS AND ORGANIZATION

This thesis presents a series of results in high-dimensional structured sparse learning from both computational and statistical aspects. We organize the thesis as follows.

1.3.1 *Part ii: Background*

In Part ii, we review some background of structured sparse learning, including sparse regression, structured sparse regression and its multi-task extensions, sparse canonical correlation analysis, sparse Gaussian graphical model and some basics of first-order optimization methods.

1.3.2 *Part iii: Optimization for Sparse Learning*

Variable selection plays an important role in high dimensional regression, which not only improves the prediction accuracy but also leads to better interpretation of the resulting model. A popular approach for variable selection is to jointly optimize the empirical risk function with non-smooth ℓ_1-regularization (e.g., Lasso [133]). However, the simple ℓ_1-regularization does not take advantage of prior knowledge of the structures among the variables (e.g., similarity among variables encoded as graph structures or pathways among genes represented as group structures). Recently, various structured sparsity-inducing regularizers have been proposed to encode prior structural information in high-dimensional data. However, due to the non-smoothness and non-separability of structured regularizers, the corresponding optimizations for solving the so-called *structured sparse regression* have been widely recognized as a challenging problem.

1.3.2.1 *Smoothing Proximal Gradient Methods*

To address computational challenge in *structured sparse regression*, we proposed a *unified smoothing proximal-gradient (SPG) optimization method* which can deal with a wide spectrum of structured regularizers, including (potentially overlapping) groups, hierarchical trees and graphs [27]. Utilizing the dual norm, we made a key observation that a variety of structured regularizers can be reformulated into a special form. Capitalizing on this form, we introduced its smooth approximation and solved this approximation via an accelerated gradient descent scheme. Since our method only utilizes the gradient information, it is much more scalable than other optimization methods (e.g., Newton method or interior-point method) and can be applied to ultra high-dimensional problems with millions of variables. We further proved that it enjoys a fast theoretical convergence rate of $O(1/t)$ where t is the number of iterations, which is much faster than the standard subgradient methods with the rate $O(1/\sqrt{t})$. Moreover, we proposed several variants of SPG, tailored to address *multivariate response / multi-task regression* where the correlated responses also lie in a high dimensional space with certain structures [29], as well as *structured canonical correlation analysis* [31].

We applied the proposed methods to study associations between genomic and phenotypic variations, known as *genome-wide association study*, which is a fundamental problem in computational biology. The data that we investigated are ultra high-dimensional with millions of variables and rich structural information. Our methods fully utilize the pathway and regulatory network information as prior knowledge and successfully uncover many important structures among genes, which provides new insights into gene regulation as well as potential controlling factors of diseases [29, 25].

This part of work was done through collaboration with Jaime Carbonell, Seyoung Kim, Qihang Lin, Han Liu and Eric Xing.

1.3.2.2 *Uniformly-Optimal Stochastic Optimization*

When the data are not only high-dimensional but also of enormous size, or when the data are collected in an online fashion, batch optimization methods usually cannot scale up. We proposed a new *stochastic optimization* method [30], *optimal regularized dual averaging*, which can provide real-time services for web data arriving at a high-rate. Instead of minimizing the empirical loss, the proposed method directly minimizes the regularized *expected loss*. In particular, we adopted the data sampling technique to construct an unbiased estimator of the gradient of the expected loss, which is the so-called *stochastic gradient*. Instead of directly using stochastic gradient as the descent direction, our method uses the weighted average of all historical stochastic gradients so that the noise in each gradient can be partially canceled and the algorithm provably converges to the optimal solution. Theoretically, our method achieves the *uniformly-optimal convergence rate* in the stochastic first-order optimization framework. That is, this con-

vergence rate cannot be further improved regardless of the problem being smooth or non-smooth, convex or strongly convex.

This work was done through collaboration with Qihang Lin and Javier Peña.

1.3.3 *Part iv: Learning Dynamic Graphical Models*

Exploring structural information is a critical step to understand complex data. One important structure to explore is the network structure, e.g., the dependency relationship among various features (e.g. profession, education, hobby, etc) in social network data. Graphical models, particularly Markov Random Fields, are widely used in modeling *dependency relationships* as *sparse network structures*. Most of the existing work in learning graphical models assume that the structures are static. However, this assumption is clearly violated in the real world, since the structures are often changing gradually but smoothly depending on other conditions, (e.g. time, space, etc). We formulated the problem of learning dynamic structures of graphical models as a supervised learning problem, where we predict the network structure among high-dimensional features as responses to a set of inputs. To address this so-called "graph-valued regression" problem, we proposed a new statistical method named Go-CART (Graph-Optimized Classification And Regression Tree) [92]. Our method can efficiently learn *dynamic sparse network structures* evolving over a set of dimensions and can scale up to very large networks. Theoretically, the oracle inequalities on excess risk and structure estimation consistency are established. We applied the proposed Go-CART method to study a climatological dataset and our method successfully explored dynamic dependency structures among different climatological factors evolving with time and location. We further extended Go-CART to Forest-Optimized CART where we could learn dynamic forest structured graphical models from the data and effectively deal with the discrete data.

This work was done with collaboration with Han Liu, John Lafferty and Larry Wasserman.

1.3.4 *Part v: Sparse Learning for Text Mining*

Text data is often highly sparse in the sense that many features (words, links, phrases, named entities, etc) are irrelevant for the modeling task. Therefore, the developed sparse learning methods and stochastic optimization techniques are powerful tools for various web-scale text mining tasks, e.g., *learning to rank* [26], *topic modeling* [28].

1. *Learning to Rank [26]:* We developed a scalable learning method to address ranking problem with *millions of raw features* (e.g. English words, n-grams, or tags such as POS and named-entities). Our method carefully models the pair-wise relationship to capture the synonymys and polysemes in text. We addressed the

scalability issue by adding sparsity constraint and then training the model with the developed stochastic optimization method.

2. *Topic Modeling [28]:* We developed a new latent semantic analysis method, sparse LSA, for learning a compact representation of the topic-word relationship. Compared to the standard topic modeling techniques (e.g, latent semantic analysis and latent Dirichlet allocation [14]), our method requires very small amount of memory and has low computational cost. Using sparse LSA, we can efficiently learn topics from a large corpus with tens of millions of documents.

This part of work was done when I interned at NEC Research Lab, in collaboration with Yanjun Qi and Bing Bai.

1.3.5 *Part vi: Conclusions and Future Work*

The conclusions of the thesis and future directions are provided in the last part of the thesis.

Part II

BACKGROUND

2 BACKGROUND

2.1 STRUCTURED SPARSE REGRESSION

In a high-dimensional regression problem, our goal is to predict the output y from a high-dimensional input vector $\mathbf{x} \in \mathbb{R}^J$. Given a dataset of N input/output pairs: $\{\mathbf{x}_i, y_i\}$ for $i = 1, \ldots, N$, let $\mathbf{y}$ denote the vector of outputs and $\mathbf{X}$ denote the matrix of inputs of N samples. The ℓ_1-regularized estimator can be obtained by solving the following convex optimization problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^J} f(\boldsymbol{\beta}) \equiv g(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1 = \sum_{i=1}^{n} \ell(y_i, \boldsymbol{\beta}^T \mathbf{x}_i) + \lambda \|\boldsymbol{\beta}\|_1, \tag{2.1}$$

where the loss function $\ell : \mathbb{R}^J \to \mathbb{R}$ is assumed to be a convex differentiable function with Lipschitz continuous gradient; and the regularizer is a convex non-smooth ℓ_1-norm penalty $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{J} |\beta_j|$, which is adopted to enforce sparsity among $\boldsymbol{\beta}$ [133, 24]. Typical examples of of loss function include (1) the squared loss for least squares regression:

$$\ell(y_i, \boldsymbol{\beta}^T \mathbf{x}_i) = \frac{1}{2}(y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2. \tag{2.2}$$

and the corresponding loss $g(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$; (2) the logistic loss for classification problem with $y \in \{-1, +1\}$:

$$\ell(y_i, \boldsymbol{\beta}^T \mathbf{x}_i) = \log(1 + \exp(-y_i \boldsymbol{\beta}^T \mathbf{x}_i)). \tag{2.3}$$

In the last decade, numerous methods have been proposed to study the ℓ_1-regularization regression problem, from the theoretical perspective [156, 143, 11, 155] to efficient computational methods [44, 110, 47, 147, 6]. The readers can refer to [55] (Chapter 18) for more discussions on ℓ_1-regularized methods.

The standard ℓ_1-norm penalty does not assume any dependencies, grouping or other structures among the input variables, which limits its applicability to complex high-dimensional scenarios in many real problems. More structured constraints on the input variables such as groupness or pairwise similarities can be introduced by employing a more sophisticated sparsity-inducing penalty that induces joint sparsity patterns among related inputs. We generically denote the structured penalty by $\Omega(\boldsymbol{\beta})$ and the corresponding regression problem can be formulated as:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^J} f(\boldsymbol{\beta}) \equiv g(\boldsymbol{\beta}) + \Omega(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1. \tag{2.4}$$

Note that we keep the ℓ_1-norm to explicitly enforce sparsity on every individual feature. If one is only interested in structured sparsity, the

parameter λ in (2.4) can be simply set to 0, which can be viewed as a special case of the above formulation.

In this section, we briefly overview some widely adopted structured penalties.

1. **Group Lasso Penalty with Disjoint Groups**

   In some situations, the variables are partitioned into groups and it is desirable to jointly include or exclude all the variables within a group. A typical example is when dealing with categorical data, each variable can be expressed via a group of dummy variables; and one should conduct variable selection on a group level instead of individual level. To achieve group level variable selection, one can adopt the ℓ_1/ℓ_q mixed-norm based group Lasso penalty with any $q > 1$ [150]:

   $$\Omega(\boldsymbol{\beta}) \equiv \gamma \sum_{g \in \mathcal{G}} w_g \|\boldsymbol{\beta}_g\|_q \equiv \gamma \sum_{g \in \mathcal{G}} w_g \{\sum_{j \in g} |\beta_j|^q\}^{1/q}, \tag{2.5}$$

   where $\mathcal{G}$ denotes a partition of $\{1, \ldots, J\}$, $\boldsymbol{\beta}_g \in \mathbb{R}^{|g|}$ is the subvector of $\boldsymbol{\beta}$ for the variables in group g; w_g is the predefined weight for group g; and $\|\cdot\|_q$ is the vector ℓ_q-norm . This ℓ_1/ℓ_q mixed-norm penalty plays the role of jointly setting all of the coefficients within each group to zero or non-zero values. In practice, the ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ are the most popular norms. Theoretically, it has been demonstrated that if the group structure is consistent with the true sparsity pattern, ℓ_1/ℓ_2 group Lasso penalty has the potential to improve the accuracy of the estimator [63].

2. **Group Lasso Penalty with Overlapping Groups**

   To model more complex group structures, the groups in $\mathcal{G}$ in (2.5) are allowed to overlap [66]. In other words, the coefficient for a variable β_j can appear in different ℓ_q-norms. Due to the sigularity of the ℓ_q-norm, the penalty in (2.5) will set some $\boldsymbol{\beta}_g$ to zeros. If we denote the set of groups that $\boldsymbol{\beta}_g = \mathbf{0}$ by $\mathcal{G}_0 \subset \mathcal{G}$, the support of the estimated $\widehat{\boldsymbol{\beta}}$ is:

   $$\text{supp}(\widehat{\boldsymbol{\beta}}) \subset \left(\bigcup_{g \in \mathcal{G}_0} g\right)^c \tag{2.6}$$

   A commonly used special case of general overlapping group structure is the hierarchial structure (e.g. a tree or a forest) [157]. Specially, we assume that variables correspond to nodes of a tree and a given variable is included in the model only if all its ancestors in the tree has already been selected. The general overlapping group structure has an important application in pathway selection for gene expression data. In more details, a biological pathway is a *group* of genes that participate in a particular biological process to perform certain functionality in a cell. To find the controlling factors related to a disease, it is more meaningful to study the genes by considering their pathways. We will discuss more about this application in Chapter 3.

3. **Chain-structured Fusion Penalty**

   If the variables are ordered in some meaningful way (e.g. on a timeline), using the (chain-structured) fusion penalty, we can learn a piece-wise constant coefficient vector [134]. The fusion penalty, which is the ℓ_1-norm of the coefficients' successive differences, takes the following form:

$$\Omega(\boldsymbol{\beta}) = \gamma \sum_{j=1}^{J-1} |\beta_{j+1} - \beta_j|. \tag{2.7}$$

   It has been widely applied to hot-spot detection for comparative genomic hybridization (CGH) data [136] and time-series data analysis [75].

4. **Graph-guided Fusion Penalty**

   The work in [73] extends the the chain-structure fusion penalty in (2.7) to a more general graph-guided fusion penalty. The graph-guided fusion penalty can encode the prior knowledge about the structural constraints over features in the form of pairwise relatedness described by a graph $G \equiv (V, E)$, where $V = \{1, \ldots, J\}$ denotes the variables of interest, and E denotes the set of edges among V. Additionally, we let $r_{ml} \in \mathbb{R}$ denote the weight of the edge $e = (m, l) \in E$, corresponding to correlation or other proper similarity measures between features m and l. The graph-guided fusion penalty, which encourages the coefficients of related features to share similar magnitude, is defined as follows:

$$\Omega(\boldsymbol{\beta}) = \gamma \sum_{e=(m,l)\in E, m<l} \tau(r_{ml})|\beta_m - \text{sign}(r_{ml})\beta_l|, \tag{2.8}$$

   where $\tau(r_{ml})$ represent a general weight function that enforces a fusion effect over coefficients β_m and β_l of relevant features. It can be any monotonically increasing function of the absolute values of correlations and the most popular examples include $\tau(r) = |r|$ or $\tau(r) = |r|^2$. The $\text{sign}(r_{ml})$ in (2.8) ensures that two positively correlated inputs would tend to influence the output in the same direction, whereas two negatively correlated inputs impose opposite effect. Since the fusion effect is calibrated by the edge weight, the graph-guided fusion penalty in (2.8) encourages highly inter-correlated inputs corresponding to a densely connected subnetwork in G to be jointly selected as relevant.

   It is noteworthy that when $r_{ml} = 1$ for all $e = (m, l) \in E$, and G is simply a chain over nodes, the graph-guided fusion penalty is reduced to the chain-structured fusion penalty in (2.7).

2.2 MULTI-TASK STRUCTURED SPARSE REGRESSION

In multi-task learning, we are interested in learning multiple related tasks jointly by analyzing data from all of the tasks at the same time

instead of considering each task individually [132, 20, 149, 154, 112]. When data are scarce, it is greatly advantageous to borrow the information in the data from other related tasks to learn each task more effectively. More specifically, we consider the multi-task sparse regression problem, where each task is to learn a functional mapping from a high-dimensional input space to a continuous-valued output space and only a small number of inputs are relevant to the output. In multi-task regression, it is often assumed that parameters for different tasks share the same sparsity pattern [139, 2, 113]; and the task of conducting variable selection can be achieved via learning the joint sparsity pattern of parameters.

For the simplicity of illustration, we assume all different tasks share the same input matrix. Let $\mathbf{X} \in \mathbb{R}^{N\times J}$ denote the matrix of input data for J inputs and $\mathbf{Y} \in \mathbb{R}^{N\times K}$ denote the matrix of output data for K outputs over N samples. We assume a linear regression model for each of the k-th output: $\mathbf{y}_k = \mathbf{X}\boldsymbol{\beta}_k + \boldsymbol{\epsilon}_k, \quad \forall k = 1, \dots K$, where $\boldsymbol{\beta}_k = [\beta_{1k}, \dots, \beta_{Jk}]^T$ is the regression coefficient vector for the k-th output and $\boldsymbol{\epsilon}_k$ is Gaussian noise. Let $\mathbf{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K] \in \mathbb{R}^{J\times K}$ be the matrix of regression coefficients for all of the K outputs. Then, the multi-task (or multivariate-response) structured sparse regression problem can be naturally formulated as the following optimization problem:

$$\min_{\mathbf{B}\in\mathbb{R}^{J\times K}} f(\mathbf{B}) \equiv \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\mathbf{B}\|_F^2 + \Omega(\mathbf{B}) + \lambda\|\mathbf{B}\|_1, \tag{2.9}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm, $\|\cdot\|_1$ denotes the matrix entry-wise ℓ_1 norm, and $\Omega(\mathbf{B})$ is a structured sparsity-inducing penalty with a structure over the outputs.

A popular approach is to adopt a joint sparsity regularization to encourage sparsity across all tasks. In particular, one can adopt the l_1/l_q mixed-norm penalty with $q > 1$ [2, 113, 104]:

$$\lambda\|\mathbf{B}\|_{q,1} = \gamma \sum_{j=1}^{J} \|\boldsymbol{\beta}_j\|_q, \tag{2.10}$$

where $\boldsymbol{\beta}_j = (\beta_j^1, \beta_j^2, \dots, \beta_j^K) \in \mathbb{R}^K$ is the j-th row of $\mathbf{B}$ and γ is a positive regularization parameter. The l_1/l_q mixed-norm penalty has the effect that each entire $\boldsymbol{\beta}_j$ is shrunk to zero all together. However, this penalty cannot be incorporate a complex structure in how the outputs themselves are correlated. In many applications, it is advantageous to utilize the prior structural information among outputs to guide the variable selection. For example, in genetic association analysis, where the goal is to discover few genetic variants or single nucleotide polymorphisms (SNPs) out of millions of SNPs (inputs) that influence phenotypes (outputs) such as gene expression measurements, the correlation structure of the phenotypes can be naturally represented as a graph, which can be used to guide the selection of SNPs as shown in Figure 2.1.

By extending the structured penalties introduced in Section 2.1, the two most widely used structured penalty for multi-task regression are defined as follows.

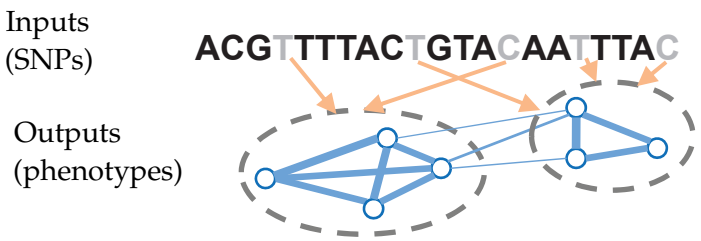


Figure 2.1: Illustration of the multi-task regression with graph structure on outputs.

1. **Group Lasso Penalty in Multi-task Regression**

   We define the group lasso penalty for a structured multi-task regression as follows:

$$\Omega(\mathbf{B}) \equiv \gamma \sum_{j=1}^{J} \sum_{g \in \mathcal{G}} w_g \|\boldsymbol{\beta}_j^g\|_q, \tag{2.11}$$

   where $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\}$ is a subset of the power set of $\{1, \ldots, K\}$ and $\boldsymbol{\beta}_j^g$ is the vector of regression coefficients correspond to outputs in group g: $\{\beta_j^k, k \in g\}$. The ℓ_1/ℓ_q mixed-norm penalty for multi-task regression in (2.10) is a special case of (2.11) where $\mathcal{G}$ only has one group $g = \{1, \ldots, J\}$. The tree-structured group-lasso penalty introduced in [72] is also a special case of (2.11).

2. **Graph-guided Fusion Penalty in Multi-task Regression**

   Assuming that a graph structure over the K outputs is given as G with a set of nodes $V = \{1, \ldots, K\}$ each corresponding to an output variable and a set of edges E, the graph-guided fusion penalty for a structured multi-task regression is given as:

$$\Omega(\mathbf{B}) = \gamma \sum_{e=(m,l) \in E} \tau(r_{ml}) \sum_{j=1}^{J} |\boldsymbol{\beta}_j^m - \text{sign}(r_{ml}) \boldsymbol{\beta}_j^l|. \tag{2.12}$$

2.3 SPARSE CANONICAL CORRELATION ANALYSIS

Given two datasets $\mathbf{X}$ and $\mathbf{Y}$ on the same set of observations, we assume that each column of $\mathbf{X}$ and $\mathbf{Y}$ is normalized to have mean zero and standard deviation one. Canonical correlation analysis (sparse CCA) [146, 145] provides a more "symmetric" solution in which it finds two *sparse* canonical vectors $\mathbf{u}$ and $\mathbf{v}$ to maximize the correlation between $\mathbf{Xu}$ and $\mathbf{Yv}$.

The sparse CCA proposed in [146, 145] takes the following form:

$$\begin{aligned} \max_{\mathbf{u},\mathbf{v}} \quad & \mathbf{u}^T \mathbf{X}^T \mathbf{Y} \mathbf{v} \\ \text{s.t.} \quad & \|\mathbf{u}\|_2 \leqslant 1, \ \|\mathbf{v}\|_2 \leqslant 1, \\ & P_1(\mathbf{u}) \leqslant c_1, \ P_2(\mathbf{v}) \leqslant c_2, \end{aligned} \tag{2.13}$$

where P_1 and P_2 are convex and non-smooth sparsity-inducing penalties that yield sparse $\mathbf{u}$ and $\mathbf{v}$. Witten et al. (2009) studied two specific forms of the penalty P (either P_1 or P_2): (1) ℓ_1-norm penalty $P(\mathbf{w}) = \|\mathbf{w}\|_1$, which will result in a sparse $\mathbf{w}$ vector. (2) chain-structured fusion penalty $P(\mathbf{w}) = \|\mathbf{w}\|_1 + \gamma \sum_j |w_j - w_{j-1}|$, which assumes that variables have a natural ordering and will result in $\mathbf{w}$ sparse and smooth along the ordering.

2.4 SPARSE GAUSSIAN GRAPHICAL MODEL

Undirected graphical models (a.k.a. Markov Random Fields) have been widely adopted for modeling dependency relationships [74]. Let Y be a p-dimensional random vector with distribution P. A common way to study the structure of P is to construct the undirected graph $G = (V, E)$, where the vertex set V corresponds to the p components of the vector Y. The edge set E is a subset of the pairs of vertices, where an edge between Y_u and Y_v is absent if and only if Y_u is conditionally independent of Y_v given all the other variables: $V_{\setminus u,v} \equiv \{Y_i, 1 \leqslant i \leqslant p, i \neq u, v\}$

$$(u, v) \notin E \Leftrightarrow Y_u \perp\!\!\!\perp Y_v \,|\, V_{\setminus u,v} \tag{2.14}$$

Let $y_1, \ldots, y_n$ be a random sample of vectors from P, where each $y_i \in \mathbb{R}^p$. We are interested in the case where p is large and, in fact, may diverge with n asymptotically. One way to estimate G from the sample is the *graphical lasso* or *glasso* [151, 48, 5], where one assumes that P is Gaussian with mean μ and covariance matrix Σ. Missing edges in the graph correspond to zero elements in the precision matrix $\Omega = \Sigma^{-1}$ [83]. A sparse estimate of Ω is obtained by solving

$$\widehat{\Omega} = \underset{\Omega \succ 0}{\arg\min} \{ \operatorname{tr}(S\Omega) - \log|\Omega| + \lambda \|\Omega\|_1 \} \tag{2.15}$$

where Ω is positive definite, S is the sample covariance matrix, and $\|\Omega\|_1 = \sum_{j,k} |\Omega_{jk}|$ is the elementwise ℓ_1-norm of Ω. A fast algorithm for finding $\widehat{\Omega}$ was given by Friedman et al. [48], which involves estimating a single row (and column) of Ω in each iteration by solving a lasso regression. The theoretical properties of $\widehat{\Omega}$ have been studied by Rothman et al. [120] and Ravikumar et al. [116].

If Y does not follow a multivariate Gaussian distribution, we can use the technique in [90] to find a set of a set of univariate functions $\{f_j\}_{j=1}^p$ such that $f(Y) \equiv (f_1(Y_1), \ldots, f_p(Y_p)) \sim N(\mu, \Sigma)$ and apply glasso on the transformed data.

2.5 FIRST-ORDER OPTIMIZATION

The aforementioned sparse regression problems can always be formulated as a composite convex optimization problem:

$$\min_{\beta} f(\beta) = g(\beta) + P(\beta), \tag{2.16}$$

where $g(\boldsymbol{\beta})$ is a smooth convex loss function with Lipschitz continuous gradient. $P(\boldsymbol{\beta})$ is a general non-smooth convex penalty function. As shown in previous sections, it can be the ℓ_1-norm $\lambda\|\boldsymbol{\beta}\|_1$; or structured penalty $\Omega(\boldsymbol{\beta})$ in Eq. (2.5), (2.7) or (2.8); or the sum of ℓ_1-norm and structured penalty $\Omega(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1$.

The traditional generic solvers include (1) subgradient descent method and (2) interior point method (IPM). For subgradient method, it converges very slowly with the rate (i.e. the number of iterations) of $O(\frac{1}{\epsilon^2})$, where ϵ is the desired accuracy and the obtained solutions are usually not sparse. For all the structured penalties that we considered here, the corresponding regression problem can always be cast to a semidefinite programming or its simpler special form (e.g. second-order cone programming (SOCP) or quadratic programming (QP)) and solved by interior point methods (IPM). Although IPM has a logarithmic convergence rate $O(\log(\frac{1}{\epsilon}))$, it is computationally prohibitive for problems of even a moderate size.

Due to the *separability* of some non-smooth penalties (e.g. ℓ_1-norm penalty), coordinate descent methods can be directly applied where we optimize the objective with one variable (or a block of variables) at a time while keeping all others fixed [47]. Although it has surprisingly good empirical performance, it is limited in that the convergence cannot be guaranteed when nondifferential terms are not separable [137].

Another class of optimization methods, proximal methods (as a special first-order methods) have become increasingly popular in the past few years in the machine learning communities. They enjoy the optimal convergence under the first-order black-box model; and more importantly, since they only use the gradient information, they are much more scalable than second-order methods or IPM and hence more suitable for large-scale applications. Although proximal methods have many variations, including Nesterov's composite method [110] and fast-iterative shrinkage-thresholding algorithm [6] (see [138] for a survey of different proximal methods), all of them need to solve a so-called proximal problem (or proximal operator, proximal mapping, generalized gradient update) at each iteration. More specifically, the proximal operator, which is based on the linearization of the smooth loss function g at the current estimate $\mathbf{w}$, takes the following form:

$$\arg\min_{\boldsymbol{\beta}} Q_L(\boldsymbol{\beta}, \mathbf{w}) \equiv g(\mathbf{w}) + \langle \nabla g(\mathbf{w}), \boldsymbol{\beta} - \mathbf{w} \rangle + \frac{L}{2}\|\boldsymbol{\beta} - \mathbf{w}\|_2^2 + P(\boldsymbol{\beta}), \tag{2.17}$$

where L is the Lipschitz constant of ∇g and hence the problem in Eq. (2.17) is a quadratic upper bound of the original problem in Eq. (2.16). We also note that the term $\frac{L}{2}\|\boldsymbol{\beta} - \mathbf{w}\|_2^2$ can be replaced by any Bregman divergence between $\boldsymbol{\beta}$ and $\mathbf{w}$. The proximal operator can be written as:

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta} - (\mathbf{w} - \frac{1}{L}\nabla g(\mathbf{w}))\|_2^2 + \frac{1}{L}P(\boldsymbol{\beta}) \tag{2.18}$$

We note that if there is no non-smooth term $P(\boldsymbol{\beta})$, Eq. (2.18) simply reduces to the standard gradient descent update rule with the step size $1/L$: $\boldsymbol{\beta} = \mathbf{w} - \frac{1}{L}\nabla g(\mathbf{w})$. It is known that when the proximal operator admits a closed-form or exact solution, proximal methods enjoy $O(1/\epsilon)$ convergence rate and accelerated techniques can further improve the rate to $O(1/\sqrt{\epsilon})$, which has already been optimal for solving any smooth convex function under the first-order black-box model [108]. Therefore, to guarantee the convergence of proximal methods, it is crucial that the structure of penalty is simple enough so that the proximal operator can be computed *exactly*; otherwise, error introduced in each proximal operator will be accumulated over iterations and the convergence is hard to analyze.

Let $\mathbf{v} = (\mathbf{w} - \frac{1}{L}\nabla g(\mathbf{w}))$, when $P(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$, it is well-known that the proximal operator is simply component-wise soft-thresholding operation [47]:

$$\boldsymbol{\beta}_j = \text{sign}(v_j)\max(0, |v_j| - \frac{\lambda}{L}). \tag{2.19}$$

Recently, a number of works have been devoted to address the issue of how to exactly compute the proximal operator. For non-overlapping groups with the ℓ_1/ℓ_2 or ℓ_1/ℓ_∞ mixed-norms, the proximal operator can be solved via a simple projection [96, 39]. A one-pass coordinate ascent method has been developed for hierarchical groups with the ℓ_1/ℓ_2 or ℓ_1/ℓ_∞ [67, 95], and quadratic min-cost network flow for arbitrary overlapping groups with the ℓ_1/ℓ_∞ [101]. However, for the ℓ_1/ℓ_2 with overlapping groups and general graph-guided fusion penalty, there is no exact solution for proximal operator and it is the exact challenge that I will address in this thesis.

In the next Part, we will first discuss how to solve the structured sparse learning for large-scale data in an efficient and scalable manner.

Part III

OPTIMIZATION FOR SPARSE LEARNING

3

SMOOTHING PROXIMAL GRADIENT METHOD FOR STRUCTURED SPARSE REGRESSION

As we discussed in the introduction part, estimating high dimensional regression models regularized by a structured sparsity-inducing penalty that encodes prior structural information on either the input or output variables remains a challenging problem from the computational aspect. In this chapter, we consider two widely adopted types of penalties of this kind as motivating examples: 1) the general overlapping-group-lasso penalty, generalized from the group-lasso penalty; and 2) the graph-guided-fused-lasso penalty, generalized from the fused-lasso penalty. We propose a general optimization approach, the *smoothing proximal gradient* (SPG) method, which can solve structured sparse regression problems with any smooth convex loss under a wide spectrum of structured sparsity-inducing penalties. Our approach combines Nesterov's smoothing technique with the accelerated proximal gradient method. It achieves a convergence rate significantly faster than the standard first-order methods, subgradient methods, and is much more scalable than the most widely used interior-point methods.

3.1 INTRODUCTION AND MOTIVATION

As outlined in Chapter 2, in recent years, various extensions of the ℓ_1-norm lasso penalty have been introduced to take advantage of the prior knowledge of the structures among inputs to encourage closely related inputs to be selected jointly [150, 134, 66]. Similar ideas have also been explored to leverage the output structures in multivariate-response regression (or multi-task regression), where one is interested in estimating multiple related functional mappings from a common input space to multiple outputs [112, 71, 72]. In this case, the structure over the outputs is available as prior knowledge, and the closely related outputs according to this structure are encouraged to share a similar set of relevant inputs. These progresses notwithstanding, the development of efficient optimization methods for solving the estimation problems resultant from the *structured sparsity-inducing penalty functions* remains a challenge for reasons we will discuss bellow. In this paper, we address the problem of developing efficient optimization methods that can handle a broad family of structured sparsity-inducing penalties with complex structures.

When the structure to be imposed during shrinkage has a relatively simple form, such as non-overlapping groups over variables (e.g., group lasso [150]), or a linear-ordering (a.k.a., chain) of variables (e.g., fused lasso [134]), efficient optimization methods have been developed. For example, under group lasso, due to the separability among

groups, a *proximal operator*[1] associated with the penalty can be computed in closed-form; thus, a number of composite gradient methods [6, 110, 96] that leverage the proximal operator as a key step (so-called "proximal gradient method") can be directly applied. For fused lasso, although the penalty is not separable, a coordinate descent algorithm was shown feasible by explicitly leveraging the linear ordering of the inputs [47].

Unfortunately, these algorithmic advancements have been outpaced by the emergence of more complex structures one would like to impose during shrinkage. For example, in order to handle a more general class of structures such as a tree or a graph over variables, various regression models that further extend the group lasso and fused lasso ideas have been recently proposed. Specifically, rather than assuming the variable groups to be non-overlapping as in the standard group lasso, the *overlapping group lasso* [66] allows each input variable to belong to multiple groups, thereby introducing overlaps among groups and enabling incorporation of more complex prior knowledge on the structure. Going beyond the standard fused lasso, the *graph-guided fused lasso* extends the original chain structure over variables to a general graph over variables, where the fused-lasso penalty is applied to each edge of the graph [73]. Due to the non-separability of the penalty terms resultant from the overlapping group or graph structures in these new models, the aforementioned fast optimization methods originally tailored for the standard group lasso or fused lasso cannot be readily applied here, due to, for example, unavailability of a closed-form solution of the proximal operator. In principle, generic convex optimization solvers such as the interior-point methods (IPM) could always be used to solve either a second-order cone programming (SOCP) or a quadratic programming (QP) formulation of the aforementioned problems; but such approaches are computationally prohibitive for problems of even a moderate size.

In this Chapter, we propose a generic optimization approach, the *smoothing proximal gradient* (SPG) method, for dealing with a broad family of sparsity-inducing penalties of complex structures. We use the overlapping-group-lasso penalty and graph-guided-fused-lasso penalty mentioned above as our motivating examples. Although these two types of penalties are seemingly very different, we show that it is possible to decouple the non-separable terms in both penalties via the dual norm; and reformulate them into a common form to which the proposed method can be applied. We call our approach a "smoothing" proximal gradient method because instead of optimizing the original objective function directly as in other proximal gradient methods, we introduce a *smooth* approximation to the structured sparsity-inducing penalty using the technique from Nesterov [109]. Then, we solve the smoothed surrogate problem by a first-order proximal gradient method known as the fast iterative shrinkage-thresholding algorithm (FISTA)[6]. We show that although we solve a smoothed

1 The proximal operator associated with the penalty is defined as: $\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta} - \mathbf{v}\|_2^2 + P(\boldsymbol{\beta})$, where $\mathbf{v}$ is any given vector and $P(\boldsymbol{\beta})$ is the non-smooth penalty.

problem, when the smoothness parameter is carefully chosen, SPG achieves a convergence rate of $O(\frac{1}{\epsilon})$ for the original objective for any desired accuracy ϵ. Below, we summarize the main advantages of this approach:

(a) It is a first-order method, as it uses only the gradient information. Thus, it is significantly more scalable than IPM for SOCP or QP. Since it is gradient-based, it allows warm restarts, thereby potentiates solving the problem along the entire regularization path [47].

(b) It is applicable to a wide class of optimization problems with a smooth convex loss and a non-smooth non-separable structured sparsity-inducing penalty. Additionally, it is applicable to both uni- and multi-task sparse structured regression, with structures on either (or both) inputs/outputs.

(c) Theoretically, it enjoys a convergence rate of $O(\frac{1}{\epsilon})$, which dominates that of the standard first-order method such as subgradient method whose rate is of $O(\frac{1}{\epsilon^2})$.

(d) Finally, SPG is very easy to implement.

3.2 SMOOTHING PROXIMAL GRADIENT

In this section, we present the smoothing proximal gradient method. The main difficulty in optimizing the objective function of structured sparse regression as defined in (2.4) arises from the non-separability of $\boldsymbol{\beta}$ in the non-smooth penalty $\Omega(\boldsymbol{\beta})$. For both types of penalties, we show that using the dual norm, the non-separable structured-sparsity-inducing penalties $\Omega(\boldsymbol{\beta})$ can be formulated as $\Omega(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}\in Q} \boldsymbol{\alpha}^T C\boldsymbol{\beta}$. Based on that, we introduce a smooth approximation to $\Omega(\boldsymbol{\beta})$ using the technique from Nesterov [109] such that its gradient with respect to $\boldsymbol{\beta}$ can be easily calculated.

3.2.1 *Reformulation of Structured Sparsity-inducing Penalty*

In this section, we show that utilizing the dual norm, the non-separable structured sparsity-inducing penalty in both (2.5) and (2.8) can be decoupled; and reformulated into a common form as a maximization problem over the auxiliary variables.

1. **Reformulating overlapping-group-lasso penalty**

   Since the dual norm of an ℓ_2-norm is also ℓ_2-norm, we can write $\|\boldsymbol{\beta}_g\|_2$ as $\|\boldsymbol{\beta}_g\|_2 = \max_{\|\boldsymbol{\alpha}_g\|_2\leqslant 1} \boldsymbol{\alpha}_g^T\boldsymbol{\beta}_g$, where $\boldsymbol{\alpha}_g \in \mathbb{R}^{|g|}$ is a vector of auxiliary variables associated with $\boldsymbol{\beta}_g$. Let $\boldsymbol{\alpha} = \left[\boldsymbol{\alpha}_{g_1}^T, \ldots, \boldsymbol{\alpha}_{g_{|\mathcal{G}|}}^T\right]^T$. Then, $\boldsymbol{\alpha}$ is a vector of length $\sum_{g\in G}|g|$ with domain $Q \equiv \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha}_g\|_2 \leqslant 1,\ \forall g \in \mathcal{G}\}$, where Q is the Cartesian product of unit balls in Euclidean space and therefore, a

closed and convex set. We can rewrite the overlapping-group-lasso penalty in (2.5) as:

$$\Omega(\boldsymbol{\beta}) = \gamma \sum_{g \in \mathcal{G}} w_g \max_{\|\boldsymbol{\alpha}_g\|_2 \leqslant 1} \boldsymbol{\alpha}_g^T \boldsymbol{\beta}_g = \max_{\boldsymbol{\alpha} \in \mathcal{Q}} \sum_{g \in \mathcal{G}} \gamma w_g \boldsymbol{\alpha}_g^T \boldsymbol{\beta}_g = \max_{\boldsymbol{\alpha} \in \mathcal{Q}} \boldsymbol{\alpha}^T C \boldsymbol{\beta}, \tag{3.1}$$

where $C \in \mathbb{R}^{\sum_{g \in \mathcal{G}} |g| \times J}$ is a matrix defined as follows. The rows of C are indexed by all pairs of $(i, g) \in \{(i, g) | i \in g, i \in \{1, \ldots, J\}, g \in \mathcal{G}\}$, the columns are indexed by $j \in \{1, \ldots, J\}$, and each element of C is given as:

$$C_{(i,g),j} = \begin{cases} \gamma w_g & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Then, we have $C\boldsymbol{\beta} = \left[\gamma w_{g_1} \boldsymbol{\beta}_{g_1}^T, \ldots, \gamma w_{g_{|\mathcal{G}|}} \boldsymbol{\beta}_{g_{|\mathcal{G}|}}^T\right]^T$.

Example. We give a concrete example of C. Assume $\boldsymbol{\beta} \in \mathbb{R}^3$ (i.e., $J = 3$) with groups $\mathcal{G} = \{g_1 = \{1, 2\}, g_2 = \{2, 3\}\}$. Then, the matrix C is defined as follows:

$$\begin{array}{r} \\ i = 1 \in g_1 \\ i = 2 \in g_1 \\ i = 2 \in g_2 \\ i = 3 \in g_2 \end{array} \begin{array}{c} \begin{array}{ccc} j = 1 & j = 2 & j = 3 \end{array} \\ \begin{pmatrix} \gamma w_{g_1} & 0 & 0 \\ 0 & \gamma w_{g_1} & 0 \\ 0 & \gamma w_{g_2} & 0 \\ 0 & 0 & \gamma w_{g_2} \end{pmatrix} \end{array}$$

Note that C is a highly sparse matrix with only a single non-zero element in each row and $\sum_{g \in \mathcal{G}} |g|$ non-zero elements in the entire matrix, and hence, can be stored with only a small amount of memory during the optimization procedure.

2. **Reformulating graph-guided-fused-lasso penalty**

First, we rewrite the graph-guided-fused-lasso penalty in (2.8) as follows:

$$\gamma \sum_{e=(m,l) \in E, m < l} \tau(r_{ml}) |\beta_m - \text{sign}(r_{ml}) \beta_l| \equiv \|C\boldsymbol{\beta}\|_1,$$

where $C \in \mathbb{R}^{|E| \times J}$ is the edge-vertex incident matrix:

$$C_{e=(m,l),j} = \begin{cases} \gamma \cdot \tau(r_{ml}) & \text{if } j = m \\ -\gamma \cdot \text{sign}(r_{ml}) \tau(r_{ml}) & \text{if } j = l \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

Again, we note that C is a highly sparse matrix with $2 \cdot |E|$ non-zero elements. Since the dual norm of the ℓ_∞-norm is the ℓ_1-norm, we can further rewrite the graph-guided-fused-lasso penalty as:

$$\|C\boldsymbol{\beta}\|_1 \equiv \max_{\|\boldsymbol{\alpha}\|_\infty \leqslant 1} \boldsymbol{\alpha}^T C \boldsymbol{\beta}, \tag{3.4}$$

where $\boldsymbol{\alpha} \in \mathcal{Q} = \{\boldsymbol{\alpha} | \|\boldsymbol{\alpha}\|_\infty \leqslant 1, \boldsymbol{\alpha} \in \mathbb{R}^{|E|}\}$ is a vector of auxiliary variables associated with $\|C\boldsymbol{\beta}\|_1$, and $\|\cdot\|_\infty$ is the ℓ_∞-norm defined as the maximum absolute value of all entries in the vector.

Remark 3.1. *As a generalization of graph-guided-fused-lasso penalty, the proposed optimization method can be applied to the ℓ_1-norm of any linear mapping of β (i.e., $\Omega(\beta) = \|C\beta\|_1$ for any given C).*

To provide a deeper insight into this reformulation, we show that (3.1) can be viewed as *Fenchel Conjugate*[58] of the indicator function.

Definition 3.1. *The Fenchel conjugate of a function $f(x)$ is the function f^* defined by:*

$$f^*(y) = \sup_{x \in dom(f)} \left(x^T y - f(x)\right). \tag{3.5}$$

Let $\delta_Q(x)$ be the indicator function:

$$\delta_Q(x) = \begin{cases} 0 & x \in Q, \\ +\infty & x \notin Q; \end{cases}$$

the penalty function $\Omega(\beta)$ is the Fenchel conjugate of δ_Q at Cv:

$$\Omega(\beta) = \max_{\alpha \in Q} \alpha^T C\beta = \delta_Q^*(C\beta). \tag{3.6}$$

3.2.2 *Smooth Approximation to Structured Sparsity-inducing Penalty*

The common formulation of $\Omega(\beta)$ given above (i.e., $\Omega(\beta) = \max_{\alpha \in Q} \alpha^T C\beta$) is still a non-smooth function of β, and this makes the optimization challenging. To tackle this problem, using the technique from Nesterov [109], we construct a smooth approximation to $\Omega(\beta)$ as following:

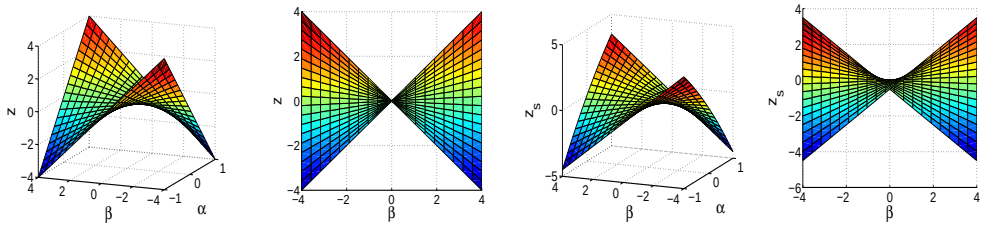
$$f_\mu(\beta) = \max_{\alpha \in Q} \left(\alpha^T C\beta - \mu d(\alpha)\right), \tag{3.7}$$

where μ is a positive smoothness parameter and $d(\alpha)$ is a smoothing function defined as $\frac{1}{2}\|\alpha\|_2^2$. The original penalty term can be viewed as $f_\mu(\beta)$ with $\mu = 0$; and one can verify that $f_\mu(\beta)$ is a lower bound of $f_0(\beta)$. In order to bound the gap between $f_\mu(\beta)$ and $f_0(\beta)$, let $D = \max_{\alpha \in Q} d(\alpha)$. In our problems, $D = |\mathcal{G}|/2$ for the overlapping-group-lasso penalty and $D = |E|/2$ for the graph-guided-fused-lasso penalty. Then, it is easy to verify that the maximum gap between $f_\mu(\beta)$ and $f_0(\beta)$ is μD:

$$f_0(\beta) - \mu D \leqslant f_\mu(\beta) \leqslant f_0(\beta).$$

From Theorem 1 as presented below, we know that $f_\mu(\beta)$ is a smooth function for any $\mu > 0$. Therefore, $f_\mu(\beta)$ can be viewed as a *smooth approximation* to $f_0(\beta)$ with a maximum gap of μD; and the μ controls the gap between $f_\mu(\beta)$ and $f_0(\beta)$. Given a desired accuracy ϵ, the convergence result in Section 3.2.5 suggests $\mu = \frac{\epsilon}{2D}$ to achieve the best convergence rate. When $\mu = \frac{\epsilon}{2D}$, we have $f_0(\beta) - \frac{\epsilon}{2} \leqslant f_\mu(\beta) \leqslant f_0(\beta)$.

Now we present the key theorem [109] to show that $f_\mu(\beta)$ is smooth in β with a simple form of the gradient.



(a) (b) (c) (d)

Figure 3.1: A geometric illustration of the smoothness of $f_\mu(\beta)$. (a) The 3-D plot of $z(\alpha, \beta)$, (b) the projection of (a) onto the β-z space, (c) the 3-D plot of $z_s(\alpha, \beta)$, and (d) the projection of (c) onto the β-z space.

Theorem 3.1. *For any $\mu > 0$, $f_\mu(\boldsymbol{\beta})$ is a convex and continuously-differentiable function in $\boldsymbol{\beta}$, and the gradient of $f_\mu(\boldsymbol{\beta})$ takes the following form:*

$$\nabla f_\mu(\boldsymbol{\beta}) = C^T \boldsymbol{\alpha}^*, \tag{3.8}$$

where $\boldsymbol{\alpha}^*$ is the optimal solution to (3.7). Moreover, the gradient $\nabla f_\mu(\boldsymbol{\beta})$ is Lipschitz continuous with the Lipschitz constant $L_\mu = \frac{1}{\mu}\|C\|^2$, where $\|C\|$ is the matrix spectral norm of C defined as $\|C\| \equiv \max_{\|\mathbf{v}\|_2 \leqslant 1} \|C\mathbf{v}\|_2$.

By viewing $f_\mu(\boldsymbol{\beta})$ as the *Fenchel Conjugate* of $d(\cdot)$ at $\frac{C\boldsymbol{\beta}}{\mu}$, the smoothness can be obtained by applying Theorem 26.3 in Rockafellar [119]. The gradient in (3.8) can be derived from the Danskin's Theorem [10] and the Lipschitz constant is shown in Nesterov [109]. For the purpose of completeness, the details of the proof are given in the appendix.

Geometric illustration of Theorem 3.1 To provide insights on why $f_\mu(\boldsymbol{\beta})$ is a smooth function as Theorem 1 suggests, in Figure 3.1, we show a geometric illustration for the case of one-dimensional parameter (i.e., $\beta \in \mathbb{R}$) with μ and C set to 1. First, we show geometrically that $f_0(\beta) = \max_{\alpha \in [-1,1]} z(\alpha, \beta)$ with $z(\alpha, \beta) \equiv \alpha\beta$ is a non-smooth function. The three-dimensional plot for $z(\alpha, \beta)$ with α restricted to $[-1, 1]$ is shown in Figure 3.1(a). We project the surface in Figure 3.1(a) onto the $\beta - z$ space as shown in Figure 3.1(b). For each β, the value of $f_0(\beta)$ is the highest point along the z-axis since we maximize over α in $[-1, 1]$. We can see that $f_0(\beta)$ is composed of two segments with a sharp point at $\beta = 0$ and hence is *non-smooth*. Now, we introduce $d(\alpha) = \frac{1}{2}\alpha^2$, let $z_s(\alpha, \beta) \equiv \alpha\beta - \frac{1}{2}\alpha^2$ and $f_\mu(\beta) = \max_{\alpha \in [-1,1]} z_s(\alpha, \beta)$. The three-dimensional plot for $z_s(\alpha, \beta)$ with α restricted to $[-1, 1]$ is shown in Figure 3.1(c). Similarly, we project the surface in Figure 3.1(c) onto the $\beta - z_s$ space as shown in Figure 3.1(d). For fixed β, the value of $f_\mu(\beta)$ is the highest point along the z-axis. In Figure 3.1(d), we can see that the sharp point at $\beta = 0$ is removed and $f_\mu(\beta)$ becomes *smooth*.

To compute the $\nabla f_\mu(\boldsymbol{\beta})$ and L_μ, we need to know $\boldsymbol{\alpha}^*$ and $\|C\|$. We present the closed-form equations for $\boldsymbol{\alpha}^*$ and $\|C\|$ for the overlapping-group-lasso penalty and graph-guided-fused-lasso penalty in the following propositions. The proof is presented in the appendix.

1. **$\boldsymbol{\alpha}^*$ under overlapping-group-lasso penalty**

Proposition 3.1. *Let $\boldsymbol{\alpha}^*$, which is composed of $\{\boldsymbol{\alpha}_g^*\}_{g\in\mathcal{G}}$, be the optimal solution to (3.7) for the overlapping-group-lasso penalty in (2.5). For any $g \in \mathcal{G}$,*

$$\boldsymbol{\alpha}_g^* = S_2(\frac{\gamma w_g \boldsymbol{\beta}_g}{\mu}),$$

where S_2 is the projection operator which projects any vector $\mathbf{u}$ to the ℓ_2 ball:

$$S_2(\mathbf{u}) = \begin{cases} \frac{\mathbf{u}}{\|\mathbf{u}\|_2} & \|\mathbf{u}\|_2 > 1, \\ \mathbf{u} & \|\mathbf{u}\|_2 \leqslant 1. \end{cases} \tag{3.9}$$

In addition, we have $\|C\| = \gamma \max_{j\in\{1,\ldots,J\}} \sqrt{\sum_{g\in\mathcal{G} \text{ s.t. } j\in g} (w_g)^2}$.

2. **$\boldsymbol{\alpha}^*$ under graph-guided-fused-lasso penalty**

Proposition 3.2. *Let $\boldsymbol{\alpha}^*$ be the optimal solution of (3.7) for graph-guided-fused-lasso penalty in (2.8). Then, we have:*

$$\boldsymbol{\alpha}^* = S_\infty(\frac{C\boldsymbol{\beta}}{\mu}),$$

where S_∞ is the projection operator defined as follows:

$$S_\infty(x) = \begin{cases} x, & \text{if} \quad -1 \leqslant x \leqslant 1 \\ 1, & \text{if} \quad x > 1 \\ -1, & \text{if} \quad x < -1. \end{cases}$$

For any vector $\boldsymbol{\alpha}$, $S_\infty(\boldsymbol{\alpha})$ is defined as applying S_∞ on each and every entry of $\boldsymbol{\alpha}$.

$\|C\|$ is upper-bounded by $\sqrt{2\gamma^2 \max_{j\in V} d_j}$, where

$$d_j = \sum_{e\in E \text{ s.t. } e \text{ incident on } j} (\tau(r_e))^2 \tag{3.10}$$

for $j \in V$ in graph G, and this bound is tight. Note that when $\tau(r_e) = 1$ for all $e \in E$, d_j is simply the degree of the node j.

3.2.3 *Smoothing Proximal Gradient Descent*

Given the smooth approximation to the non-smooth structured sparsity-inducing penalties, now, we apply the fast iterative shrinkage-thresholding algorithm (FISTA) [6, 138] to solve a generically reformulated optimization problem, using the gradient information from Theorem 3.1. We substitute the penalty term $\Omega(\boldsymbol{\beta})$ in (2.4) with its smooth approximation $f_\mu(\boldsymbol{\beta})$ to obtain the following optimization problem:

$$\min_{\boldsymbol{\beta}} \widetilde{f}(\boldsymbol{\beta}) \equiv g(\boldsymbol{\beta}) + f_\mu(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1. \tag{3.11}$$

Let

$$h(\boldsymbol{\beta}) = g(\boldsymbol{\beta}) + f_\mu(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + f_\mu(\boldsymbol{\beta}). \tag{3.12}$$

Algorithm 3.1 Smoothing Proximal Gradient Descent (SPG) for Structured Sparse Regression

Input: $\mathbf{X}$, $\mathbf{y}$, C, $\boldsymbol{\beta}^0$, Lipschitz constant L, desired accuracy ϵ.
Initialization: set $\mu = \frac{\epsilon}{2D}$ where $D = \max_{\boldsymbol{\alpha}\in Q} \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2$ ($D = |\mathcal{G}|/2$ for the overlapping-group-lasso penalty and $D = |E|/2$ for the graph-guided-fused-lasso penalty), $\theta_0 = 1$, $\mathbf{w}^0 = \boldsymbol{\beta}^0$.
Iterate For $t = 0, 1, 2, \ldots$, until convergence of $\boldsymbol{\beta}^t$:

1. Compute $\nabla h(\mathbf{w}^t)$ according to (3.13).
2. Solve the proximal operator associated with the ℓ_1-norm:

$$\begin{aligned} \boldsymbol{\beta}^{t+1} &= \arg\min_{\boldsymbol{\beta}} Q_L(\boldsymbol{\beta}, \mathbf{w}^t) \\ &\equiv \arg\min_{\boldsymbol{\beta}} h(\mathbf{w}^t) + \langle \boldsymbol{\beta} - \mathbf{w}^t, \nabla h(\mathbf{w}^t)\rangle + \lambda\|\boldsymbol{\beta}\|_1 + \frac{L}{2}\|\boldsymbol{\beta} - \mathbf{w}^t\|_2^2 \end{aligned} \tag{3.15}$$

3. Set $\theta_{t+1} = \frac{2}{t+3}$.
4. Set $\mathbf{w}^{t+1} = \boldsymbol{\beta}^{t+1} + \frac{1-\theta_t}{\theta_t}\theta_{t+1}(\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t)$.

Output: $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{t+1}$.

be the smooth part of $\widetilde{f}(\boldsymbol{\beta})$. According to Theorem 3.1, the gradient of $h(\boldsymbol{\beta})$ is given as:

$$\nabla h(\boldsymbol{\beta}) = \mathbf{X}^T(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}) + C^T\boldsymbol{\alpha}^*. \tag{3.13}$$

Moreover, $\nabla h(\boldsymbol{\beta})$ is Lipschitz-continuous with the Lipschitz constant:

$$L = \lambda_{\max}(\mathbf{X}^T\mathbf{X}) + L_\mu = \lambda_{\max}(\mathbf{X}^T\mathbf{X}) + \frac{\|C\|^2}{\mu}, \tag{3.14}$$

where $\lambda_{\max}(\mathbf{X}^T\mathbf{X})$ is the largest eigenvalue of $(\mathbf{X}^T\mathbf{X})$.

Since $\widetilde{f}(\boldsymbol{\beta})$ only involves a very simple non-smooth part (i.e., the ℓ_1-norm penalty), we can adopt FISTA [6] to minimize $\widetilde{f}(\boldsymbol{\beta})$ as shown in Algorithm 3.1. Algorithm 3.1 alternates between the sequences $\{w^t\}$ and $\{\boldsymbol{\beta}^t\}$ and θ_t can be viewed as a special "step-size", which determines the relationship between $\{w^t\}$ and $\{\boldsymbol{\beta}^t\}$ as in Step 4 of Algorithm 3.1. As shown in Beck and Teboulle [6], such a way of setting θ_t leads to Lemma 1 in Appendix, which further guarantees the convergence result in Theorem 3.2.

Rewriting $Q_L(\boldsymbol{\beta}, \mathbf{w}^t)$ in (3.15):

$$Q_L(\boldsymbol{\beta}, \mathbf{w}^t) = \frac{1}{2}\|\boldsymbol{\beta} - (\mathbf{w}^t - \frac{1}{L}\nabla h(\mathbf{w}^t))\|_2^2 + \frac{\lambda}{L}\|\boldsymbol{\beta}\|_1.$$

Let $\mathbf{v} = (\mathbf{w}^t - \frac{1}{L}\nabla h(\mathbf{w}^t))$, the closed-form solution for $\boldsymbol{\beta}^{t+1}$ can be obtained by soft-thresholding [47] as presented in the next proposition.

Proposition 3.3. *The closed-form solution of*

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta} - \mathbf{v}\|_2^2 + \frac{\lambda}{L}\|\boldsymbol{\beta}\|_1$$

can be obtained by the soft-thresholding operation:

$$\beta_j = \text{sign}(v_j)\max(0, |v_j| - \frac{\lambda}{L}), \quad j = 1, \ldots, J. \tag{3.16}$$

An important advantage of using the proximal operator associated with the ℓ_1-norm $Q_L(\boldsymbol{\beta}, \mathbf{w}^t)$ is that it can provide us with sparse solutions, where the coefficients for irrelevant inputs are set *exactly* to zeros, due to the soft-thresholding operation in (3.16). When the term $\lambda\|\boldsymbol{\beta}\|_1$ is not included in the objective, for overlapping group lasso, we can only obtain the group level sparsity but not the individual feature level sparsity inside each group. However, as for optimization, Algorithm 3.1 still applies in the same way. The only difference is that Step 2 of Algorithm 3.1 becomes $\boldsymbol{\beta}^{t+1} = \arg\min_{\boldsymbol{\beta}} h(\mathbf{w}^t) + \langle \boldsymbol{\beta} - \mathbf{w}^t, \nabla h(\mathbf{w}^t)\rangle + \frac{L}{2}\|\boldsymbol{\beta} - \mathbf{w}^t\|_2^2 = \mathbf{w}^t - \frac{1}{L}\nabla h(\mathbf{w}^t)$. Since there is no soft-thresholding step, the obtained solution $\widehat{\boldsymbol{\beta}}$ has no exact zeros. We then need to set a threshold (e.g., 10^{-5}) and select the relevant groups which contain the variables with the parameter above this threshold.

3.2.4 *Issues on the Computation of the Lipschitz Constant*

When J is large, the computation of $\lambda_{\max}(\mathbf{X}^T\mathbf{X})$ and hence the Lipschitz constant L could be very expensive. To further accelerate Algorithm 3.1, a line search backtracking step could be used to dynamically assign a constant L_t for the proximal operator in each iteration [6]. More specifically, given any positive constant R, let

$$Q_R(\boldsymbol{\beta}, \mathbf{w}^t) = h(\mathbf{w}^t) + \langle \boldsymbol{\beta} - \mathbf{w}^t, \nabla h(\mathbf{w}^t)\rangle + \lambda\|\boldsymbol{\beta}\|_1 + \frac{R}{2}\|\boldsymbol{\beta} - \mathbf{w}^t\|_2^2,$$

and

$$\boldsymbol{\beta}^{t+1} \equiv \boldsymbol{\beta}_R(\mathbf{w}^t) = \arg\min_{\boldsymbol{\beta}} Q_R(\boldsymbol{\beta}, \mathbf{w}^t).$$

The key to guarantee the convergence rate of Algorithm 3.1 is to ensure that the following inequality holds for each iteration:

$$\widetilde{f}(\boldsymbol{\beta}^{t+1}) = h(\boldsymbol{\beta}^{t+1}) + \lambda\|\boldsymbol{\beta}^{t+1}\|_1 \leqslant Q_R(\boldsymbol{\beta}^{t+1}, \mathbf{w}^t). \tag{3.17}$$

It is easy to check when R is equal to the Lipschitz constant L, it will satisfy the above inequality for any $\boldsymbol{\beta}^{t+1}$ and $\mathbf{w}^t$. However, when it is difficult to compute the Lipschitz constant, instead of using a global constant L, we could find a sequence $\{L_t\}_{t=0}^T$ such that L_{t+1} satisfies the inequality (3.17) for the t-th iteration. In particular, we start with any small constant L_0. For each iteration, we find the smallest integer $a \in \{0, 1, 2, \ldots\}$ such that by setting $L_{t+1} = \tau^a L_t$ where $\tau > 1$ is a pre-defined scaling factor, we have:

$$\widetilde{f}(\boldsymbol{\beta}_{L_{t+1}}(\mathbf{w}^t)) \leqslant Q_{L_{t+1}}(\boldsymbol{\beta}_{L_{t+1}}(\mathbf{w}^t), \mathbf{w}^t). \tag{3.18}$$

Then we set $\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}_{L_{t+1}}(\mathbf{w}^t) \equiv \arg\min Q_{L_{t+1}}(\boldsymbol{\beta}, \mathbf{w}^t)$.

3.2.5 *Convergence Rate and Time Complexity*

Although we optimize the approximation function $\widetilde{f}(\beta)$ rather than the original $f(\beta)$ directly, it can be proven that $f(\widehat{\beta})$ is sufficiently close to the optimal objective value of the *original* function $f(\beta^*)$. The convergence rate of Algorithm 3.1 is presented in the next theorem.

Theorem 3.2. *Let β^* be the optimal solution to (2.4) and β^t be the approximate solution at the t-th iteration in Algorithm 3.1. If we require $f(\beta^t) - f(\beta^*) \leqslant \epsilon$ where f is the original objective, and set $\mu = \frac{\epsilon}{2D}$, then the number of iterations t is upper-bounded by*

$$\sqrt{\frac{4\|\beta^* - \beta^0\|_2^2}{\epsilon}\left(\lambda_{\max}(X^TX) + \frac{2D\|C\|^2}{\epsilon}\right)}. \tag{3.19}$$

The key idea behind the proof of this theorem is to decompose $f(\beta^t) - f(\beta^*)$ into three parts: (i) $f(\beta^t) - \widetilde{f}(\beta^t)$, (ii) $\widetilde{f}(\beta^t) - \widetilde{f}(\beta^*)$, and (iii) $\widetilde{f}(\beta^*) - f(\beta^*)$. (i) and (iii) can be bounded by the gap of the approximation μD; and (ii) only involves the function $\widetilde{f}$ and can be upper bounded by $O(\frac{1}{t^2})$ as shown in Beck and Teboulle [6]. We obtain (3.19) by balancing these three terms. The details of the proof are presented in the appendix. According to Theorem 3.2, Algorithm 3.1 converges in $O(\frac{\sqrt{2D}}{\epsilon})$ iterations, which is much faster than the subgradient method with the convergence rate of $O(\frac{1}{\epsilon^2})$. Note that the convergence rate depends on D through the term $\sqrt{2D}$, and the D depends on the problem size.

Remark 3.2. *Since there is no line search in Algorithm 3.1, we cannot guarantee that the objective values are monotonically decreasing over iterations theoretically. One simple strategy to guarantee the monotone decreasing property is to first compute $\widetilde{\beta}^{t+1} = arg\,min_\beta\, Q_L(\beta, w^t)$ and then set $\beta^{t+1} = arg\,min_{\beta \in \{\widetilde{\beta}^{t+1}, \beta^t\}} f(\beta)$.*

Remark 3.3. *Theorem 3.2 only shows the convergence rate for the objective value. As for the estimator β^t, since it is a convex optimization problem, it is well known that β^t will eventually converge to β^*. However, the speed of convergence of β^t to β^* depends on the structure of the input X. If $h(\beta)$ is a strongly convex function with the strongly convexity parameter $\sigma > 0$. In our problem, it is equivalent to saying that X^TX is a non-singular matrix with the smallest eigenvalue $\sigma > 0$. Then we can show that if $f(\beta^t) - f(\beta^*) \leqslant \epsilon$ at the convergence, then $\|\beta^t - \beta^*\|_2 \leqslant \sqrt{\frac{2\epsilon}{\sigma}}$. In other words, β^t converges to β^* in ℓ_2-distance at the rate of $O(\frac{1}{\epsilon^2})$. For general high-dimensional sparse learning problems with $J > N$, X^TX is singular and hence the optimal solution β^* is not unique. In such a case, one can only show that β^t will converge to one of the optimal solutions. But the speed of the convergence of $\|\beta^t - \beta^*\|_2$ or its relationship with $f(\beta^t) - f(\beta^*)$ is widely recognized as an open problem in optimization community.*

As for the time complexity, the main computational cost in each iteration comes from calculating the gradient $\nabla h(w_t)$. Therefore, SPG shares almost the same per-iteration time as the subgradient descent

Table 3.1: Comparison of Per-iteration Time Complexity

	Overlapping Group Lasso	Graph-guided Fused Lasso
SPG	$O(J \min(J, N) + \sum_{g \in \mathcal{G}} \lvert g \rvert)$	$O(J \min(J, N) + \lvert E \rvert)$
IPM	$O\left((J + \lvert\mathcal{G}\rvert)^2 (N + \sum_{g \in \mathcal{G}} \lvert g \rvert)\right)$	$O\left((J + \lvert E \rvert)^3\right)$

but with a faster convergence rate. In more details, if $J < N$ and $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$ can be pre-computed and stored in memory, the computation of first part of $\nabla h(\mathbf{w}_t)$, $(\mathbf{X}^T\mathbf{X})\mathbf{w}_t - (\mathbf{X}^T\mathbf{y})$, takes the time complexity of $O(J^2)$. Otherwise, if $J > N$, we can compute this part by $\mathbf{X}^T(\mathbf{X}\mathbf{w}_t - \mathbf{y})$ which takes the time complexity of $O(JN)$. As for the generic solver, IPM for SOCP for overlapping group lasso or IPM for QP for graph-guided fused lasso, although it converges in fewer iterations (i.e., $\log(\frac{1}{\epsilon})$), its per-iteration complexity is higher by orders of magnitude than ours as shown in Table 3.1. In addition to time complexity, IPM requires the pre-storage of $\mathbf{X}^T\mathbf{X}$ and each IPM iteration requires significantly more memory to store the Newton linear system. Therefore, the SPG is much more efficient and scalable for large-scale problems.

3.3 RELATED OPTIMIZATION METHODS

Recently, many first-order approaches have been developed for various subclasses of overlapping group lasso and graph-guided fused lasso. Below, we provide a survey of these methods.

3.3.1 *Related work for mixed-norm based group-lasso penalty*

Most of the existing optimization methods developed for mixed-norm penalties can handle only a specific subclass of the general overlapping-group-lasso penalties. Most of these methods use the proximal gradient framework [6, 110] and focus on the issue of how to *exactly* solve the proximal operator. For non-overlapping groups with the ℓ_1/ℓ_2 or ℓ_1/ℓ_∞ mixed-norms, the proximal operator can be solved via a simple projection [96, 39]. A one-pass coordinate ascent method has been developed for tree-structured groups with the ℓ_1/ℓ_2 or ℓ_1/ℓ_∞ [67, 95], and quadratic min-cost network flow for arbitrary overlapping groups with the ℓ_1/ℓ_∞ [101].

Table 3.2 summarizes the applicability, the convergence rate, and the per-iteration time complexity for the available first-order methods for different subclasses of group lasso penalties. More specifically, the methods in the first three rows adopt the proximal gradient framework. The first column of these rows gives the solver for the proximal operator. Each entry in Table 3.2 contains the convergence rate and the per-iteration time complexity. For the sake of simplicity, for all methods, we omit the time for computing the gradient of the loss function which is required for all of the methods (i.e., $\nabla g(\boldsymbol{\beta})$ with

Table 3.2: Comparisons of different first-order methods for optimizing mixed-norm based overlapping-group-lasso penalties.

Method	No overlap ℓ_1/ℓ_2	No overlap ℓ_1/ℓ_∞	Overlap Tree ℓ_1/ℓ_2	Overlap Tree ℓ_1/ℓ_∞	Overlap Arbitrary ℓ_1/ℓ_2	Overlap Arbitrary ℓ_1/ℓ_∞
Projection [96]	$O(\frac{1}{\sqrt{\epsilon}})$ $O(J)$	$O(\frac{1}{\sqrt{\epsilon}})$ $O(J \log J)$	N.A.	N.A.	N.A.	N.A.
Coordinate Ascent [67, 95]	$O(\frac{1}{\sqrt{\epsilon}})$ $O(J)$	$O(\frac{1}{\sqrt{\epsilon}})$ $O(J \log J)$	$O(\frac{1}{\sqrt{\epsilon}})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert)$	$O(\frac{1}{\sqrt{\epsilon}})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert \log \lvert g\rvert)$	N.A.	N.A.
Network Flow [101]	N.A.	$O(\frac{1}{\sqrt{\epsilon}})$ quadratic min-cost flow	N.A.	$O(\frac{1}{\sqrt{\epsilon}})$ quadratic min-cost flow	N.A.	$O(\frac{1}{\sqrt{\epsilon}})$ quadratic min-cost flow
FOBOS [39]	$O(\frac{1}{\epsilon})$ $O(J)$	$O(\frac{1}{\epsilon})$ $O(J \log J)$	$O(\frac{1}{\epsilon})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert)$	$O(\frac{1}{\epsilon})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert \log \lvert g\rvert)$	$O(\frac{1}{\epsilon^2})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert)$ (subgradient)	$O(\frac{1}{\epsilon})$ quadratic min-cost flow
SPG	$O(\frac{1}{\epsilon})$ $O(J)$	$O(\frac{1}{\epsilon})$ $O(J \log J)$	$O(\frac{1}{\epsilon})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert)$	$O(\frac{1}{\epsilon})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert \log \lvert g\rvert)$	$O(\frac{1}{\epsilon})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert)$	$O(\frac{1}{\epsilon})$ $O(\sum_{g\in\mathcal{G}} \lvert g\rvert \log \lvert g\rvert)$

$O(J^2)$). The per-iteration time complexity in the table may come from the computation of proximal operator or subgradient of the penalty. "N.A." stands for "not applicable" or no guarantee in the convergence. As we can see from Table 3.2, although our method is not the most ideal one for some of the special cases, our method along with FOBOS [39] are the only generic first-order methods that can be applied to all subclasses of the penalties.

As we can see from Table 3.2, for arbitrary overlaps with the ℓ_1/ℓ_∞, although the method proposed in Mairal et al. [101] achieves $O(\frac{1}{\sqrt{\epsilon}})$ convergence rate, the per-iteration complexity can be high due to solving a quadratic min-cost network flow problem. From the worst-case analysis, the per-iteration time complexity for solving the network flow problem in Mairal et al. [101] is at least $O(|V||E|) = O((J + |\mathcal{G}|)(|\mathcal{G}| + J + \sum_{g\in\mathcal{G}} |g|))$, which is much higher than our method with $O(\sum_{g\in\mathcal{G}} |g| \log |g|)$. More importantly, for the case of arbitrary overlaps with the ℓ_1/ℓ_2, our method has a superior convergence rate to all the other methods.

3.3.2 *Related work for fused lasso*

For the graph-guided-fused-lasso penalty, when the structure is a simple chain, the pathwise coordinate descent method [47] can be applied. For the general graph structure, a first-order method that approximately solves the proximal operator was proposed in Liu et al. [97]. However, the convergence cannot be guaranteed due to the errors introduced in computing the proximal operator over iterations.

Recently, two different path algorithms have been proposed [135, 159] that can be used to solve the graph-guided fused lasso as a special case. Unlike the traditional optimization methods that solve the

Table 3.3: Comparisons of different methods for optimizing graph-guided fused lasso

Method & Condition	Pre-processing Time	Per-iteration Time Complexity	No. of Iterations
H. Zhou & K. Lange [159] ($\mathbf{X}$ full column rank, entire path)	$O(J^3)$	$O\left((\lvert E\rvert + J)^2\right)$	$O(\lvert E\rvert + J)$
R. Tibshirani & J. Taylor [135] ($\mathbf{X}$ full column rank, entire path)	$O\left(J^3 + N(\lvert E\rvert + J)\min((\lvert E\rvert + J), N)\right)$	$O\left(\min((\lvert E\rvert + J)^2, N^2)\right)$	$O(\lvert E\rvert + J)$ (lower bound)
R. Tibshirani & J. Taylor [135] ($\mathbf{X}$ not full column rank, entire path)	$O\left(J^3 + J^2N + (\lvert E\rvert + J)^2N\right)$	$O(N^2)$	$O(\lvert E\rvert + J)$ (lower bound)
SPG (single regularization parameter)	$O(NJ^2)$	$O(J^2 + \lvert E\rvert)$	$O(\frac{1}{\epsilon})$

problem for a fixed regularization parameter, they solve the entire path of solutions, and thus, has great practical advantages. In addition, for both methods, updating solutions from one hitting time to another is computationally very cheap. More specifically, a QR decomposition based updating scheme was proposed in Tibshirani and Taylor [135] and the updating in Zhou and Lange [159] can be done by an efficient sweep operation.

However, for high-dimensional data with $J \gg N$, the path algorithms have the following problems:

1. For a general design matrix $\mathbf{X}$ other than the identity matrix, the method in Tibshirani and Taylor [135] needs to first compute the pseudo-inverse of $\mathbf{X}$: $\mathbf{X}^+ = (\mathbf{X}^T\mathbf{X})^+\mathbf{X}^T$, which could be computationally expensive for large J.

2. The original version of the algorithms in Tibshirani and Taylor [135], Zhou and Lange [159] requires that $\mathbf{X}$ has a full column rank. When $J > N$, although one can add an extra $\epsilon\|\beta\|_2^2$ term, this changes the original objective value especially when ϵ is large. For smaller ϵ, the matrix $(\mathbf{X}^*)^T\mathbf{X}^*$ with $\mathbf{X}^* = \begin{bmatrix} \mathbf{X} \\ \\ \epsilon I \end{bmatrix}$ is highly ill-conditioned; and hence computing its inverse as the initilization step in Tibshirani and Taylor [135] is very difficult. There is no known result on how to balance this trade-off.

3. In both Tibshirani and Taylor [135] and Zhou and Lange [159], the authors extend their algorithm to deal with the case when $\mathbf{X}$ does not have a full column rank. The extended version requires a Gramm-Schmidt process as the initialization which could take some extra time.

In Table 3.3, we present the comparisons for different methods. From our analysis, the method in [159] is more efficient than the one in Tibshirani and Taylor [135] since it avoids the heavy computation of the pseudo-inverse of $\mathbf{X}$. In practice, if $\mathbf{X}$ has a full column rank and one is interested in solutions on the entire path, the method in [159] is very efficient and faster than our method.

3.4 EXTENSIONS TO MULTI-TASK REGRESSION WITH STRUCTURES ON OUTPUTS

The structured sparsity-inducing penalties as discussed in the previous section can be similarly used in the multi-task regression setting [72, 73] where the prior structural information is available for the outputs instead of inputs. In a sparse multi-task regression with structure on the output side, we encounter the same difficulties of optimizing with non-smooth and non-separable penalties as in the previous section, and the SPG can be extended to this problem in a straightforward manner. Due to the importance of this class of problems and its applications, in this section, we briefly discuss how our method can be applied to the multi-task regression with structured-sparsity-inducing penalties.

Using the similar techniques in Section 3.2.1, $\Omega(\mathbf{B})$ can be reformulated as:

$$\Omega(\mathbf{B}) = \max_{\mathbf{A}\in\mathcal{Q}} \langle C\mathbf{B}^T, \mathbf{A}\rangle, \tag{3.20}$$

where $\langle \mathbf{U}, \mathbf{V}\rangle \equiv \mathrm{Tr}(\mathbf{U}^T\mathbf{V})$ denotes a matrix inner product. C is constructed in the similar way as in (3.2) or (3.3) just by replacing the index of the input variables with the output variables, and $\mathbf{A}$ is the matrix of auxiliary variables. In particular,

1. Group Structure: $\mathbf{A} = \left[[\boldsymbol{\alpha}_{1g_1}^T \dots \boldsymbol{\alpha}_{1g_{|G|}}^T]^T \dots [\boldsymbol{\alpha}_{Jg_1}^T \dots \boldsymbol{\alpha}_{Jg_{|G|}}^T]^T\right]$, where each subvector $\boldsymbol{\alpha}_{jg}$ is the auxiliary variables for $\|\boldsymbol{\beta}_{jg}\|_2$ such that $\|\boldsymbol{\beta}_{jg}\|_2 = \max_{\|\boldsymbol{\alpha}_{jg}\|_2\leqslant 1} \boldsymbol{\alpha}_{jg}^T\boldsymbol{\beta}_{jg}$. $\mathbf{A}$ is a $(\sum_{g\in\mathcal{G}}|g|)\times J$ matrix with domain $\mathcal{Q} \equiv \{\mathbf{A} \mid \|\boldsymbol{\alpha}_{jg}\|_2 \leqslant 1,\ \forall\, j \in \{1,\dots,J\}, g \in \mathcal{G}\}$.

2. Graph Structure: $\mathbf{A}$ is the auxiliary matrix for $\|C\mathbf{B}^T\|_1$ such that

$$\|C\mathbf{B}^T\|_1 \equiv \max_{\|\mathbf{A}\|_\infty\leqslant 1} \langle C\mathbf{B}^T\mathbf{A}\rangle,$$

   where $\|\cdot\|_\infty$ is the matrix entry-wise infinity norm. Therefore, we define $\mathbf{A}$ as a J by $K + |E|$ matrix with the domain $\mathcal{Q} = \{\mathbf{A} | \|\mathbf{A}\|_\infty \leqslant 1\}$.

Then we introduce the smooth approximation of (3.20):

$$f_\mu(\mathbf{B}) = \max_{\mathbf{A}\in\mathcal{Q}} \left(\langle C\mathbf{B}^T, \mathbf{A}\rangle - \mu d(\mathbf{A})\right), \tag{3.21}$$

where $d(\mathbf{A}) \equiv \frac{1}{2}\|\mathbf{A}\|_F^2$. Following a proof strategy similar to that in Theorem 3.1, we can show that $f_\mu(\mathbf{B})$ is convex and smooth with gradient $\nabla f_\mu(\mathbf{B}) = (\mathbf{A}^*)^T C$, where $\mathbf{A}^*$ is the optimal solution to (3.21). The closed-form solution of $\mathbf{A}^*$ and the Lipschitz constant for $\nabla f_\mu(\mathbf{B})$ can be derived in the same way.

By substituting $\Omega(\mathbf{B})$ in (2.9) with $f_\mu(\mathbf{B})$, we can adopt Algorithm 3.1 to solve (2.9) with convergence rate of $O(\frac{1}{\epsilon})$. The per-iteration time complexity of SPG as compared to IPM for SOCP or QP formulation is presented int Table 3.4. As we can see, the per-iteration complexity for SPG is linear in $\max(|K|, \sum_{g\in\mathcal{G}}|g|)$ or $\max(|K|, |E|)$ while traditional approaches based on IPM scape at least cubically to the size of outputs K.

Table 3.4: Comparison of Per-iteration Time Complexity for Multi-task Regression

	Overlapping Group Lasso	Graph-guided Fused Lasso
SPG	$O(JK\min(J,N) + J\sum_{g\in\mathcal{G}}\|g\|)$	$O(JK\min(J,N) + J\|E\|)$
IPM	$O\left(J^2(K+\|\mathcal{G}\|)^2(KN + J(\sum_{g\in\mathcal{G}}\|g\|))\right)$	$O\left(J^3(K+\|E\|)^3\right)$

3.5 EXPERIMENT

In this section, we evaluate the scalability and efficiency of the smoothing proximal gradient method (SPG) on a number of structured sparse regression problems via simulation, and apply SPG to an overlapping group lasso problem on a real genetic datable.

On an overlapping group lasso problem, we compare the SPG with FOBOS [39] and IPM for SOCP.[2] On a multi-task graph-guided fused lasso problem, we compare the running time of SPG with that of the FOBOS [39] and IPM for QP.[3] Note that for FOBOS, since the proximal operator associated with $\Omega(\beta)$ cannot be solved exactly, we set the "loss function" to $l(\beta) = g(\beta) + \Omega(\beta)$ and the penalty to $\lambda\|\beta\|_1$. According to Duchi and Singer [39], for the non-smooth loss $l(\beta)$, FOBOS achieves $O\left(\frac{1}{\epsilon^2}\right)$ convergence rate, which is slower than our method.

All experiments are performed on a standard PC with 4GB RAM and the software is written in MATLAB. The main difficulty in comparisons is a fair stopping criterion. Unlike IPM, SPG and FOBOS do not generate a dual solution, and therefore, it is not possible to compute a primal-dual gap, which is the traditional stopping criterion for IPM. Here, we adopt a widely used approach for comparing different methods in optimization literature. Since it is well known that IPM usually gives more accurate (i.e., lower) objective, we set the objective obtained from IPM as the optimal objective value and stop the first-order methods when the objective is below 1.001 times the optimal objective. For large datasets for which IPM cannot be applied, we stop the first-order methods when the relative change in the objective is below 10^{-6}. In addition, maximum iterations is set to 20,000.

Since our main focus is on the optimization algorithm, for the purpose of simplicity, we assume that each group in the overlapping group lasso problem receives the same amount of regularization and hence set the weights w_g for all group to be 1. In principle, more sophisticated prior knowledge of the importance for each group can be naturally incorporated into w_g. In addition, we notice that each variable j with the regularization $\lambda|\beta_j|$ in $\lambda\|\beta\|_1$ can be viewed as a singleton group. To ease the tuning of parameters, we again assume that each group (including the singleton group) receives the same

2 We use the state-of-the-art MATLAB package SDPT3 [140] for SOCP.

3 We use the commercial package MOSEK (http://www.mosek.com/) for QP. Graph-guided fused lasso can also be solved by SOCP but it is less efficient than QP.

amount of regularization and hence constrain the regularization parameters $\lambda = \gamma$.

The smoothing parameter μ is set to $\frac{\epsilon}{2D}$ according to Theorem 3.2, where D is determined by the problem size. It is natural that for large-scale problems with large D, a larger ϵ can be adopted without affecting the recovery quality significantly. Therefore, instead of setting ϵ, we directly set $\mu = 10^{-4}$, which provided us with reasonably good approximation accuracies for different scales of problems based on our experience for a range of μ in simulations. As for FOBOS, we set the stepsize rate to $\frac{c}{\sqrt{t}}$ as suggested in Duchi and Singer [39], where c is carefully tuned to be $\frac{0.1}{\sqrt{NJ}}$ for univariate regression and $\frac{0.1}{\sqrt{NJK}}$ for multi-task regression.

3.5.1 *Simulation Study I: Overlapping Group Lasso*

We simulate data for a univariate linear regression model with the overlapping group structure on the inputs as described below. Assuming that the inputs are ordered, we define a sequence of groups of 100 adjacent inputs with an overlap of 10 variables between two successive groups so that

$$\mathcal{G} = \{\{1, \ldots, 100\}, \{91, \ldots, 190\}, \ldots, \{J - 99, \ldots, J\}\},$$

with $J = 90|\mathcal{G}| + 10$. We set $\beta_j = (-1)^j \exp(-(j-1)/100)$ for $1 \leqslant j \leqslant J$. We sample each element of $\mathbf{X}$ from *i.i.d.* Gaussian distribution, and generate the output data from $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim N(0, I_{N\times N})$.

To demonstrate the efficiency and scalability of SPG, we vary J, N and γ and report the total CPU time in seconds and the objective value in Table 3.5. The regularization parameter γ is set to either $|\mathcal{G}|/5$ or $|\mathcal{G}|/20$. As we can see from Table 3.5, firstly, both SPG and FOBOS are more efficient and scalable by orders of magnitude than IPM for SOCP. For larger J and N, we are unable to collect the results for SOCP. Secondly, SPG is more efficient than FOBOS for almost all different scales of the problems.[4] Thirdly, for SPG, a smaller γ leads to faster convergence. This result is consistent with Theorem 3.2, which shows that the number of iterations is linear in γ through the term $\|C\|$. Moreover, we notice that a larger N does not increase the computational time for SPG. This is also consistent with the time complexity analysis, which shows that for linear regression, the per-iteration time complexity is independent of N.

However, we find that the solutions from IPM are more accurate and in fact, it is hard for first-order approaches to achieve the same precision as IPM. Assuming that we require $\epsilon = 10^{-6}$ for the accuracy of the solution, it takes IPM about $O(\log(\frac{1}{\epsilon})) \approx 14$ iterations to converge while $O(\frac{1}{\epsilon}) = 10^6$ iterations for SPG. This is the drawback for any first-order method. However, in many real applications, we do

4 In some entries in Table 3.5, the Obj. from FOBOS is much larger than other methods. This is because that FOBOS has reached the maximum number of iterations before convergence. Instead, for our simulations, SPG generally converges in hundreds or at most, a few thousands, iterations and never pre-terminates.

Table 3.5: Comparisons of different optimization methods on the overlapping group lasso. SPG is more efficient than the first-order method FOBOS in terms of CPU time. As compared to IPM for solving SOCP, although SPG has slightly worse objective values, it is much more scalable and efficient.

$\|\mathcal{G}\| = 10$		N=1,000		N=5,000		N=10,000	
($J = 910$)		CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)	Obj.
$\gamma = 2$	SOCP	103.71	266.683	493.08	917.132	3777.46	1765.518
	FOBOS	27.12	266.948	1.71	918.019	1.48	1765.613
	SPG	0.87	266.947	0.71	917.463	1.28	1765.692
$\gamma = 0.5$	SOCP	106.02	83.304	510.56	745.102	3585.77	1596.418
	FOBOS	32.44	82.992	4.98	745.788	4.65	1597.531
	SPG	0.42	83.386	0.41	745.104	0.69	1596.452
$\|\mathcal{G}\| = 50$		N=1,000		N=5,000		N=10,000	
($J = 4510$)		CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)	Obj.
$\gamma = 10$	SOCP	4144.20	1089.014	-	-	-	-
	FOBOS	476.91	1191.047	394.75	1533.314	79.82	2263.494
	SPG	56.35	1089.052	77.61	1533.318	78.90	2263.601
$\gamma = 2.5$	SOCP	3746.43	277.911	-	-	-	-
	FOBOS	478.62	286.327	867.94	559.251	183.72	1266.728
	SPG	33.09	277.942	30.13	504.337	26.74	1266.723
$\|\mathcal{G}\| = 100$		N=1,000		N=5,000		N=10,000	
($J = 9010$)		CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)	Obj.
$\gamma = 20$	FOBOS	1336.72	2090.808	2261.36	3132.132	1091.20	3278.204
	SPG	234.71	2090.792	225.28	2692.981	368.52	3278.219
$\gamma = 5$	FOBOS	1689.69	564.209	2287.11	1302.552	3342.61	1185.661
	SPG	169.61	541.611	192.92	736.559	176.72	1114.933

not require the objective to be extremely accurate (e.g., $\epsilon = 10^{-3}$ is sufficiently accurate in general) and first-order methods are more suitable. More importantly, first-order methods can be applied to large-scale high-dimensional problems while IPM can only be applied to small or moderate scale problems due to the expensive computation necessary for solving the Newton linear system.

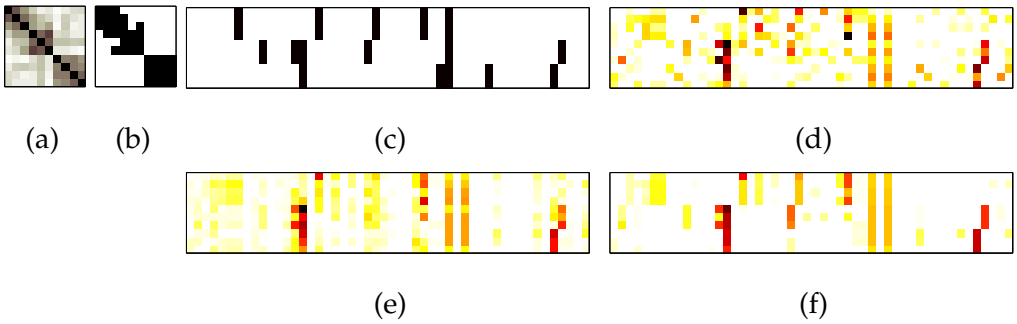


Figure 3.2: Regression coefficients estimated by different methods based on a single simulated datable. $b = 0.8$ and threshold $\rho = 0.3$ for the output correlation graph are used. Red pixels indicate large values. (a) The correlation coefficient matrix of phenotypes, (b) the edges of the phenotype correlation graph obtained at threshold 0.3 are shown as black pixels, (c) the true regression coefficients used in simulation. Absolute values of the estimated regression coefficients are shown for (d) lasso, (e) ℓ_1/ℓ_2 regularized multi-task regression, (f) Graph-guided fused lasso. Rows correspond to outputs and columns to inputs.

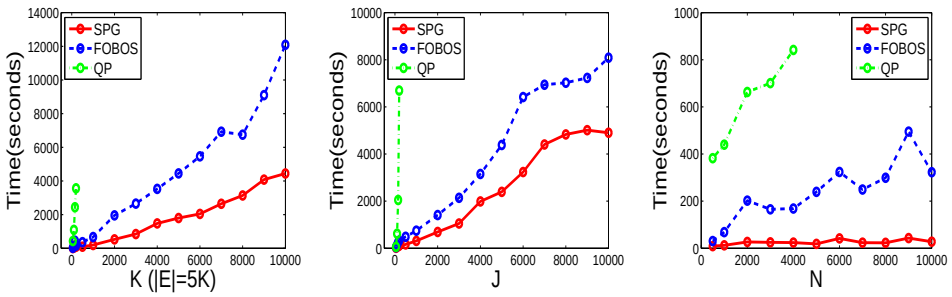


Figure 3.3: Comparisons of SPG, FOBOS and QP. (a) Vary K from 50 to 10,000, fixing $N = 500, J = 100$; (b) Vary J from 50 to 10,000, fixing $N = 1000, K = 50$; and (c) Vary N from 500 to 10000, fixing $J = 100, K = 50$.

3.5.2 *Simulation Study II: Multi-task Graph-guided Fused Lasso*

We simulate data using the following scenario analogous to the problem of genetic association mapping, where we are interested in identifying a small number of genetic variations (inputs) that influence the phenotypes (outputs). We use $K = 10$, $J = 30$ and $N = 100$. To simulate the input data, we use the genotypes of the 60 individuals from the parents of the HapMap CEU panel [131], and generate genotypes for additional 40 individuals by randomly mating the original 60 individuals. We generate the regression coefficients β_k's such that the outputs y_k's are correlated with a block-like structure in the correlation matrix. We first choose input-output pairs with non-zero regression coefficients as we describe below. We assume three groups of correlated output variables of sizes 3, 3, and 4. We randomly select inputs that are relevant jointly among the outputs within each

group, and select additional inputs relevant across multiple groups to model the situation of a higher-level correlation structure across two subgraphs as in Figure 3.2(a). Given the sparsity pattern of $\mathbf{B}$, we set all non-zero β_{ij} to a constant $b = 0.8$ to construct the true coefficient matrix $\mathbf{B}$. Then, we simulate output data based on the linear regression model with noise distributed as standard Gaussian, using the simulated genotypes as inputs. We threshold the output correlation matrix in Figure 3.2(a) at $\rho = 0.3$ to obtain the graph in Figure 3.2(b), and use this graph as prior structural information for graph-guided fused lasso. As an illustrative example, the estimated regression coefficients from different regression models for recovering the association patterns are shown in Figures 3.2(d)–(f). While the results of lasso and ℓ_1/ℓ_2-regularized multi-task regression with $\Omega(\mathbf{B}) = \sum_{j=1}^{J} \|\beta_{j,:}\|_2$ [112] in Figures 3.2 (d) and (e) contain many false positives, the results from graph-guided fused lasso in Figure 3.2(f) show fewer false positives and reveal clear block structures. Thus, the graph-guided fused lasso proves to be a superior regression model for recovering the true regression pattern that involves structured sparsity in the input/output relationships.

To compare SPG with FOBOS and IPM for QP in solving such a structured sparse regression problem, we vary K, J, N, and present the computation time in seconds in Figures 3.3(a)-(c), respectively. We select the regularization parameter γ using a separate validation datable, and report the CPU time for graph-guided fused lasso with the selected γ. The input/output data and true regression coefficient matrix $\mathbf{B}$ are generated in the way similar as above. More precisely, we assume that each group of correlated output variables is of size 10. For each group of the outputs, We randomly select 10% of the input variables as relevant. In addition, we randomly select 5% of the input variables as relevant to every two consecutive groups of outputs and 1% of the input variables as relevant to every three consecutive groups. We set the ρ for each datable so that the number of edges is 5 times the number of the nodes (i.e. $|E| = 5K$). Figure 3.3 shows that SPG is substantially more efficient and can scale up to very high-dimensional and large-scale datasets. Moreover, we notice that the increase of N almost does not affect the computation time of SPG, which is consistent with the complexity analysis in Section 3.2.5.

3.5.3 *Real Data Analysis: Pathway Analysis of Breast Cancer Data*

In this section, we apply the SPG to an overlapping group lasso problem with a logistic loss on a real-world datable collected from breast cancer tumors [142, 65].The main goal is to demonstrate the importance of employing structured sparsity-inducing penalties for performance enhancement in real life high-dimensional regression problems, thereby further exhibit and justify the needs of efficient solvers such as SPG for such problems.

The data are given as gene expression measurements for 8,141 genes in 295 breast-cancer tumors (78 metastatic and 217 non-metastatic).

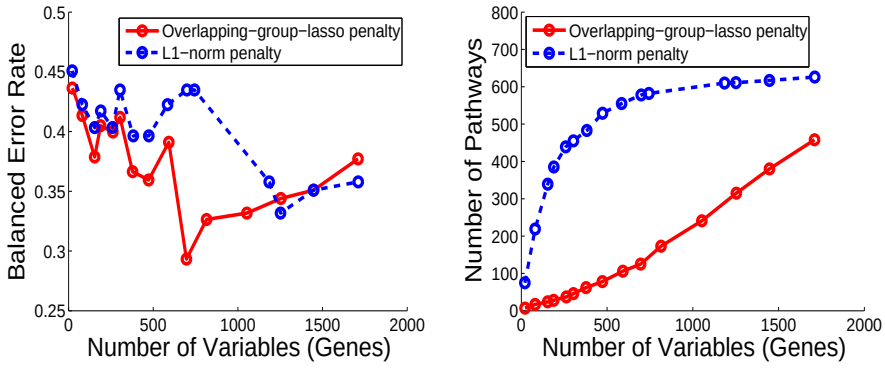


Figure 3.4: Results from the analysis of breast cancer datable. (a) Balanced error rate for varying the number of selected genes, and (b) the number of pathways for varying the number of selected genes.

A lot of research efforts in biology have been devoted to identifying biological pathways that consist of a group of genes participating in a particular biological process to perform a certain functionality in the cell. Thus, a powerful way of discovering genes involved in a tumor growth is to consider groups of interacting genes in each pathway rather than individual genes independently [100]. The overlapping-group-lasso penalty provides us with a natural way to incorporate these known pathway information into the biological analysis, where each group consists of the genes in each pathway. This approach can allow us to find pathway-level gene groups of significance that can distinguish the two tumor types. In our analysis of the breast cancer data, we cluster the genes using the canonical pathways from the Molecular Signatures Database [126], and construct the overlapping-group-lasso penalty using the pathway-based clusters as groups. Many of the groups overlap because genes can participate in multiple pathways. Overall, we obtain 637 pathways over 3,510 genes, with each pathway containing 23.47 genes on average and each gene appearing in four pathways on average. Instead of analyzing all 8,141 genes, we focus on these 3,510 genes which belong to certain pathways. We set up the optimization problem of minimizing the logistic loss with the overlapping-group-lasso penalty to classify the tumor types based on the gene expression levels, and solve it with SPG.

Since the number of positive and negative samples are imbalanced, we adopt the balanced error rate defined as the average error rate of the two classes.[5] We split the data into the training and testing sets with the ratio of 2:1, and vary the $\lambda = \gamma$ from large to small to obtain the full regularization path.

In Figure 3.4, we compare the results from fitting the logistic regression with the overlapping-group-lasso penalty with a baseline model with only the ℓ_1-norm penalty. Figure 3.4(a) shows the balanced error rates for the different numbers of selected genes along

5 See http://www.modelselect.inf.ethz.ch/evaluation.php for more details.

the regularization path. As we can see, the balanced error rate for the model with the overlapping-group-lasso penalty is lower than the one with the ℓ_1-norm, especially when the number of selected genes is between 500 to 1000. The model with the overlapping-group-lasso penalty achieves the best error rate of 29.23% when 696 genes are selected, and these 696 genes belong to 125 different pathways. In Figure 3.4(b), for the different numbers of selected genes, we show the number of pathways to which the selected genes belong. From Figure 3.4(b), we see that when the group structure information is incorporated, fewer pathways are selected. This indicates that regression with the overlapping-group-lasso penalty selects the genes at the pathway level as a functionally coherent groups, leading to an easy interpretation for functional analysis. On the other hand, the genes selected via the ℓ_1-norm penalty are scattered across many pathways as genes are considered independently for selection. The total computational time for computing the whole regularization path with 20 different values for the regularization parameters is 331 seconds for the overlapping group lasso.

We perform functional enrichment analysis on the selected pathways, using the functional annotation tool [62], and verify that the selected pathways are significant in their relevance to the breast-cancer tumor types. For example, in a highly sparse model obtained with the group-lasso penalty at the very left end of Figure 3.4(b), the selected gene markers belong to only seven pathways, and many of these pathways appear to be reasonable candidates for an involvement in breast cancer. For instance, all proteins in one of the selected pathways are involved in the activity of *proteases* whose function is to degrade unnecessary or damaged proteins through a chemical reaction that breaks peptide bonds. One of the most important malignant properties of cancer involves the uncontrolled growth of a group of cells, and *protease inhibitors*, which degrade misfolded proteins, have been extensively studied in the treatment of cancer. Another interesting pathway selected by overlapping group lasso is known for its involvement in *nicotinate and nicotinamide metabolism*. This pathway has been confirmed as a marker for breast cancer in previous studies [100]. In particular, the gene *ENPP1* (ectonucleotide pyrophosphatase/phosphodiesterase 1) in this pathway has been found to be overly expressed in breast tumors [1]. Other selected pathways include the one related to ribosomes and another related to DNA polymerase, which are critical in the process of generating proteins from DNA and relevant to the property of uncontrolled growth in cancer cells.

We also examine the number of selected pathways that gives the lowest error rate in Figure 3.4. At the error rate of 29.23%, 125 pathways (696 genes) are selected. It is interesting to notice that among these 125 pathways, one is closely related to *apoptosis*, which is the process of programmed cell death that occurs in multicellular organisms and is widely known to be involved in un-controlled tumor growth in cancer. Another pathway involves the genes *BRCA1*,

BRCA2, and *ATR*, which have all been associated with cancer susceptibility.

For comparison, we examine the genes selected with the ℓ_1-norm penalty that does not consider the pathway information. In this case, we do not find any meaningful functional enrichment signals that are relevant to breast cancer. For example, among the 582 pathways that involve 687 genes at 37.55% error rate, we find two large pathways with functional enrichments, namely *response to organic substance* (83 genes with p-value 3.3E-13) and the process of *oxidation reduction* (73 genes with p-value 1.7E-11). However, both are quite large groups and matched to relatively high-level biological processes that do not provide much insight on cancer-specific pathways.

3.6 APPENDIX: TECHNICAL PROOFS

Proof of Theorem 3.1

Recall that $d(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\alpha}\|^2$ with the $\text{dom}(\boldsymbol{\alpha}) = Q$. According to Definition 3.1, the conjugate of $d(\cdot)$ at $\frac{C\boldsymbol{\beta}}{\mu}$ is: $d^*\left(\frac{C\boldsymbol{\beta}}{\mu}\right) = \sup_{\boldsymbol{\alpha}\in Q}\left(\boldsymbol{\alpha}^T\frac{C\boldsymbol{\beta}}{\mu} - d(\boldsymbol{\alpha})\right)$, and hence

$$f_\mu(\boldsymbol{\beta}) \equiv \arg\max_{\boldsymbol{\alpha}\in Q}\left(\boldsymbol{\alpha}^T C\boldsymbol{\beta} - \mu d(\boldsymbol{\alpha})\right) = \mu d^*\left(\frac{C\boldsymbol{\beta}}{\mu}\right).$$

According to Theorem 26.3 in Rockafellar [119] "a closed proper convex function is essentially strictly convex if and only if its conjugate is essentially smooth", since $d(\boldsymbol{\alpha})$ is a closely proper strictly convex function, its conjugate is smooth. Therefore, $f_\mu(\boldsymbol{\beta})$ is a smooth function.

Now we apply Danskin's Theorem (Prop B.25 in Bertsekas [10]) to derive $\nabla f_\mu(\boldsymbol{\beta})$. Let $\phi(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \boldsymbol{\alpha}^T C\boldsymbol{\beta} - \mu d(\boldsymbol{\alpha})$. Since $d(\cdot)$ is a strongly convex function, $\arg\max_{\boldsymbol{\alpha}\in Q}\phi(\boldsymbol{\alpha}, \boldsymbol{\beta})$ has a unique optimal solution and we denote it as $\boldsymbol{\alpha}^*$. According to Danskin's Theorem:

$$\nabla f_\mu(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}}\phi(\boldsymbol{\alpha}^*, \boldsymbol{\beta}) = C^T\boldsymbol{\alpha}^*. \tag{3.22}$$

As for the proof of Lipschitz constant of $f_\mu(\boldsymbol{\beta})$, readers may refer to Nesterov [109].

Proof of Proposition 3.1

$$\begin{aligned}
\boldsymbol{\alpha}^* &= \arg\max_{\boldsymbol{\alpha}\in Q}\left(\boldsymbol{\alpha}^T C\boldsymbol{\beta} - \frac{\mu}{2}\|\boldsymbol{\alpha}\|_2^2\right) \\
&= \arg\max_{\boldsymbol{\alpha}\in Q}\sum_{g\in\mathcal{G}}\left(\gamma w_g \boldsymbol{\alpha}_g^T\boldsymbol{\beta}_g - \frac{\mu}{2}\|\boldsymbol{\alpha}_g\|_2^2\right) \\
&= \arg\min_{\boldsymbol{\alpha}\in Q}\sum_{g\in\mathcal{G}}\|\boldsymbol{\alpha}_g - \frac{\gamma w_g\boldsymbol{\beta}_g}{\mu}\|_2^2
\end{aligned} \tag{3.23}$$

Therefore, (3.23) can be decomposed into $|\mathcal{G}|$ independent problems: each one is the Euclidean projection onto the ℓ_2-ball:

$$\boldsymbol{\alpha}_g^* = \underset{\boldsymbol{\alpha}_g:\|\boldsymbol{\alpha}_g\|_2\leqslant 1}{\arg\min} \|\boldsymbol{\alpha}_g - \frac{\gamma w_g \boldsymbol{\beta}_g}{\mu}\|_2^2,$$

and $\boldsymbol{\alpha}^* = \left[(\boldsymbol{\alpha}_{g_1}^*)^T, \ldots, (\boldsymbol{\alpha}_{g_{|\mathcal{G}|}}^*)^T\right]^T$. According to the property of ℓ_2-ball, it can be easily shown that:

$$\boldsymbol{\alpha}_g^* = S_2(\frac{\gamma w_g \boldsymbol{\beta}_g}{\mu}),$$

where $S_2(\mathbf{u}) = \begin{cases} \frac{\mathbf{u}}{\|\mathbf{u}\|_2} & \|\mathbf{u}\|_2 > 1, \\ \mathbf{u} & \|\mathbf{u}\|_2 \leqslant 1. \end{cases}$

As for $\|C\|$,

$$\|C\mathbf{v}\|_2 = \gamma\sqrt{\sum_{g\in\mathcal{G}}\sum_{j\in g}(w_g)^2 v_j^2} = \lambda\sqrt{\sum_{j=1}^{J}(\sum_{g\in\mathcal{G}\ \text{s.t.}\ j\in g}(w_g)^2)v_j^2},$$

the maximum value of $\|C\mathbf{v}\|_2$, given $\|\mathbf{v}\|_2 \leqslant 1$, can be achieved by setting $v_{\hat{j}}$ for j corresponding to the largest summation $\sum_{g\in\mathcal{G}\ \text{s.t.}\ j\in g}(w_g)^2$ to one, and setting other v_j's to zeros. Hence, we have

$$\|C\mathbf{v}\|_2 = \gamma \max_{j\in\{1,\ldots,J\}} \sqrt{\sum_{g\in\mathcal{G}\ \text{s.t.}\ j\in g}(w_g)^2}.$$

Proof of Proposition 3.2

Similar to the proof technique of Proposition 1, we reformulate the problem of solving $\boldsymbol{\alpha}^*$ as a Euclidean projection:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}\in\mathcal{Q}}{\arg\max}\left(\boldsymbol{\alpha}^T C\boldsymbol{\beta} - \frac{\mu}{2}\|\boldsymbol{\alpha}\|_2^2\right) = \underset{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_\infty\leqslant 1}{\arg\min} \|\boldsymbol{\alpha} - \frac{C\boldsymbol{\beta}}{\mu}\|_2^2,$$

and the optimal solution $\boldsymbol{\alpha}^*$ can be obtained by projecting $\frac{C\boldsymbol{\beta}}{\mu}$ onto the ℓ_∞-ball.

According to the construction of matrix C, we have for any vector $\mathbf{v}$:

$$\|C\mathbf{v}\|_2^2 = \gamma^2 \sum_{e=(m,l)\in E}(\tau(r_{ml}))^2(v_m - \text{sign}(r_{ml})v_l)^2 \tag{3.24}$$

By the simple fact that $(a \pm b)^2 \leqslant 2a^2 + 2b^2$ and the inequality holds as equality if and only if $a = \pm b$, for each edge $e = (m, l) \in E$, the value $(v_m - \text{sign}(r_{ml})v_l)^2$ is upper bounded by $2v_m^2 + 2v_l^2$. Hence, when $\|\mathbf{v}\|_2 = 1$, the right-hand side of (3.24) can be further bounded by:

$$\begin{aligned} \|C\mathbf{v}\|_2^2 &\leqslant \gamma^2 \textstyle\sum_{e=(m,l)\in E} 2(\tau(r_{ml}))^2(v_m^2 + v_l^2) \\ &= \gamma^2 \textstyle\sum_{j\in V}(\sum_{e\ \text{incident on}\ k} 2(\tau(r_e))^2)v_j^2 \\ &= \gamma^2 \textstyle\sum_{j\in V} 2d_j v_j^2 \\ &\leqslant 2\gamma^2 \max_{j\in V} d_j, \end{aligned} \tag{3.25}$$

where

$$d_j = \sum_{e \in E \text{ s.t. } e \text{ incident on } j} (\tau(r_e))^2.$$

Therefore, we have

$$\|C\| \equiv \max_{\|v\|_2 \leqslant 1} \|Cv\|_2 \leqslant \sqrt{2\gamma^2 \max_{j \in V} d_j}.$$

Note that this upper bound is tight because the first inequality in (3.25) is tight.

Proof of Theorem 3.2

Based on the result from Beck and Teboulle [6], we have the following lemma:

Lemma 3.1. *For the function* $\widetilde{f}(\beta) = h(\beta) + \lambda\|\beta\|_1$, *where* $h(\beta)$ *is an arbitrary convex smooth function and its gradient* $\nabla h(\beta)$ *is Lipschitz continuous with the Lipschitz constant* L. *We apply Algorithm 3.1 to minimize* $\widetilde{f}(\beta)$ *and let* β^t *be the approximate solution at the* t*-th iteration. For any* β, *we have the following bound:*

$$\widetilde{f}(\beta^t) - \widetilde{f}(\beta) \leqslant \frac{2L\|\beta - \beta^0\|_2^2}{t^2}. \tag{3.26}$$

In order to use the bound in (3.26), we use the similar proof scheme as in Lan [79] and decompose $f(\beta^t) - f(\beta^*)$ into three terms:

$$f(\beta^t) - f(\beta^*) = \left(f(\beta^t) - \widetilde{f}(\beta^t)\right) + \left(\widetilde{f}(\beta^t) - \widetilde{f}(\beta^*)\right) + \left(\widetilde{f}(\beta^*) - f(\beta^*)\right) \tag{3.27}$$

According to the definition of $\widetilde{f}$, we know that for any β

$$\widetilde{f}(\beta) \leqslant f(\beta) \leqslant \widetilde{f}(\beta) + \mu D,$$

where $D \equiv \max_{\alpha \in Q} d(\alpha)$. Therefore, the first term in (3.27), $f(\beta^t) - \widetilde{f}(\beta^t)$, is upper-bounded by μD, and the last term in (3.27) is less than or equal to 0 (i.e., $\widetilde{f}(\beta^*) - f(\beta^*) \leqslant 0$). Combining (3.26) with these two simple bounds, we have:

$$\begin{aligned} f(\beta^t) - f(\beta^*) &\leqslant \mu D + \frac{2L\|\beta^* - \beta^0\|_2^2}{t^2} \\ &\leqslant \mu D + \frac{2\|\beta^* - \beta^0\|_2^2}{t^2}\left(\lambda_{\max}(X^T X) + \frac{\|C\|^2}{\mu}\right). \end{aligned} \tag{3.28}$$

By setting $\mu = \frac{\epsilon}{2D}$ and plugging this into the right-hand side of (3.28), we obtain

$$f(\beta^t) - f(\beta^*) \leqslant \frac{\epsilon}{2} + \frac{2\|\beta^*\|_2^2}{t^2}\left(\lambda_{\max}\left(X^T X\right) + \frac{2D\|C\|^2}{\epsilon}\right). \tag{3.29}$$

If we require the right-hand side of (3.29) to be equal to ϵ and solve it for t, we obtain the bound of t in (3.19).

4

STRUCTURED SPARSE CANONICAL CORRELATION ANALYSIS

In the previous chapter, we propose SPG to solve linear regression with structured-sparsity-inducing penalty. In fact, the idea of smoothing the structured-sparsity-inducing penalty has wide applications in various structured statistical models. In this chapter, we propose to solve structured sparse canonical correlation analysis (structured sparse CCA) with the application to an important genome-wide association study problem, eQTL mapping. We extend the sparse CCA so that it could exploit either the pre-given or unknown group structure via the structured-sparsity-inducing penalty. Since each subproblem of *structured sparse CCA* is strongly convex in nature, by combining Nesterov's excessive gap method with the smoothed structured-sparsity-inducing penalty, we can achieve a fast rate of convergence of $O(1/\sqrt{\epsilon})$ as compared to $O(1/\epsilon)$ of SPG. We demonstrate the effectiveness of our models on both simulated and genetic datasets.

4.1 INTRODUCTION AND MOTIVATION

A fundamental problem in genome-wide association study (GWA study) is to understand associations between genomic and phenotypic variations, which is referred as *expression quantitative trait loci (eQTLs) mapping*. The eQTL mapping discovers genetic associations between genotype data of single nucleotide polymorphisms (SNPs) and phenotype data of gene expression levels to provide insights into gene regulation, and potentially, controlling factors of a disease. More formally, we have two datasets $\mathbf{X}$ (e.g. SNPs data) and $\mathbf{Y}$ (e.g. gene expression data) of dimensions $n \times d$ and $n \times p$ collected on the same set of n observations. Both p and d could be much larger than n in an eQTL study. Our goal is to investigate the relationship between $\mathbf{X}$ and $\mathbf{Y}$.

A popular approach for eQTL mapping is to formulate the problem into a sparse multivariate regression [87, 73]. These methods treat $\mathbf{X}$ as input, $\mathbf{Y}$ as output and try to identify a small subset of input variables that simultaneously related to all the responses. Despite the promising aspects of these models, such multivariate-regression approaches are not symmetric in that the regression coefficients are only put on the $\mathbf{X}$ side. There is no clear reason why one wants to regress $\mathbf{Y}$ on $\mathbf{X}$ but instead of $\mathbf{X}$ on $\mathbf{Y}$ for an association study. Also, in the study of eQTL mapping, it is difficult to find a small subset of SNPs which can explain the expression levels for all the involved genes.

In contrast to sparse multivariate regression approach, sparse canonical correlation analysis (sparse CCA) [146, 145] provides a more "symmetric" solution in which it finds two *sparse* canonical vectors $\mathbf{u}$ and $\mathbf{v}$ to maximize the correlation between $\mathbf{Xu}$ and $\mathbf{Yv}$. In eQTL

mapping, sparse CCA automatically selects a subset of SNP genotypes and a subset of gene expression levels and maximizes their linear combination correlations. Although sparse CCA has been successfully applied to some genomic datasets (e.g. CGH data [145]), it has not well been studied for eQTL mapping. In a study of eQTL mapping, it is a great interest for biologists to seek for a subset of SNP genotypes and a subset of gene expression levels that are closely related. To address this problem, we apply sparse CCA to eQTL mapping; and show that by incorporating the proper structural information, sparse CCA can be a useful tool for GWA study.

It is well known that when dealing with high-dimensional data, prior structural knowledge is crucial for the analysis, which facilities model's interpretability. For example, a biological pathway is a *group* of genes that participate in a particular biological process to perform certain functionality in a cell. To find the controlling factors related to a disease, it is more meaningful to study the genes by considering their pathways. However, the existing sparse CCA models use the ℓ_1-regularization and do not incorporate the rich structural information among variables (e.g. genetic pathways). In this chapter, we propose a *structured sparse CCA framework* that can naturally incorporate the group structural information. In particular, we consider two scenarios: (1) when the group structure is pre-given, we propose to incorporate such prior knowledge using the *overlapping-group-lasso penalty* [66]. Compare to the standard group lasso [150], we allow arbitrary overlaps among groups which reflects the fact that a gene may belong to multiple pathways. We refer to this model as the *group-structured sparse CCA*; (2) if such structural information is not available as a *priori*, we propose the *group pursuit sparse CCA* using a *group pursuit penalty* based on a fusion penalty, which simultaneously conducts variable selection and structure estimation.

We formulate the structured sparse CCA into a biconvex problem and adopt an alternative strategy to conduct the optimization. However, unlike in [146] where the simple ℓ_1-norm penalty is used, our formulation involves the non-separable *overlapping-group-lasso* penalty and *group pursuit* penalty. Such non-separability poses great challenge on optimization techniques.

Although many methods have been proposed [39, 66, 67, 101, 94, 29] for solving the related sparse learning problems, they either cannot be applied to our problem or achieve a sub-optimal convergence rate. In this work, we propose to apply the excessive gap method [107] to solve the sparse CCA with a wide class of *structured-sparsity-inducing penalties*. Since it is a first-order method, the per-iteration time complexity is very low (e.g. linear in the sum of group sizes) and the method can scale up to millions of variables. It is a primal-dual approach which diminishes the primal-dual gap over iterations. For each subproblem in the alternating optimization procedure, the algorithm achieves the convergence rate of $O(1/t^2)$ (i.e. the duality gap is less than $O(1/t^2)$) in t iterations.

4.2 GROUP STRUCTURED SPARSE CCA

Here, we extend the sparse CCA introduced in Section 2.3 to more general forms of P to incorporate the group structural information. In eQTL mapping, the structural knowledge among genes on $\mathbf{Y}$ side is often of more interest. For the ease of illustration, we assume that $P_1(\mathbf{u}) = \|\mathbf{u}\|_1$ and mainly focus on $P_2(\mathbf{v})$, which incorporates the structural information. Since (2.13) is biconvex in $\mathbf{u}$ and $\mathbf{v}$ individually, a natural optimization strategy is the alternating approach: fix $\mathbf{u}$ and optimize over $\mathbf{v}$; then fix $\mathbf{v}$ and optimize over $\mathbf{u}$; and iterate over these two steps. In our a setting, the optimization with respect to $\mathbf{u}$ with $P_1(\mathbf{u}) = \|\mathbf{u}\|_1$ is relatively simple and the closed-form solution has been obtained in [146]. However, due to the complicated structure of $P_2(\mathbf{v})$, the optimization with respect to $\mathbf{v}$ cannot be easily solved and we will address this challenge in the following.

In particular, we study the problem in which the group structural information among variables in $\mathbf{Y}$ is pre-given from the domain knowledge; and our goal is to identify a small subset of relevant groups under the sparse CCA framework. More formally, let us assume that the set of groups of variables in $\mathbf{Y}$: $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\}$ is defined as a subset of the power set of $\{1, \ldots, p\}$, and is available as prior knowledge. Note that the members (groups) of $\mathcal{G}$ are allowed to overlap. Inspired by the *group-lasso penalty* [150] and the *elastic-net penalty* [162], we define our penalty $P_2(\mathbf{v})$ as follows:

$$P_2(\mathbf{v}) = \sum_{g \in \mathcal{G}} w_g \|\mathbf{v}_g\|_2 + \frac{c}{2}\mathbf{v}^T\mathbf{v}, \tag{4.1}$$

where $\mathbf{v}_g \in \mathbb{R}^{|g|}$ is the subvector of $\mathbf{v}$ in group g, w_g is the predefined weight for group g; c is the tuning parameter and $\|\cdot\|_2$ is the vector ℓ_2-norm. The ℓ_1/ℓ_2 mixed-norm penalty in $P_2(\mathbf{v})$ plays the role of group selection. Since some gene expression levels are highly correlated, the ridge penalty $\frac{c}{2}\mathbf{v}^T\mathbf{v}$ addresses the problem of the collinearity, enforcing strongly correlated variables to be in or out of the model together. In addition, according to [162, 103], the ridge penalty is crucial to ensure the stable variable selection when $p \gg n$, which is a typical setting of eQTL mapping.

Rather than solving the constraint form of $P_2(\mathbf{v})$, we solve the regularized problem using the Lagrange form:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{v}} \quad & -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} + \frac{\tau}{2}\mathbf{v}^T\mathbf{v} + \theta \sum_{g \in \mathcal{G}} w_g \|\mathbf{v}_g\|_2 \\ \text{s.t.} \quad & \|\mathbf{u}\|_2 \leqslant 1,\ \|\mathbf{v}\|_2 \leqslant 1,\ \|\mathbf{u}\|_1 \leqslant c_1, \end{aligned} \tag{4.2}$$

where there exists a one to one correspondence between (θ, τ) and (c, c_2) (c_2 is the upper bound of $P_2(\mathbf{v})$). We refer to this model (4.2) as the *group-structured sparse CCA*.

4.2.1 *Optimization Algorithm*

The main difficulty in solving (4.2) arises from optimizing with respect to $\mathbf{v}$. Let the domain of $\mathbf{v}$ be denoted as $Q_1 = \{\mathbf{v} \,|\, \|\mathbf{v}\|_2 \leqslant 1\}$,

$\boldsymbol{\beta} = \frac{1}{\tau}\mathbf{Y}^T\mathbf{X}\mathbf{u}$ and $\gamma = \frac{\theta}{\tau}$, the optimization of (4.2) with respect to $\mathbf{v}$ can be written as:

$$\min_{\mathbf{v}\in Q_1} f(\mathbf{v}) \equiv l(\mathbf{v}) + P(\mathbf{v}), \tag{4.3}$$

where $l(\mathbf{v}) = \frac{1}{2}\|\mathbf{v} - \boldsymbol{\beta}\|_2^2$ is the Euclidean distance loss function and $P(\mathbf{v})$ is the overlapping-group-lasso penalty: $P(\mathbf{v}) = \gamma \sum_{g\in\mathcal{G}} w_g \|\mathbf{v}_g\|_2$. The optimization problem in (4.3) is so-called *proximal mapping* (or *proximal operator*) associated with the function $P(\mathbf{v})$.

As shown in Section 3.2.1, we can rewrite $P(\mathbf{v})$ as:

$$P(\mathbf{v}) = \max_{\boldsymbol{\alpha}\in Q_2} \boldsymbol{\alpha}^T C\mathbf{v} = \delta^*_{Q_2}(C\boldsymbol{\beta}). \tag{4.4}$$

Here $\boldsymbol{\alpha} = \left[\boldsymbol{\alpha}_{g_1}^T, \ldots, \boldsymbol{\alpha}_{g_{|\mathcal{G}|}}^T\right]^T$ is the concatenation of the vectors $\{\boldsymbol{\alpha}_g\}_{g\in\mathcal{G}}$ and δ is the indicator function. We denote the domain of $\boldsymbol{\alpha}$ as:

$$Q_2 \equiv \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha}_g\|_2 \leqslant 1,\ \forall g \in \mathcal{G}\}.$$

The matrix $C \in \mathbb{R}^{(\sum_{g\in\mathcal{G}}|g|)\times p}$ is defined as:

$$C_{(i,g),j} = \begin{cases} \gamma w_g & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

where the rows of C are indexed by all pairs of $(i, g) \in \{(i, g) | i \in g, i \in \{1, \ldots, p\}\}$, the columns are indexed by $j \in \{1, \ldots, p\}$. Using the Nesterov's smoothing technique [109], we construct the smooth approximation as shown in Section 3.2.2:

$$P_\mu(\mathbf{v}) = \max_{\boldsymbol{\alpha}\in Q_2} \left(\boldsymbol{\alpha}^T C\mathbf{v} - \mu d(\boldsymbol{\alpha})\right), \tag{4.6}$$

with its gradient $\nabla P_\mu(\mathbf{v}) = C^T\boldsymbol{\alpha}_\mu(\mathbf{v})$, where $\boldsymbol{\alpha}_\mu(\mathbf{v})$ is the optimal solution to: $\boldsymbol{\alpha}_\mu(\mathbf{v}) = \operatorname{argmax}_{\boldsymbol{\alpha}\in Q_2} \boldsymbol{\alpha}^T C\mathbf{v} - \mu d(\boldsymbol{\alpha})$. Here μ is the positive smoothness parameter and $d(\boldsymbol{\alpha})$ is defined as $\frac{1}{2}\|\boldsymbol{\alpha}\|_2^2$.

We substitute $P(\mathbf{v})$ in the original objective function $f(\mathbf{v})$ with $P_\mu(\mathbf{v})$ and construct the smooth approximation of $f(\mathbf{v})$:

$$f_\mu(\mathbf{v}) \equiv l(\mathbf{v}) + P_\mu(\mathbf{v}). \tag{4.7}$$

The relationship of $f_\mu(\mathbf{v})$ and $f(\mathbf{v})$ can be characterized by the following inequality:

$$f(\mathbf{v}) - \mu D \leqslant f_\mu(\mathbf{v}) \leqslant f(\mathbf{v}), \tag{4.8}$$

where $D = \max_{\boldsymbol{\alpha}\in Q_2} d(\boldsymbol{\alpha}) = |\mathcal{G}|/2$, where $|\mathcal{G}|$ is the number of groups.

The fundamental idea of the excessive gap method [107] is to diminish the duality gap between the objective $f(\mathbf{v})$ and its *Fenchel dual* over iterations. In this section, we derive the Fenchel dual of $f(\mathbf{v})$ and study its property. According to Theorem 3.3.5 in [15], the Fenchel dual problem of $f(\mathbf{v})$, $\phi(\boldsymbol{\alpha})$, takes the following form:

$$\phi(\boldsymbol{\alpha}) = -l^*(-C^T\boldsymbol{\alpha}) - \delta_{Q_2}(\boldsymbol{\alpha}), \tag{4.9}$$

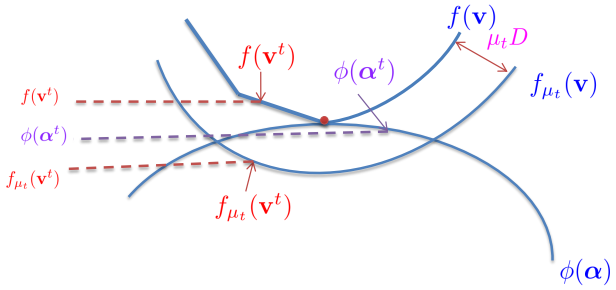


Figure 4.1: Illustration of the excessive gap method

where l^* is the Fenchel Conjugate of l and

$$-l^*(-C^T\boldsymbol{\alpha}) = -\max_{\mathbf{v}\in Q_1} -\mathbf{v}^T C^T\boldsymbol{\alpha} - l(\mathbf{v}) = \min_{\mathbf{v}\in Q_1} \mathbf{v}^T C^T\boldsymbol{\alpha} + \frac{1}{2}\|\mathbf{v}-\boldsymbol{\beta}\|_2^2.$$

As shown in [109], The gradient of $\phi(\boldsymbol{\alpha})$ takes the following form:

$$\nabla\phi(\boldsymbol{\alpha}) = C\mathbf{v}(\boldsymbol{\alpha}), \tag{4.10}$$

where

$$\mathbf{v}(\boldsymbol{\alpha}) = \operatorname{argmin}_{\mathbf{v}\in Q_1} \mathbf{v}^T C^T\boldsymbol{\alpha} + \frac{1}{2}\|\mathbf{v}-\boldsymbol{\beta}\|_2^2. \tag{4.11}$$

Moreover, $\nabla\phi(\boldsymbol{\alpha})$ is Lipschitz continuous with the Lipschitz constant $L(\phi) = \frac{1}{\sigma}\|C\|^2$, where $\sigma = 1$ is the strongly convex parameter for function $l(\mathbf{v})$ and $\|C\|$ is the matrix spectral norm of C: $\|C\| \equiv \max_{\|\mathbf{x}\|_2=1} \|C\mathbf{x}\|_2$.

According to the next proposition, the closed-form equations for $\mathbf{v}(\boldsymbol{\alpha})$ and $\|C\|$ can be written as follows:

Proposition 4.1. $\mathbf{v}(\boldsymbol{\alpha})$ *takes the following form:*

$$\mathbf{v}(\boldsymbol{\alpha}) = S_2\left(\boldsymbol{\beta} - C^T\boldsymbol{\alpha}\right), \tag{4.12}$$

where S_2 is the projection operator (shrinking to the ℓ_2- ball) and $\|C\|$ takes the following form:

$$\|C\| = \gamma \max_{j\in\{1,\ldots,p\}} \sqrt{\sum\nolimits_{g\in\mathcal{G}\ s.t.\ j\in g} (w_g)^2}. \tag{4.13}$$

According to Proposition 4.1, the value of the Lipschitz constant for $\nabla\phi(\boldsymbol{\alpha})$ is:

$$L(\phi) = \|C\|^2 = \gamma^2 \max_{j\in\{1,\ldots,p\}} \sum\nolimits_{g\in\mathcal{G}\ \text{s.t.}\ j\in g} (w_g)^2. \tag{4.14}$$

According to the Fenchel duality theorem [15], we know that under certain mild conditions which hold for our problem: $\min_{\mathbf{v}\in Q_1} f(\mathbf{v}) = \max_{\boldsymbol{\alpha}\in Q_2} \phi(\boldsymbol{\alpha})$, and for any $\mathbf{v} \in Q_1$ and $\boldsymbol{\alpha} \in Q_2$:

$$\phi(\boldsymbol{\alpha}) \leqslant f(\mathbf{v}). \tag{4.15}$$

The key idea of the *excessive gap method* [107] is to simultaneously maintain two sequences $\{\mathbf{v}^t\}$, $\{\boldsymbol{\alpha}^t\}$ and a diminishing smoothness parameter sequence $\{\mu_t\}$ such that:

$$f_{\mu_t}(\mathbf{v}^t) \leqslant \phi(\boldsymbol{\alpha}^t); \quad \mu_{t+1} \leqslant \mu_t; \quad \text{and } \lim_{t\to\infty} \mu_t = 0 \tag{4.16}$$

The geometric illustration of this idea is presented in Figure 4.1. Combining (4.8), (4.15) and (4.16), we have that:

$$f_{\mu_t}(\mathbf{v}^t) \leqslant \phi(\boldsymbol{\alpha}^t) \leqslant f(\mathbf{v}^t) \leqslant f_{\mu_t}(\mathbf{v}^t) + \mu_t D \tag{4.17}$$

From (4.17), when $\mu_t \to 0$, we have $f(\mathbf{v}^t) \approx \phi(\boldsymbol{\alpha}^t)$, which are hence the optimal primal and dual solutions.

Moreover, in the excessive gap method, a gradient mapping operator $\psi : Q_2 \to Q_2$ is defined as follows: for any $\mathbf{z} \in Q_2$,

$$\psi(\mathbf{z}) = \arg\max_{\boldsymbol{\alpha}\in Q_2} \left\{ \langle \nabla\phi(\mathbf{z}), \boldsymbol{\alpha} - \mathbf{z}\rangle - \frac{1}{2} L(\phi)\|\boldsymbol{\alpha} - \mathbf{z}\|_2^2 \right\}. \tag{4.18}$$

The gradient mapping operator ψ for our problem can also be computed in closed-form as presented in the next proposition as shown in the next Proposition.

Proposition 4.2. *For any $\mathbf{z} \in Q_2$, $\psi(\mathbf{z})$ in (4.18) is the concatenation of subvectors $[\psi(\mathbf{z})]_g$ for all groups $g \in \mathcal{G}$. For any group g,*

$$[\psi(\mathbf{z})]_g = S_2\left(\mathbf{z}_g + \frac{[\nabla\phi(\mathbf{z})]_g}{L(\phi)}\right),$$

where $[\nabla\phi(\mathbf{z})]_g = \gamma w_g [\mathbf{v}(\mathbf{z})]_g$ is the g-th subvector of $\nabla\phi(\mathbf{z})$ and the projection operator S_2 is defined in (3.9).

Propositions and 4.1 and 4.2 have shown that, for our problem, all the essential ingredients of the excessive gap framework can be computed in closed-form. We present the excessive gap method to solve proximal mapping associated with overlapping-group-lasso penalty in Algorithm 4.1.

The Lemma 7.4 and Theorem 7.5 in [107] guarantee that both the starting points, $\mathbf{v}^0$ and $\boldsymbol{\alpha}^0$, and the sequences, $\{\mathbf{v}^t\}$ and $\{\boldsymbol{\alpha}^t\}$ in Algorithm 4.1 satisfy the key condition $f_{\mu_t}(\mathbf{v}^t) \leqslant \phi(\boldsymbol{\alpha}^t)$ in (4.16). Using (4.17), the convergence rate can be established via the duality gap:

$$f(\mathbf{v}^t) - \phi(\boldsymbol{\alpha}^t) \leqslant f_{\mu_t}(\mathbf{v}^t) + \mu_t D - \phi(\boldsymbol{\alpha}^t) \leqslant \mu_t D. \tag{4.19}$$

From (4.19), we can see that the duality gap which characterizes the convergence rate is reduced at the same rate at which μ_t approaches to 0. According to Step 4 in Algorithm 4.1, the closed-form equation of μ_t can be written as:

$$\begin{aligned} \mu_t &= (1-\tau_{t-1})\mu_{t-1} = \frac{t}{t+2}\mu_{t-1} = \frac{t}{t+2}\cdot\frac{t-1}{t+1}\cdots\frac{2}{4}\cdot\frac{1}{3}\cdot\mu_0 \\ &= \frac{2}{(t+1)(t+2)}\mu_0 = \frac{4L(\phi)}{(t+1)(t+2)} = \frac{4\|C\|^2}{(t+1)(t+2)}. \end{aligned} \tag{4.20}$$

Combining (4.19) and (4.20), we immediately obtain the convergence rate of Algorithm 4.1 [107].

Algorithm 4.1 Excessive Gap Algorithm for Proximal Mapping Associated with Overlapping-group-lasso Penalty

Input: β, γ, $\mathcal{G}$ and $\{w_g\}_{g\in\mathcal{G}}$
Initialization: (1) Construct C; (2) Compute $L(\phi)$ as in (4.14) and set $\mu_0 = 2L(\phi)$; (3) Set $\mathbf{v}^0 = \mathbf{v}(\mathbf{0}) = S_2(\boldsymbol{\beta})$; (4) Set $\boldsymbol{\alpha}^0 = \psi(\mathbf{0})$
Iterate For $t = 0, 1, 2, \ldots$, until convergence of $\mathbf{v}^t$:

1. Set $\tau_t = \frac{2}{t+3}$
2. Compute $\boldsymbol{\alpha}_{\mu_t}(\mathbf{v}^t)$.
3. Set $\mathbf{z}^t = (1-\tau_t)\boldsymbol{\alpha}^t + \tau_t \boldsymbol{\alpha}_{\mu_t}(\mathbf{v}^t)$
4. Update $\mu_{t+1} = (1-\tau_t)\mu_t$
5. Compute $\mathbf{v}(\mathbf{z}^t) = S_2(\boldsymbol{\beta} - C^T\mathbf{z}^t)$ as in (4.12).
6. Update $\mathbf{v}^{t+1} = (1-\tau_t)\mathbf{v}^t + \tau_t\mathbf{v}(\mathbf{z}^t)$
7. Update $\boldsymbol{\alpha}^{t+1} = \psi(\mathbf{z}^t)$ as in Proposition 4.2.

Output: $\mathbf{v}^{t+1}$.

Theorem 4.1. *(Rate of convergence for duality gap) The duality gap between the primal solution $\{\mathbf{v}^t\}$ and dual solution $\{\boldsymbol{\alpha}^t\}$ generated from Algorithm 4.1 satisfies:*

$$f(\mathbf{v}^t) - \phi(\boldsymbol{\alpha}^t) \leqslant \mu_t D = \frac{4\|C\|^2 D}{(t+1)(t+2)}, \tag{4.21}$$

where $D = \max_{\boldsymbol{\alpha}\in Q_2} d(\boldsymbol{\alpha})$. In other words, if we require that the duality gap is less than ϵ, Algorithm 4.1 needs at most $\left\lceil 2\|C\|\sqrt{\frac{D}{\epsilon}} - 1 \right\rceil$ iterations.

According to [108], the convergence rate in Theorem 4.1 has already achieved the optimal rate for solving any convex smooth problem using only the first-order information.

As for time complexity, it is easy to verify that the per-iteration complexity time of Algorithm 4.1 is *linear* in $p + \sum_{g\in\mathcal{G}} |g|$, which is very cheap and hence can easily scale up to high-dimensional data.

Remark 4.1. *As compared to the smoothing proximal gradient method (SPG) introduced in Chapter 3, the excessive gap method achieves a faster convergence rate of $O(1/t^2)$ instead of $O(1/t)$ of SPG. This is mainly because that the loss function in CCA (Euclidean distance loss) is strongly convex while the loss in high-dimensional regression is not. The excessive gap method replies on the strong convexity of the loss function to achieve the faster $O(1/t^2)$ rate.*

4.3 GROUP PURSUIT IN SPARSE CCA

When the group information is not given as prior information, it is of desire to automatically group the relevant variables into clusters under the sparse CCA framework. For this purpose, we propose the group pursuit sparse CCA model in this section. Our group pursuit approach is based on pairwise comparisons between v_i and v_j for all $1 \leqslant i < j \leqslant p$: when $v_i = v_j$, the i-th and j-th variables are grouped together. We identify all subgroups among p variables by conducting pairwise comparisons and applying transitivity rule, i.e. $v_i = v_j$ and $v_j = v_k$ implies that the i-th, j-th, and k-th variables are clustered into the same group. The pairwise comparisons can be naturally encoded in the *graph-guided fusion penalty* as introduced in Section 2.1, $\sum_{i<j} |v_i - v_j|$, where the ℓ_1-norm will enforce $v_i - v_j = 0$ for closely related (i,j) pairs. The idea of using such a penalty arises from the *fused lasso* in [134], where it uses the penalty $\sum_{j=1}^{p-1} |v_{j+1} - v_j|$ to capture the *changing point* of the parameters on an ordered chain and the learned parameters are piece-wise constant.

In practice, instead of using the simple penalty $\sum_{i<j} |v_i - v_j|$ which treats each pair of variables equally, we could add the weight w_{ij} to incorporate the prior knowledge that how likely that the i-th and j-th variables are in the same group. Moreover, the ℓ_1-norm of $\mathbf{v}$ is also incorporated in the penalty to enforce sparse solution as in the fused lasso [134]. Then, our *group pursuit penalty* can be formulated as:

$$P_2(\mathbf{v}) = \sum_{i<j} w_{ij}|v_i - v_j| + c'\|\mathbf{v}\|_1 + \frac{c}{2}\mathbf{v}^T\mathbf{v}, \tag{4.22}$$

where c' is the tuning parameter to balance the ℓ_1-norm penalty and the fusion penalty. This penalty function will simultaneously select the relevant variables and cluster them in to groups in an automatic manner. A natural way for assigning w_{ij} is to set $w_{ij} = |r_{ij}|^q$, where r_{ij} is the correlation between the i-th and j-th variable; q models the strength of the prior: a larger q results in a stronger belief of the correlation based group structure. For the purpose of simplicity, we set $w_{ij} = |r_{ij}|$ with $q = 1$ throughout this chapter; while in principle, any prior knowledge of the possibility of being in the same group can be incorporated into w.

In some cases, we have the prior knowledge that the i-th and j-th variables do not belong to the same group; then the term $|v_i - v_j|$ should not appear in the group pursuit penalty (4.22). Therefore, rather than having $|v_i - v_j|$ for all (i,j) pairs which forms a complete graph, we generalize the group pursuit penalty with the fusion penalty defined on an *arbitrary graph* with the edge set E. In summary, the *group pursuit sparse CCA* is defined as follows:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{v}} \quad & -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} + \frac{\tau}{2}\mathbf{v}^T\mathbf{v} + \theta_1\|\mathbf{v}\|_1 + \theta_2 \sum_{(i,j)\in E} w_{ij}|v_i - v_j| \quad (4.23) \\ \text{s.t.} \quad & \|\mathbf{u}\|_2 \leqslant 1,\ \|\mathbf{v}\|_2 \leqslant 1,\ P_1(\mathbf{u}) \leqslant c_1. \end{aligned}$$

It is straightforward to specialize excessive gap method for solving the group pursuit sparse CCA with a similar approach. . Note that

Table 4.1: Comparison between ExGap with Grad and IPM for SOCP

$\lvert\mathcal{G}\rvert = 20,\quad p = 20, 100$					$\lvert\mathcal{G}\rvert = 40,\quad p = 40, 100$				
$\gamma = 0.2$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 0.4$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	4.059E-3	4.4063E+3	4.900E-12	2	ExGap	1.960E-2	8.8682E+3	5.259E-11	3
Grad	3.135E+1	4.4063E+3	—	69	Grad	1.753E-1	8.8682E+3	—	19
SOCP	4.866E+1	4.4063E+3	8.723E-8	17	SOCP	9.642E+2	8.8682E+3	7.048E-9	20
$\gamma = 2$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 4$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	2.591E-2	4.4123E+3	1.197E-9	9	ExGap	1.626E-1	8.8851E+3	8.384E-7	18
Grad	3.496E+1	4.4123E+3	—	773	Grad	9.462E+1	8.8851E+3	—	1036
SOCP	4.440E+1	4.4123E+3	8.267E-5	13	SOCP	2.952E+3	8.8851E+3	3.844E-8	18
$\lvert\mathcal{G}\rvert = 100,\quad p = 100, 100$					$\lvert\mathcal{G}\rvert = 500,\quad p = 500, 100$				
$\gamma = 1$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 5$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	2.172E-1	2.2296E+4	6.646E-8	9	ExGap	4.381E+1	1.1211E+5	4.816E-8	51
Grad	5.219E+1	2.2296E+4	—	199	Grad	1.021E+2	1.1211E+5	—	723
$\gamma = 10$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 50$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	1.288E+0	2.2362E+4	7.891E-7	48	ExGap	1.855E+2	1.1250E+5	9.757E-7	2144
Grad	9.463E+1	2.2363E+4	—	1293	Grad	2.831E+3	1.1286E+5	—	*20000*
$\lvert\mathcal{G}\rvert = 1000,\quad p = 1,000, 100$					$\lvert\mathcal{G}\rvert = 5000,\quad p = 5,000, 100$				
$\gamma = 10$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 50$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	2.232E+2	2.2456E+5	9.376E-7	102	ExGap	1.579E+3	1.1245E+6	9.615E-7	1752
Grad	2.432E+2	2.2456E+5	—	867	Grad	2.977E+5	1.1261E+6	—	*20000*
$\gamma = 100$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 500$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	8.108E+3	2.2500E+5	8.677E-7	3872	ExGap	7.432E+3	1.1250E+6	9.851E-7	9080
Grad	5.718E+3	2.2668E+5	—	*20000*	Grad	2.981E+5	1.1498E+6	—	*20000*

other group pursuit penalties [125, 158] have also been recently proposed in the regression setting. However, they either cannot obtain sparse solutions or is computationally very difficult under the sparse CCA framework.

4.4 EXPERIMENT

In this section, we present the numerical results on both simulated and real datasets to illustrate the performance of the proposed algorithm.

4.4.1 *Computational Efficiency of Excessive Gap Method*

In this section, we evaluate the scalability and efficiency of the excessive gap method (ExGap) for solving the proximal mapping associated with the overlapping-group-lasso penalty:

$$\arg\min_{\mathbf{v}:\|\mathbf{v}\|_2\leqslant 1} f(\mathbf{v}) = \frac{1}{2}\|\mathbf{v}-\boldsymbol{\beta}\|_2^2 + \gamma \sum_{g\in\mathcal{G}} w_g \|\mathbf{v}_g\|_2,$$

where $\boldsymbol{\beta}$ is given and all w_g are assumed to be 1 for simplicity. We compare ExGap with two widely used optimization methods: (1) formulating the problem into a second-order cone programming (SOCP) and solving by interior-point method (IPM) using the state-of-the-art MATLAB package SPDT3 [140] and (2) projected subgradient-descent method (Grad) The stepsize of the Grad is set to $\frac{\eta}{\sqrt{t}}$ as suggested in [39], where the constant η is carefully tuned to be $\frac{0.1}{\sqrt{p}}$. All of the experiments are performed on a PC with Intel Core 2 Quad Q6600 2.4GHz CPU and 4GB RAM. The software is written in MATLAB. We terminate the optimization procedure of ExGap and SOCP when the relative duality gap (Rel_Gap) is less than 10^{-6}: Rel_Gap $= \frac{|f(\mathbf{v}^t)-\phi(\boldsymbol{\alpha}^t)|}{1+|f(\mathbf{v}^t)|+|\phi(\boldsymbol{\alpha}^t)|} \leqslant 10^{-6}$. For Grad, since there is no dual solutions, we use objective of ExGap as the "optimal" objective for Grad and stop Grad when its objective is less than 1.00001 times the objective of ExGap. We set the maximum iteration for all methods to be 20,000.

More specifically, we generate the data using the similar approach as in [65], with an overlapping group structure imposed on $\boldsymbol{\beta}$ as described below. Assuming that inputs are ordered and each group is of size 1000, we define a sequence of groups of 1000 adjacent inputs with an overlap of 100 variables between two successive groups, i.e. $\mathcal{G} = \{\{1,\ldots,1000\},\{901,\ldots,1900\},\ldots,\{p-999,\ldots,p\}\}$ with $p = 1000|\mathcal{G}| + 100$. We set the support of $\boldsymbol{\beta}$ to be the first half of the variables and set the values of $\boldsymbol{\beta}$ in the support to be 1 and otherwise 0.

We vary the number of the groups $|\mathcal{G}|$ and report the CPU time in *seconds* (CPU), primal objective value (Primal Obj), relative duality gap (Rel_Gap) and the number of iterations (Iter) in Table 4.1. For each setting of $|\mathcal{G}|$, we use two levels of regularization: (1) $\gamma = \frac{|\mathcal{G}|}{100}$ and (2) $\gamma = \frac{|\mathcal{G}|}{10}$. Note that when $|\mathcal{G}| \geqslant 50$ ($p \geqslant 50,100$), we are unable to collect results for SOCP, because they lead to out-of-memory errors due to the large storage requirement for solving the Newton linear system. In addition, for large $|\mathcal{G}|$, Grad cannot converge in 20,000 iterations. From Table 4.1, we can see that ExGap achieves the same objective value as SOCP with small relative duality gap. For all different scales of the problem, ExGap is much more efficient than SOCP and Grad. It can easily scale up to high-dimensional data with millions of variables. Another interesting observation is that: for smaller γ which leads to smaller $\|C\|$ and $L(\phi)$, the convergence of ExGap is much faster. This observation is consistent with convergence result in

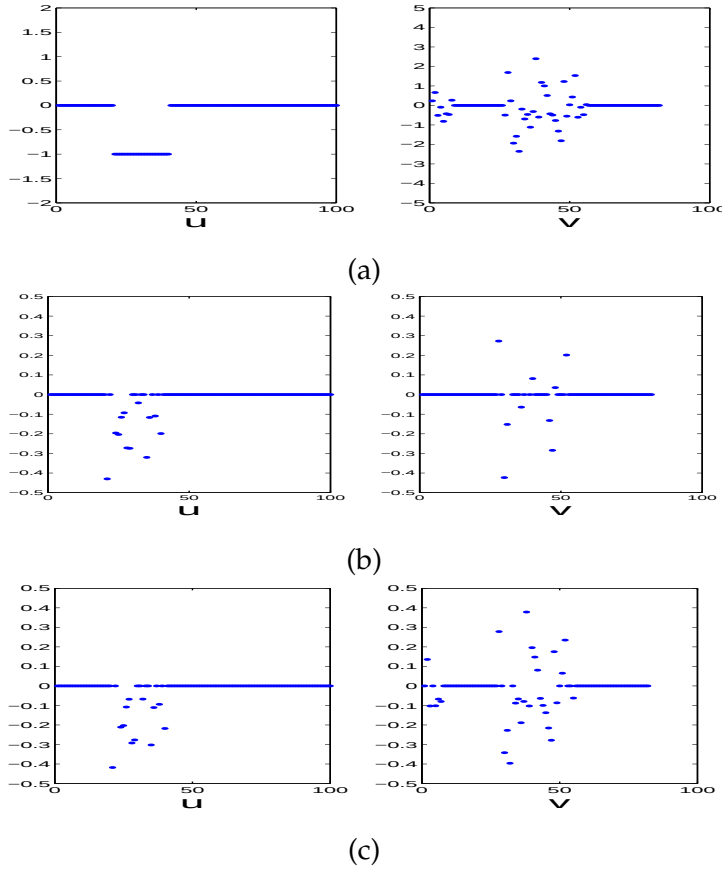


Figure 4.2: (a) True $\mathbf{u}$ and $\mathbf{v}$; (b) Estimated $\mathbf{u}$ and $\mathbf{v}$ using the ℓ_1-regularized sparse CCA; (c) Estimated $\mathbf{u}$ and $\mathbf{v}$ using the group-structured sparse CCA.

Theorem 4.1. It suggests that ExGap is more efficient when the non-smooth part plays less important role in the optimization problem.

4.4.2 *Simulations*

In this and next subsections, we use simulated data and a real eQTL dataset to investigate the performance of the overlapping group-structured and network-structured sparse CCA. All the regularization parameters are chosen from $\{0.01, 0.02, \ldots, 0.09, 0.1, 0.2, \ldots, 0.9, 1, 2, 10\}$ and set using the permutation-based method in [145]. Instead of tuning all the parameters on a multi-dimensional grid which is computationally heavy, we first train the ℓ_1-regularized sparse CCA (i.e. $P_1(\mathbf{u}) = \|\mathbf{u}\|_1$, $P_2(\mathbf{v}) = \|\mathbf{v}\|_1$) and the tuned regularization parameter c_1 in (2.13) is used for all structured models. For the overlapping-group-lasso penalty in (4.2), all the group weights $\{w_g\}$ are set to 1. In addition, we observe that the learned sparsity pattern is quite insensitive to the parameter τ (the regularization parameter for the quadratic penalty); and therefore we set it to 1 for simplicity. For all algorithms, we use 10 random initializations of $\mathbf{u}$ and select the results that lead to the largest correlation.

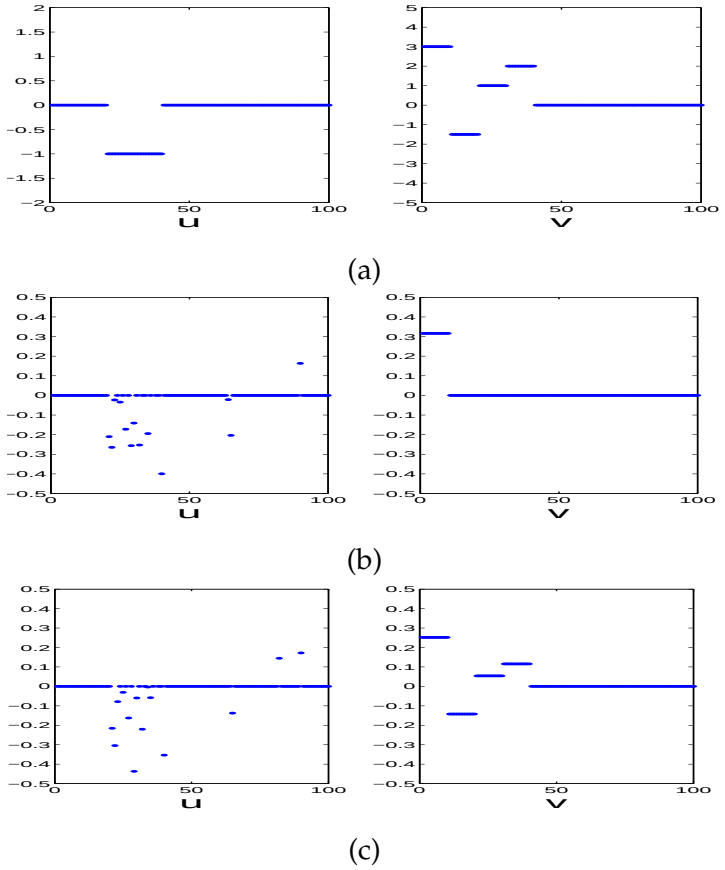


Figure 4.3: (a) True $\mathbf{u}$ and $\mathbf{v}$; (b) Estimated $\mathbf{u}$ and $\mathbf{v}$ using the ℓ_1-regularized sparse CCA; (c) Estimated $\mathbf{u}$ and $\mathbf{v}$ using the group pursuit sparse CCA.

4.4.2.1 *Group-structured Sparse CCA*

In this section, we conduct the simulation where the overlapping group structure in $\mathbf{v}$ is given as *a priori*. We generate the data $\mathbf{X}$ and $\mathbf{Y}$ with $n = 50$, $d = 100$ and $p = 82$ as follows. Let $\mathbf{u}$ be a vector of length d with 20 0s, 20 -1s, and 60 0s. We construct $\mathbf{v}$ with $p = 82$ variables using the same approach as in [65]: assuming that $\mathbf{v}$ is covered by 10 groups; each group has 10 variables with 2 variables overlapped between every two successive groups, i.e. $\mathcal{G} = \{\{1,\ldots,10\},\{9,\ldots,18\},\ldots,\{73,\ldots,82\}\}$. For the indices of the 2nd, 3rd, 8th, 9th and 10th groups, we set the corresponding entries of $\mathbf{v}$ to be zeros and the other entries are sampled from i.i.d. $N(0,1)$. In addition, we randomly generate a latent vector $\mathbf{z}$ of length n and normalize it to unit length.

We generate the data matrix $\mathbf{X}$ with each $X_{ij} \sim N(z_i u_j, 1)$ and $\mathbf{Y}$ with each $Y_{ij} \sim N(z_i v_j, 1)$. The true and estimated vectors for $\mathbf{u}$ and $\mathbf{v}$ are presented in Figure 4.2. For the group-structured sparse CCA, we add the regularization $\sum_{g\in\mathcal{G}} \|\mathbf{v}_g\|_2$ on $\mathbf{v}$ where $\mathcal{G}$ is taken from the prior knowledge. It can be seen that the group-structured sparse CCA recovers the true $\mathbf{v}$ much better while the simple ℓ_1-regularized sparse CCA leads to an over-sparsified $\mathbf{v}$ vector.

4.4.2.2 *Group Pursuit Sparse CCA*

In this simulation we assume that the group structure over $\mathbf{v}$ is unknown and the goal is to uncover the group structure using the network-structured sparse CCA. We generate the data $\mathbf{X}$ and $\mathbf{Y}$ with $n = 50$ and $p = d = 100$ as follows. Let $\mathbf{u}$ be a vector of length d with 20 0s, 20 -1s, and 60 0s as in the previous simulation study; and $\mathbf{v}$ be a vector of length p with 10 3s, 10 -1.5s, 10 1s, 10 2s and 60 0s. In addition, we randomly generate a latent vector $\mathbf{z}$ of length n and normalize it to unit length.

We generate $\mathbf{X}$ with each sample $\mathbf{x}_i \sim N(z_i\mathbf{u}, 0.1I_{d\times d})$; and $\mathbf{Y}$ with each sample $\mathbf{y}_i \sim N(z_i\mathbf{v}, 0.1\Sigma_y)$ where $(\Sigma_y)_{jk} = \exp^{-|\nu_j-\nu_k|}$. We conduct the group pursuit via the network-structured sparse CCA in (4.23), where we add the fusion penalty for each pair of variables in $\mathbf{v}$, i.e., E is the edge set of the complete graph. The estimated vector $\mathbf{u}$ and $\mathbf{v}$ are presented in Figure 4.3 (c). It can be easily seen that the network-structured sparse CCA correctly captures the group structure in the $\mathbf{v}$ vector. This experiment demonstrates that network-structured sparse CCA could be a useful tool for conducting group pursuit in the CCA framework.

4.4.3 *Real eQTL Data*

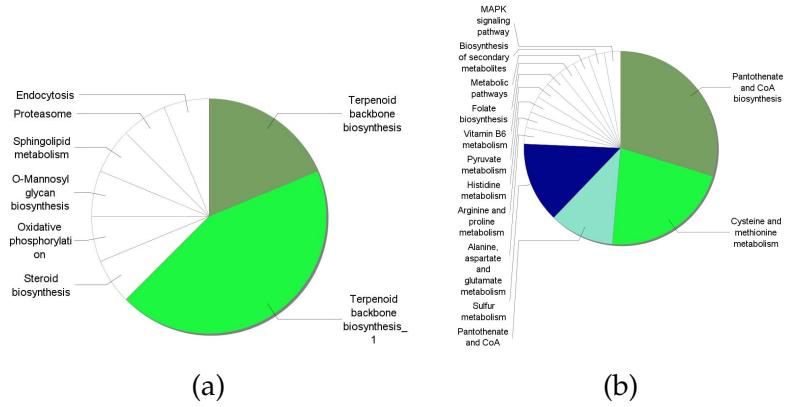
4.4.3.1 *Group-structured Sparse CCA: Pathway Selection*

In this section, we report experiment results on a yeast eQTL data[18, 45]. In particular, we have two data matrices, $\mathbf{X}$ and $\mathbf{Y}$. $\mathbf{X}$ contains $d = 1260$ SNPs from the chromosomes 1–16 for $n = 124$ yeast strains. $\mathbf{Y}$ is the gene expression data of $p = 1,155$ genes for the same 124 yeast strains. All these $p = 1,155$ genes are from the KEGG database [69]. According to KEGG, these genes belong to 92 pathways. We treat each pathway as a group . The statistics of the 92 pathways are summarized as follows: the average number of genes in each group is 25.78; and the largest group has 475 genes. One thing to note is that there are a lot of overlapping genes among the 92 pathways and the average appearance frequency of each gene in the pathways is 2.05. To achieve more refined resolution of gene selection, besides the 92 pathway groups, we also add in $p = 1,155$ groups where each group only has one singleton gene. Therefore, instead of just selecting genes at the pathway level, we could also select genes within each pathway.

Using the group-structured sparse CCA, we selected altogether 121 SNPs (i.e. the number of nonzero elements in estimated $\mathbf{u}$) and 47 genes (i.e. the number of nonzero elements in estimated $\mathbf{v}$). These 47 genes spread over 32 pathways. Such a high coverage of pathways is mainly due to the fact that several selected genes are important in different biological processes and hence each of them belongs to multiple pathways. There are 14 pathways that contain at least two selected genes as listed in Table 4.2. Among these 14 pathways, 5 of them are highly significant with the p-value less than 0.001. Using the tool ClueGo [12], the overview chart of KEGG enrichment on the

Table 4.2: List of pathways with at least 2 selected genes in the pathway: the first two columns are the pathway ID and annotation from KEGG, the third column is the number of selected genes in this pathway; the fourth column is the ratio of the number of selected genes in the pathway (third column) over the number of genes in the dataset in the pathway; the last column gives the p-values which is calculated as the hypergeometric probability to get so many genes for a KEGG pathway annotation.

ID	Annotation	No. Genes	Ratio	p-value
00072	Synthesis and degradation of ketone bodies	2	100.0	7.65E-4**
00280	Valine, leucine and isoleucine degradation	2	18.18	3.58E-2
00620	Pyruvate metabolism	2	6.06	2.35E-1
00640	Propanoate metabolism	2	18.18	3.58E-2
00650	Butanoate metabolism	2	10.00	1.05E-2
00900	Terpenoid backbone biosynthesis	7	53.85	1.26E-8**
01100	Metabolic pathways	29	4.58	1.00E-3*
01110	Biosynthesis of secondary metabolites	15	6.64	7.24E-4**
00100	Steroid biosynthesis	10	66.67	2.84E-13**
00190	Oxidative phosphorylation	4	5.26	1.59E-1
00514	O-Mannosyl glycan biosynthesis	3	23.07	4.81E-3
00600	Sphingolipid metabolism	2	15.38	4.91E-2
03050	Proteasome	2	5.71	2.56E-1
04144	Endocytosis	2	5.56	2.66E-1



(a) (b)

Figure 4.4: Overview chart of KEGG functional enrichment using (a) the group-structured sparse CCA; (b) ℓ_1-regularized sparse CCA

selected genes is presented in Figure 4.4 (a). We can see from Figure 4.4 (a) that the *Terpenoid backbone biosynthesis* is the most important functional group, which is a large class of natural products consisting of isoprene (C5) units. In fact, the first 8 pathways in Table 4.2 are all closely related to *Terpenoid backbone biosynthesis*. As a comparison, the ℓ_1-regularized sparse CCA selects 173 SNPs and 71 genes and

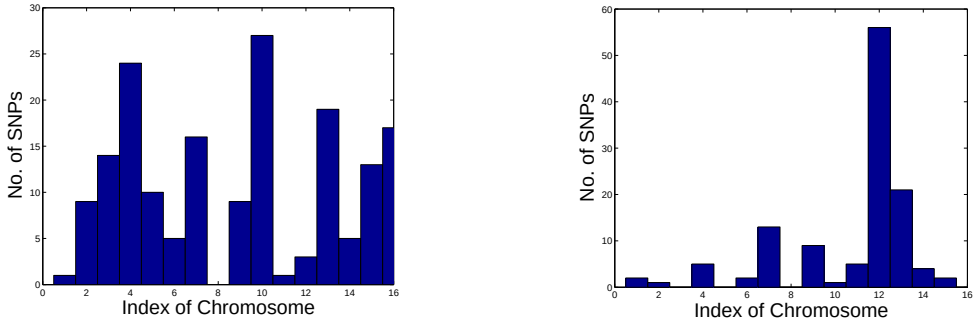
Table 4.3: GO enrichment analysis for the selected genes using the group-structured sparse CCA: the first two columns are the GO ID (category) and annotation, the third column is the number of selected genes having the GO annotation, the fourth column is the GO cluster size and the last column gives the p-value. The rows are ranked according to the increasing order of p-values.

GO ID	GO Attribute	N	X	p-value
0006696	ergosterol biosynthetic process	17	25	2.71E-32
0008204	ergosterol metabolic process	17	27	2.09E-31
0006694	steroid biosynthetic process	17	34	5.61E-29
0016126	sterol biosynthetic process	17	34	5.61E-29
0016125	sterol metabolic process	17	45	2.52E-26
0008202	steroid metabolic process	17	49	1.46E-25
0008610	lipid biosynthetic process	20	149	4.71E-21
0006066	cellular alcohol metabolic process	21	210	2.09E-19
0044255	cellular lipid metabolic process	21	259	1.71E-17
0006629	lipid metabolic process	21	279	8.00E-17
0003824	catalytic activity	42	2195	9.20E-15
0006720	isoprenoid metabolic process	7	13	1.35E-12
0008299	isoprenoid biosynthetic process	7	13	1.35E-12
0005783	endoplasmic reticulum	20	410	2.28E-12
0043094	cellular metabolic compound salvage	12	159	1.08E-09
0005789	endoplasmic reticulum membrane	15	293	1.42E-09
0044432	endoplasmic reticulum part	15	317	4.23E-09
0006695	cholesterol biosynthetic process	4	5	1.36E-08
0008203	cholesterol metabolic process	4	5	1.36E-08
0031090	organelle membrane	23	954	4.61E-08
0044444	cytoplasmic part	39	2810	6.43E-08
0044237	cellular metabolic process	46	4187	1.08E-07
0044238	primary metabolic process	43	3581	1.86E-07
0055114	oxidation reduction	13	309	2.32E-07
0016491	oxidoreductase activity	13	318	3.23E-07
0008152	metabolic process	46	4321	4.35E-07

these 71 genes belong to 50 pathways. The KEGG enrichment for ℓ_1-regularized sparse CCA is presented Figure 4.4(b).

In addition to the above pathway analysis, we also perform the enrichment analysis on the selected genes according to the standard Gene Ontology (GO). The enrichment is carried out using the standard annotation tool from [9] and the result is shown in Table 4.3 with the cutoff point set to 1E-3. We see that most of clusters are significantly enriched and many of them are known to be explicitly relevant. For example, *isoprenoid metabolic process* and *isoprenoid biosynthetic process* are very closely related to *terpenoid backbone biosynthesis*.

The group-structured sparsity-inducing penalty on genes will also affect the selection of SNPs. As a comparison, the ℓ_1-regularized sparse CCA selects 173 SNPs. while the group-structured sparse CCA selects only 121 SNPs. The number of selected SNPs in each chromosome is



(a) (b)

Figure 4.5: The number of selected SNPs in each chromosome using (a) the ℓ_1-regularized sparse CCA; (b) the group-structured sparse CCA.

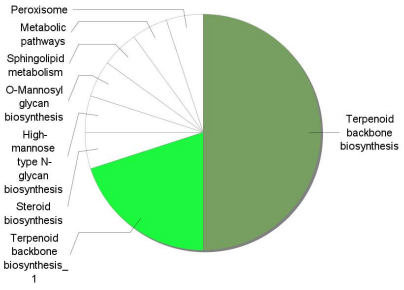


Figure 4.6: Overview chart of KEGG functional enrichment using the tree-structured sparse CCA;

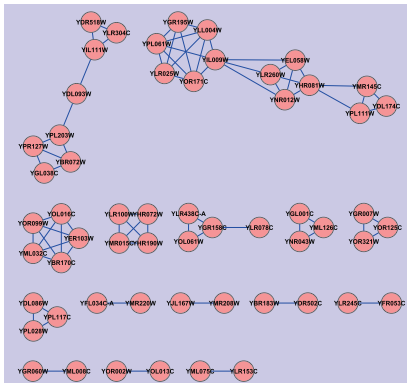


Figure 4.7: Selected genes and their relationship estimated by the network-structured sparse CCA

presented in Figure 4.5. As we can see, most of the selected SNPs using the group-structured sparse CCA belong to Chromosome 12 and 13.

4.4.3.2 *Tree-structured Sparse CCA*

In this experiment, rather than utilizing the group information extracted from the KEGG pathways [69], we learn a hierarchical tree structure on the same yeast dataset and then use the learned tree structure to define the groups. In more details, we run the hierarchical agglomerative clustering on the $p \times p$ correlation matrix of $\mathbf{Y}$ where each leaf node of the tree corresponds to a single gene. We discard the tree nodes for weak correlations near the root of the tree. In particularly, we calculate the distance of each tree node to the root; normalize them by the maximum distance and discard those nodes with the distance less than 0.3. Each node of the tree defines a group which contains the genes represented by its leaf children nodes. Finally, we obtain 973 groups corresponding to internal nodes and 1,155 singleton group corresponding to leaf nodes. After obtaining the group

structure induced from the clustering tree, we then apply the group-structured sparse CCA as in the last section. We name the obtained model as *tree-structured sparse CCA*.

The tree-structured sparse CCA selects altogether 123 SNPs and 66 genes. An overview chart of functional enrichment using the KEGG pathways is presented in Figure 4.6. As we can see, most of the functions are identical to those learned by the group-structured sparse CCA with the group information obtained from KEGG, e.g. *Terpenoid backbone biosynthesis*, *Steroid biosynthesis*, *O-Mannosyl glycan biosynthesis*, *Sphingolipid*, etc. We also perform the GO enrichment analysis on the selected genes and obtained 30 GO clusters. All of these clusters are significantly enriched and 25 of them are the same as the GO clusters obtained by the group-structured sparse CCA with the groups from KEGG. This experiment suggests that, even without any prior knowledge of the pathway information as group structure, our correlation based tree-structured sparse CCA can also select the relevant genes and provide the similar enrichment results.

4.4.3.3 *Group Pursuit Sparse CCA*

Now, we do not assume any prior information of the group structure among the genes. Our goal is to simultaneously select the relevant genes and group them into clusters. Using the group pursuit sparse CCA in (4.23) and thresholding the absolute value of the pair-wise correlation at 0.8 to construct the edge set E, we selected 61 genes in total. We present the obtained clusters among these 61 genes in Figure 4.4 (c) where two selected genes are connected if the absolute value of difference between the estimated parameters ($|v_i - v_j|$) is less than 1E-3 (singleton nodes are not plotted due to space limitations) . We observe that there are two obvious clusters. With the learned clustering structure, we can study the functional enrichment of one cluster by another which leads to more elaborate analysis as compared to the analysis of all the genes.

5

STOCHASTIC OPTIMIZATION: OPTIMAL REGULARIZED DUAL AVERAGING METHODS

In the previous chapters, we focus on deterministic optimization problems where we assume that the data points are pre-given and each gradient needs to be computed on all the data points. In this chapter, we study the stochastic optimization which directly minimizes the expectation of the loss function. In particular, we consider a wide spectrum of *regularized stochastic optimization problems* for large-scale online optimization where both the loss function and regularizer can be non-smooth. We develop a new algorithm based on the regularized dual averaging (RDA) method, that can simultaneously achieve the optimal convergence rates for both convex and strongly convex loss. In particular, for strongly convex loss, it achieves the optimal rate of $O(\frac{1}{N} + \frac{1}{N^2})$ for N iterations, which improves the rate $O(\frac{\log N}{N})$ for previous regularized dual averaging algorithms. In addition, our method constructs the final solution directly from the proximal mapping instead of averaging of all previous iterates. For widely used sparsity-inducing regularizers (e.g., ℓ_1-norm), it has the advantage of encouraging sparser solutions. We further develop a multi-stage extension using the proposed algorithm as a subroutine, which achieves the uniformly-optimal rate $O(\frac{1}{N} + \exp\{-N\})$ for strongly convex loss.

5.1 INTRODUCTION AND MOTIVATION

Many risk minimization problems in machine learning can be formulated into a regularized stochastic optimization problem of the following form [1]:

$$\min_{x \in \mathcal{X}} \{\phi(x) := f(x) + h(x)\}. \tag{5.1}$$

Here, the set of feasible solutions $\mathcal{X}$ is a convex set in $\mathbb{R}^n$, which is endowed with a norm $\|\cdot\|$ and the dual norm $\|\cdot\|_*$. The regularizer $h(x)$ is assumed to be convex, but could be non-differentiable. Popular examples of $h(x)$ include ℓ_1-norm and related sparsity-inducing regularizers. The loss function $f(x)$ takes the form:

$$f(x) := \mathbb{E}_\xi(F(x, \xi)) = \int F(x, \xi) dP(\xi),$$

where ξ is a random vector with the distribution P. In typical regression or classification tasks, ξ is the input and response (or class label) pair. We assume that for every random vector ξ, $F(x, \xi)$ is a convex

1 In this chapter, following standard notations in the field of stochastic optimization, we use x to denote decision variables, i.e., the regression coefficient vector (β in previous chapters) in the regression setting.

and continuous function in $x \in \mathcal{X}$. Therefore, $f(x)$ is also convex. Furthermore, we assume that there exist constants $L \geqslant 0$, $M \geqslant 0$ and $\widetilde{\mu} \geqslant 0$ such that

$$\begin{aligned} \frac{\widetilde{\mu}}{2}\|x-y\|^2 &\leqslant f(y) - f(x) - \langle y - x, f'(x)\rangle \\ &\leqslant \frac{L}{2}\|x-y\|^2 + M\|x-y\|, \quad \forall x, y \in \mathcal{X}, \end{aligned} \tag{5.2}$$

where $f'(x) \in \partial f(x)$, the subdifferential of f. We note that this assumption allows us to adopt a wide class of loss functions. For example, if $f(x)$ is smooth and its gradient $f'(x) = \nabla f(x)$ is Lipschitz continuous, we have $L > 0$ and $M = 0$ (e.g., squared or logistic loss). If $f(x)$ is non-smooth but Lipschitz continuous, we have $L = 0$ and $M > 0$ (e.g., hinge loss). If $\widetilde{\mu} > 0$, $f(x)$ is strongly convex and $\widetilde{\mu}$ is the so-called strong convexity parameter.

In general, the optimization problem in (5.1) is challenging since the integration in $f(x)$ is computationally intractable for high-dimensional P. In many learning problems, we do not even know the underlying distribution P but can only generate *i.i.d.* samples ξ from P. A traditional approach is to consider empirical loss minimization problem where the expectation in $f(x)$ is replaced by its empirical average on a set of training samples $\{\xi_1, \ldots, \xi_m\}$: $f_{\text{emp}}(x) := \frac{1}{m}\sum_{i=1}^{m} F(x, \xi_i)$. However, for modern data-intensive applications, minimization of empirical loss with an off-line optimization solver could suffer from very poor scalability.

In the past few years, many stochastic (sub)gradient methods [39, 41, 61, 82, 106, 52, 68, 80, 56, 115] have been developed to directly solve the stochastic optimization problem in (5.1), which enjoy low per-iteration complexity and the capability of scaling up to very large data sets. In particular, at the t-th iteration with the current iterate x_t, these methods randomly draw a sample ξ_t from P; then compute the so-called "stochastic subgradient" $G(x_t, \xi_t) \in \partial_x F(x_t, \xi_t)$ where $\partial_x F(x_t, \xi_t)$ denotes the subdifferential of $F(x, \xi_t)$ with respect to x at x_t; and update x_t using $G(x_t, \xi_t)$. These algorithms fall into the class of *stochastic approximation* methods. More rigourously, a stochastic gradient of $f(x)$ at x (denoted as $G(x, \xi)$) is defined as a vector-valued vector such that $\mathbb{E}_\xi G(x, \xi) = f'(x) \in \partial f(x)$. Recently, Xiao [148] proposed the *regularized dual averaging (RDA)* method and its accelerated version (AC-RDA) based on Nesterov's primal-dual method [111]. Instead of only utilizing a single stochastic subgradient $G(x_t, \xi_t)$ of the current iteration, it updates the parameter vector using the average of all past stochastic subgradients $\{G(x_i, \xi_i)\}_{i=1}^{t}$ and hence leads to improved empirical performances.

In this chapter, we propose a novel regularized dual averaging method, called *optimal RDA* or *ORDA*, which achieves the optimal expected convergence rate of $\mathbb{E}[\phi(\widehat{x}) - \phi(x^*)]$, where $\widehat{x}$ is the solution from ORDA and x^* is the optimal solution of (5.1). As compared to previous dual averaging methods, it has three main advantages:

1. For strongly convex $f(x)$, ORDA improves the convergence rate of stochastic dual averaging methods $O(\frac{\sigma^2 \log N}{\widetilde{\mu} N}) \approx O(\frac{\log N}{\widetilde{\mu} N})$

[111, 148] to an optimal rate $O\left(\frac{\sigma^2+M^2}{\widetilde{\mu}N}+\frac{L}{N^2}\right)\approx O\left(\frac{1}{\widetilde{\mu}N}\right)$, where σ^2 is the variance of the stochastic subgradient, N is the number of iterations, and the parameters $\widetilde{\mu}$, M and L of $f(x)$ are defined in (5.2).

2. ORDA is a *self-adaptive* and *optimal* algorithm for solving both convex and strongly convex $f(x)$ with the strong convexity parameter $\widetilde{\mu}$ as an input. When $\widetilde{\mu}=0$, ORDA reduces to a variant of AC-RDA in [148] with the optimal rate for solving convex $f(x)$. Furthermore, our analysis allows $f(x)$ to be non-smooth while AC-RDA requires the smoothness of $f(x)$. For strongly convex $f(x)$ with $\widetilde{\mu}>0$, our algorithm achieves the optimal rate of $\left(\frac{1}{\widetilde{\mu}N}\right)$ while AC-RDA does not utilize the advantage of strong convexity.

3. Existing RDA methods [148] and many other stochastic gradient methods (e.g., [106, 52]) can only show the convergence rate for the averaged iterates: $\bar{x}_N=\sum_{t=1}^N \rho_t x_t/\sum_{t=1}^N \rho_t$, where the $\{\rho_t\}$ are nonnegative weights. However, in general, the average iterates $\bar{x}_N$ cannot keep the structure that the regularizer tends to enforce (e.g., sparsity, low-rank, etc). For example, when $h(x)$ is a sparsity-inducing regularizer (ℓ_1-norm), although x_t computed from proximal mapping will be sparse as t goes large, the averaged solution could be non-sparse. In contrast, our method directly generates the final solution from the proximal mapping, which leads to sparser solutions.

In addition to the rate of convergence, we also provide *variance bounds* and *high probability bounds* on the error of objective values. The variance bound is important for characterizing the the uncertainty of the error. By showing that $\mathrm{Var}[\phi(\widehat{x})-\phi(x^*)]$ converges to zero at the rate of $O(1/N)$ for ORDA, we conclude that each single run of our algorithm is reliable when N is large enough. For high probability bounds, utilizing a technical lemma from [42], we could show the same bound as in RDA [148] but under a weaker assumption.

Furthermore, using ORDA as a subroutine, we develop the multi-stage ORDA which obtains the convergence rate of $O\left(\frac{\sigma^2+M^2}{\widetilde{\mu}N}+\exp\{-\sqrt{\widetilde{\mu}/L}N\}\right)$ for strongly convex $f(x)$. Recall that ORDA has the rate $O\left(\frac{\sigma^2+M^2}{\widetilde{\mu}N}+\frac{L}{N^2}\right)$ for strongly convex $f(x)$. The rate of muli-stage ORDA improves the second term in the rate of ORDA from $O\left(\frac{L}{N^2}\right)$ to $O\left(\exp\{-\sqrt{\widetilde{\mu}/L}N\}\right)$ and achieves the so-called "*uniformly-optimal*" rate [105]. Although the improvement is on the non-dominating term, multi-stage ORDA is an optimal algorithm for both stochastic and deterministic optimization. In particular, for *deterministic* strongly convex and smooth $f(x)$ ($M=0$), one can use the same algorithm but only replaces the stochastic subgradient $G(x,\xi)$ by the deterministic gradient $\nabla f(x)$. Then, the variance of the stochastic subgradient $\sigma=0$. Now the term $\frac{\sigma^2+M^2}{\widetilde{\mu}N}$ in the rate equals to o and multi-stage ORDA becomes an optimal deterministic solver with the exponential rate $O\left(\exp\{-\sqrt{\widetilde{\mu}/L}N\}\right)$.

This is the reason why such a rate is "uniformly-optimal", i.e., optimal with respect to both stochastic and deterministic optimization.

5.2 PRELIMINARY AND NOTATIONS

In the framework of first-order stochastic optimization, the only available information of $f(x)$ is the stochastic subgradient. Formally speaking, stochastic subgradient of $f(x)$ at x, $G(x,\xi)$, is a vector-valued function such that $\mathbb{E}_\xi G(x,\xi) = f'(x) \in \partial f(x)$. Following the existing literature in stochastic optimization [79, 52, 80] , a standard assumption on $G(x,\xi)$ is made throughout the chapter : there exists a constant σ such that for all $x \in \mathcal{X}$,

$$\mathbb{E}_\xi(\|G(x,\xi) - f'(x)\|_*^2) \leqslant \sigma^2. \tag{5.3}$$

We note that this assumption is weaker than the standard assumption in online learning setting. In particular, in most online learning literature (e.g., [161, 148, 124]), it either assumes that: there exists a constant B such that for all $x \in \mathcal{X}$ and ξ:

$$\|G(x,\xi)\|_* \leqslant B. \tag{5.4}$$

or for all $x \in \mathcal{X}$:

$$\mathbb{E}_\xi(\|G(x,\xi)\|_*^2) \leqslant B^2. \tag{5.5}$$

It can be seen that the assumption in (5.4) implies the one in (5.5). By some simple derivation as follows, we could see that (5.5) implies our assumption in (5.3). In particular, by the fact that

$$\|G(x,\xi) - f'(x)\|_*^2 \leqslant \left(\|G(x,\xi)\|_* + \|f'(x)\|_*\right)^2 \leqslant 2\|G(x,\xi)\|_*^2 + 2\|f'(x)\|_*^2$$

we have:

$$\begin{aligned}\mathbb{E}_\xi(\|G(x,\xi) - f'(x)\|_*^2) &\leqslant 2\mathbb{E}_\xi\left(\|G(x,\xi)\|_*^2\right) + 2\|\mathbb{E}_\xi(G(x,\xi))\|_*^2 \\ &\leqslant 2B^2 + 2\mathbb{E}_\xi(\|G(x,\xi)\|_*^2) \leqslant 4B^2,\end{aligned}$$

where we use the convexity of $\|\cdot\|_*^2$ and Jensen's inequality.

A key updating step in dual averaging methods, the proximal mapping, utilizes the Bregman divergence. Let $\omega(x) : \mathcal{X} \to \mathbb{R}$ be a strongly convex and differentiable function, the Bregman divergence associated with $\omega(x)$ is defined as:

$$V(x,y) := \omega(x) - \omega(y) - \langle \nabla\omega(y), x - y\rangle. \tag{5.6}$$

One typical and simple example is $\omega(x) = \frac{1}{2}\|x\|_2^2$ together with $V(x,y) = \frac{1}{2}\|x-y\|_2^2$. One may refer to [148] for more examples. We can always scale $\omega(x)$ so that $V(x,y) \geqslant \frac{1}{2}\|x-y\|^2$ for all $x, y \in \mathcal{X}$. Following the assumption in [52]: we assume that $V(x,y)$ *grows quadratically* with the parameter $\tau > 1$, i.e., $V(x,y) \leqslant \frac{\tau}{2}\|x-y\|^2$ with $\tau > 1$ for all $x, y \in \mathcal{X}$. In fact, we could simply choose $\omega(x)$ with a τ-Lipschitz continuous gradient so that the quadratic growth assumption will be automatically satisfied.

Furthermore, we define $\mu = \frac{\widetilde{\mu}}{\tau}$, which scales the strong convexity parameter $\widetilde{\mu}$ by $\frac{1}{\tau}$, where τ is the quadratic growth constant. Therefore, for any $x, y \in \mathcal{X}$, $\mu V(x,y) \leqslant \frac{\widetilde{\mu}}{2}\|x-y\|^2$.

Algorithm 5.1 Optimal Regularized Dual Averaging Method: ORDA(x_0, N, Γ, c)

Input Parameters: Starting point $x_0 \in \mathcal{X}$, the number of iterations N, constants $\Gamma \geqslant L$ and $c \geqslant 0$.
Parameters for $f(x)$**:** Constants L, M and $\widetilde{\mu}$ for $f(x)$ in (5.2) and set $\mu = \widetilde{\mu}/\tau$.
Initialization: Set $\theta_t = \frac{2}{t+2}$; $\nu_t = \frac{2}{t+1}$; $\gamma_t = c(t+1)^{3/2} + \tau\Gamma$; $z_0 = x_0$.
Iterate for $t = 0, 1, 2, \ldots, N$:

1. $y_t = \frac{(1-\theta_t)(\mu+\theta_t^2\gamma_t)}{\theta_t^2\gamma_t+(1-\theta_t^2)\mu} x_t + \frac{(1-\theta_t)\theta_t\mu+\theta_t^3\gamma_t}{\theta_t^2\gamma_t+(1-\theta_t^2)\mu} z_t$
2. Sample ξ_t from the distribution $P(\xi)$ and compute the stochastic subgradient $G(y_t, \xi_t)$.
3. $g_t = \theta_t\nu_t \left(\sum_{i=0}^{t} \frac{G(y_i,\xi_i)}{\nu_i}\right)$
4. $z_{t+1} = \arg\min_{x\in\mathcal{X}} \left\{\langle x, g_t\rangle + h(x) + \theta_t\nu_t \left(\sum_{i=0}^{t} \frac{\mu V(x,y_i)}{\nu_i}\right) + \theta_t\nu_t\gamma_{t+1}V(x, x_0)\right\}$
5. $x_{t+1} = \arg\min_{x\in\mathcal{X}} \left\{\langle x, G(y_t, \xi_t)\rangle + h(x) + \left(\frac{\mu}{\tau\theta_t^2} + \frac{\gamma_t}{\tau}\right) V(x, y_t)\right\}$

Output: x_{N+1}

5.3 OPTIMAL REGULARIZED DUAL AVERAGING METHOD

In dual averaging methods [111, 148], the key proximal mapping step utilizes the average of all past stochastic subgradients to update the parameter vector. In particular, it takes the form:

$$z_{t+1} = \arg\min_{x\in\mathcal{X}} \Big\{\langle g_t, x\rangle + h(x) + \frac{\beta_t}{t} V(x, x_0)\Big\},$$

where β_t is the step-size and

$$g_t = \frac{1}{t+1}\sum_{i=0}^{t} G(z_i, \xi_i).$$

For strongly convex $f(x)$, the current dual averaging methods achieve a rate of $O(\frac{\sigma^2 \log N}{\widetilde{\mu} N})$, which is suboptimal. In this section, we propose a new dual averaging algorithm which adapts to both strongly and non-strongly convex $f(x)$ via the strong convexity parameter $\widetilde{\mu}$ and achieves optimal rates in both cases. In addition, for previous dual averaging methods, to guarantee the convergence, the final solution takes the form: $\widehat{x} = \frac{1}{N+1}\sum_{t=0}^{N} z_t$ and hence is not sparse in nature for sparsity-inducing regularizers. Instead of taking the average, we introduce another proximal mapping and generate the final solution directly from the second proximal mapping. This strategy will provide us sparser solutions in practice. It is worthy to note that in RDA, z_N has been proved to achieve the desirable sparsity pattern (i.e., manifold identification property) [86]. However, according to [86], the convergence of $\phi(z_N)$ to the optimal $\phi(x^*)$ is established only under a more restrictive assumption that x^* is a strong local minimizer of ϕ relative to the optimal manifold and the convergence rate

is quite slow. Without this assumption, the convergence of $\phi(z_N)$ is still unknown.

The proposed optimal RDA (ORDA) method is presented in Algorithm 5.1. In general, the constant Γ which defines the step-size parameter γ_t is set to L. However, we allow Γ to be an arbitrary constant greater than or equal to L to facilitate the introduction of the multi-stage ORDA in the later section. The parameter c is set to achieve the optimal rates for both convex and strongly convex loss. When $\mu > 0$ (or equivalently, $\widetilde{\mu} > 0$), c is set to 0 so that $\gamma_t \equiv \tau\Gamma \geqslant \tau L$; while for $\mu = 0$, $c = \frac{\sqrt{\tau}(\sigma+M)}{2\sqrt{V(x^*,x_0)}}$. Since x^* is unknown in practice, one might replace $V(x^*, x_0)$ in c by a tuning parameter.

Here, we make a few more explanations of Algorithm 5.1. In Step 1, the intermediate point y_t is a convex combination of x_t and z_t and when $\mu = 0$, $y_t = (1-\theta_t)x_t + \theta_t z_t$. The choice of the combination weights is inspired by [52]. Second, with our choice of θ_t and ν_t, it is easy to prove that $\sum_{i=0}^{t} \frac{1}{\nu_i} = \frac{1}{\theta_t \nu_t}$. Therefore, g_t in Step 3 is a convex combination of $\{G(y_i, \xi_i)\}_{i=0}^{t}$. As compared to RDA which uses the average of past subgradients, g_t in ORDA is a *weighted average* of all past stochastic subgradients and the subgradient from the larger iteration has a larger weight (i.e., $G(y_i, \xi_i)$ has the weight $\frac{2(i+1)}{(t+1)(t+2)}$). In practice, instead of storing all past stochastic subgradients, g_t could be simply updated based on g_{t-1}:

$$g_t = \theta_t \nu_t \left(\frac{g_{t-1}}{\theta_{t-1}\nu_{t-1}} + \frac{G(y_t, \xi_t)}{\nu_t} \right).$$

We also note that since the error in the stochastic subgradient $G(y_t, \xi_t)$ will affect the sparsity of x_{t+1} via the second proximal mapping, to obtain stable sparsity recovery performances, it would be better to construct the stochastic subgradient with a small batch of samples [148, 35]. This could help to reduce the noise of the stochastic subgradient.

5.3.1 *Convergence Rate*

We present the convergence rate for ORDA. We start by presenting a general theorem without plugging the values of the parameters. To simplify our notations, we define $\Delta_t := G(y_t, \xi_t) - f'(y_t)$.

Theorem 5.1. *For ORDA, if we require $c > 0$ when $\widetilde{\mu} = 0$, then for any $t \geqslant 0$: the gap between $\phi(x_{t+1})$ and $\phi(x^*)$ can be characterized by:*

$$\begin{aligned} &\phi(x_{t+1}) - \phi(x^*) \qquad (5.7)\\ &\leqslant \theta_t \nu_t \gamma_{t+1} V(x^*, x_0) + \frac{\theta_t \nu_t}{2} \sum_{i=0}^{t} \frac{(\|\Delta_i\|_* + M)^2}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i \gamma_i}{\tau} - \theta_i L\right)\nu_i} + \theta_t \nu_t \sum_{i=0}^{t} \frac{\langle x^* - \widehat{z}_i, \Delta_i \rangle}{\nu_i}, \end{aligned}$$

where

$$\widehat{z}_t = \frac{\theta_t \mu}{\mu + \gamma_t \theta_t^2} y_t + \frac{(1-\theta_t)\mu + \gamma_t \theta_t^2}{\mu + \gamma_t \theta_t^2} z_t, \qquad (5.8)$$

is a convex combination of y_t *and* z_t*; and* $\widehat{z}_t = z_t$ *when* $\mu = 0$*. Taking the expectation on both sides of* (5.7)*:*

$$\mathbb{E}\phi(x_{t+1}) - \phi(x^*) \leqslant \theta_t \nu_t \gamma_{t+1} V(x^*, x_0) + (\sigma^2 + M^2)\theta_t \nu_t \sum_{i=0}^{t} \frac{1}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i \gamma_i}{\tau} - \theta_i L\right)\nu_i}. \quad (5.9)$$

The proof of Theorem 5.1 is given in Appendix. In the next two corollaries, we establish the rates of convergence in expectation for ORDA by choosing different values for c based on $\widetilde{\mu}$.

Corollary 5.1. *For convex* $f(x)$ *with* $\widetilde{\mu} = 0$ *, by setting* $c = \frac{\sqrt{\tau}(\sigma+M)}{2\sqrt{V(x^*,x_0)}}$ *and* $\Gamma = L$*, we obtain:*

$$\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \leqslant \frac{4\tau L V(x^*, x_0)}{N^2} + \frac{8(\sigma + M)\sqrt{\tau V(x^*, x_0)}}{\sqrt{N}}. \quad (5.10)$$

Based on (5.9), the proof of Corollary 5.1 is straightforward with the details in Appendix. Since x^* is unknown in practice, one could set c by replacing $V(x^*, x_0)$ in c with any value $D^* \geqslant V(x^*, x_0)$. By doing so, (5.10) remains valid after replacing all $V(x^*, x_0)$ by D^*. For convex $f(x)$ with $\widetilde{\mu} = 0$, the rate in (5.10) has achieved the *uniformly-optimal* rate according to [105]. In fact, if $f(x)$ is a deterministic and smooth function with $\sigma = M = 0$ (e.g., smooth empirical loss), one only needs to change the stochastic subgradient $G(y_t, \xi_t)$ to $\nabla f(y_t)$. The resulting algorithm, which reduces to Algorithm 3 in [138], is an optimal deterministic first-order method with the rate $O(\frac{LV(x^*,x_0)}{N^2})$.

We note that the quadratic growth assumption of $V(x, y)$ is not necessary for convex $f(x)$. If one does not assume this assumption and replaces the last step in ORDA by

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle x, G(y_t, \xi_t) \rangle + h(x) + \left(\frac{\mu}{2\theta_t^2} + \frac{\gamma_t}{2} \right) \|x - y_t\|^2 \right\},$$

we can achieve the same rate as in (5.10) but just removing all τ from the right hand side. But the quadratic growth assumption is indeed required for showing the convergence for strongly convex $f(x)$ as in the next corollary.

Corollary 5.2. *For strongly convex* $f(x)$ *with* $\widetilde{\mu} > 0$*, we set* $c = 0$ *and* $\Gamma = L$ *and obtain that:*

$$\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \leqslant \frac{4\tau L V(x^*, x_0)}{N^2} + \frac{4\tau(\sigma^2 + M^2)}{\mu N}. \quad (5.11)$$

The dominating term in (5.11), $O\left(\frac{1}{\mu N}\right)$, is optimal and better than the $O\left(\frac{\log N}{\mu N}\right)$ rate for previous dual averaging methods. However, ORDA has not achieved the uniformly-optimal rate, which takes the form of $O(\frac{\sigma^2+M^2}{\mu N} + \exp(-\sqrt{\frac{\mu}{L}}N))$. In particular, for deterministic smooth and strongly convex $f(x)$ (i.e., empirical loss with $\sigma = M = 0$), ORDA

only achieves the rate of $O(\frac{L}{N^2})$ while the optimal deterministic rate should be $O\left(\exp(-\sqrt{\frac{\mu}{L}}N)\right)$ [108]. Inspired by the multi-restart technique in [56, 80], we present a multi-stage extension of ORDA in Section 5.4 which achieves the uniformly-optimal convergence rate.

5.3.2 *Mini-batch Strategy and Distributed Computing*

Since our algorithms only utilize the stochastic gradient information, the distributed mini-batch strategy in [37] can be directly applied to further accelerated the algorithms. In particular, we assume for each iteration t, a query of the oracle returns b stochastic gradients $\{G(y_t, \xi_{ti})\}_{i=1}^b$, where b is the mini-batch size and each ξ_{ti} is drawn i.i.d. from P. Let $G(y_t, \xi_t) := \frac{1}{b}\sum_{i=1}^b G(x, \xi_{ti})$ be the average of all these stochastic gradients at y_t. If we replace $G(y_t, \xi_t)$ by $G(y_t, \xi_t)$ in either Algorithm 5.1, it is easy to verify that the variance of the stochastic gradient is reduced by a factor of b, i.e., $\|G(x, \xi_t) - f'(x)\|_*^2 \leqslant \sigma^2/b$, $\forall\, x$. Assume that we can process each query of the oracle (i.e., b stochastic gradients) in a distributed manner with a subsequent vector-sum operation based averaging step. The averaging step can be done via a vector-sum operation based on the minimum-depth spanning-tree of the distributed network topology as in [37]. Then according to Section 5.2 in [37], such a distributed mini-batch strategy achieves an asymptotic optimal (linear) speed-up, i.e., the distributed algorithm is b times faster to achieve the same optimization error than the serial algorithm. Put another way, in the same amount of wall-clock time, serial algorithm achieves optimization error of $O(1/\sqrt{N})$; while distributed algorithm achieves error of $O(1/\sqrt{Nb})$.

5.3.3 *Variance Bounds*

Corollary 5.1 and 5.2 imply that $\phi(x_{N+1})$ converges to $\phi(x^*)$ *on average*. However, it does not provide the accuracy of the solution from a single run of ORDA. We prove that under certain assumptions, the variance of the error of the objective value, $\mathrm{Var}[\phi(x_{N+1}) - \phi(x^*)]$, also converges to zero at a rate of $O(\frac{1}{N})$. Therefore, the error of the objective value will eventually converge to a distribution that concentrates close to zero and we can conclude that any single run of ORDA is reliable if N is large enough.

Theorem 5.2. *We assume that there exists a constant* D *such that* $\|x^* - \widehat{z}_t\| \leqslant D$ *for all* t, *where* $\widehat{z}_t$ *is defined in* (5.8). *We also assume the fourth moment of* $G(x, \xi) - f'(x)$ *is bounded by* σ^4 *for any* $x \in \mathcal{X}$, *i.e.,*

$$\mathbb{E}\|G(x, \xi) - f'(x)\|_*^4 \leqslant \sigma^4. \tag{5.12}$$

Then for any $t \geqslant 0$, the variance of the error is bounded by:

$$\begin{aligned} Var[\phi(x_{t+1}) - \phi(x^*)] \quad \leqslant \quad & 3\theta_t^2 v_t^2 \gamma_{t+1}^2 V(x^*, x_0)^2 \\ & + 6\theta_t^2 v_t^2 (\sigma^4 + M^4) \left(\sum_{i=0}^{t} \frac{1}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i \gamma_i}{\tau} - \theta_i L \right) v_i} \right)^2 \\ & + 3\theta_t^2 v_t^2 D^2 \sigma^2 \sum_{i=0}^{t} \frac{1}{v_i^2}. \end{aligned} \tag{5.13}$$

Based on (5.13)*, for convex $f(x)$ with $\widetilde{\mu} = 0$ and $c = \frac{\sqrt{\tau}(\sigma+M)}{2\sqrt{V(x^*,x_0)}}$, we have:*

$$Var[\phi(x_{N+1}) - \phi(x^*)] \leqslant \frac{96\tau^2 L^2 V(x^*, x_0)^2}{N^4} + \frac{120(\sigma + M)^2 \tau V(x^*, x_0) + 4D^2\sigma^2}{N} \tag{5.14}$$

For strongly convex $f(x)$ with $c = 0$, we have:

$$Var[\phi(x_{N+1}) - \phi(x^*)] \leqslant \frac{48\tau^2 L^2 V(x^*, x_0)^2}{N^4} + \frac{96\tau^2(\sigma^4 + M^4)}{N^2\mu^2} + \frac{4D^2\sigma^2}{N} \tag{5.15}$$

Because z_t and y_t always stay inside $\mathcal{X}$ and $\widehat{z}_t$ is their convex combination, the condition $\|x^* - \widehat{z}_t\| \leqslant D$ for all t is automatically satisfied when $\mathcal{X}$ is bounded and D is the diameter of $\mathcal{X}$. Even if $\mathcal{X}$ is unbounded, we can try to confine the search to a bounded set which contains the optimal solution. We also note that the boundedness of the 4-th moment condition implies our basic assumption on the variance of the stochastic subgradient in (5.3) according to Jensen's inequality, i.e., $\mathbb{E}\|G(x,\xi) - f'(x)\|_*^2 \leqslant (\mathbb{E}\|G(x,\xi) - f'(x)\|_*^4)^{\frac{1}{2}} \leqslant \sigma^2$. In other words, we need a stronger assumption here in order to provide a bound on the variance. The detailed proof is in Appendix.

5.3.4 *High Probability Bounds*

For stochastic optimization problems, another evaluation criterion is the confidence level of the objective value. In particular, it is of great interest to find $\epsilon(N, \delta)$ as a monotonically decreasing function in both N and $\delta \in (0, 1)$ such that the solution x_{N+1} satisfies

$$\Pr\left(\phi(x_{N+1}) - \phi(x^*) \geqslant \epsilon(N, \delta)\right) \leqslant \delta.$$

In other words, we want to show that with probability at least $1 - \delta$, $\phi(x_{N+1}) - \phi(x^*) < \epsilon(N, \delta)$.

According to Markov inequality, for any $\epsilon > 0$, $\Pr(\phi(x_{N+1}) - \phi(x^*) \geqslant \epsilon) \leqslant \frac{\mathbb{E}(\phi(x_{N+1}) - \phi(x^*))}{\epsilon}$. Therefore, we have $\epsilon(N, \delta) = \frac{\mathbb{E}\phi(x_{N+1}) - \phi(x^*)}{\delta}$. Under the basic assumption in (5.3), namely $\mathbb{E}_\xi(\|G(x,\xi) - f'(x)\|_*^2) \leqslant \sigma^2$, and according to Corollary 5.1 and 5.2, $\epsilon(N, \delta) = O\left(\frac{(\sigma+M)\sqrt{V(x^*,x_0)}}{\sqrt{N}\delta}\right)$ for convex $f(x)$, and $\epsilon(N, \delta) = O\left(\frac{\sigma^2+M^2}{\mu N \delta}\right)$ for strongly convex $f(x)$.

However, the above bounds are quite loose. To obtain tighter bounds, we strengthen the basic assumption of the stochastic subgradient in (5.3) to the "light-tail" assumption [106]. In particular, we assume that $\mathbb{E}\left(\exp\left\{\|G(x,\xi) - f'(x)\|_*^2/\sigma^2\right\}\right) \leqslant \exp\{1\}$, $\forall x \in \mathcal{X}$. By further making

the boundedness assumption ($\|x^* - \widehat{z}_t\| \leqslant D$) and utilizing a technical lemma from [42], we obtain a much tighter high probability bound with $\epsilon(N,\delta) = O\left(\frac{\sqrt{\ln(1/\delta)}D\sigma}{\sqrt{N}}\right)$ for both convex and strongly convex $f(x)$.

Theorem 5.3. *We assume that (1)* $\mathbb{E}\left(\exp\left\{\|G(x,\xi) - f'(x)\|_*^2/\sigma^2\right\}\right) \leqslant \exp\{1\}$, $\forall x \in \mathcal{X}$ *(i.e., "light-tail" assumption) and (2) there exists a constant* D *such that* $\|x^* - \widehat{z}_t\| \leqslant D$ *for all* t. *By setting* $\Gamma = L$ *in ORDA, for any iteration* t *and* $\delta \in (0,1)$, *we have, with probability at least* $1-\delta$*:*

$$\phi(x_{t+1}) - \phi(x^*)) \leqslant \epsilon(t,\delta) \tag{5.16}$$

with

$$\begin{aligned}\epsilon(t,\delta) &= \theta_t\nu_t\gamma_{t+1}V(x^*,x_0) + \theta_t\nu_t\sum_{i=0}^{t}\frac{M^2}{\eta_i\nu_i} \\ &\quad + \theta_t\left[\sum_{i=0}^{t}\frac{\sigma^2}{\eta_i} + \frac{8\sigma^2\ln(2/\delta)}{(\frac{\mu+\gamma_0}{\tau}-L)} + 16\sigma^2\sqrt{\sum_{i=0}^{t}\frac{\ln(2/\delta)}{\eta_i^2}}\right] \\ &\quad + \sqrt{3\ln\frac{2}{\delta}}\theta_t\nu_t D\sigma\left(\sum_{i=0}^{t}\frac{1}{\nu_i^2}\right)^{1/2},\end{aligned} \tag{5.17}$$

where $\eta_i = \left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i\gamma_i}{\tau} - \theta_i L\right)$.

For convex $f(x)$ *with* $\widetilde{\mu} = 0$, *by setting* $c = \frac{\sqrt{\tau}(\sigma+M)}{2\sqrt{V(x^*,x_0)}}$ *and* $\Gamma = L$, *we have*

$$\begin{aligned}\epsilon(N,\delta) &= \frac{4\tau LV(x^*,x_0)}{N^2} + \frac{24\sqrt{\tau V(x^*,x_0)(\sigma+M)}}{\sqrt{N}} + \frac{16\ln(2/\delta)\sqrt{\tau V(x^*,x_0)}\sigma}{N} \\ &\quad + \frac{16\sigma\sqrt{\ln(2/\delta)\ln(N+3)V(x^*,x_0)}}{N} + \frac{2\sqrt{\ln(2/\delta)}D\sigma}{\sqrt{N}}.\end{aligned} \tag{5.18}$$

For convex $f(x)$ *with* $\widetilde{\mu} > 0$ *(or equivalently* $\mu > 0$*), by setting* $c = 0$ *and* $\Gamma = L$, , *we have*

$$\begin{aligned}\epsilon(N,\delta) &= \frac{4\tau LV(x^*,x_0)}{N^2} + \frac{16\tau(\sigma^2+M^2)\ln(N+2)}{\mu N} \\ &\quad + \frac{48\sigma^2\ln(2/\delta)}{\mu N} + \frac{2\sqrt{\ln(2/\delta)}D\sigma}{\sqrt{N}}.\end{aligned} \tag{5.19}$$

By making stronger assumptions as in Theorem 5.3, we obtain much tighter bounds with $\epsilon(N,\delta) = O\left(\frac{\sqrt{\ln(1/\delta)}D\sigma}{\sqrt{N}}\right)$ in (5.18) and (5.19) as compared to the simple bounds obtained by Markov inequality. The proof of Theorem 5.3 is presented in Appendix.

5.4 MULTI-STAGE ORDA FOR STOCHASTIC STRONGLY CONVEX OPTIMIZATION

As we show in Section 5.3.1, for convex $f(x)$, ORDA achieves the uniformly-optimal rate. However, for strongly convex $f(x)$, although the dominating term of the convergence rate in (5.11) is optimal,

Algorithm 5.2 Multi-stage ORDA for Stochastic Strongly Convex Optimization

Initialization: $x_0 \in \mathcal{X}$, a constant $\mathcal{V}_0 \geqslant \phi(x_0) - \phi(x^*)$ and the number of stages K.

Iterate for $k = 1, 2, \ldots, K$:

1. Set $N_k = \max\left\{4\sqrt{\frac{\tau L}{\mu}}, \frac{2^{k+9}\tau(\sigma^2+M^2)}{\mu\mathcal{V}_0}\right\}$
2. Set $\Lambda_k = N_k^{3/2}\sqrt{\frac{2^{k-1}\mu(\sigma^2+M^2)}{\tau\mathcal{V}_0}}$
3. Generate $\widetilde{x}_k$ by calling the sub-routine ORDA$(\widetilde{x}_{k-1}, N_k, \Gamma = \Lambda_k + L, c = 0)$

Output: $\widetilde{x}_K$

the overall rate is not uniformly-optimal. Inspired by the multi-stage stochastic approximation methods [56, 68, 80], we propose the multi-stage extension of ORDA in Algorithm 5.2 for stochastic strongly convex optimization. For each stage $1 \leqslant k \leqslant K$, we run ORDA in Algorithm 5.1 as a sub-routine for N_k iterations with the parameter $\gamma_t = c(t+1)^{3/2} + \tau\Gamma$ with $c = 0$ and $\Gamma = \Lambda_k + L$. Roughly speaking, we set $N_k = 2N_{k-1}$ and $\Lambda_k = 4\Lambda_{k-1}$. In other words, we double the number of iterations for the next stage but reduce the step-size. The multi-stage ORDA has achieved uniformly-optimal convergence rate as shown in Theorem 5.4 with the proof in Appendix. The proof technique follows the one in [80]. Due this specialized proof technique, instead of showing $\mathbb{E}(\phi(x_N)) - \phi(x^*) \leqslant \epsilon(N)$ as in ORDA, we show the number of iterations $N(\epsilon)$ to achieve the ϵ-accurate solution: $\mathbb{E}(\phi(x_{N(\epsilon)})) - \phi(x^*) \leqslant \epsilon$. But the two convergence rates are equivalent.

Theorem 5.4. *If we run multi-stage ORDA for K stages with $K = \log_2\left(\frac{\mathcal{V}_0}{\epsilon}\right)$ for any given ϵ, we have $\mathbb{E}(\phi(\widetilde{x}_K)) - \phi(x^*) \leqslant \epsilon$ and the total number of iterations is upper bounded by:*

$$N = \sum_{k=1}^{K} N_k \leqslant 4\sqrt{\frac{\tau L}{\mu}}\log_2\left(\frac{\mathcal{V}_0}{\epsilon}\right) + \frac{1024\tau(\sigma^2+M^2)}{\mu\epsilon}. \tag{5.20}$$

5.5 RELATED WORKS

In the last few years, a number of stochastic gradient methods [39, 41, 61, 82, 106, 148, 52, 80, 56, 40, 42] have been developed to solve (5.1), especially for a sparsity-inducing $h(x)$. In Table 5.1, we compare the proposed ORDA and its multi-stage extension with some widely used stochastic gradient methods using the following metrics. For the ease of comparison, we assume $f(x)$ is smooth with $M = 0$.

1. The convergence rate for solving (non-strongly) convex $f(x)$ and whether this rate has achieved the uniformly-optimal (Uni-opt) rate.

Table 5.1: Summary for different stochastic gradient algorithms. V is short for $V(x^*, x_0)$; AC for "accelerated"; M for "multi-stage" and NA stands for either "not applicable" or "no analysis of the rate".

	Convex $f(x)$		Strongly Convex $f(x)$			Final $\widehat{x}$	Bregman
	Rate	Uni-opt	Rate	Opt	Uni-opt		
FOBOS [39]	$O\left(\frac{G\sqrt{V}}{\sqrt{N}}\right)$	NO	$O\left(\frac{G^2 \log N}{\widetilde{\mu} N}\right)$	NO	NO	Prox	NO
COMID [41]	$O\left(\frac{G\sqrt{V}}{\sqrt{N}}\right)$	NO	$O\left(\frac{G^2 \log N}{\widetilde{\mu} N}\right)$	NO	NO	Prox	YES
SAGE [61]	$O\left(\frac{\sigma\sqrt{D}}{\sqrt{N}} + \frac{LD}{N^2}\right)$	NEARLY	$O\left(\frac{\sigma^2}{\widetilde{\mu} N} + \frac{LD}{N^2}\right)$	YES	NO	Prox	NO
AC-SA [52]	$O\left(\frac{\sigma\sqrt{V}}{\sqrt{N}} + \frac{LV}{N^2}\right)$	YES	$O\left(\frac{\sigma^2}{\widetilde{\mu} N} + \frac{LV}{N^2}\right)$	YES	NO	Avg	YES
M-AC-SA [80]	NA	NA	$O\left(\frac{\sigma^2}{\widetilde{\mu} N} + \exp\{-\sqrt{\frac{\widetilde{\mu}}{L}} N\}\right)$	YES	YES	Avg	YES
Epoch-GD [56]	NA	NA	$O\left(\frac{G^2}{\widetilde{\mu} N}\right)$	YES	NO	Avg	NO
RDA [148]	$O\left(\frac{G\sqrt{V}}{\sqrt{N}}\right)$	NO	$O\left(\frac{G^2 \log N}{\widetilde{\mu} N}\right)$	NO	NO	Avg	YES
AC-RDA [148]	$O\left(\frac{\sigma\sqrt{V}}{\sqrt{N}} + \frac{LV}{N^2}\right)$	YES	NA	NA	NA	Avg	YES
ORDA	$O\left(\frac{\sigma\sqrt{V}}{\sqrt{N}} + \frac{LV}{N^2}\right)$	YES	$O\left(\frac{\sigma^2}{\widetilde{\mu} N} + \frac{LV}{N^2}\right)$	YES	NO	Prox	YES
M-ORDA	NA	NA	$O\left(\frac{\sigma^2}{\widetilde{\mu} N} + \exp\{-\sqrt{\frac{\widetilde{\mu}}{L}} N\}\right)$	YES	YES	Prox	YES

2. The convergence rate for solving strongly convex $f(x)$ and whether (1) the dominating term of rate is optimal, i.e., $O\left(\frac{\sigma^2}{\widetilde{\mu} N}\right)$ and (2) the overall rate is uniformly-optimal.

3. Whether the final solution $\widehat{x}$, on which the results of convergence are built, is generated from the weighted average of previous iterates (Avg) or from the proximal mapping (Prox). For sparsity-inducing regularizers, the solution directly from the proximal mapping is often sparser than the averaged solution.

4. Whether an algorithm allows to use a general Bregman divergence in proximal mapping or it only allows the Euclidean distance $V(x, y) = \frac{1}{2}\|x - y\|_2^2$.

In Table 5.1, the algorithms in the first 7 rows are stochastic approximation algorithms where only the current stochastic gradient is used at each iteration. The last 4 rows are dual averaging methods where all past subgradients are used. Some algorithms in Table 5.1 make a more restrictive assumption on the stochastic gradient: $\exists G > 0, \mathbb{E}\|G(x, \xi)\|_*^2 \leqslant G^2, \forall x \in \mathcal{X}$. It is easy to verify that this assumption implies our basic assumption in (5.3) by Jensen's inequality.

As we can see from Table 5.1, the proposed ORDA possesses all good properties except that the convergence rate for strongly convex $f(x)$ is not uniformly-optimal. Multi-stage ORDA further improves this rate to be uniformly-optimal. In particular, SAGE [61] achieves a nearly optimal rate since the parameter D in the convergence rate is chosen such that $\mathbb{E}\left(\|x_t - x^*\|_2^2\right) \leqslant D$ for all $t \geqslant 0$ and it could be much larger than $V \equiv V(x^*, x_0)$. In addition, SAGE requires the boundedness of the domain $\mathcal{X}$, the smoothness of $f(x)$, and only allows the Euclidean distance in proximal mapping. As compared to

AC-SA [52] and multi-stage AC-SA [80], our methods do not require the final averaging step; and as shown in our experiments, ORDA has better empirical performances due to the usage of all past stochastic subgradients. Furthermore, we improve the rates of RDA and extend AC-RDA to an optimal algorithm for both convex and strongly convex $f(x)$. Another highly relevant work is [68]. Juditsky et al. [68] proposed multi-stage algorithms to achieve the optimal strongly convex rate based on non-accelerated dual averaging methods. However, the algorithms in [68] assume that $\phi(x)$ is a Lipschitz continuous function, i.e., the subgradient of $\phi(x)$ is bounded. Therefore, when the domain $\mathcal{X}$ is unbounded, the algorithms in [68] cannot be directly applied. Recently, the paper [115] develops another stochastic gradient method which achieves the rate $O(\frac{G^2}{\widetilde{\mu}N})$ for strongly convex $f(x)$. However, for non-smooth $f(x)$, it requires the averaging of the last a few iterates and this rate is not uniformly-optimal.

5.6 EXPERIMENTS

In this section, we conduct simulated experiments to demonstrate the performance of ORDA and its multi-stage extension (M_ORDA). We compare our ORDA and M_ORDA (only for strongly convex loss) with several state-of-the-art stochastic gradient methods, including RDA and AC-RDA [148], AC-SA [52], FOBOS [39] and SAGE [61]. For a fair comparison, we compare all different methods using solutions which have expected convergence guarantees. For all algorithms, we tune the parameter related to step-size (e.g., c in ORDA for convex loss) within an appropriate range and choose the one that leads to the minimum objective value.

5.6.1 *Simulated Experiments*

In this experiment, we solve a sparse linear regression problem:

$$\min_{x\in\mathbb{R}^n} f(x) + h(x),$$

where

$$f(x) = \frac{1}{2}\mathbb{E}_{a,b}((a^T x - b)^2) + \frac{\rho}{2}\|x\|_2^2,$$

and $h(x) = \lambda\|x\|_1$. The input vector a is generated from $N(0, I_{n\times n})$ and the response $b = a^T x^* + \epsilon$, where $x_i^* = 1$ for $1 \leqslant i \leqslant n/2$ and 0 otherwise and the noise $\epsilon \sim N(0,1)$. When $\rho = 0$, the problem is the well known Lasso [133] and when $\rho > 0$, it is known as Elastic-net [162]. The regularization parameter λ is tuned so that a deterministic solver on all the samples can correctly recover the underlying sparsity pattern. We set $n = 100$ and create a large pool of samples for generating stochastic gradients and evaluating objective values. The number of iterations N is set to 500. We note that it is fair to run each algorithm for $N = 500$ iterations for the comparison since all the competitors in this experiment are stochastic first-other methods. In particular, in every iteration, each algorithm receives one new data

	Obj (std)	F1-score (std)
RDA	2.087e+1 (2.760e-2)	0.67 (0.00)
AC-RDA	2.067e+1 (3.144e-2)	0.67 (0.00)
AC-SA	2.066e+1 (1.661e-2)	0.67 (0.00)
FOBOS	2.098e+1 (3.151e-2)	0.83 (0.02)
SAGE	2.065e+1 (3.162e-2)	0.82 (0.02)
ORDA	**2.056e+1 (1.761e-2)**	**0.92 (0.02)**

Table 5.2: Comparisons for different algorithms in objective value and F1-score for solving Lasso problem.

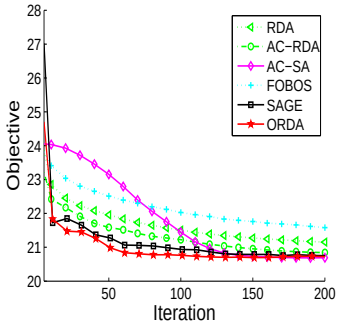


Figure 5.1: Objective values v.s. Iterations. Only the first 200 iterations are plotted for better visualization and the ease of comparisons.

	Obj (std)	F1-score (std)
RDA	2.157e+1 (2.998e-2)	0.67 (0.00)
AC-RDA	2.112e+1 (2.525e-2)	0.67 (0.00)
AC-SA	2.101e+1 (8.306e-3)	0.67 (0.00)
FOBOS	2.119e+1 (2.216e-3)	0.84 (0.02)
SAGE	2.109e+1 (4.749e-3)	0.73 (0.02)
ORDA	**2.097e+1** (6.248e-3)	**0.87 (0.02)**
M_ORDA	**2.098e+1** (6.248e-3)	**0.88 (0.02)**

Table 5.3: Comparisons for different algorithms in objective and F1-score for solving Elastic-net problem.

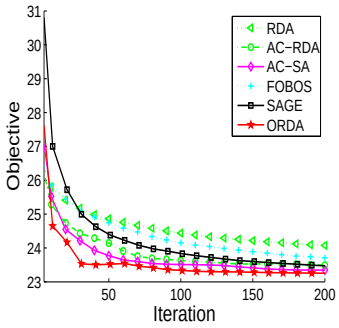


Figure 5.2: Objective values v.s. Iterations. Only the first 200 iterations are plotted for better visualization and the ease of comparisons.

sample from the underlying distribution and main per-iteration computational cost are the same (i.e., computing the stochastic gradient).

Since we focus on stochastic optimization instead of online learning, we could randomly draw samples from an underlying distribution. So we construct the stochastic gradient using the mini-batch strategy [37, 35] with the batch size 50. We run each algorithm for 100 times and report the mean of the objective value and the F1-score for sparsity recovery performance. F1-score is defined as $2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$ where

$$\text{precision} = \sum_{i=1}^{p} \mathbf{1}_{\{\widehat{x}_i=1, x_i^*=1\}} / \sum_{i=1}^{p} \mathbf{1}_{\{\widehat{x}_i=1\}}$$

and

$$\text{recall} = \sum_{i=1}^{p} \mathbf{1}_{\{\widehat{x}_i=1, x_i^*=1\}} / \sum_{i=1}^{p} \mathbf{1}_{\{x_i^*=1\}}.$$

The higher the F1-score is, the better the recovery ability of the sparsity pattern.

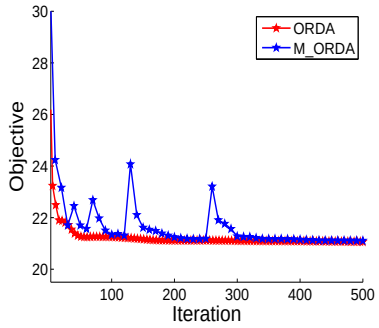


Figure 5.3: ORDA v.s. M_ORDA.

Table 5.4: The statistics of the experimental datasets.

	# Training Samples (m)	# Testing Samples	# Features (n)
MNIST (6 vs 7)	12,183	1,986	784
20 Newsgroup	1,027	400	17,390

We first set $\rho = 0$ to test algorithms for (non-strongly) convex $f(x)$. The result is presented in Table 5.2 (the first two columns). We also plot the decrease of the objective values for the first 200 iterations in Figure 5.1. From Table 5.2, ORDA performs the best in both objective value and recovery ability of sparsity pattern. For those optimal algorithms (e.g., AC-RDA, AC-SA, SAGE, ORDA), they achieve lower final objective values and the rates of the decrease are also faster. We note that for dual averaging methods, the solution generated from the (first) proximal mapping (e.g., z_t in ORDA) has almost perfect sparsity recovery performance. However, since here is no convergence guarantee for that solution, we do not report results here.

Then we set $\rho = 1$ to test algorithms for solving strongly convex $f(x)$. The results are presented in Table 5.2 (the last two columns) and Figure 5.2 and 5.3. As we can see from Table 5.2, ORDA and M_ORDA perform the best. Although M_ORDA achieves the theoretical uniformly-optimal convergence rate, the empirical performance of M_ORDA is almost identical to that of ORDA. This observation is consistent with our theoretical analysis since the improvement of the convergence rate only appears on the non-dominating term. In addition, ORDA, M_ORDA, AC-SA and SAGE with the convergence rate $O(\frac{1}{\bar{\mu}N})$ achieve lower objective values as compared to other algorithms with the rate $O(\frac{\log N}{\bar{\mu}N})$. For better visualization, we do not include the comparison between M_ORA and ORDA in Figure 5.2. Instead, we present the comparison separately in Figure 5.3. From Figure 5.3, the final objective values of both algorithms are very close. An interesting observation is that, for M_ORDA, each time when a new stage starts, it leads to a sharp increase in the objective value following by a quick drop.

Table 5.5: Experimental results for MNIST in terms of objective value, density of the final solution and testing error.

MNIST	Obj	Density(%)	Err(%)
RDA	570.26	62.50	0.106
AC-RDA	527.95	71.42	0.098
AC-SA	514.40	80.73	0.106
FOBOS	560.73	53.44	0.123
SAGE	547.54	55.86	0.114
ORDA	**497.62**	**41.70**	**0.065**

Table 5.6: Experimental results for 20-newsgroup in terms of objective value, density of the final solution and testing error.

20 Newsgroups	Obj	Density(%)	Err(%)
RDA	272.07	43.42	3.310
AC-RDA	252.36	68.36	3.213
AC-SA	249.13	89.39	3.213
FOBOS	281.54	33.99	3.408
SAGE	232.56	33.15	3.310
ORDA	**229.30**	**22.23**	**2.823**

5.6.2 *Real Data Experiments*

In this section, we compare different stochastic gradient methods for binary classification task using two real datasets: MNIST data for digital recognition[2] and 20-newsgroup for document categorization[3]. The MNIST data consists of images for digits from 0 to 9, where each image is represented by a $28 \times 28 = 784$ gray-scale pixel-map. We normalize the value of each gray-scale by dividing 255 so that each feature value is between 0 and 1. Following the original RDA paper in [148], we construct a binary classification task of distinguishing between digit 6 and 7. For the 20-newsgroup, we classify the postings from two related newsgroups *alt.atheism* and *talk.religion.misc* using the tf-idf of the vocabulary as features. The summary statistics of the data are presented in Table 5.4. We note that two datasets are qualitatively very different: for MNIST, the number of features is more than the number of samples; while for 20-newsgroup, there are many more features than samples. We train the classifier via the sparse logistic regression:

$$\min_{x \in \mathbb{R}^n} \underbrace{m\mathbb{E}_{a,b} \log(1 + \exp(-b(a^T x)))}_{f(x)} + \underbrace{\lambda \|x\|_1}_{h(x)}, \tag{5.21}$$

where $a \in \mathbb{R}^n$ denotes the sample, $b \in \{-1, +1\}$ is the class label and m is the total number of training samples. We multiply the expected

2 http://yann.lecun.com/exdb/mnist/

3 http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 5.7: Speed-up for distributed implementation

No. of Cores	4	8	12	24	48
Speed-up	1	1.62	2.16	2.71	3.27

loss by m to avoid a too small objective value. The parameter λ is set to 10 for MNIST and 0.1 for 20-newsgroup to achieve reasonable amount of sparsity. We run each algorithm for 500 iterations with the batch size 50 and report the objective value, density of the final solution and testing error in Table 5.5 and Table 5.6. As we can see, on both datasets, ORDA achieves the best performance in all three different evaluation metrics. DP-SA leads to the second smallest testing error. Uniformly-optimal methods achieve smaller objective values and testing errors as compared to those non-uniformly-optimal methods.

5.7 MORE DISCUSSIONS ON SCALABILITY ISSUE AND DISTRIBUTED IMPLEMENTATION

We note that in our experiments, the scale of the datasets are relatively small and thus deterministic optimization algorithms (e.g., glmnet [47], FISTA [6]) can be applied and lead to faster convergence. However, for some large-scale datasets which can not fit into memory or for some online data (e.g., twitter data or search log which are collected at a real time), the deterministic algorithms cannot be easily applied. On the other hand, for our methods, since each iteration only utilizes one (or a few) data samples, they can be easily distributed and hence applied to large-scale datasets with millions of samples and features or streaming data.

Another advantage for our methods is that there is a simple distributed implementation using the technique from [37]. Since our algorithms only utilize the stochastic gradient, the distributed mini-batch strategy in [37] can be directly applied to further accelerate the algorithms. In particular, we assume for each iteration t, a query of the stochastic oracle returns b stochastic gradients $\{G(x, \xi_{ti})\}_{i=1}^{b}$, where b is the mini-batch size and each ξ_{ti} is drawn *i.i.d.* from P. Let $G(x, \xi_t) := \frac{1}{b}\sum_{i=1}^{b} G(x, \xi_{ti})$ be the average of all these stochastic gradients. If we replace $G(x, \xi_t)$ by $G(x, \xi_t)$ in either ORDA, it is easy to verify that the variance of the stochastic gradient is reduced by a factor of b, i.e., $\mathbb{E}_\xi\left(\|G(x, \xi_t) - f'(x)\|_*^2\right) \leqslant \sigma^2/b$, $\forall\, x \in \mathcal{X}$. Assume that we can process each query of the oracle (i.e., b stochastic gradients) in a distributed manner with a subsequent averaging step based on the vector-sum operation. Then according to Section 5.2 in [37], such a distributed mini-batch strategy achieves an asymptotic optimal (linear) speed-up. Put another way, in the same amount of wall-clock time, the serial ORDA achieves an optimization error of $O(1/\sqrt{N})$; while a distributed implementation achieves an error of $O(1/\sqrt{Nb})$. We tested the distributed implementation on a web-spam classification dataset [144] using the NIMLE distributed computing toolkit developed by IBM [53]. The dataset contains 350,000 samples with about

16.6 million of tri-gram features and we use the ℓ_1-regularized logistic regression as the objective function. The algorithm is implemented on a 12 machine cluster with 4 cores on each machine. In Table 5.7, we report the speed-up by varying the number of cores (as compared to using a single machine with 4 cores). As we can see, the distributed implementation indeed leads to significant improvement on the computing time. However, it is still far from the linear speed-up. This is mainly due to communication latency time among different machines and some other overhead (e.g., pre-processing).

5.8 APPENDIX: TECHNICAL PROOFS

Proof of Theorem 5.1

We first state a basic property for Bregman distance functions in the following Proposition. This proposition generalizes Lemma 1 in [81] by extending one distance function to a sequence of functions.

Proposition 5.1. *Given any proper lsc convex function $\psi(x)$ and a sequence of $\{z_i\}_{i=0}^t$ with each $z_i \in \mathcal{X}$, if $z_+ = arg\,min_{x\in\mathcal{X}} \left\{\psi(x) + \sum_{i=0}^t \eta_i V(x, z_i)\right\}$, where $\{\eta_i \geqslant 0\}_{i=0}^t$ is a sequence of parameters, then $\forall x \in \mathcal{X}$:*

$$\psi(x) + \sum_{i=0}^{t} \eta_i V(x, z_i) \geqslant \psi(z_+) + \sum_{i=0}^{t} \eta_i V(z_+, z_i) + \left(\sum_{i=0}^{t} \eta_i\right) V(x, z_+). \tag{5.22}$$

Proof of Proposition 5.1. For a Bregman distance function $V(x, y)$, let $\nabla_1 V(x, y)$ denote the gradient of $V(\cdot, y)$ at the point x. It is easy to show that:

$$V(x, y) \equiv V(z, y) + \langle \nabla_1 V(z, y), x - z\rangle + V(x, z), \qquad \forall\, x, y, z \in \mathcal{X},$$

which further implies that:

$$\begin{aligned} &\sum_{i=0}^{t} \eta_i V(x, z_i) \\ &= \sum_{i=0}^{t} \eta_i V(z_+, z_i) + \sum_{i=0}^{t} \eta_i \langle \nabla_1 V(z_+, z_i), x - z_+\rangle + \left(\sum_{i=0}^{t} \eta_i\right) V(x, z_+). \end{aligned} \tag{5.23}$$

Since z_+ is the minimizer of the convex function $\psi(x) + \sum_{i=0}^t \eta_i V(x, z_i)$, it is known that there exists a subgradient g of ψ at z_+ ($g \in \partial\psi(z_+)$) such that:

$$\langle g + \sum_{i=0}^{t} \eta_i \nabla_1 V(z_+, z), x - z_+\rangle \geqslant 0 \qquad \forall x \in \mathcal{X}. \tag{5.24}$$

Using the above two relations and the definition of subgradient ($\psi(x) \geqslant \psi(z_+) + \langle g, x - z_+ \rangle$ for all $x \in \mathcal{X}$), we conclude that:

$$\begin{aligned}
& \psi(x) + \sum_{i=0}^{t} \eta_i V(x, z_i) \\
\geqslant\ & \psi(z_+) + \sum_{i=0}^{t} \eta_i V(z_+, z_i) \\
& + \langle g + \sum_{i=0}^{t} \eta_i \nabla_1 V(z_+, z_i), x - z_+ \rangle + \left(\sum_{i=0}^{t} \eta_i \right) V(x, z_+) \\
\geqslant\ & \psi(z_+) + \sum_{i=0}^{t} \eta_i V(z_+, z_i) + \left(\sum_{i=0}^{t} \eta_i \right) V(x, z_+).
\end{aligned}$$

□

To better present the proof of Theorem 5.1, we denote $G(y_t, \xi_t)$ by $G(y_t)$ and define:

$$\Delta_t := G(y_t) - f'(y_t) = G(y_t, \xi_t) - f'(y_t) \tag{5.25}$$

We first show some basic properties Δ_t. Let $\xi_{[t]}$ denote the collection of *i.i.d.* random vectors $\{\xi_i\}_{i=0}^{t}$. Since both random vectors y_t and z_t are functions of $\xi_{[t-1]}$ and are independent of $\{\xi_i\}_{i=t}^{N}$, we have that for any $t \geqslant 1$ and any α, β:

$$\mathbb{E}\Delta_t = \mathbb{E}_{\xi_{[t-1]}}[\mathbb{E}_{\xi_t}(\Delta_t | \xi_{[t-1]})] = \mathbb{E}_{\xi_{[t-1]}} 0 = 0; \tag{5.26}$$

$$\mathbb{E}\|\Delta_t\|_*^2 = \mathbb{E}_{\xi_{[t-1]}}[\mathbb{E}_{\xi_t}(\|\Delta_t\|_*^2 | \xi_{[t-1]})] \leqslant \mathbb{E}_{\xi_{[t-1]}} \sigma^2 = \sigma^2; \tag{5.27}$$

$$\begin{aligned}
\mathbb{E}\langle \alpha y_t + \beta z_t, \Delta_t \rangle &= \mathbb{E}_{\xi_{[t-1]}}[\langle \alpha y_t + \beta z_t, \mathbb{E}_{\xi_t} \Delta_t \rangle | \xi_{[t-1]}] \\
&= \mathbb{E}_{\xi[t-1]}[\langle \alpha y_t + \beta z_t, 0 \rangle | \xi_{[t-1]}] = 0,
\end{aligned} \tag{5.28}$$

Proof of Theorem 5.1. With our choice of θ_t, ν_t, γ_t, it is easy to show (see [138]) that:

$$\sum_{i=0}^{t} \frac{1}{\nu_i} = \frac{1}{\theta_t \nu_t}, \qquad \frac{1 - \theta_t}{\theta_t \nu_t} = \frac{1}{\theta_{t-1} \nu_{t-1}}, \qquad \theta_t \leqslant \nu_t. \tag{5.29}$$

We further define $\frac{1}{\theta_{-1}\nu_{-1}} = 0$. We first bound the objective value $\phi(x_{t+1})$ by:

$$\begin{aligned}
\phi(x_{t+1}) &= f(x_{t+1}) + h(x_{t+1}) \leqslant f(y_t) + \langle x_{t+1} - y_t, f'(y_t) \rangle \\
&\quad + \frac{L}{2}\|x_{t+1} - y_t\|^2 + M\|x_{t+1} - y_t\| + h(x_{t+1}) \\
&\leqslant \underbrace{f(y_t) + \langle x_{t+1} - y_t, G(y_t) \rangle + \left(\frac{\mu}{\tau \theta_t^2} + \frac{\gamma_t}{\tau} \right) V(x_{t+1}, y_t) + h(x_{t+1})}_{C_1} \\
&\quad \underbrace{- \frac{1}{2}\left(\frac{\mu}{\tau \theta_t^2} + \frac{\gamma_t}{\tau} - L \right) \|x_{t+1} - y_t\|^2 - \langle x_{t+1} - y_t, \Delta_t \rangle + M\|x_{t+1} - y_t\|}_{C_2}
\end{aligned}$$

We bound the terms C_1 and C_2 respectively. Let $\widehat{x}_{t+1}$ be the convex combination of x_t and z_{t+1}:

$$\widehat{x}_{t+1} = (1-\theta_t)x_t + \theta_t z_{t+1}.$$

Then we have $\widehat{x}_{t+1} - y_t = \theta_t(z_{t+1} - \widehat{z}_t)$, where

$$\widehat{z}_t = \frac{\theta_t \mu}{\mu + \gamma_t \theta_t^2} y_t + \frac{(1-\theta_t)\mu + \gamma_t\theta_t^2}{\mu + \gamma_t\theta_t^2} z_t,$$

which is a convex combination of y_t and z_t. By the fact that x_{t+1} is the minimizer of C_1 and utilizing the relationship $V(x_{t+1}, y_t) \leqslant \frac{\tau\|x_{t+1}-y_t\|^2}{2}$ and $\widehat{x}_{t+1} - y_t = \theta_t(z_{t+1} - \widehat{z}_t)$:

$$\begin{aligned} C_1 \leqslant \quad & f(y_t) + \langle \widehat{x}_{t+1} - y_t, f'(y_t)\rangle + \theta_t \langle z_{t+1} - \widehat{z}_t, \Delta_t\rangle \\ & + \left(\frac{\mu + \gamma_t\theta_t^2}{2}\right)\|z_{t+1} - \widehat{z}_t\|^2 + h(\widehat{x}_{t+1}). \end{aligned} \tag{5.30}$$

By the convexity of $\|\cdot\|^2$ and the fact that $\frac{1}{2}\|x-y\|^2 \leqslant V(x,y)$ for any $x, y \in \mathcal{X}$:

$$\begin{aligned} & \left(\frac{\mu + \gamma_t\theta_t^2}{2}\right)\|z_{t+1} - \widehat{z}_t\|^2 \\ \leqslant \quad & \theta_t \mu V(z_{t+1}, y_t) + \left((1-\theta_t)\mu + \theta_t^2\gamma_t\right) V(z_{t+1}, z_t). \end{aligned} \tag{5.31}$$

We plug (5.31) back into RHS of (5.30) and substitute $\widehat{x}_{t+1}$ with $(1-\theta_t)x_t + \theta_t z_{t+1}$. By the convexity of $h(\cdot)$:

$$\begin{aligned} C_1 \quad \leqslant \quad & (1-\theta_t)\left(f(y_t) + \langle x_t - y_t, f'(y_t)\rangle + h(x_t)\right) \\ & + \underbrace{\theta_t\left(f(y_t) + \langle z_{t+1} - y_t, G(y_t)\rangle + h(z_{t+1}) + \mu V(z_{t+1}, y_t) + \left(\frac{(1-\theta_t)\mu}{\theta_t} + \gamma_t\theta_t\right) V(z_{t+1}, z_t)\right)}_{C_3} \\ & + \theta_t\langle z_{t+1} - \widehat{z}_t, \Delta_t\rangle + \theta_t\langle y_t - z_{t+1}, \Delta_t\rangle \\ \leqslant \quad & (1-\theta_t)\phi(x_t) + C_3 + \theta_t\langle y_t - \widehat{z}_t, \Delta_t\rangle. \end{aligned} \tag{5.32}$$

Now we bound C_3 using Proposition 5.1. Utilizing the first equality in (5.29), we can re-write z_t as

$$z_t = \arg\min_{x\in\mathcal{X}}\left\{\widetilde{\psi}_t(x) + \sum_{i=0}^{t-1}\frac{\mu}{\nu_i}V(x, y_i) + \gamma_t V(x, x_0)\right\},$$

where

$$\widetilde{\psi}_t(x) := \sum_{i=0}^{t-1}\frac{f(y_i) + \langle x - y_i, G(y_i)\rangle + h(x)}{\nu_i}.$$

Furthermore, we define $\psi_t(x) := \sum_{i=0}^{t-1}\frac{f(y_i)+\langle x-y_i, G(y_i)\rangle + h(x) + \mu V(x,y_i)}{\nu_i}$ and apply Proposition 5.1 with $x = z_{t+1}$:

$$\begin{aligned} \left(\sum_{i=0}^{t-1}\frac{\mu}{\nu_i} + \gamma_t\right) V(z_{t+1}, z_t) \quad \leqslant \quad & \left(\widetilde{\psi}_t(z_{t+1}) + \sum_{i=0}^{t-1}\frac{\mu}{\nu_i}V(z_{t+1}, y_i) + \gamma_t V(z_{t+1}, x_0)\right) \\ & - \left(\widetilde{\psi}_t(z_t) + \sum_{i=0}^{t-1}\frac{\mu}{\nu_i}V(z_t, y_i) + \gamma_t V(z_t, x_0)\right) \\ = \quad & \psi_t(z_{t+1}) + \gamma_t V(z_{t+1}, x_0) \\ & - \psi_t(z_t) - \gamma_t V(z_t, x_0) \end{aligned} \tag{5.33}$$

We can bound the last term in C_3 by (5.33). In particular, according to (5.29):

$$\begin{aligned}\left(\frac{(1-\theta_t)\mu}{\theta_t}+\gamma_t\theta_t\right)V(z_{t+1},z_t) &\leqslant \nu_t\left(\sum_{i=0}^{t-1}\frac{\mu}{\nu_i}+\gamma_t\right)V(z_{t+1},z_t)\\ &\leqslant \nu_t\left(\psi_t(z_{t+1})+\gamma_t V(z_{t+1},x_0)-\psi_t(z_t)-\gamma_t V(z_t,x_0)\right).\end{aligned}$$

With the above inequality, we immediately obtain an upper bound for C_3. Therefore, by the definition of $\psi_t(\cdot)$, we bound the term C_1 by:

$$\begin{aligned}C_1 \leqslant\quad & (1-\theta_t)\phi(x_t) \\ & +\theta_t\nu_t\left(\psi_{t+1}(z_{t+1})-\psi_t(z_t)+\gamma_t V(z_{t+1},x_0)-\gamma_t V(z_t,x_0)\right)+\theta_t\langle y_t-\widehat{z}_t,\Delta_t\rangle.\end{aligned} \tag{5.34}$$

To bound C_2, since the parameter $c>0$ whenever $\mu=0$, we always have $\frac{\mu}{\tau\theta_t^2}+\frac{\gamma_t}{\tau}-L>0$. Using a simple inequality: $-\frac{\alpha}{2}\kappa^2+\beta\kappa\leqslant\frac{\beta^2}{2\alpha}$ ($\alpha>0$), with $\alpha=\frac{\mu}{\tau\theta_t^2}+\frac{\gamma_t}{\tau}-L$, $\beta=\|\Delta_t\|_*+M$ and $\kappa=\|x_{t+1}-y_t\|$, we have:

$$\begin{aligned}C_2 &\leqslant -\frac{1}{2}\left(\frac{\mu}{\tau\theta_t^2}+\frac{\gamma_t}{\tau}-L\right)\|x_{t+1}-y_t\|^2+\|x_{t+1}-y_t\|(\|\Delta_t\|_*+M)\\ &\leqslant \frac{(\|\Delta_t\|_*+M)^2}{2\left(\frac{\mu}{\tau\theta_t^2}+\frac{\gamma_t}{\tau}-L\right)}.\end{aligned} \tag{5.35}$$

By summing up the upper bound for C_1 in (5.34) and the bound for C_2 in (5.35), we obtain an upper bound for $\phi(x_{t+1})$ according to (5.30). Utilizing the second relation in (5.29), we build up the following recursive inequality:

$$\begin{aligned}\frac{\phi(x_{t+1})}{\theta_t\nu_t} &\leqslant \frac{\phi(x_t)}{\theta_{t-1}\nu_{t-1}}+\left(\psi_{t+1}(z_{t+1})-\psi_t(z_t)+\gamma_t V(z_{t+1},x_0)-\gamma_t V(z_t,x_)\right)\\ &\quad+\frac{(\|\Delta_i\|_*+M)^2}{2\left(\frac{\mu}{\tau\theta_t}+\frac{\theta_t\gamma_t}{\tau}-\theta_t L\right)\nu_t}+\frac{\langle y_t-\widehat{z}_t,\Delta_t\rangle}{\nu_t}\leqslant\cdots\\ &\leqslant \frac{\phi(x_0)}{\theta_{-1}\nu_{-1}}+\psi_{t+1}(z_{t+1})-\psi_0(z_0)+\gamma_t V(z_{t+1},x_0)-\gamma_t V(z_t,x_)\\ &\quad+\sum_{i=0}^{t}\frac{(\|\Delta_i\|_*+M)^2}{2\left(\frac{\mu}{\tau\theta_i}+\frac{\theta_i\gamma_i}{\tau}-\theta_i L\right)\nu_i}+\sum_{i=0}^{t}\frac{\langle y_i-\widehat{z}_i,\Delta_i\rangle}{\nu_i}\\ &= \psi_{t+1}(z_{t+1})+\gamma_t V(z_{t+1},x_0)\\ &\quad+\sum_{i=0}^{t}\frac{(\|\Delta_i\|_*+M)^2}{2\left(\frac{\mu}{\tau\theta_i}+\frac{\theta_i\gamma_i}{\tau}-\theta_i L\right)\nu_i}+\sum_{i=0}^{t}\frac{\langle y_i-\widehat{z}_i,\Delta_i\rangle}{\nu_i},\end{aligned} \tag{5.36}$$

where the last inequality is obtained by the fact that $\frac{1}{\theta_{-1}\nu_{-1}}=0$, $V(z_0,x_0)=0$, $\psi_0(z_0)=0$. Using the fact that

$$z_{t+1}=\arg\min_{x\in\mathcal{X}}\{\psi_{t+1}(x)+\gamma_{t+1}V(x,x_0)\}$$

and $\gamma_t \leqslant \gamma_{t+1}$, (5.36) further implies that:

$$\begin{aligned}
\frac{\phi(x_{t+1})}{\theta_t \nu_t} &\leqslant \psi_{t+1}(x^*) + \gamma_{t+1} V(x^*, x_0) + \sum_{i=0}^{t} \frac{(\|\Delta_i\|_* + M)^2}{2\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i \gamma_i}{\tau} - \theta_i L\right)\nu_i} + \sum_{i=0}^{t} \frac{\langle y_i - \widehat{z}_i, \Delta_i\rangle}{\nu_i} \\
&= \sum_{i=0}^{t} \frac{f(y_i) + \langle x^* - y_i, f'(y_i)\rangle + h(x^*) + \mu V(x^*, y_i)}{\nu_i} + \sum_{i=0}^{t} \frac{\langle x^* - y_i, \Delta_i\rangle}{\nu_i} \\
&\quad + \gamma_{t+1} V(x^*, x_0) + \sum_{i=0}^{t} \frac{(\|\Delta_i\|_* + M)^2}{2\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i \gamma_i}{\tau} - \theta_i L\right)\nu_i} + \sum_{i=0}^{t} \frac{\langle y_i - \widehat{z}_i, \Delta_i\rangle}{\nu_i} \\
&\leqslant \sum_{i=0}^{t} \frac{\phi(x^*)}{\nu_i} + \gamma_{t+1} V(x^*, x_0) \\
&\quad + \sum_{i=0}^{t} \frac{(\|\Delta_i\|_* + M)^2}{2\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i \gamma_i}{\tau} - \theta_i L\right)\nu_i} + \sum_{i=0}^{t} \frac{\langle x^* - \widehat{z}_i, \Delta_i\rangle}{\nu_i}.
\end{aligned} \tag{5.37}$$

Multiplying by $\theta_t \nu_t$ on both sides of (5.37), we obtain the result in (5.7). From the properties of Δ_i in (5.26)–(5.28), we conclude that for all i, $\mathbb{E}\langle x^* - \widehat{z}_i, \Delta_i\rangle = 0$ and $\mathbb{E}(\|\Delta_i\|_* + M)^2 \leqslant 2\sigma^2 + 2M^2$. By taking the expectation on both sides of (5.7) and using the aforementioned properties for Δ_i, we obtain the result in (5.9). □

Proof of Corollary 5.1

Proof. When $\mu = 0$, the expected gap in the objective function in (5.9) for the last iterate becomes:

$$\begin{aligned}
&\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \\
\leqslant\ & \theta_N \nu_N \gamma_{N+1} V(x^*, x_0) + (\sigma^2 + M^2)\theta_N \nu_N \sum_{t=0}^{N} \frac{1}{\left(\frac{\gamma_t}{\tau} - L\right)\theta_t \nu_t}
\end{aligned} \tag{5.38}$$

With choice of $\theta_N = \frac{2}{N+2}$, $\nu_N = \frac{2}{N+1}$ and $\gamma_{N+1} = c(N+2)^{3/2} + \tau L$, the first term in (5.38) is bounded by:

$$\theta_N \nu_N \gamma_{N+1} V(x^*, x_0) \leqslant \frac{4\tau L V(x^*, x_0)}{N^2} + \frac{8c\, V(x^*, x_0)}{\sqrt{N}} \tag{5.39}$$

Similarly, the second term in (5.38) can be bounded by:

$$(\sigma^2 + M^2)\theta_N \nu_N \sum_{t=0}^{N} \frac{1}{\left(\frac{\gamma_t}{\tau} - L\right)\theta_t \nu_t} \leqslant \frac{2\tau(\sigma + M)^2}{c\sqrt{N}} \tag{5.40}$$

By summing the above two inequalities, we obtain that:

$$\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \leqslant \frac{4\tau L V(x^*, x_0)}{N^2} + \frac{8c\, V(x^*, x_0)}{\sqrt{N}} + \frac{2\tau(\sigma + M)^2}{c\sqrt{N}} \tag{5.41}$$

We minimize the RHS of (5.41) with respect to c and obtain the convergence rate result in Corollary 5.1 and the corresponding optimal $c = \frac{\sqrt{\tau}(\sigma + M)}{2\sqrt{V(x^*, x_0)}}$. □

Proof of Corollary 5.2

Proof. When $\mu > 0$, we set $c = 0$ and $\gamma_t \equiv \tau L$ and then (5.9) becomes:

$$\begin{aligned}\mathbb{E}\phi(x_{N+1}) - \phi(x^*) &\leqslant \theta_N \nu_N \tau L V(x^*, x_0) + \frac{\tau(\sigma^2 + M^2)}{\mu}\theta_N \nu_N \sum_{t=0}^{N} \frac{\theta_t}{\nu_t} \\ &\leqslant \frac{4\tau L V(x^*, x_0)}{N^2} + \frac{4\tau(\sigma^2 + M^2)}{\mu N}. \end{aligned} \tag{5.42}$$

This gives the result in (5.11) in Corollary 5.1. □

Proof of Theorem 5.2

Proof. Since $\mathrm{Var}[\phi(x_{t+1}) - \phi(x^*)] \leqslant \mathbb{E}[(\phi(x_{t+1}) - \phi(x^*))^2]$, utilizing (5.7), we can derive the bound for $\mathrm{Var}[\phi(x_{t+1}) - \phi(x^*)]$. Following the elementary inequality $(a+b+c)^2 \leqslant 3a^2 + 3b^2 + 3c^2$ for any $a, b, c \in \mathbb{R}$, we have:

$$\begin{aligned}\mathrm{Var}[\phi(x_{t+1}) - \phi(x^*)] &\leqslant \mathbb{E}[(\phi(x_{t+1}) - \phi(x^*))^2] \\ &\leqslant 3\theta_t^2 \nu_t^2 \underbrace{\gamma_{t+1}^2 V(x^*, x_0)^2}_{C_1} \\ &\quad + 3\theta_t^2 \nu_t^2 \underbrace{\frac{1}{4}\mathbb{E}\left(\left(\sum_{i=0}^{t} \frac{(\|\Delta_i\|_* + M)^2}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i\gamma_i}{\tau} - \theta_i L\right)\nu_i}\right)^2\right)}_{C_2} \\ &\quad + 3\theta_t^2 \nu_t^2 \underbrace{\mathbb{E}\left(\left(\sum_{i=0}^{t} \frac{\langle x^* - \widehat{z}_i, \Delta_i\rangle}{\nu_i}\right)^2\right)}_{C_3}. \end{aligned} \tag{5.43}$$

We first bound C_2. According to our assumption $\mathbb{E}\|G(x,\xi) - f'(x)\|_*^4 \leqslant \sigma^4, \forall x \in \mathcal{X}$, we have for any t:

$$\begin{aligned}\mathbb{E}(\|\Delta_t\|_* + M)^4 &\leqslant 8\mathbb{E}(\|\Delta_t\|_*^4 + M^4) \\ &= 8\mathbb{E}_{\xi_{[t-1]}}[\mathbb{E}_{\xi_t}(\|\Delta_t\|_*^4 | \xi_{[t-1]})] + 8M^4 \\ &\leqslant 8(\sigma^4 + M^4). \end{aligned}$$

According to Cauchy-Schwarz's inequality:

$$\begin{aligned}\mathbb{E}((\|\Delta_i\|_* + M)^2(\|\Delta_j\|_* + M)^2) &\leqslant \sqrt{\mathbb{E}(\|\Delta_i\|_* + M)^4}\sqrt{\mathbb{E}(\|\Delta_j\|_* + M)^4} \\ &\leqslant 8(\sigma^4 + M^4), \quad \forall\, i, j, \end{aligned}$$

we have:

$$\begin{aligned} C_2 &= \frac{1}{4}\mathbb{E}\left(\left(\sum_{i=0}^{t} \frac{(\|\Delta_i\|_* + M)^2}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i\gamma_i}{\tau} - \theta_i L\right)\nu_i}\right)^2\right) \\ &\leqslant \frac{1}{4}\sum_{i=0}^{t}\sum_{j=0}^{t} \frac{\mathbb{E}((\|\Delta_i\|_* + M)^2(\|\Delta_j\|_* + M)^2)}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i\gamma_i}{\tau} - \theta_i L\right)\nu_i \left(\frac{\mu}{\tau\theta_j} + \frac{\theta_j\gamma_j}{\tau} - \theta_j L\right)\nu_j} \\ &\leqslant 2(\sigma^4 + M^4)\left(\sum_{i=0}^{t} \frac{1}{\left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i\gamma_i}{\tau} - \theta_i L\right)\nu_i}\right)^2. \end{aligned} \tag{5.44}$$

To bound C_3. First note that

$$C_3 = \sum_{i=0}^{t} \frac{1}{\nu_i^2} \left(\mathbb{E}\langle x^* - \widehat{z}_i, \Delta_i\rangle\right)^2 + 2 \sum_{0 \leqslant i < j \leqslant t} \frac{1}{\nu_i^2 \nu_j^2} \mathbb{E}\left(\langle x^* - \widehat{z}_i, \Delta_i\rangle \langle x^* - \widehat{z}_j, \Delta_j\rangle\right) \quad (5.45)$$

For the quadratic terms in C_3, by the assumption that $\|x^* - \widehat{z}_i\| \leqslant D$, we have:

$$\mathbb{E}\left[\left(\langle x^* - \widehat{z}_i, \Delta_i\rangle\right)^2\right] \leqslant \mathbb{E}\left[\|x^* - \widehat{z}_i\|^2 \|\Delta_i\|_*^2\right] \leqslant D^2\sigma^2. \quad (5.46)$$

For the cross terms, since $\langle x^* - \widehat{z}_i, \Delta_i\rangle$ only depends on $\xi_{[i]}$ and $\widehat{z}_j$ only on $\xi_{[j-1]}$, by (5.28):

$$\begin{aligned} \mathbb{E}\left(\langle x^* - \widehat{z}_i, \Delta_i\rangle \langle x^* - \widehat{z}_j, \Delta_j\rangle\right) &= \mathbb{E}_{\xi_{[j-1]}}\left(\langle x^* - \widehat{z}_i, \Delta_i\rangle \mathbb{E}_{\xi_j} \langle x^* - \widehat{z}_j, \Delta_j\rangle | \xi_{[j-1]}\right) \\ &= 0. \end{aligned} \quad (5.47)$$

By (5.46) and (5.47), we have:

$$C_3 = \mathbb{E}\left(\left(\sum_{i=0}^{t} \frac{\langle x^* - \widehat{z}_i, \Delta_i\rangle}{\nu_i}\right)^2\right) \leqslant D^2\sigma^2 \sum_{i=0}^{t} \frac{1}{\nu_i^2}. \quad (5.48)$$

By plugging (5.44) and (5.48) back into (5.43), we immediately obtain (5.13) in Theorem 5.2. By plugging all the parameters according to Corollary 5.1 and 5.2, we have the results in (5.14) and (5.15).

□

Proof of Theorem 5.3

We prove Theorem 5.3 using the following two lemmas.

Lemma 5.1 (Lemma 6 in [79]). *Let $\xi_0, \xi_1, \ldots$ be a sequence of* i.i.d. *random variables and $\varphi_i = \varphi_i(\xi_{[i]})$ be deterministic Borel functions of $\xi_{[i]}$ such that:*

1. $\mathbb{E}(\varphi_i | \xi_{[i-1]}) = 0$;
2. *There exists a positive deterministic sequence $\{\sigma_i\}$:*

$$\mathbb{E}\left(\exp\left\{\varphi_i^2/\sigma_i^2\right\} | \xi_{[i-1]}\right) \leqslant \exp\{1\}.$$

Then for any $\delta \in (0,1)$, Prob $\left(\sum_{i=0}^{t} \varphi_i \geqslant \sqrt{3\ln(1/\delta)}(\sum_{i=0}^{t} \sigma_i^2)^{1/2}\right) \leqslant \delta$.

Lemma 5.2 (Lemma 5 in [42]). *Under the assumptions in Theorem 5.3, for any positive and nondecreasing sequence η_i, we have*

$$\sum_{i=0}^{t} \frac{\|\Delta_i\|_*^2}{\eta_i} \geqslant \sum_{i=0}^{t} \frac{\mathbb{E}\|\Delta_i\|_*^2}{\eta_i} + \max\left\{\frac{8\sigma^2\ln(1/\delta)}{\eta_0}, 16\sigma^2\sqrt{\sum_{i=0}^{t} \frac{\ln(1/\delta)}{\eta_i^2}}\right\}$$

holds with probability at most $\delta \in (0,1)$.

We note that although Lemma 5 in [42] assumes that $\eta_i = \eta\sqrt{i+1}$, its proof and conclusion remain valid for any positive nondecreasing sequence $\{\eta_i\}$.

Proof of Theorem 5.3. To simply notations, let $\eta_i = \left(\frac{\mu}{\tau\theta_i} + \frac{\theta_i\gamma_i}{\tau} - \theta_i L\right)$. For both convex and strongly convex $f(x)$, according to our setting of parameters, it is easy to verifty that $\{\eta_i\}$ is a positive monotonically increasing sequence. According to Theorem 5.1:

$$\begin{aligned}\phi(x_{t+1}) - \phi(x^*) \leqslant\ & \underbrace{\theta_t\nu_t\gamma_{t+1}V(x^*, x_0) + \theta_t\nu_t\sum_{i=0}^{t}\frac{M^2}{\eta_i\nu_i}}_{C_1} \\ & \underbrace{+\theta_t\nu_t\sum_{i=0}^{t}\frac{\|\Delta_i\|_*^2}{\eta_i\nu_i}}_{C_2} \\ & \underbrace{+\theta_t\nu_t\sum_{i=0}^{t}\frac{\langle x^* - \widehat{z}_i, \Delta_i\rangle}{\nu_i}}_{C_3},\end{aligned}$$

Firstly, we analyze the last term C_3 using Lemma 5.1. Let $\varphi_i(\xi_{[i]}) := \frac{\langle x^* - \widehat{z}_i, \Delta_i\rangle}{\nu_i}$ and hence $C_3 = \theta_t\nu_t\sum_{i=0}^{t}\varphi_i$. It is easy to verify that $\mathbb{E}(\varphi_i|\xi_{[i-1]}) = 0$ and there exists a sequence $\sigma_i = \frac{D\sigma}{\nu_i}$ such that:

$$\begin{aligned}\mathbb{E}(\exp\{\varphi_i^2/\sigma_i^2\}|\xi_{[i-1]}) \equiv\ & \mathbb{E}\left(\exp\left\{\left(\frac{\langle x^* - \widehat{z}_i, \Delta_i\rangle}{\nu_i}\right)^2 / \frac{D^2\sigma^2}{\nu_i^2}\right\}\right) \\ \leqslant\ & \mathbb{E}\left(\exp\left\{\frac{\|x^* - \widehat{z}_i\|^2\|\Delta_i\|_*^2}{D^2\sigma^2}\right\}\right) \leqslant \exp\{1\},\end{aligned}$$

where the last inequality holds because $\|x^* - \widehat{z}_t\| \leqslant D$ and our "light-tail" assumption. By Lemma 5.1, we conclude that for any $\delta \in (0, 1)$,

$$\Pr\Big(C_3 \geqslant \underbrace{\sqrt{3\ln\frac{2}{\delta}}\theta_t\nu_t D\sigma\big(\sum_{i=0}^{t}\frac{1}{\nu_i^2}\big)^{1/2}}_{D_3}\Big) \leqslant \frac{\delta}{2}. \tag{5.49}$$

Secondly, we bound the term C_2 using Lemma 5.2. Since ν_i is decreasing in i, we have

$$C_2 = \theta_t\nu_t\sum_{i=0}^{t}\frac{\|\Delta_i\|_*^2}{\eta_i\nu_i} \leqslant \theta_t\nu_t\sum_{i=0}^{t}\frac{\|\Delta_i\|_*^2}{\eta_i\nu_t} = \theta_t\sum_{i=0}^{t}\frac{\|\Delta_i\|_*^2}{\eta_i}. \tag{5.50}$$

Since η_i is increasing in i when $\Gamma = L$, we can directly apply Lemma 5.2 as follows:

$$\begin{aligned}&\Pr\left(C_2 \geqslant \underbrace{\theta_t\left[\sum_{i=0}^{t}\frac{\sigma^2}{\eta_i} + \frac{8\sigma^2\ln(2/\delta)}{(\frac{\mu+\gamma_0}{\tau} - L)} + 16\sigma^2\sqrt{\sum_{i=0}^{t}\frac{\ln(2/\delta)}{\eta_i^2}}\right]}_{D_2}\right) \\ \leqslant\ & \Pr\left(\theta_t\sum_{i=0}^{t}\frac{\|\Delta_i\|_*^2}{\eta_i} \geqslant \theta_t\left[\sum_{i=0}^{t}\frac{\mathbb{E}\|\Delta_i\|_*^2}{\eta_i} + \max\left\{\frac{8\sigma^2\ln(2/\delta)}{(\frac{\mu+\gamma_0}{\tau} - L)}, 16\sigma^2\sqrt{\sum_{i=0}^{t}\frac{\ln(2/\delta)}{\eta_i^2}}\right\}\right]\right) \\ \leqslant\ & \frac{\delta}{2}\end{aligned}$$

where the first inequality is from (5.50), $a+b \geqslant \max\{a,b\}$ and the fact $\mathbb{E}\|\Delta_i\|_*^2 \leqslant \sigma^2 \ln\left(\mathbb{E}\exp\left(\frac{\|\Delta_i\|_*^2}{\sigma^2}\right)\right) \leqslant \sigma^2\ln(e) = \sigma^2$ and the second inequality is due to Lemma 5.2.

Combining (5.51) and (5.49), by the union bound:

$$\begin{aligned} & \Pr\left(\phi(x_{t+1}) - \phi(x^*) \geqslant C_1 + D_2 + D_3\right) \\ \leqslant\ & \Pr\left(C_1 + C_2 + C_3 \geqslant C_1 + D_2 + D_3\right) \\ \leqslant\ & \Pr\left(C_2 \geqslant D_2\right) + \Pr\left(C_3 \geqslant D_3\right) \leqslant \frac{\delta}{2} + \frac{\delta}{2} = \delta, \end{aligned} \tag{5.51}$$

we immediately obtain (5.17). The bounds in (5.18) and (5.19) can be derived by plugging all the parameters into (5.17).

□

Proof of Theorem 5.4

To prove theorem 5.4, we first state a corollary of Theorem 5.1.

Corollary 5.3. *For strongly convex $f(x)$, by setting $c = 0$ and $\Gamma = \Lambda + L$ in ORDA, we obtain that:*

$$\mathbb{E}\phi(x_{N+1}) - \phi(x^*) \leqslant \frac{4\tau(\Lambda + L)V(x^*, x_0)}{N^2} + \frac{(N+3)(\sigma^2 + M^2)}{\Lambda}. \tag{5.52}$$

The proof technique follows the proof in [80]. The main idea is to show that $\mathbb{E}(\phi(\widetilde{x}_k)) - \phi(x^*) \leqslant \mathcal{V}_0 2^{-k}$, where $\widetilde{x}_k$ is the solution from the k-th stage.

Proof. We show by induction that

$$\mathbb{E}(\phi(\widetilde{x}_k)) - \phi(x^*) \leqslant \mathcal{V}_0 2^{-k}. \tag{5.53}$$

By the definition of $\mathcal{V}_0$ ($\mathcal{V}_0 > \phi(\widetilde{x}_0) - \phi(x^*)$), this inequality holds for $k = 0$.

Assuming (5.53) holds for the $(k-1)$-th stage, by the strong convexity of $f(x)$, we have

$$\mathbb{E}[V(x^*, \widetilde{x}_{k-1})] \leqslant \mathbb{E}\left[\frac{\tau}{2}\|\widetilde{x}_{k-1} - x^*\|^2\right] \leqslant \mathbb{E}\left[\frac{\tau}{\widetilde{\mu}}(\phi(\widetilde{x}_{k-1}) - \phi(x^*))\right] \leqslant \frac{\mathcal{V}_0 2^{-(k-1)}}{\mu}$$

According to Corollary 5.3 and the setting of N_k and Γ_k, we have

$$\begin{aligned} \mathbb{E}[\phi(\widetilde{x}_k) - \phi(x^*)] &\leqslant \frac{4\tau(\Lambda_k + L)\mathbb{E}V(x^*, \widetilde{x}_{k-1})}{N_k^2} + \frac{(N_k+3)(\sigma^2+M^2)}{\Lambda_k} \\ &\leqslant \frac{4\tau L\mathcal{V}_0 2^{-(k-1)}}{\mu N_k^2} + \frac{4\tau\Lambda_k\mathcal{V}_0 2^{-(k-1)}}{\mu N_k^2} + \frac{4N_k(\sigma^2+M^2)}{\Lambda_k} \\ &\leqslant \frac{4\tau L\mathcal{V}_0 2^{-(k-1)}}{\mu N_k^2} + \frac{8\sqrt{(\sigma^2+M^2)\tau\mathcal{V}_0 2^{-(k-1)}}}{\sqrt{\mu N_k}} \\ &\leqslant \frac{\mathcal{V}_0 2^{-k}}{2} + \frac{\mathcal{V}_0 2^{-k}}{2} = \mathcal{V}_0 2^{-k}. \end{aligned}$$

Therefore, we prove that $\mathbb{E}[\phi(\widetilde{x}_k) - \phi(x^*)] \leqslant \mathcal{V}_0 2^{-k}$ for $k \geqslant 1$.

After running K stages of multi-stage ORDA with $K = \log_2\left(\frac{\mathcal{V}_0}{\epsilon}\right)$, we have $\mathbb{E}[\phi(\widetilde{x}_k) - \phi(x^*)] \leqslant \mathcal{V}_0 2^{-K} = \epsilon$. The total number of iterations from these K stages is upper bounded by:

$$\begin{aligned}
\sum_{k=1}^{K} N_k &\leqslant \sum_{k=1}^{K} \max\left\{4\sqrt{\frac{\tau L}{\mu}}, \frac{2^{k+9}\tau(\sigma^2+M^2)}{\mu\mathcal{V}_0}\right\} \\
&\leqslant \sum_{k=1}^{K}\left[4\sqrt{\frac{\tau L}{\mu}} + \frac{2^{k+9}\tau(\sigma^2+M^2)}{\mu\mathcal{V}_0}\right] \\
&= 4\sqrt{\frac{\tau L}{\mu}}K + \frac{1024\tau(\sigma^2+M^2)(2^K-1)}{\mu\mathcal{V}_0} \\
&\leqslant 4\sqrt{\frac{\tau L}{\mu}}\log_2\left(\frac{\mathcal{V}_0}{\epsilon}\right) + \frac{1024\tau(\sigma^2+M^2)}{\mu\epsilon}
\end{aligned}$$

□

Part IV

LEARNING DYNAMIC SPARSE GRAPHICAL MODELS

6

GRAPH-VALUED REGRESSION

In the first part of the thesis, we discussed efficient optimization techniques for multi-task regression, where the goal is to predict $\mathbb{E}(Y|X = x)$. Here Y and X denote the random response and input vectors, respectively. In many applications, instead of estimating the conditional mean of Y given X as in multi-task regression, we are interested in estimating conditional structures of Y given X, i.e., learning dynamic structures of Y varying with X. One of the most important structures in machine learning is the undirected graphical model (a.k.a., Markov network), which decode the dependency structure of a random vector into a graph G. In this chapter, we study the problem of estimating the conditional undirected graphical model of Y give X as input, denoted as $G(x)$. We refer to this problem as "graph-valued regression". We propose a semiparametric method for estimating $G(x)$ that builds a tree on the X space just as in CART (classification and regression trees), but at each leaf of the tree estimates a graph. We call the method "Graph-optimized CART," or Go-CART. We study the theoretical properties of graph-valued regression using dyadic partitioning trees, establishing oracle inequalities on risk minimization and graph estimation consistency. We also demonstrate the application of Go-CART to a meteorological dataset, showing how graph-valued regression can provide an interesting tool for analyzing high dimensional data.

6.1 INTRODUCTION AND MOTIVATION

Let Y be a p-dimensional random vector with distribution P. A common way to study the structure of P is to construct the undirected graph $G = (V, E)$, where the vertex set V corresponds to the p components of the vector Y. The edge set E is a subset of the pairs of vertices, where an edge between Y_j and Y_k is absent if and only if Y_j is conditionally independent of Y_k given all the other variables. G is so-called the undirected graphical model with respect to the distribution P of Y.

Suppose now that Y and X are both random vectors, and let $P(\cdot\,|\,X)$ denote the conditional distribution of Y given X. In a typical regression or classification problem, we are interested in the conditional mean $\mu(x) = \mathbb{E}\,(Y\,|\,X = x)$. But if Y is multivariate, we may be also interested in how the structure of $P(\cdot\,|\,X)$ varies as a function of X. In particular, let $G(x)$ be the undirected graph corresponding to $P(\cdot\,|\,X = x)$. We refer to the problem of estimating $G(x)$ as *graph-valued regression*.

Let $\mathcal{G} = \{G(x) : x \in \mathcal{X}\}$ be a set of graphs indexed by $x \in \mathcal{X}$, where $\mathcal{X}$ is the domain of X. Then $\mathcal{G}$ induces a partition of $\mathcal{X}$, denoted as $\mathcal{X}_1, \ldots, \mathcal{X}_m$, where x_1 and x_2 lie in the same partition element if and only if $G(x_1) = G(x_2)$. Graph-valued regression thus reduces to esti-

mating the partition and estimating the graph within each partition element.

We present three different partition-based graph estimators; two that use global optimization, and one based on a greedy splitting procedure. One of the optimization based schemes uses penalized empirical risk minimization, the other uses held-out risk minimization. As we show, both methods enjoy strong theoretical properties under relatively weak assumptions; in particular, we establish oracle inequalities on the excess risk of the estimators, and partition selection consistency (under stronger assumptions) in Section 6.4. While the optimization based estimates are attractive, they do not scale well computationally when the input dimension is large. An alternative is to adapt the greedy algorithms of classical CART, as we describe in Section 6.3.1. In Section 6.5 we present experimental results on both synthetic data and a meteorological dataset, demonstrating how graph-valued regression can be an effective tool for analyzing high dimensional data with covariates.

6.2 GRAPH-VALUED REGRESSION

Let $y_1, \ldots, y_n$ be a random sample of vectors from P, where each $y_i \in \mathbb{R}^p$. We are interested in the case where p is large and, in fact, may diverge with n asymptotically. One way to estimate G from the sample is the *graphical lasso* or *glasso* [49, 5], where one assumes that P is Gaussian with mean μ and covariance matrix Σ. Missing edges in the graph correspond to zero elements in the precision matrix $\Omega = \Sigma^{-1}$. A sparse estimate of Ω is obtained by solving

$$\widehat{\Omega} = \underset{\Omega \succ 0}{\arg\min} \{ \operatorname{tr}(S\Omega) - \log|\Omega| + \lambda \|\Omega\|_1 \} \tag{6.1}$$

where Ω is positive definite, S is the sample covariance matrix, and $\|\Omega\|_1 = \sum_{j,k} |\Omega_{jk}|$ is the elementwise ℓ_1-norm of Ω. A fast algorithm for finding $\widehat{\Omega}$ was given by Friedman et al. [49], which involves estimating a single row (and column) of Ω in each iteration by solving a lasso regression. The theoretical properties of $\widehat{\Omega}$ have been studied by Rothman et al. [120] and Ravikumar et al. [117]. In practice, it seems that the glasso yields reasonable graph estimators even if Y is not Gaussian; however, proving conditions under which this happens is an open problem.

We briefly mention three different strategies for estimating $G(x)$, the graph of Y conditioned on $X = x$, each of which builds upon the glasso.

Parametric Estimators. Assume that $Z = (X, Y)$ is jointly multivariate Gaussian with covariance matrix $\Sigma = \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_Y \end{pmatrix}$. We can estimate Σ_X, Σ_Y, and Σ_{XY} by their corresponding sample quantities $\widehat{\Sigma}_X$, $\widehat{\Sigma}_Y$, and $\widehat{\Sigma}_{XY}$, and the marginal precision matrix of X, denoted Ω_X, can be estimated using the glasso. The conditional distribution of Y given $X = x$ is obtained by standard Gaussian formulas. In particular, the conditional covariance matrix of $Y \mid X$ is $\widehat{\Sigma}_{Y|X} = \widehat{\Sigma}_Y - \widehat{\Sigma}_{YX}\widehat{\Omega}_X\widehat{\Sigma}_{XY}$ and

a sparse estimate of the conditional precision $\widehat{\Omega}_{Y|X}$ can be obtained by directly plugging in $\widehat{\Sigma}_{Y|X}$ into glasso. However, the estimated graph does not vary with different values of X.

Kernel Smoothing Estimators. We assume that Y given X is Gaussian, but without making any assumption about the marginal distribution of X. Thus $Y|X = x \sim N(\mu(x), \Sigma(x))$. Under the assumption that both $\mu(x)$ and $\Sigma(x)$ are smooth functions of x, we estimate $\Sigma(x)$ via kernel smoothing:

$$\widehat{\Sigma}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{\|x-x_i\|}{h}\right)(y_i - \widehat{\mu}(x))(y_i - \widehat{\mu}(x))^T}{\sum_{i=1}^{n} K\left(\frac{\|x-x_i\|}{h}\right)}$$

where K is a kernel (e.g. the probability density function of the standard Gaussian distribution), $\|\cdot\|$ is the Euclidean norm, $h > 0$ is a bandwidth and

$$\widehat{\mu}(x) = \sum_{i=1}^{n} K\left(\frac{\|x-x_i\|}{h}\right) y_i \Big/ \sum_{i=1}^{n} K\left(\frac{\|x-x_i\|}{h}\right).$$

Now we apply glasso in (6.1) with $S = \widehat{\Sigma}(x)$ to obtain an estimate of $G(x)$. This method is appealing because it is simple and very similar to nonparametric regression smoothing; the method was analyzed for one-dimensional X in [160]. However, while it is easy to estimate $G(x)$ at any given x, it requires global smoothness of the mean and covariance functions.

Partition Estimators. In this approach, we partition $\mathcal{X}$ into finitely many connected regions $\mathcal{X}_1, \ldots, \mathcal{X}_m$. Within each $\mathcal{X}_j$, we apply the glasso to get an estimated graph $\widehat{G}_j$. We then take $\widehat{G}(x) = \widehat{G}_j$ for all $x \in \mathcal{X}_j$. To find the partition, we appeal to the idea used in CART (classification and regression trees) [17]. We take the partition elements to be recursively defined rectangles with sides parallel to the axes. As is well-known, we can then represent the partition by a tree, where each leaf node corresponds to a single partition element. In CART, the leaves are associated with the means within the partitions; while in our case, there will be an estimated undirected graph for each leaf node. We refer to this method as Graph-optimized CART, or Go-CART. The remainder of this chapter is devoted to the details of this method.

Discussions on Kernel Smoothing v.s. Partition Estimators. Although both kernel smoothing and partition estimators can be applied for the estimation of the conditional Gaussian graphical models, one should choose different estimators for different applications. In particular, when the underlying graphs changes smoothly with respect to input x as evidenced in many time-varying applications, kernel smoothing estimator is a more suitable choice due to its local smoothing effect. However, there are two drawbacks for the kernel smoothing estimator. The first one is that since one has to apply glasso for each data point x with its $\widehat{\Sigma}(x)$, the computational cost of kernel smoothing estimator is prohibitive when applied to a large

number of data points. The second drawback is that when the dimensionality of x is high, the kernel smoothing estimator without dimensionality reduction displays a high variability.

On the other hands, when using partition estimators, one advantage is that it could help us find abruptly changing points in the x space. This will be particularly useful for many applications. For example, for climate data analysis where x represents locations and y represents a set of climatological factors, changing points in the x space might correspond to some interesting geographical features (e.g., a mountain between two adjacent locations might lead to very different climate phenomena of these locations). To put it another way, when using partition estimators, we normally do not assume graphs for adjacent x are close and our goal is to detect those changing points. In addition, the partition based estimator can easily scale to large dataset with many data points and works well even when the dimensionality of x is high. In fact, when the dimensionality of x is high, our Go-CART estimator will only make splits on just a few relevant dimensions of x and leave other dimensions unsplit.

6.3 GRAPH-OPTIMIZED CART

Let $X \in \mathbb{R}^d$ and $Y \in \mathbb{R}^p$ be two random vectors, and let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be n i.i.d. samples from the joint distribution of (X, Y). The domains of X and Y are denoted by $\mathcal{X}$ and $\mathcal{Y}$ respectively; and for simplicity we take $\mathcal{X} = [0,1]^d$. We assume that

$$Y \,|\, X = x \sim N_p(\mu(x), \Sigma(x))$$

where $\mu : \mathbb{R}^d \to \mathbb{R}^p$ is a vector-valued mean function and $\Sigma : \mathbb{R}^d \to \mathbb{R}^{p\times p}$ is a matrix-valued covariance function. We also assume that for each x, $\Omega(x) = \Sigma(x)^{-1}$ is a sparse matrix, i.e., many elements of $\Omega(x)$ are zero. In addition, $\Omega(x)$ may also be a sparse function of x, i.e., $\Omega(x) = \Omega(x_R)$ for some $R \subset \{1, \ldots, d\}$ with cardinality $|R| \ll d$. The task of graph-valued regression is to find an inverse covariance $\widehat{\Omega}(x)$ to estimate $\Omega(x)$ for any $x \in \mathcal{X}$; in some situations the graph of $\Omega(x)$ is of greater interest than the entries of $\Omega(x)$ themselves.

Go-CART is a partition based conditional graph estimator. We partition $\mathcal{X}$ into finitely many connected regions $\mathcal{X}_1, \ldots, \mathcal{X}_m$, and within each $\mathcal{X}_j$ we apply the graphical lasso to estimate a graph $\widehat{G}_j$. We then take $\widehat{G}(x) = \widehat{G}_j$ for all $x \in \mathcal{X}_j$. To find the partition, we restrict ourselves to dyadic splits, as studied by [123, 13]. The primary reason for such a choice is the computational and theoretical tractability of dyadic partition based estimators.

Let $\mathcal{T}$ denote the set of dyadic partitioning trees (DPTs) defined over $\mathcal{X} = [0,1]^d$, where each DPT $T \in \mathcal{T}$ is constructed by recursively dividing $\mathcal{X}$ by means of axis-orthogonal dyadic splits. Each node of a DPT corresponds to a hyperrectangle in $[0,1]^d$. If a node is associated to the hyperrectangle $A = \prod_{l=1}^{d} [a_l, b_l]$, then after being dyadically split along dimension k, the two children are associated with the sub-hyperrectangles $A_L^{(k)} = \prod_{l<k} [a_l, b_l] \times [a_k, \frac{a_k+b_k}{2}] \times$

$\prod_{l>k}[a_l, b_l]$ and $\mathcal{A}_R^{(k)} = \mathcal{A} \backslash \mathcal{A}_L^{(k)}$. Given a DPT T, we denote by $\Pi_T = \{\mathcal{X}_1, \ldots, \mathcal{X}_{m_T}\}$ the partition of $\mathcal{X}$ induced by the leaf nodes of T. For a dyadic integer $N = 2^K$, we define $\mathcal{T}_N$ to be the collection of all DPTs such that no partition has a side length smaller than 2^{-K}. We use $\mu_T(x)$ and $\Omega_T(x)$ to denote the piecewise constant mean and precision functions associated with T:

$$\mu_T(x) = \sum_{j=1}^{m_T} \mu_{\mathcal{X}_j} \cdot I\left(x \in \mathcal{X}_j\right) \quad \text{and} \quad \Omega_T(x) = \sum_{j=1}^{m_T} \Omega_{\mathcal{X}_j} \cdot I\left(x \in \mathcal{X}_j\right),$$

where $\mu_{\mathcal{X}_j} \in \mathbb{R}^p$ and $\Omega_{\mathcal{X}_j} \in \mathbb{R}^{p \times p}$ are the mean vector and precision matrix for $\mathcal{X}_j$.

Before formally defining our graph-valued regression estimators, we require some further definitions. Given a DPT T with an induced partition $\Pi_T = \{\mathcal{X}_j\}_{j=1}^{m_T}$ and corresponding mean and precision functions $\mu_T(x)$ and $\Omega_T(x)$, the negative conditional log-likelihood risk $R(T, \mu_T, \Omega_T)$ and its sample version $\widehat{R}(T, \mu_T, \Omega_T)$ are defined as follows:

$$R(T, \mu_T, \Omega_T) = \sum_{j=1}^{m_T} \mathbb{E}\left[\left(\mathrm{tr}\left[\Omega_{\mathcal{X}_j}\left((Y - \mu_{\mathcal{X}_j})(Y - \mu_{\mathcal{X}_j})^T\right)\right] - \log|\Omega_{\mathcal{X}_j}|\right) \cdot I\left(X \in \mathcal{X}_j\right)\right],$$

$$\widehat{R}(T, \mu_T, \Omega_T) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m_T}\left[\left(\mathrm{tr}\left[\Omega_{\mathcal{X}_j}\left((y_i - \mu_{\mathcal{X}_j})(y_i - \mu_{\mathcal{X}_j})^T\right)\right] - \log|\Omega_{\mathcal{X}_j}|\right) \cdot I\left(x_i \in \mathcal{X}_j\right)\right].$$

Let $[[T]] > 0$ denote a prefix code over all DPTs $T \in \mathcal{T}_N$; thus, $\sum_{T \in \mathcal{T}_N} 2^{-[[T]]} \leqslant 1$. One such prefix code $[[T]]$ is proposed in [123], and takes the form $[[T]] = 3|\Pi_T| - 1 + (|\Pi_T| - 1)\log d / \log 2$. A simple upper bound for $[[T]]$ is

$$[[T]] \leqslant (3 + \log d / \log 2)|\Pi_T|. \tag{6.2}$$

Our analysis will assume that the conditional means and precision matrices are bounded in the $\|\cdot\|_\infty$ and $\|\cdot\|_1$ norms; specifically we suppose there is a positive constant B and a sequence $L_{1,n}, \ldots, L_{m_T,n} > 0$, where each $L_{j,n} \in \mathbb{R}^+$ is a function of the sample size n, and we define the domains of each $\mu_{\mathcal{X}_j}$ and $\Omega_{\mathcal{X}_j}$ as

$$\begin{aligned} M_j &= \{\mu \in \mathbb{R}^p : \ \|\mu\|_\infty \leqslant B\}, \\ \Lambda_j &= \left\{\Omega \in \mathbb{R}^{p \times p} : \ \Omega \text{ is positive } \Omega \succ \mathbf{0}, \|\Omega\|_1 \leqslant L_{j,n}\right\}. \end{aligned} \tag{6.3}$$

With this notation in place, we can now define two estimators.

Definition 6.1. *The **penalized empirical risk minimization Go-CART estimator** is defined as*

$$\widehat{T}, \left\{\widehat{\mu}_{\widehat{\mathcal{X}}_j}, \widehat{\Omega}_{\widehat{\mathcal{X}}_j}\right\}_{j=1}^{m_{\widehat{T}}} = \mathrm{argmin}_{T \in \mathcal{T}_N, \mu_{\mathcal{X}_j} \in M_j, \Omega_{\mathcal{X}_j} \in \Lambda_j}\left\{\widehat{R}(T, \mu_T, \Omega_T) + \mathrm{pen}(T)\right\}$$

where $\widehat{R}$ is defined in (6.2) and

$$\mathrm{pen}(T) = \gamma_n \cdot m_T \sqrt{\frac{[[T]]\log 2 + 2\log(np)}{n}}$$

where $\gamma_n > 0$ is a regularization parameter.

Empirically, we may always set the dyadic integer N to be a reasonably large value; the tuning parameter γ_n is responsible for selecting a suitable DPT $T \in \mathcal{T}_N$.

We also formulate an estimator that minimizes held-out risk. For this purpose, suppose that $\{(x'_1, y'_1), \ldots, (x'_{n_2}, y'_{n_2})\}$ is an i.i.d. sample of held-out data. The held-out negative log-likelihood risk is then given by

$$\widehat{R}_{\text{out}}(T, \mu_T, \Omega_T) = \frac{1}{n_2} \sum_{i=1}^{n_2} \sum_{j=1}^{m_T} \left\{ \left(\text{tr}\left[\Omega_{\mathcal{X}_j} \left((y'_i - \mu_{\mathcal{X}_j})(y'_i - \mu_{\mathcal{X}_j})^T \right) \right] - \log|\Omega_{\mathcal{X}_j}| \right) \cdot I\left(x'_i \in \mathcal{X}_j\right) \right\}.$$

Definition 6.2. *For each DPT T define*

$$\widehat{\mu}_T, \widehat{\Omega}_T = \text{argmin}_{\mu_{\mathcal{X}_j} \in M_j, \Omega_{\mathcal{X}_j} \in \Lambda_j} \widehat{R}(T, \mu_T, \Omega_T)$$

*where $\widehat{R}$ is defined in (6.2) but only evaluated on $\mathcal{D}_1 = \{(x_1, y_1), \ldots, (x_{n_1}, y_{n_1})\}$. The **held-out risk minimization Go-CART estimator** is*

$$\widehat{T} = \text{argmin}_{T \in \mathcal{T}_N} \widehat{R}_{\text{out}}(T, \widehat{\mu}_T, \widehat{\Omega}_T).$$

6.3.1 *Greedy Partitioning*

The procedures in the last section require us to find an optimal dyadic partitioning tree within $\mathcal{T}_N$. Although dynamic programming can be applied, as in [13], the computation does not scale to large input dimensions d. In this section, we propose a simple yet effective greedy algorithm to find an approximate solution $(\widehat{T}, \widehat{\mu}_T, \widehat{\Omega}_T)$. We focus on the held-out risk minimization form as in Definition 6.2, due to its superior empirical performance. But note that our greedy approach is generic and can easily be adapted to the penalized empirical risk minimization form.

First, consider the simple case that we are given a dyadic tree structure T which induces a partition $\Pi_T = \{\mathcal{X}_1, \ldots, \mathcal{X}_{m_T}\}$ on $\mathcal{X}$. For any partition element $\mathcal{X}_j$, we estimate the sample mean using the training dataset $\mathcal{D}_1$:

$$\widehat{\mu}_{\mathcal{X}_j} = \frac{1}{\sum_{i=1}^{n_1} I\left(x_i \in \mathcal{X}_j\right)} \sum_{i=1}^{n_1} y_i \cdot I\left(x_i \in \mathcal{X}_j\right).$$

The glasso is then used to estimate a sparse precision matrix $\widehat{\Omega}_{\mathcal{X}_j}$. More precisely, let $\widehat{\Sigma}_{\mathcal{X}_j}$ be the sample covariance matrix for the partition element $\mathcal{X}_j$, given by

$$\widehat{\Sigma}_{\mathcal{X}_j} = \frac{1}{\sum_{i=1}^{n_1} I\left(x_i \in \mathcal{X}_j\right)} \sum_{i=1}^{n_1} \left(y_i - \widehat{\mu}_{\mathcal{X}_j}\right)\left(y_i - \widehat{\mu}_{\mathcal{X}_j}\right)^T \cdot I\left(x_i \in \mathcal{X}_j\right).$$

The estimator $\widehat{\Omega}_{\mathcal{X}_j}$ is obtained by optimizing

$$\widehat{\Omega}_{\mathcal{X}_j} = \arg\min_{\Omega \succ 0} \left\{ \text{tr}(\widehat{\Sigma}_{\mathcal{X}_j} \Omega) - \log|\Omega| + \lambda_j \|\Omega\|_1 \right\},$$

where λ_j is in one-to-one correspondence with $L_{j,n}$ in (6.3). In practice, we run the full regularization path of the glasso, from large λ_j,

which yields very sparse graph, to small λ_j, and select the graph that minimizes the held-out negative log-likelihood risk. (Note that if there is no additional held-out validation data, the Bayesian information criterion (BIC) suggested by [151] could be adopted. In practice, compared to BIC, the held-out risk minimization is more stable and has better sparsity pattern recovery quality.) To further improve the model selection performance, we refit the parameters of the precision matrix after the graph has been selected. That is, the glasso typically results in a large estimation bias as a consequence of the ℓ_1-regularization. To reduce the bias, we first estimate the sparse precision matrix using ℓ_1-regularization, and then we refit the Gaussian model without ℓ_1-regularization, but enforcing the sparsity pattern obtained in the first step.

The natural, standard greedy procedure starts from the coarsest partition $\mathcal{X} = [0,1]^d$ and then computes the decrease in the held-out risk by dyadically splitting each hyperrectangle $\mathcal{A}$ along dimension $k \in \{1,\dots d\}$. The dimension k^* that results in the largest decrease in held-out risk is selected, where the change in risk is given by

$$\begin{aligned} & \Delta\widehat{R}^{(k)}_{out}(\mathcal{A}, \widehat{\mu}_{\mathcal{A}}, \widehat{\Omega}_{\mathcal{A}}) \\ = \quad & \widehat{R}_{out}(\mathcal{A}, \widehat{\mu}_{\mathcal{A}}, \widehat{\Omega}_{\mathcal{A}}) - \widehat{R}_{out}(\mathcal{A}^{(k)}_L, \widehat{\mu}_{\mathcal{A}^{(k)}_L}, \widehat{\Omega}_{\mathcal{A}^{(k)}_L}) - \widehat{R}_{out}(\mathcal{A}^{(k)}_R, \widehat{\mu}_{\mathcal{A}^{(k)}_R}, \widehat{\Omega}_{\mathcal{A}^{(k)}_R}). \end{aligned}$$

If splitting any dimension k of $\mathcal{A}$ leads to an increase in the held-out risk, the element $\mathcal{A}$ should no longer be split and hence becomes a partition element of Π_T.

This greedy partitioning method parallels the classical algorithms for classification and regression that have been used in statistical learning for decades. However, the strength of the procedures given in Definitions 6.1 and 6.2 is that they lend themselves to a theoretical analysis under relatively weak assumptions, as we show in the following section. The theoretical properties of greedy Go-CART are left to future work.

6.4 THEORETICAL PROPERTIES

We define the oracle risk R^* over $\mathcal{T}_N$ as

$$R^* = R(T^*, \mu^*_T, \Omega^*_T) = \inf_{T \in \mathcal{T}_N, \mu_{\mathcal{X}_j} \in M_j, \Omega_{\mathcal{X}_j} \in \Lambda_j} R(T, \mu_T, \Omega_T).$$

Note that T^*, $\mu^*_{T^*}$, and $\Omega^*_{T^*}$ might not be unique, since the finest partition always achieves the oracle risk. To obtain oracle inequalities on our estimation procedure, we make the following two technical assumptions.

Assumption 6.1. *Let $T \in \mathcal{T}_N$ be an arbitrary DPT which induces a partition $\mathcal{X}_1, \dots, \mathcal{X}_{m_T}$ on $\mathcal{X}$, we assume that there exists a constant B, such that*

$$\max_{1 \leqslant j \leqslant m_T} \|\mu_{\mathcal{X}_j}\|_\infty \leqslant B \quad \text{and} \quad \max_{1 \leqslant j \leqslant m_T} \sup_{\Omega \in \Lambda_j} \log|\Omega| \leqslant L_n$$

where Λ_j *is defined in* (6.3) *and* $L_n = \max_{1 \leqslant j \leqslant m_T} L_{j,n}$, *where* $L_{j,n}$ *is the same as in* (6.3). *We also assume that*

$$L_n = o(\sqrt{n}).$$

Assumption 6.2. *Let* $Y = (Y_1, \ldots, Y_p)^T \in \mathbb{R}^p$. *For any for any* $A \subset \mathcal{X}$, *we define*

$$\begin{aligned} Z_{k\ell}(A) &= Y_k Y_\ell \cdot I(X \in A) - \mathbb{E}(Y_k Y_\ell \cdot I(X \in A)) \\ Z_j(A) &= Y_j \cdot I(X \in A) - \mathbb{E}(Y_j \cdot I(X \in A)). \end{aligned}$$

We assume there exist constants M_1, M_2, v_1, *and* v_2, *such that*

$$\sup_{k,\ell,A} \mathbb{E}|Z_{k\ell}(A)|^m \leqslant \frac{m! M_2^{m-2} v_2}{2} \quad \text{and} \quad \sup_{j,A} \mathbb{E}|Z_j(A)|^m \leqslant \frac{m! M_1^{m-2} v_1}{2}.$$

for all $m \geqslant 2$.

Theorem 6.1. *Let* $T \in \mathcal{T}_N$ *be a DPT that induces a partition* $\mathcal{X}_1, \ldots, \mathcal{X}_{m_T}$ *on* $\mathcal{X}$. *For any* $\delta \in (0, 1/4)$, *let* $\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}$ *be the estimator obtained using the penalized empirical risk minimization objective of Definition 6.1, with a penalty term* $\text{pen}(T)$ *of the form*

$$\text{pen}(T) = (C_1 + 1) L_n m_T \sqrt{\frac{[[T]] \log 2 + 2 \log p + \log(48/\delta)}{n}}$$

where $C_1 = 8\sqrt{v_2} + 8B\sqrt{v_1} + B^2$. *Then for sufficiently large* n, *the excess risk inequality*

$$R(\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) - R^* \leqslant \inf_{T \in \mathcal{T}_N} \left\{ 2\text{pen}(T) + \inf_{\mu_{\mathcal{X}_j} \in M_j, \Omega_{\mathcal{X}_j} \in \Lambda_j} (R(T, \mu_T, \Omega_T) - R^*) \right\}$$

holds with probability at least $1 - 4\delta$.

A similar oracle inequality holds when using the held-out risk minimization form.

Theorem 6.2. *Let* $T \in \mathcal{T}_N$ *be a DPT which induces a partition* $\mathcal{X}_1, \ldots, \mathcal{X}_{m_T}$ *on* $\mathcal{X}$. *We define* $\phi_n(T)$ *to be a function of* n *and* T *such that*

$$\phi_n(T) = \text{pen}(T) = (C_2 + \sqrt{2}) L_n m_T \sqrt{\frac{[[T]] \log 2 + 2 \log p + \log(384/\delta)}{n}}$$

where $C_2 = 8\sqrt{2v_2} + 8B\sqrt{2v_1} + \sqrt{2}B^2$ *and* $L_n = \max_{1 \leqslant j \leqslant m_T} L_{j,n}$. *Partition the data into* $\mathcal{D}_1 = \{(x_1, y_1), \ldots, (x_{n_1}, y_{n_1})\}$ *and* $\mathcal{D}_2 = \{((x'_1, y'_1), \ldots, (x'_{n_2}, y'_{n_2}))\}$ *with sizes* $n_1 = n_2 = n/2$. *Let* $\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}$ *be the estimator constructed using the held-out risk minimization criterion of Definition 6.2. Then, for sufficiently large* n, *the excess risk inequality*

$$R(\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) - R^* \leqslant \inf_{T \in \mathcal{T}_N} \left\{ 3\phi_n(T) + \inf_{\mu_{\mathcal{X}_j} \in M_j, \Omega_{\mathcal{X}_j} \in \Lambda_j} (R(T, \mu_T, \Omega_T) - R^*) \right\} + \phi_n(\widehat{T})$$

with probability at least $1 - \delta$.

Note that in contrast to the statement in Theorem 6.1, Theorem 6.2 results in a stochastic upper bound due to the extra $\phi_n(\widehat{T})$ term, which depends on the complexity of the final estimate $\widehat{T}$. Due to space limitations, the proofs of both theorems are detailed in the supplementary materials.

We now temporarily make the strong assumption that the model is correct, so that Y given X is conditionally Gaussian, with a partition structure that is given by a dyadic tree. We show that with high probability, the true dyadic partition structure can be correctly recovered.

Assumption 6.3. *The true model is*

$$Y \,|\, X = x \sim N_p(\mu^*_{T^*}(x), \Omega^*_{T^*}(x)) \tag{6.4}$$

where $T^* \in \mathcal{T}_N$ *is a DPT with induced partition* $\Pi(T^*) = \{\mathcal{X}^*_j\}_{j=1}^{m_{T^*}}$*, such that the precision matrix satisfies*

$$\Omega^*_{T^*}(x) = \sum_{j=1}^{m_{T^*}} \Omega^*_j \, I(x \in \mathcal{X}^*_j).$$

Under this assumption, clearly

$$\inf_{\mu^*_{T^*}, \Omega^*_{T^*} \in \mathcal{M}_{T^*}} R(T^*, \mu^*_{T^*}, \Omega^*_{T^*}) = \inf_{T \in \mathcal{T}_N, \mu_T, \Omega_T \in \mathcal{M}_T} R(T, \mu_T, \Omega_T),$$

where $\mathcal{M}_T$ is given by

$$\mathcal{M}_T = \Big\{ \mu(x) = \sum_{j=1}^{m_T} \mu_{\mathcal{X}_j} \, I(x \in \mathcal{X}_j), \; \Omega(x) = \sum_{j=1}^{m_T} \Omega_{\mathcal{X}_j} \, I(x \in \mathcal{X}_j) : \; \mu_{\mathcal{X}_j} \in M_j, \; \Omega_{\mathcal{X}_j} \in \Lambda_j \Big\}.$$

We have the following definitions:

Definition 6.3. *A tree estimation procedure* $\widehat{T}$ *is* ***tree partition consistent*** *in case*

$$\mathbb{P}\left(\Pi(\widehat{T}) \subset \Pi(T^*) \right) \to 1$$

as $n \to \infty$.

Note that the estimated partition may be finer than the true partition. Establishing a tree partition consistency result requires further technical assumptions. The following assumption specifies that for arbitrary adjacent subregions of the true dyadic partition, either the means or the variances should be sufficiently different. Without such an assumption, of course, it is impossible to detect the boundaries of the true partition.

Assumption 6.4. *Let* $\mathcal{X}^*_i$ *and* $\mathcal{X}^*_j$ *be adjacent partition elements of* T^**, so that they have a common parent node within* T^**. Let* $\Sigma^*_{\mathcal{X}^*_i} = (\Omega^*_{\mathcal{X}^*_i})^{-1}$*. We assume there exist positive constants* c_1, c_2, c_3, c_4*, such that either*

$$2 \log \left| \frac{\Sigma^0_{\mathcal{X}^*_i} + \Sigma^0_{\mathcal{X}^*_j}}{2} \right| - \log |\Sigma^0_{\mathcal{X}^*_i}| - \log |\Sigma^0_{\mathcal{X}^*_j}| \geqslant c_4$$

or $\|\mu^*_{\mathcal{X}^*_i} - \mu^*_{\mathcal{X}^*_j}\|_2^2 \geqslant c_3$*. We also assume*

$$\rho_{\min}(\Omega^*_{\mathcal{X}^*_j}) \geqslant c_1, \quad \forall j = 1, \ldots, m_{T^*},$$

where $\rho_{\min}(\cdot)$ *denotes the smallest eigenvalue. Furthermore, for any* $T \in \mathcal{T}_N$ *and any* $\mathcal{A} \in \Pi_T$*, we have* $\mathbb{P}(X \in \mathcal{A}) \geqslant c_2$.

Theorem 6.3. *Under the above assumptions, we have*

$$\begin{aligned} & \inf_{T \in \mathcal{T}_N,\, \Pi(T^*) \nsubseteq \Pi(T)} \; \inf_{\mu_T,\, \Omega_T \in \mathcal{M}_T} R(T, \mu_T, \Omega_T) - \inf_{\mu,\, \Omega \in \mathcal{M}_{T^*}} R(T^*, \mu^0_{T^*}, \Omega^0_{T^*}) \\ > \quad & \min\{\frac{c_1 c_2 c_3}{2}, c_2 c_4\} \end{aligned}$$

where c_1, c_2, c_3, c_4 are defined in Assumption 6.4. Moreover, the Go-CART estimator in both the penalized risk minimization and held-out risk minimization form is tree partition consistent.

This result shows that, with high probability, we obtain a finer partition than T^*; the assumptions do not, however, control the size of the resulting partition. The proof of this result appears in the supplementary material.

6.5 EXPERIMENT

We now present the performance of the greedy DPT learning algorithm of Section 6.3.1 on both synthetic data and a real meteorological dataset. In each experiment, we set the dyadic integer to $N = 2^{10}$ to ensure that we can obtain sufficiently fine partitions of the covariate space $\mathcal{X}$. Furthermore, we always ensure that the region (hyperrectangle) represented by each leaf node contains no fewer than 10 data points to guarantee reasonable estimates of the sample means and sparse inverse covariance matrices.

6.5.1 *Synthetic Data*

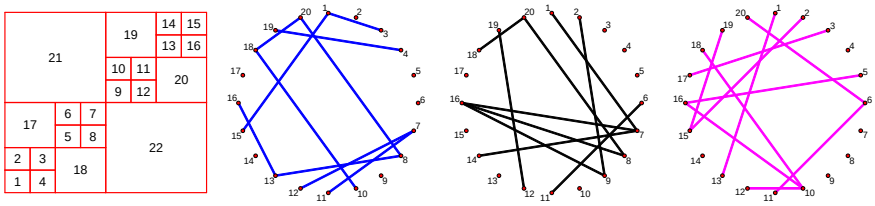
6.5.1.1 *Dyadic Underlying Partition*

We generate n data points $x_1, \ldots, x_n \in \mathbb{R}^d$ with $n = 10,000$ and $d = 10$ uniformly distributed on the unit hypercube $[0,1]^d$. We split the square $[0,1]^2$ defined by the first two dimension of the unit hypercube into 22 subregions as shown in Figure 6.1 (a). For the t-th subregion where $1 \leqslant t \leqslant 22$, we generate an Erdös-Rényi random graph $G^t = (V^t, E^t)$ with the number of vertices $p = 20$, the number of edges $|E| = 10$ and the maximum node degree is 4. As an illustration example, the random graphs for subregion 4 (smallest region), 17 (middle region) and 22 (large region) are presented in Figure 6.1 (b), (c) and (d) respectively. For each graph G^t, we generate the inverse covariance matrix Ω^t according to:

$$\Omega^t_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0.245 & \text{if } (i,j) \in E^t, \\ 0 & \text{otherwise,} \end{cases}$$

where 0.245 guarantees the positive definiteness of Ω^t when the maximum node degree is 4.

For each data point x_i in the t-th subregion, we associate it with a 20-dimensional response vector y_i generated from a multivariate



(a) (b) (c) (d)

Figure 6.1: (a) The 22 subregions defined on $[0,1]^2$. The horizontal axis corresponds to the first dimension denoted as X_1 while the vertical axis corresponds to the second dimension denoted as X_2. The bottom left point corresponds to $[0,0]$ and the upper right point corresponds to $[1,1]$. (b) The ground true graph for subregion 4. (c) The ground true graph for subregion 17. (d) The ground true graph for subregion 22.

Gaussian distribution $N_{20}\left(0, \left(\Omega^t\right)^{-1}\right)$. We also create an equally-sized held-out dataset in the same manner based on $\{\Omega^t\}_{t=1}^{22}$.

We apply the greedy algorithm on this synthetic dataset. The learned dyadic tree structure and its induced partitions are presented in Figure 6.2. Estimated graphs for some nodes are also illustrated. Note that the label for each subregion in the subplot (c) is the leaf node ID of tree in subplot (a). We conduct 100 Monte Carlo simulations and find that 82 times out of 100 runs our algorithm perfectly recover the ground true partitions on the X_1-X_2 plane and never wrongly split any irrelevant dimensions ranging from X_3 to X_{10}. Moreover, the estimated graphs have interesting patterns. Even though the graphs within each subregion are sparse, the estimated graph by pooling all the data together is highly dense. With the progress of our algorithm, the estimated graphs become sparser and sparser. However, for the immediate parent nodes of the true subregions, the graphs become denser again. Out of the 82 simulations where we correctly identify the tree structure, we list the graph estimation performance for subregions 1, 4, 17, 18, 21, 22 in terms of precision, recall, and F1-score. Let $\widehat{E}$ be the estimated edge set while E be the true edge set. These criteria are defined as:

$$\text{precision} = \frac{|\widehat{E} \cap E|}{|\widehat{E}|}, \quad \text{recall} = \frac{|\widehat{E} \cap E|}{|E|}, \quad \text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

We see that for a larger subregion, it is easier for us to get a better recovery performance. While good recovery for a very small region becomes more challenging. This makes sense. In fact, for subregion 1 (the smallest one), we have only approximately $10000/64 \approx 156$ data points. It's more challenging to perfectly estimate the sparsity pattern of in total $p(p-1)/2 = 190$ edges. In contrast, for the subregion 18, we have around $10000/16 = 625$ data points fall inside, which makes graph estimation much easier. We also plot the held-out risk in the subplot (c). As we can see, the first few splits lead to the most significant decreases in term of the held-out risk. The whole risk curve illustrates a diminishing return behavior. Correctly splitting the large rectangle into middle ones leads to significant decrease of the risk; in contrast, splitting the middle rectangles into the smaller ones does not reduce the risk as much. We have more simulated experiments

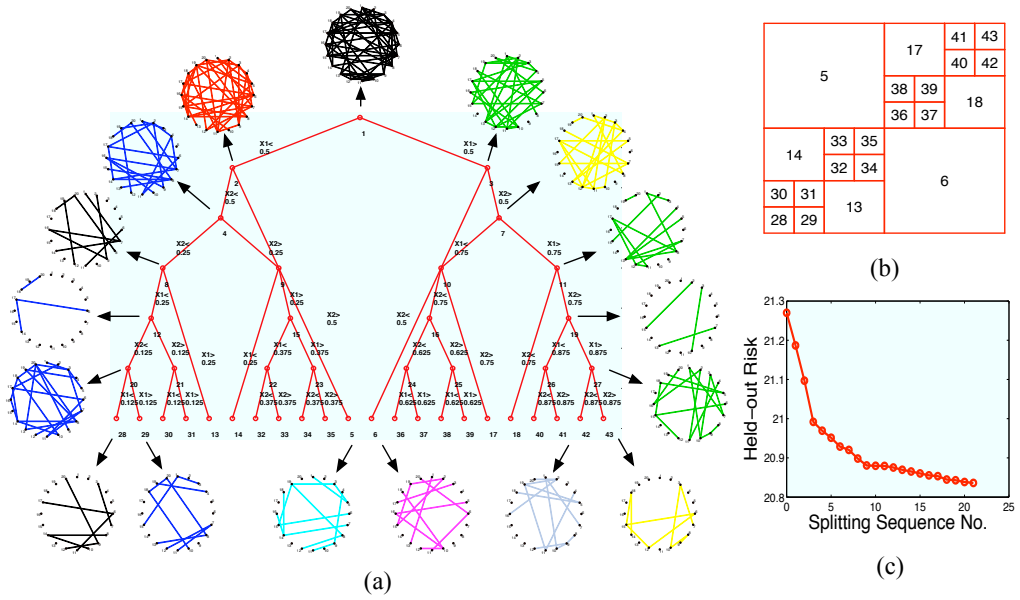


Figure 6.2: (a) The learned dyadic tree structure; (b) The induced partition on $[0,1]^2$ and the number labeled on each subregion corresponds to each leaf node ID of the tree in (a); (c) The held-out negative log-likelihood risk for each split. The order of the splits corresponds the ID of the tree node (from small to large)

when the ground true conditional covariance matrix is a continuous function of the x. We present them in the appendix.

Table 6.1: The graph estimation performance over different subregions

	Mean values over 100 runs (Standard deviation)					
subregion	region 1	region 4	region 17	region 18	region 21	region 22
Precision	0.8327 (0.15)	0.8429 (0.15)	0.9821 (0.05)	0.9853 (0.04)	0.9906 (0.04)	0.9899 (0.05)
Recall	0.7890 (0.16)	0.7990 (0.18)	1.0000 (0.00)	1.0000 (0.00)	1.0000 (0.00)	1.0000 (0.00)
F_1-score	0.7880 (0.11)	0.7923 (0.12)	0.9904 (0.03)	0.9921 (0.02)	0.9949 (0.02)	0.9913 (0.02)

6.5.1.2 *Non-Dyadic Underlying Partition*

To further demonstrate recovery quality of our method, in this section, we simulate the data where the ground true conditional covariance matrix is continuous in X; and we compare the graphs estimated by our method to the one by applying glasso directly on the entire data.

Chain Structure

We consider the case where X lies on a one dimensional chain. More precisely, we generate n equally spaced points $x_1, \ldots, x_n \in \mathbb{R}$ with $n = 10,000$ on $[0,1]$. We generate an Erdös-Rényi random graph $G^1 = (V^1, E^1)$ with the number of vertices $p = 20$, the number of edges $|E| = 10$ and the maximum node degree to be 4 as the basis. Then we simulate the output $y_1, \ldots, y_n] \in \mathbb{R}^p$ as follows:

1. From $t = 2$ to T, we construct the graph $G^t = (V^t, E^t)$ as follows: (a) with probability 0.05, remove one edge from G^{t-1} and (b)

with probability 0.05, add one edge to the graph generated in (a). We make sure that the total number of edges is between 5 and 15; and maximum node degree is still 4.

2. For each graph G^t, generate the inverse covariance matrix Ω^t:

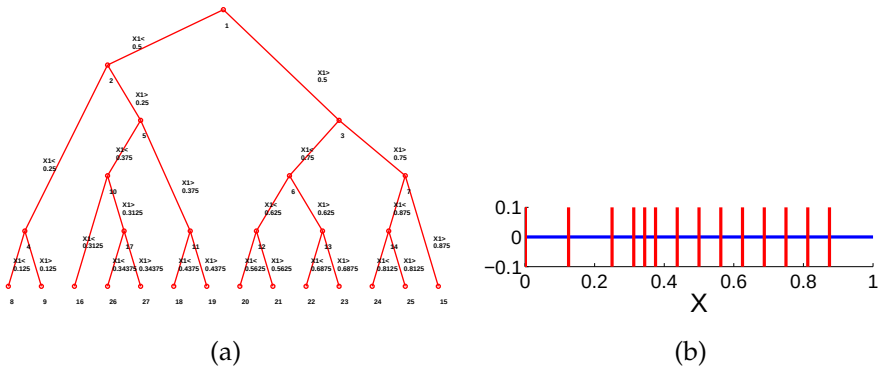
$$\Omega^t(i,j) = \begin{cases} 1 & \text{if } i = j, \\ 0.245 & \text{if } (i,j) \in E^t, \\ 0 & \text{otherwise,} \end{cases}$$

where 0.245 guarantees the positive definiteness of Ω^t when the maximum degree is 4.

3. For each t, we sample y_t from a multivariate Gaussian distribution with mean $\mu = (0,\ldots,0) \in \mathbb{R}^p$ and covariance matrix $\Sigma^t = (\Omega^t)^{-1}$:

$$y_t \sim N(\mu, \Sigma^t),$$

In addition, we generate the equal-sized held-out data in the same manner as described based with the same μ and Σ^t and apply our greedy algorithm to learn the dyadic tree structure and corresponding inverse covariance matrices. The learned tree structure and the corresponding partitions are presented in Figure 6.3.



(a) (b)

Figure 6.3: (a) Learned tree structure; (b) Corresponding partitions

To examine the recovery quality of the underlying graph structure, we compare our estimated graphs to the one estimated by directly applying glasso to the entire dataset. We present the comparison of precision, recall and F1-score in Figure 6.4 (a), (b) and (c) respectively. As we can see, out method achieves much higher precision and F1-Score. As for recall, glasso is even slightly better than us because the graphs estimated by glasso on the entire data is very dense as shown in 6.4 (d). The dense graphs lead to to fewer false negatives (thus large recall values) but many false positives (thus small precision values).

Two-way Grid Structure

We demonstrate Go-CART for a two dimensional design X. The underlying graph structures and Y are generated in the way similar to that in in the previous section as follows: We generate equally

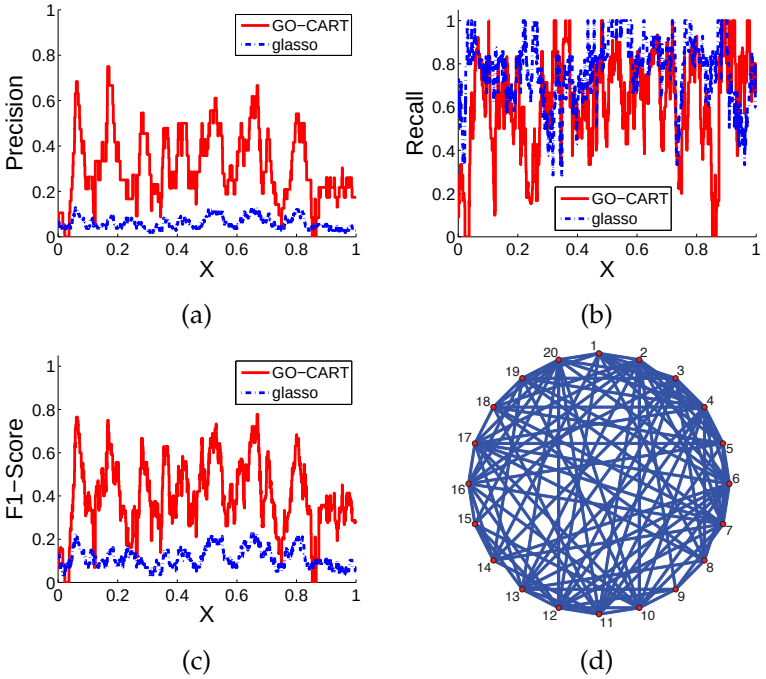


Figure 6.4: Comparison of our algorithm with glasso (a) Precision; (b) Recall; (c) F1-score; (d) Estimated graph by applying glasso on the entire dataset

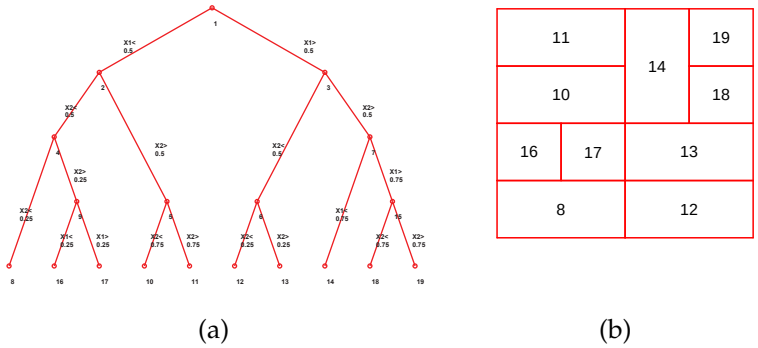
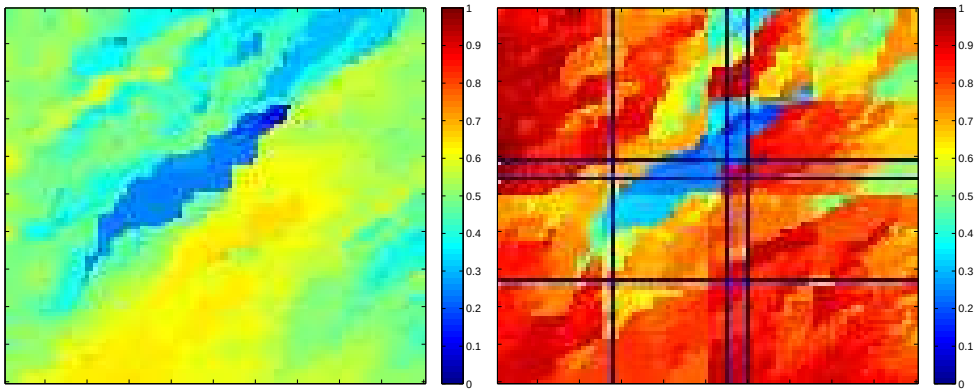


Figure 6.5: (a) Learned tree structure; (b) Learned partitions where the labels correspond to the index of the leaf node in (a)

spaced $x_1, \ldots, x_n \in \mathbb{R}^2$ with $n = 10,000$ on a unit two-way grid $[0,1]^2$. We generate an Erdös-Rényi random graph $G^{1,1} = (V^{1,1}, E^{1,1})$ with the number of vertices $p = 20$, the number of edges $|E| = 10$ and the maximum node degree to be 4 then construct the graphs for each x along diagonals. More precisely, for each pair of i, j, where $1 \leqslant i \leqslant 100$ and $1 \leqslant j \leqslant 100$, we randomly select either $G^{i-1,j}$ (if it exists) or $G^{i,j-1}$ (if it exists) with equal probability as the basis graph. Then, we construct the graph $G^{i,j} = (V^{i,j}, E^{i,j})$ by removing one edge and adding one edge with probability 0.05 based on the selected basis graph and taking care that the number of edge is between 5 and 15 and the maximum degree is still 4. With the underlying graph structures, we generate the covariance matrix and output Y in the same way as in the last section.



(a) (b)

Figure 6.6: (a) Color map of F1-score via applying glasso on the entire dataset; (b) Color map of F1-score learned by our method. Red pixels indicate large values (approaching 1) and blue pixels indicate small values (approaching 0) as shown in the color bar.

We apply the greedy algorithm to learn the dyadic tree structure and corresponding inverse covariance matrices. The learned tree structure and the corresponding partitions are presented in Figure 6.5. We plot the F1-score obtained by glasso on the entire data as compared to our method in Figure 6.6. As we can see, for most x, our method achieves significantly higher F1-score than directly applying glasso. Note that since the graphs around the middle part of the diagonal (line connecting $[0, 1]$ and $[1, 0]$) have the most variability, therefore the F1-scores for both method in this area are relatively low.

6.5.2 *Climate Data Analysis*

In this section ,we apply Go-CART on a meteorology dataset [99], which contains monthly data of 18 different meteorological factors from 1990 to 2002. We use the data from 1990 to 1995 as the training data and data from 1996 to 2002 as the held-out validation data. The observations span 125 locations in the US on an equally spaced grid with the range of latitude from 30.475 to 47.975 and the range of longitude from -119.75 to -82.25. The 18 meteorological factors measured for each month include CO_2, CH_4, H_2, CO, average temperature (TMP), diurnal temperature range (DTR), minimum temperate (TMN), maximum temperature (TMX), precipitation (PRE), vapor (VAP), cloud cover (CLD), wet days (WET), frost days (FRS), global solar radiation (GLO), direct solar radiation (DIR), extraterrestrial radiation (ETR), extraterrestrial normal radiation (ETRN) and UV aerosol index (UV). (for more details, see [99]).

As a comparison to our method, we estimate the sparse graph with all the data from 125 locations using the graphical lasso algorithm, the estimated graph is shown in Figure 6.7 (a). As we can see, there is no edge connecting to any of CO_2, CH_4, H_2 and CO. It contradicts our domain knowledge that these 4 factors are greenhouse gases and should correlate to the solar radiation factors (including GLO, DIR, ETR, ETRN,

and UV) in some way according IPCC report [64], one of the most authoritative reports in the field of meteorology. The reason why the estimated graph against the domain knowledge may be because that we pull all the data together so that the positive correlations at one location might be counteracted by the negative correlations at other locations.

To incorporate the geographical information, we treat the location information (longitude and latitude) for each site as a two-dimensional covariate x. The meteorology data of $p = 18$ factors are treated as the response y. We learn the dyadic tree structure using the greedy algorithm and obtain altogether 87 partitioned subregions as is shown in Figure 6.7. We use different colors and shapes to denote subregions so that no adjacent subregions has both the same color and shape. Note that since there are only 8 colors being utilized, it's possible that some non-adjacent parts can share the same color without causing confusion.

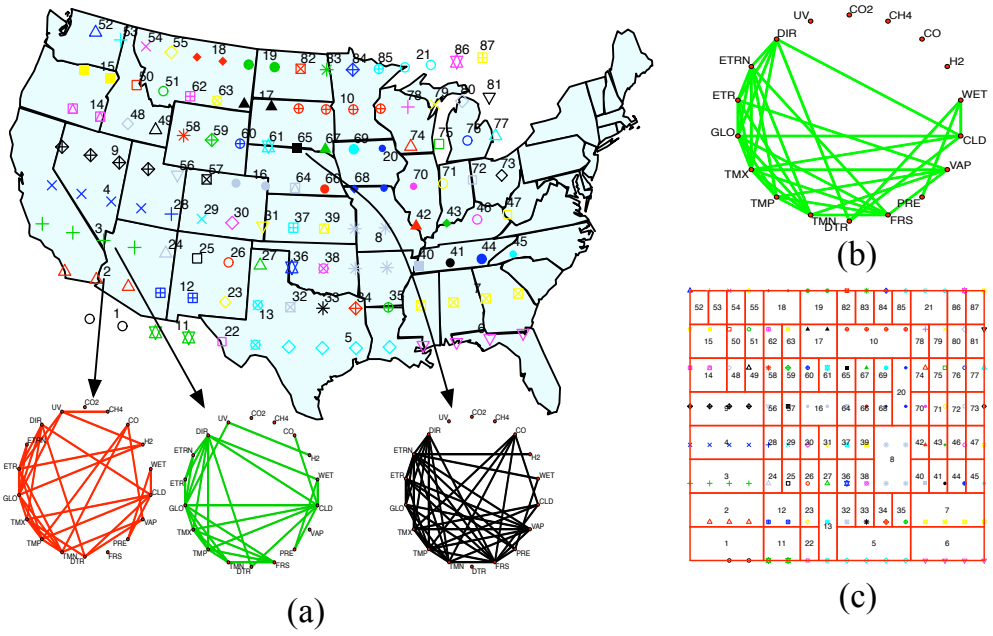


Figure 6.7: The climate data. (a) Learned partitions for the 125 locations and projected to the US map, with the estimated graphs for subregions 2, 3, and 65; (b) Estimated graph with data pooled from all 125 locations; (c): the re-scaled partition pattern induced by the learned dyadic tree structure.

As an illustrative example, we plot the estimated graphs for subregions 2 (corresponding to the strip land from Los Angles, California to Phoenix, Arizona) and subregion 3 (corresponding the strip land from Bakersfield, California to Flagstaff, Arizona) near the Pacific Ocean in the subplots (b) and (c) of Figure 6.7. First, the graphs for these two adjacent subregions are quite similar which suggests some spatial smoothness of the learned graphs. Second, for both graphs, CO is connected solar radiation factors in an either direct or indirect way. And H_2 is connected to UV which are in accordance with the Chapter 7 of IPCC report [64]. For the comparison purpose, we also plot in subplot (d) the estimated graph for subregion 65 which corresponds to the border of South Dakota and Nebraska in the mainland of the

US. Clearly, the graph in Figure 6.7 (d) is quite different from those in (b) and (c). We omit the plots for all 87 subregions due to the space limit. In fact, we find out the graphs corresponding to the locations along the coasts are sparser than those corresponding to the locations in the mainland. This interesting pattern might provide insights to help meteorologists better understand the dependency relationships among these meteorological factors at different locations.

6.6 APPENDIX: TECHNICAL PROOF

Proof of Theorem 6.1

Proof. For any $T \in \mathcal{T}_N$, we denote

$$S_{j,n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_{\mathcal{X}_j})(y_i - \mu_{\mathcal{X}_j})^\top \cdot I(x_i \in \mathcal{X}_j) \tag{6.5}$$

$$\bar{S}_j = \mathbb{E}(Y - \mu_{\mathcal{X}_j})(Y - \mu_{\mathcal{X}_j})^\top \cdot I(X \in \mathcal{X}_j). \tag{6.6}$$

We then have

$$\begin{aligned} &\left|R(T, \mu_T, \Omega_T) - \widehat{R}(T, \mu_T, \Omega_T)\right| \qquad (6.7)\\ \leqslant\ & \left|\sum_{j=1}^{m} \operatorname{tr}\left[\Omega_{\mathcal{X}_j}\left(S_{j,n} - \bar{S}_j\right)\right]\right| + \left|\sum_{j=1}^{m} \log|\Omega_{\mathcal{X}_j}| \cdot \left[\frac{1}{n}\sum_{i=1}^{n} I(x_i \in \mathcal{X}_j) - \mathbb{E}I(X \in \mathcal{X}_j)\right]\right| \\ \leqslant\ & \underbrace{\sum_{j=1}^{m} \|\Omega_{\mathcal{X}_j}\|_1 \cdot \left\|S_{j,n} - \bar{S}_j\right\|_\infty}_{A_1} + \underbrace{\sum_{j=1}^{m}\left|\log|\Omega_{\mathcal{X}_j}|\right| \cdot \left|\frac{1}{n}\sum_{i=1}^{n} I(x_i \in \mathcal{X}_j) - \mathbb{E}I(X \in \mathcal{X}_j)\right|}_{A_2}. \end{aligned}$$

We now analyze the terms A_1 and A_2 separately.

For A_2, using the Hoeffding's inequality, for $\epsilon > 0$, we get

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} I(x_i \in \mathcal{X}_j) - \mathbb{E}I(X \in \mathcal{X}_j)\right| > \epsilon\right) \leqslant 2\exp\left(-2n\epsilon^2\right), \tag{6.8}$$

which implies that,

$$\mathbb{P}\left(\sup_{T \in \mathcal{T}_N}\left|\frac{1}{n}\sum_{i=1}^{n} I(x_i \in \mathcal{X}_j) - \mathbb{E}I(X \in \mathcal{X}_j)\right| / \epsilon_T > 1\right) \leqslant 2\sum_{T \in \mathcal{T}_N}\exp\left(-2n\epsilon_T^2\right), \tag{6.9}$$

where ϵ_T means ϵ is a function of T. For any $\delta \in (0,1)$, we have, with probability at least $1 - \delta/4$,

$$\forall T \in \mathcal{T}_N,\ \left|\frac{1}{n}\sum_{i=1}^{n} I(x_i \in \mathcal{X}_j) - \mathbb{E}I(X \in \mathcal{X}_j)\right| \leqslant \sqrt{\frac{[[T]]\log 2 + \log(8/\delta)}{2n}} \tag{6.10}$$

where $[[T]] > 0$ is the prefix code of T given in (6.2).

From Assumption 6.1, since $\Omega_{\mathcal{X}_j} \in \Lambda_j$, we have that

$$\max_{1 \leqslant j \leqslant m_T} \log\left|\Omega_{\mathcal{X}_j}\right| \leqslant L_n \tag{6.11}$$

Therefore, with probability at least $1 - \delta/4$,

$$A_2 \leqslant L_n m_T \sqrt{\frac{[[T]]\log 2 + \log(8/\delta)}{2n}}. \tag{6.12}$$

Next, we analyze the term A_1. It's obvious that

$$\max_{1\leqslant j\leqslant m_T} \|\Omega_{\mathcal{X}_j}\|_1 \leqslant L_n. \tag{6.13}$$

We only need to bound the term $\left\|S_{j,n} - \bar{S}_j\right\|_\infty$. By Assumption 6.2 and the union bound, we have, for any $\epsilon > 0$,

$$\begin{aligned}
&\mathbb{P}\left(\left\|S_{j,n} - \bar{S}_j\right\|_\infty > \epsilon\right) \\
&\leqslant \mathbb{P}\left(\left\|\frac{1}{n}\sum_{i=1}^n y_i y_i^T I(x_i \in \mathcal{X}_j) - \mathbb{E}\left[YY^T I(X \in \mathcal{X}_j)\right]\right\|_\infty > \frac{\epsilon}{4}\right) && (6.14)\\
&\quad + \mathbb{P}\left(\left\|\frac{1}{n}\sum_{i=1}^n y_i \mu_{\mathcal{X}_j}^T I(x_i \in \mathcal{X}_j) - \mathbb{E}\left[Y\mu_{\mathcal{X}_j}^T I(X \in \mathcal{X}_j)\right]\right\|_\infty > \frac{\epsilon}{4}\right) && (6.15)\\
&\quad + \mathbb{P}\left(\left\|\frac{1}{n}\sum_{i=1}^n \mu_{\mathcal{X}_j} y_i^T I(x_i \in \mathcal{X}_j) - \mathbb{E}\left[\mu_{\mathcal{X}_j} Y^T I(X \in \mathcal{X}_j)\right]\right\|_\infty > \frac{\epsilon}{4}\right) && (6.16)\\
&\quad + \mathbb{P}\left(\left\|\frac{1}{n}\sum_{i=1}^n \mu_{\mathcal{X}_j} \mu_{\mathcal{X}_j}^T I(x_i \in \mathcal{X}_j) - \mathbb{E}\left[\mu_{\mathcal{X}_j} \mu_{\mathcal{X}_j}^T I(X \in \mathcal{X}_j)\right]\right\|_\infty > \frac{\epsilon}{4}\right). && (6.17)
\end{aligned}$$

Using the fact that $\|\mu\|_\infty \leqslant B$ and the Assumption 6.2, we can apply Bernstein's exponential inequality on (6.14), (6.15), and (6.16). Also, since the indicator function is bounded, we can apply Hoeffding's inequality on (6.17). We then obtain:

$$\begin{aligned}
&\mathbb{P}\left(\left\|S_{j,n} - \bar{S}_j\right\|_\infty > \epsilon\right) && (6.18)\\
\leqslant\ & 2p^2 \exp\left(-\frac{1}{32}\left(\frac{n\epsilon^2}{\nu_2 + M_2\epsilon}\right)\right)\\
&+4p^2 \exp\left(-\frac{1}{32B^2}\left(\frac{n\epsilon^2}{\nu_1 + M_1\epsilon}\right)\right)\\
&+2p^2 \exp\left(-\frac{2n\epsilon^2}{B^4}\right).
\end{aligned}$$

Therefore, for any $\delta \in (0, 1/4)$, we have, for any $\epsilon \to 0$ as n goes to infinity, with probability at least $1 - \delta/4$:

$$\begin{aligned}
\forall T \in \mathcal{T}_N,\ \left\|S_{j,n} - \bar{S}_j\right\|_\infty &\leqslant (8\sqrt{\nu_2}) \cdot \sqrt{\frac{[[T]]\log 2 + 2\log p + \log(24/\delta)}{n}} && (6.19)\\
&+ (8B\sqrt{\nu_1}) \cdot \sqrt{\frac{[[T]]\log 2 + 2\log p + \log(48/\delta)}{n}} && (6.20)\\
&+ B^2 \cdot \sqrt{\frac{[[T]]\log 2 + 2\log p + \log(24/\delta)}{2n}} && (6.21)
\end{aligned}$$

Combined with (6.13), we get that

$$A_1 \leqslant C_1 L_n m_T \sqrt{\frac{[[T]]\log 2 + 2\log p + \log(48/\delta)}{n}}. \tag{6.22}$$

where $C_1 = 8\sqrt{\nu_2} + 8B\sqrt{\nu_1} + B^2$.

Since the above analysis holds uniformly over the whole space of $\mathcal{T}_N$, when choosing

$$\text{pen}(T) = (C_1 + 1) L_n m_T \sqrt{\frac{[[T]]\log 2 + 2\log p + \log(48/\delta)}{n}}, \tag{6.23}$$

we then get, with probability at least $1 - \delta/2$,

$$\sup_{T\in\mathcal{T}_N, \mu_j \in M_j, \Omega_j \in \Lambda_j} \left|R(T, \mu_T, \Omega_T) - \widehat{R}(T, \mu_T, \Omega_T)\right| \leqslant \text{pen}(T) \tag{6.24}$$

for large enough n.

Given the uniform deviation inequality in (6.24), we have, for large enough n: for any $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\begin{aligned}
R(\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) &\leqslant \widehat{R}(\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) + \mathrm{pen}(\widehat{T}) \\
&= \inf_{T \in \mathcal{T}_N, \mu_{x_j} \in M_j, \Omega_{x_j} \in \Lambda_j} \left\{ \widehat{R}(T, \mu_T, \Omega_T) + \mathrm{pen}(T) \right\} \\
&\leqslant \inf_{T \in \mathcal{T}_N} \left\{ \widehat{R}(T, \mu_T^*, \Omega_T^*) + \mathrm{pen}(T) \right\} \\
&\leqslant \inf_{T \in \mathcal{T}_N} \{ R(T, \mu_T^*, \Omega_T^*) + 2\mathrm{pen}(T) \} \\
&= \inf_{T \in \mathcal{T}_N} \left\{ \inf_{\mu_{x_j} \in M_j, \Omega_{x_j} \in \Lambda_j} (R(T, \mu_T, \Omega_T) + 2\mathrm{pen}(T) \right\}.
\end{aligned}$$

The desired result of the theorem follows by subtracting R^* from both sides. □

Proof of Theorem 6.2

Proof. From (6.24), we have, for large enough n, on the dataset $\mathcal{D}_1$, with probability at least $1-\delta/4$

$$\sup_{T \in \mathcal{T}_N, \mu_j \in M_j, \Omega_j \in \Lambda_j} \left| R(T, \mu_T, \Omega_T) - \widehat{R}(T, \mu_T, \Omega_T) \right| \leqslant \phi_n(T). \tag{6.25}$$

Follow the same line of analysis, we can also get, on the validation dataset $\mathcal{D}_2$, with probability at least $1 - \frac{\delta}{4}$.

$$\sup_{T \in \mathcal{T}_N} \left| R(T, \widehat{\mu}_T, \widehat{\Omega}_T) - \widehat{R}_{\mathrm{out}}(T, \widehat{\mu}_T, \widehat{\Omega}_T) \right| \leqslant \phi_n(T) \tag{6.26}$$

for large enough n. Where $\widehat{\mu}_T, \widehat{\Omega}_T$ are as defined in (6.4).

Using the fact that

$$\widehat{T} = \mathrm{argmin}_{T \in \mathcal{T}_N} \widehat{R}_{\mathrm{out}}(T, \widehat{\mu}_T, \widehat{\Omega}_T), \tag{6.27}$$

we have, for large enough n: for any $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\begin{aligned}
R(\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) &\leqslant \widehat{R}_{\mathrm{out}}(\widehat{T}, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) + \phi_n(\widehat{T}) \\
&= \inf_{T \in \mathcal{T}_N} \widehat{R}_{\mathrm{out}}(T, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) + \phi_n(\widehat{T}) \\
&\leqslant \inf_{T \in \mathcal{T}_N} \left\{ R(T, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) + \phi_n(T) \right\} + \phi_n(\widehat{T}) \\
&\leqslant \inf_{T \in \mathcal{T}_N} \left\{ \widehat{R}(T, \widehat{\mu}_{\widehat{T}}, \widehat{\Omega}_{\widehat{T}}) + \phi_n(T) + \phi_n(T) \right\} + \phi_n(\widehat{T}) \\
&= \inf_{T \in \mathcal{T}_N} \left\{ 3\phi_n(T) + \inf_{\mu_{x_j} \in M_j, \Omega_{x_j} \in \Lambda_j} R(T, \mu_T, \Omega_T) \right\} + \phi_n(\widehat{T}).
\end{aligned}$$

The result follows by subtracting R^* on both sides. □

Proof of Theorem 6.3

Proof. For any $T \in \mathcal{T}_N$, $\Pi(T^*) \nsubseteq \Pi(T)$, there must exists a subregion $\mathcal{X}' \in \Pi(T)$ such that there does not exist any $\mathcal{A} \in \Pi(T^*)$ which makes $\mathcal{X}' \subset \mathcal{A}$. In this case, we can find a minimal class of disjoint subregions $\{\widetilde{\mathcal{X}}_1, \ldots, \widetilde{\mathcal{X}}_{k'}\} \in \Pi(T^*)$, such that

$$\mathcal{X}' \subset \cup_{i=1}^{k'} \widetilde{\mathcal{X}}_i, \tag{6.28}$$

where $k' \geqslant 2$. We define $\mathcal{X}_i^* = \widetilde{X}_i \cap \mathcal{X}'$ for $i = 1, \ldots, k'$. Then we have

$$\mathcal{X}' = \cup_{i=1}^{k'} \mathcal{X}_i^*. \tag{6.29}$$

Let $\{\mu^0_{\mathcal{X}_j^*}, \Omega^0_{\mathcal{X}_j^*}\}_{j=1}^{k'}$ be the corresponding true parameters on $\widetilde{\mathcal{X}}_1, \ldots, \widetilde{\mathcal{X}}_{k'}$. We denote $R(\mathcal{X}', \mu^0_{T^*}, \Omega^0_{T^*})$ to be the risk of $\mu^0_{T^*}$ and $\Omega^0_{T^*}$ on the subregion $\mathcal{X}'$, then

$$\begin{aligned} & R(\mathcal{X}', \mu^0_{T^*}, \Omega^0_{T^*}) \\ = & \sum_{j=1}^{k'} \mathbb{E}\left[\left(\operatorname{tr}\left[\Omega^0_{\mathcal{X}_j^*}\left((Y-\mu^*_{\mathcal{X}_j^*})(Y-\mu^*_{\mathcal{X}_j^*})^T\right)\right] - \log|\Omega^0_{\mathcal{X}_j^*}|\right) \cdot I(X \in \mathcal{X}_j^*)\right] \\ = & \; p\mathbb{P}\left(X \in \mathcal{X}'\right) - \sum_{j=1}^{k'} \mathbb{P}\left(X \in \mathcal{X}_j^0\right) \log|\Omega^0_{\mathcal{X}_j^*}|. \end{aligned}$$

Since the DPT T does not further partition $\mathcal{X}'$, we have, for any $\mu_T, \Omega_T \in \mathcal{M}_T$:

$$\begin{aligned} R(\mathcal{X}', \mu_T, \Omega_T) &= \sum_{j=1}^{k'} \mathbb{E}\left[\left(\operatorname{tr}\left[\Omega_T\left((Y-\mu_T)(Y-\mu_T)^T\right)\right] - \log|\Omega_T|\right) \cdot I(X \in \mathcal{X}_j^*)\right] \\ &= \sum_{j=1}^{k'} \mathbb{E}\left[\left(\operatorname{tr}\left[\Omega_T\left((Y-\mu_T)(Y-\mu_T)^T\right)\right]\right) \cdot I(X \in \mathcal{X}_j^*)\right] - \mathbb{P}(X \in \mathcal{X}') \log|\Omega_T|. \end{aligned}$$

Since

$$\begin{aligned} (Y-\mu_T)(Y-\mu_T)^T = & (Y-\mu^0_{\mathcal{X}_j^*})(Y-\mu^0_{\mathcal{X}_j^*})^T + (Y-\mu^0_{\mathcal{X}_j^*})(\mu^0_{\mathcal{X}_j^*}-\mu_T)^T \\ & + (\mu^0_{\mathcal{X}_j^*}-\mu_T)(Y-\mu^0_{\mathcal{X}_j^*})^T + (\mu^0_{\mathcal{X}_j^*}-\mu_T)(\mu^0_{\mathcal{X}_j^*}-\mu_T)^T. \end{aligned}$$

This implies that

$$\begin{aligned} & \sum_{j=1}^{k'} \mathbb{E}\left[\left(\operatorname{tr}\left[\Omega_T\left((Y-\mu_T)(Y-\mu_T)^T\right)\right]\right) \cdot I(X \in \mathcal{X}_j^*)\right] \\ = & \sum_{j=1}^{k'} \mathbb{P}\left(X \in \mathcal{X}_j^0\right)\left[\operatorname{tr}(\Omega_T(\Omega_j^*)^{-1}) + \operatorname{tr}(\Omega_T(\mu^0_{\mathcal{X}_j^*}-\mu_T)(\mu^0_{\mathcal{X}_j^*}-\mu_T)^T)\right]. \end{aligned}$$

Using the fact that

$$R(\mathcal{X}', \mu_T, \Omega_T) \geqslant \max\{R(\mathcal{X}', \mu^0_{T^*}, \Omega_T), R(\mathcal{X}', \mu_T, \Omega^0_{T^*})\}, \tag{6.30}$$

We consider the two cases on the R.H.S. separately.
Case 1: The μ's are different.

we know that

$$\begin{aligned}
&\inf_{\mu_T,\Omega_T\in\mathcal{M}_T} R(\mathcal{X}',\mu_T,\Omega_T) - R(\mathcal{X}',\mu^0_{T^*},\Omega^0_{T^*}) \qquad (6.31)\\
&\geqslant \inf_{\mu_T} R(\mathcal{X}',\mu_T,\Omega^0_{T^*}) - R(\mathcal{X}',\mu^0_{T^*},\Omega^0_{T^*})\\
&= \inf_{\mu_T} \sum_{j=1}^{k'} \mathbb{P}\left(X\in\mathcal{X}^0_j\right)(\mu^0_{\mathcal{X}^*_j}-\mu_T)^T\Omega^0_{\mathcal{X}^*_j}(\mu^0_{\mathcal{X}^*_j}-\mu_T)\\
&\geqslant c_1c_2\inf_{\mu_T}\sum_{j=1}^{k'}\|\mu^0_{\mathcal{X}^*_j}-\mu_T\|_2^2
\end{aligned}$$

where the last inequality follows from that fact that $\rho_{\min}(\Omega^0_{\mathcal{X}^*_j}) \geqslant c_1, \mathbb{P}\left(X\in\mathcal{X}^0_j\right) \geqslant c_2$. It's easy to see that a lower bound of the last term is achieved at $\bar{\mu}_T$,

$$\bar{\mu}_T = \frac{1}{k'}\sum_{j=1}^{k'}\mu^0_{\mathcal{X}^*_j}. \qquad (6.32)$$

Furthermore, for any two DPTs T and T', if $\Pi(T)\subset\Pi(T')$. it's obvious that

$$\inf_{\mu_T,\Omega_T\in\mathcal{M}_T} R(T,\mu_T,\Omega_T) \geqslant \inf_{\mu_{T'},\Omega_{T'}\in\mathcal{M}_{T'}} R(T',\mu_{T'},\Omega_{T'}). \qquad (6.33)$$

Therefore, in the sequel, without loss of generality, we only need to consider the case $k'=2$.

The result of this case then follows from the fact that

$$\sum_{j=1}^{2}\|\mu^0_{\mathcal{X}^*_j}-\bar{\mu}_T\|_2^2 = \frac{1}{2}\|\mu_{\mathcal{X}^*_1}-\mu_{\mathcal{X}^*_2}\|_2^2 \geqslant \frac{c_3}{2}. \qquad (6.34)$$

Case 2: The Ω's are different.

In this case, we have

$$\begin{aligned}
&\inf_{\mu_T,\Omega_T\in\mathcal{M}_T} R(\mathcal{X}',\mu_T,\Omega_T) - R(\mathcal{X}',\mu^0_{T^*},\Omega^0_{T^*}) \geqslant \inf_{\Omega_T} R(\mathcal{X}',\mu^0_{T^*},\Omega_T) - R(\mathcal{X}',\mu^0_{T^*},\Omega^0_{T^*})\\
&= \inf_{\Omega_T}\sum_{j=1}^{k'}\mathbb{P}\left(X\in\mathcal{X}^0_j\right)\left(\mathrm{tr}\left[\Omega^{-1}_{\mathcal{X}^*_j}(\Omega_T-\Omega^0_{\mathcal{X}^*_j})\right] - \left(\log|\Omega_T| - \log|\Omega^0_{\mathcal{X}^*_j}|\right)\right)\\
&\geqslant c_2\inf_{\Omega_T}\sum_{j=1}^{k'}\left(\mathrm{tr}\left[\Omega^{-1}_{\mathcal{X}^*_j}(\Omega_T-\Omega^0_{\mathcal{X}^*_j})\right] - \left(\log|\Omega_T| - \log|\Omega^0_{\mathcal{X}^*_j}|\right)\right)\\
&\geqslant c_2\inf_{\Sigma_T}\sum_{j=1}^{k'}\left(\mathrm{tr}\left[\Sigma^0_{\mathcal{X}^*_j}(\Sigma_T^{-1}-\Omega^0_{\mathcal{X}^*_j})\right] + \log\frac{|\Sigma_T|}{|\Sigma^0_{\mathcal{X}^*_j}|}\right)\\
&= c_2\inf_{\Sigma_T}\sum_{j=1}^{k'}\left(\mathrm{tr}\left(\Sigma^0_{\mathcal{X}^*_j}\Sigma_T^{-1}\right) + \log\frac{|\Sigma_T|}{|\Sigma^0_{\mathcal{X}^*_j}|} - p\right)
\end{aligned}$$

where $\Sigma_T = \Omega_T^{-1}$

As discussed before, we only need to consider the case $k'=2$, a lower bound of the last term is achieved at

$$\bar{\Sigma}_T = \frac{\Sigma_{\mathcal{X}^*_1}+\Sigma_{\mathcal{X}^*_2}}{2} \qquad (6.35)$$

Plug-in $\bar{\Sigma}_T$, we get

$$\begin{aligned}
&\inf_{\Sigma_T} \sum_{j=1}^{2} \left(\operatorname{tr}\left(\Sigma^0_{\mathcal{X}^*_j} \Sigma_T^{-1}\right) + \log \frac{|\Sigma_T|}{|\Sigma^0_{\mathcal{X}^*_j}|} - p \right) \geqslant \sum_{j=1}^{2} \left(\operatorname{tr}\left(\Sigma^0_{\mathcal{X}^*_j} \bar{\Sigma}_T^{-1}\right) + \log \frac{|\bar{\Sigma}_T|}{|\Sigma^0_{\mathcal{X}^*_j}|} - p \right) \\
&= \operatorname{tr}\left((2\bar{\Sigma}_T - \Sigma_{\mathcal{X}^*_2}) \bar{\Sigma}_T^{-1}\right) + \log \frac{|\bar{\Sigma}_T|}{|\Sigma_{\mathcal{X}^*_1}|} - p + \operatorname{tr}\left(\Sigma_{\mathcal{X}^*_2} \bar{\Sigma}_T^{-1}\right) + \log \frac{|\bar{\Sigma}_T|}{|\Sigma_{\mathcal{X}^*_2}|} - p \\
&= \log \frac{|\bar{\Sigma}_T|}{|\Sigma_{\mathcal{X}^*_1}|} + \log \frac{|\bar{\Sigma}_T|}{|\Sigma_{\mathcal{X}^*_2}|} \\
&= 2 \log \left| \frac{\Sigma_{\mathcal{X}^*_1} + \Sigma_{\mathcal{X}^*_2}}{2} \right| - \log |\Sigma_{\mathcal{X}^*_1}| - \log |\Sigma_{\mathcal{X}^*_2}| \\
&\geqslant c_4.
\end{aligned}$$

where the last inequality follows from the given assumption.

Therefore, we have

$$\inf_{\mu_T, \Omega_T \in \mathcal{M}_T} R(\mathcal{X}', \mu_T, \Omega_T) - R(\mathcal{X}', \mu^0_{T^*}, \Omega^0_{T^*}) \geqslant c_2 c_4. \tag{6.36}$$

The desired result of theorem is obtained by combining the above discussed two cases. □

7

MARKOV FOREST REGRESSION

In the previous chapter, we proposed a new problem of estimating the Markov network (a.k.a. undirected graphical model) of a random vector Y conditioned on another random vector X as input, referred to as "graph-valued regression". We further proposed a partition-based estimator called graph-optimized CART (Go-CART) to address this problem. However, Go-CART requires Y to be continuous and follow a conditional Gaussian distribution, which greatly restricts the model's applicability. In this chapter, we propose Markov forest regression as an important complementary approach, where a conditional forest-structured Markov network is estimated, allowing both discrete and continuous Y. In the proposed forest-optimized CART (Fo-CART) estimator, we build a dyadic partitioning tree on X where the set of leaf nodes defines a partition of the input space, and estimate a forest-structured Markov network on each leaf node of the tree. We also empirically demonstrate the usefulness of the proposed methods on both simulated and real datasets.

7.1 INTRODUCTION AND MOTIVATION

A useful way to explore the structure of a distribution P for the random vector $Y = (Y_1, \ldots, Y_d) \in \mathbb{R}^d$ is to estimate its Markov network, or undirected graph [83, 43]. This encodes the structure of P into an undirected graph $G = (V, E)$, where the vertex set V corresponds to $Y_1, \ldots, Y_d$. The edge set encodes conditional independencies among components of Y; an edge is missing between Y_u and Y_v if and only if Y_u and Y_v are conditionally independent given the rest of the variables. To avoid overfitting, it is often assumed that the graph is *sparse*, with a small number of edges.

For continuous Y, a large body of literature assumes that its corresponding distribution P is multivariate Gaussian $N(\mu, \Sigma)$. By optimizing the ℓ_1-regularized log-likelihood, one can recover the underlying Markov network via the estimated inverse covariance matrix [152, 5, 49]. For binary Y, one approach is to assume a discrete Gaussian distribution—an Ising model—and estimate the corresponding Markov network by solving an approximate sparse maximum likelihood problem [5, 118]. In addition to such parametric models, another tractable class of sparse graphical models is the set of forest-structured models. Recently, significant progress has been made on estimating forest- or tree-structured graphical models [23, 127, 129, 93, 128, 32]. For both discrete and continuous Y, the forest structure can be estimated via a Chow-Liu algorithm [33] with a thresholding procedure [129, 93].

Existing literature mainly focuses on estimating the Markov network for a high dimensional random vector Y. However, for many im-

portant applications, the data have both a high dimensional response variable Y and a vector of covariates X. In this setting, it is of interest to estimate the Markov network of Y as a function of X. This problem is known as *graph-valued regression* [92]. In particular, let $P(\cdot|X)$ be the conditional distribution of Y given X. For the case where Y is continuous, Liu et al. [92] assumed that $P(\cdot|X = x)$ follows a Gaussian distribution $N(\mu(x), \Sigma(x))$ and proposed a penalized maximum log-likelihood estimator, named graph-optimized CART (or Go-CART) [92], to estimate the conditional graph structure. A kernel smoothing based approach was further proposed in [76] for continuous Y but it is limited in that it does not partition the input space X, and requires that X is very low-dimensional.

In this paper, we address the problem of estimating a conditional discrete Markov network. It is not straightforward to extend Go-CART to handle a discrete response. The main challenge is that Go-CART adopts a likelihood-based framework. Most methods for estimating discrete Markov networks either resort to pseudo-likelihood based inference or approximation algorithms, due to the intractable computation of the partition function in likelihood [5, 118]. To handle discrete Y, we relax the conditional Gaussian assumption and propose the *Markov forest regression* as a complement to Go-CART. Markov forest regression estimates a forest-structured Markov network $F(x)$ associated with the conditional distribution $P(\cdot|X = x)$. Note that we do not assume the Markov network of the ground true conditional distribution $P(\cdot|X = x)$ is a forest; rather, we try to estimate a forest-structured conditional distribution $\widehat{P}(\cdot|X = x)$ that best approximates $P(\cdot|X = x)$. Since the likelihood of a forest-structured distribution factors into univariate and bivariate marginals and can be easily computed, a partition-based maximum log-likelihood estimator can be defined in a similar way as in Go-CART. In particular, let $\mathcal{X}$ denote the domain of X. We partition $\mathcal{X}$ into finitely many regions $\mathcal{X}_1, \dots \mathcal{X}_m$. For each partition element $\mathcal{X}_j$, we estimate a forest $\widehat{F}_{\mathcal{X}_j}$ for the corresponding Y, and take $\widehat{F}(x) = \widehat{F}_{\mathcal{X}_j}$ for all $x \in \mathcal{X}_j$. To find the partition, we adopt the basic technique behind classification and regression trees (CART), recursively splitting the domain $\mathcal{X}$ into small hyperrectangles to build up a partitioning tree. We use log-likelihood as the search criterion for the best partitioning tree structure using a forest-structured model at each leaf node. We refer to this method as Forest-optimized CART, or Fo-CART.

The Fo-CART estimator complements Go-CART [92] and can naturally handle both discrete and continuous response vectors without assuming a conditional Gaussian distribution. In this paper, we focus on the case of discrete Y, but our method easily extends to the continuous case. The only difference is that for discrete data, univariate and bivariate marginals for learning the forest structure are estimated via normalized counts; in the continuous case, they are estimated using the nonparametric kernel density estimation.

7.2 BACKGROUND

In this section, we introduce the necessary building-blocks of FoCART .

Forest Learning. We first consider the simple case where we only have Y and are interested in estimating its forest-structured Markov network. For simplicity, we assume a common domain for each component of Y, denoted as $\mathcal{Y}$. Let $\mathcal{T}_d$ be the family of trees with d nodes and $\mathcal{F}_d$ the family of forests with d nodes. From Cayley's theorem [21], we know that the number of possible forests in $\mathcal{F}_d$ is upper bounded by $(d+1)^{d-1}$. Let P_F be a forest-structured distribution for Y which is Markov to $F = (V, E(F)) \in \mathcal{F}_d$. It is known that the joint density of P_F factorizes as follows [83]:

$$p_F(y) = \text{prod}_{u=1}^{d} p(y_u) \text{prod}_{(u,v)\in E(F)} \frac{p(y_u, y_v)}{p(y_u)\, p(y_v)}, \tag{7.1}$$

where $p(y_u)$ and $p(y_u, y_v)$ are univariate and bivariate marginals. We define $\mathcal{P}(\mathcal{F}_d)$ be the family of distributions supported by graphs in $\mathcal{F}_d$: $\mathcal{P}(\mathcal{F}_d) = \{P_F$ takes the form (7.1), for some $F \in \mathcal{F}_d\}$, always assuming that the corresponding densities exist (with respect to some fixed base measure).

Suppose we are given n i.i.d. samples $y^{(1)}, \ldots, y^{(n)}$ drawn from the distribution P; the true Markov network of P does not have to be a forest, i.e., P may not belong to $\mathcal{P}(\mathcal{F}_d)$. The oracle forest density q^* is defined as

$$q^* = \underset{q\in\mathcal{P}(\mathcal{F}_d)}{\arg\min} -\mathbb{E}_P[\log q(Y)] = \underset{q\in\mathcal{P}(\mathcal{F}_d)}{\arg\min} D(p \,\|\, q), \tag{7.2}$$

where $D(p \,\|\, q)$ is the KL-divergence between p and q. Our goal is to estimate a forest structure $\widehat{F}$ having an associated density $\widehat{p}_{\widehat{F}}$ that best approximates q^*.

Let e be the edge connecting Y_u and Y_v. We denote the mutual information corresponding to e by:

$$I(e) \equiv I(u, v) = \sum_{(y_u, y_v)\in\mathcal{Y}^2} p(y_u, y_v) \log \frac{p(y_u, y_v)}{p(y_u)\, p(y_v)}. \tag{7.3}$$

The forest structure $\widehat{F}$ with the associated density $\widehat{p}_{\widehat{F}}$ can be estimated via a thresholded Chow-Liu based algorithm [129, 93]:

1. Given the data $y^{(1)}, \ldots, y^{(n)}$, estimate univariate marginals $\{\widehat{p}(y_u)\}_{u\in V}$ and bivariate marginals $\{\widehat{p}(y_u, y_v)\}_{1\leqslant u<v\leqslant d}$. Compute the set of empirical mutual information quantities:

$$I(u, v) \equiv \sum_{(y_u, y_v)\in\mathcal{Y}^2} \widehat{p}(y_u, y_v) \log \frac{\widehat{p}(y_u, y_v)}{\widehat{p}(y_u)\, \widehat{p}(y_v)} \quad \text{for all} \quad 1 \leqslant u < v \leqslant d.$$

2. Run a maximum-weight spanning tree algorithm [77] to obtain an edge set of the tree: $E(\widehat{T}) \equiv \arg\max_{E(T):T\in\mathcal{T}_d} \sum_{e\in E(T)} I(e)$ [33].

3. Define a threshold ϵ. Prune the tree $\widehat{T}$ to a forest $\widehat{F}$ by selecting those edges $\widehat{e}$ such that $I(\widehat{e}) \geqslant \epsilon$: $E(\widehat{F}) = \{\widehat{e} \in E(\widehat{T}) : I(\widehat{e}) \geqslant \epsilon\}$, The corresponding density estimator is defined as $\widehat{p}_{\widehat{F}} = \text{prod}_{u \in V} \widehat{p}(y_u) \text{prod}_{(u,v) \in E(\widehat{F})} \frac{\widehat{p}(y_u, y_v)}{\widehat{p}(y_u)\,\widehat{p}(y_v)}$.

The first two steps constitute the Chow-Liu algorithm; the third thresholding step addresses overfitting by pruning a tree to a more sparse forest. For discrete Y, the estimated marginals are simply the normalized counts of each observed symbol in $\mathcal{Y}$ and $\mathcal{Y}^2$. The threshold ϵ can be set to be $\epsilon = n^{-\beta}$ according to [129], where β is a tuning parameter between 0 and 1. For continuous Y, Liu et al. [93] propose to estimate the marginals via kernel density estimation and the threshold ϵ is tuned based on validation data. For both cases, the above algorithm is both structure consistent and risk consistent. In this paper, we mainly consider discrete Y but our algorithm can be easily extended to the continuous case.

Dyadic Partitioning Trees. To obtain the partition of the input space, we adopt dyadic partitioning trees (DPT), as in [92]. DPTs has been widely applied to regression and classification tasks, and enjoy strong theoretical properties [89, 123]. For simplicity, we assume the domain of $X \in \mathbb{R}^b$ to be $\mathcal{X} = [0,1]^b$. A DPT T defined over $\mathcal{X}$ is constructed by recursively dividing $\mathcal{X}$ by means of *axis-orthogonal dyadic splits*. Each node of T is associated with a hyperrectangle in $\mathcal{X} = [0,1]^b$ and the root node corresponds to $\mathcal{X}$ itself. Given a DPT T, the set of leaf nodes defines a partition of $\mathcal{X}$ that denote by $\Pi(T) = \{\mathcal{X}_1, \ldots, \mathcal{X}_{m_T}\}$. To restrict the complexity of the class DPTs, we introduce a dyadic integer $N = 2^K$ and define $\mathcal{T}_N$ to be the collection of all DPTs such that no partition $\mathcal{X}_j$ has a side length smaller than $\frac{1}{N} = 2^{-K}$. A prefix code $[[T]]$ on the set of DPTs $T \in \mathcal{T}_N$ satisfies $\sum_{T \in \mathcal{T}_N} 2^{-[[T]]} \leqslant 1$. A specific prefix code in [123] takes the form: $[[T]] = 3|\Pi(T)| - 1 + (|\Pi(T)| - 1)\log b / \log 2$ with a simple upper bound $[[T]] \leqslant (3 + \log b / \log 2)|\Pi(T)|$.

7.3 FOREST-OPTIMIZED CART ESTIMATOR

In this section, we introduce our Forest-optimized CART estimator (Fo-CART). Fo-CART is a partition based conditional forest estimator. Given a DPT $T \in \mathcal{T}_N$, $\mathcal{X} = [0,1]^b$ is partitioned into m_T connected hyperrectangles induced by the leaf nodes, $\Pi(T) = \{\mathcal{X}_1, \ldots, \mathcal{X}_{m_T}\}$. For each partition element $\mathcal{X}_j$, we estimate a forest structure $\widehat{F}_{\mathcal{X}_j}$ and take $\widehat{F}(x) = \widehat{F}_{\mathcal{X}_j}$ for all $x \in \mathcal{X}_j$.

Let $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ be n i.i.d. samples from the joint distribution of (X, Y). Let $F_T(x)$ and $p_{F_T}(x)$ be the forest structure and the corresponding density associated with T:

$$F_T(x) = F_{\mathcal{X}_j} \qquad \text{and} \qquad p_{F_T}(x) = p_{F_{\mathcal{X}_j}} \quad \text{if} \quad x \in \mathcal{X}_j. \tag{7.4}$$

Given a DPT T and the corresponding $F_T(x)$ and $p_{F_T}(x)$, the negative

log-likelihood risk $R(T, F_T, p_{F_T})$ and its sample version $\widehat{R}(T, F_T, p_{F_T})$ are defined as follows:

$$R(T, F_T, p_{F_T}) = \sum_{j=1}^{m_T} \mathbb{E}\Big[I(X \in \mathcal{X}_j) \cdot \Big\{ \sum_{u=1}^{d} \log p_{\mathcal{X}_j}(y_u) + \sum_{(u,v)\in E(F_{\mathcal{X}_j})} \log \frac{p_{\mathcal{X}_j}(y_u, y_v)}{p_{\mathcal{X}_j}(y_u) p_{\mathcal{X}_j}(y_v)} \Big\} \Big] \tag{7.5}$$

$$\widehat{R}(T, F_T, p_{F_T}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m_T} I(x^{(i)} \in \mathcal{X}_j) \Big\{ \sum_{u=1}^{d} \log p_{\mathcal{X}_j}(y_u^{(i)}) + \sum_{(u,v)\in E(F_{\mathcal{X}_j})} \log \frac{p_{\mathcal{X}_j}(y_u^{(i)}, y_v^{(i)})}{p_{\mathcal{X}_j}(y_u^{(i)}) p_{\mathcal{X}_j}(y_v^{(i)})} \Big\}, \tag{7.6}$$

where $p_{\mathcal{X}_j}$ is the marginal density of Y with the corresponding $X \in \mathcal{X}_j$. With this notation in place, we define our Fo-CART estimator:

Definition 7.1. ***Penalized empirical risk minimization Fo-CART estimator:***

$$\widehat{T}, \Big\{ \widehat{F}_{\widehat{\mathcal{X}}_j}, \widehat{p}_{\widehat{F}_{\widehat{\mathcal{X}}_j}} \Big\}_{j=1}^{m_{\widehat{T}}} = \operatorname{argmin}_{T \in \mathcal{T}_N, F_T, p_{F_T}} \Big\{ \widehat{R}(T, F_T, p_{F_T}) + \gamma_n \cdot \mathrm{pen}(T) \Big\}, \tag{7.7}$$

where $\widehat{R}$ is defined in (7.6) and $\mathrm{pen}(T) = d m_T \sqrt{\frac{4\log(n) + [[T]] \log 2}{2n}}$ with tuning parameter γ_n.

It is difficult to tune γ_n empirically. Therefore, we define a more practical held-out risk minimization estimator as follows. We split our dataset $\mathcal{D}$ into two sets, training data $\mathcal{D}_1 = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n_1)}, y^{(n_1)})\}$ and held-out validation data $\mathcal{D}_2 = \{(\widetilde{x}^{(1)}, \widetilde{y}^{(1)}), \ldots, (\widetilde{x}^{(n_2)}, \widetilde{y}^{(n_2)})\}$ with $n_1 + n_2 = n$. Following equation (7.6), the held-out negative log-likelihood risk is given by

$$\widehat{R}_{\mathrm{out}}(T, F_T, p_{F_T}) = \frac{1}{n_2} \sum_{i=1}^{n_2} \sum_{j=1}^{m_T} I(\widetilde{x}^{(i)} \in \mathcal{X}_j) \Big\{ \sum_{u=1}^{d} \log p_{\mathcal{X}_j}(\widetilde{y}_u^{(i)}) + \sum_{(u,v)\in E(F_{\mathcal{X}_j})} \log \frac{p_{\mathcal{X}_j}(\widetilde{y}_u^{(i)}, \widetilde{y}_v^{(i)})}{p_{\mathcal{X}_j}(\widetilde{y}_u^{(i)}) p_{\mathcal{X}_j}(\widetilde{y}_v^{(i)})} \Big\}. \tag{7.8}$$

Definition 7.2. *For each DPT $T \in \mathcal{T}_N$ with the induced partition $\Pi(T) = \{\mathcal{X}_j\}_{j=1}^{m_T}$, define*

$$\Big\{ \widehat{F}_{\mathcal{X}_j}, \widehat{p}_{\widehat{F}_{\mathcal{X}_j}} \Big\}_{j=1}^{m_T} = \operatorname{argmin}_{F_T, p_{F_T}} \widehat{R}(T, F_T, p_{F_T}),$$

where the risk $\widehat{R}$ is evaluated on the training set $\mathcal{D}_1$. The forest $\widehat{F}_T$ and density $\widehat{p}_{\widehat{F}_T}$ are constructed as in (7.4). We then search for the best DPT $\widehat{T}$ using the held-out validation data:

$$\widehat{T} = \operatorname{argmin}_{T \in \mathcal{T}_N} \widehat{R}_{\mathrm{out}}(T, \widehat{F}_T, \widehat{p}_{\widehat{F}_T}).$$

where $\widehat{R}_{\mathrm{out}}$ as defined in (7.8) is evaluated on $\mathcal{D}_2$. The held-out risk minimization Fo-CART estimator is $\widehat{T}$ with its induced partition $\Pi(\widehat{T}) = \{\widehat{\mathcal{X}}_j\}_{j=1}^{m_{\widehat{T}}}$ and $\{\widehat{F}_{\widehat{\mathcal{X}}_j} \widehat{p}_{\widehat{F}_{\widehat{\mathcal{X}}_j}}\}_{j=1}^{m_{\widehat{T}}}$.

7.4 COMPUTATIONAL ALGORITHM

Exhaustive search of the best DPT in $\mathcal{T}_N$ could be computationally very expensive for large b, even using the dynamic programming scheme in [13]. As in [92], we adopt a greedy tree learning algorithm to find a DPT $\widehat{T}$ with the corresponding $\{\widehat{F}_{\widehat{\mathcal{X}}_j}, \widehat{p}_{\widehat{F}_{\widehat{\mathcal{X}}_j}}\}_{j=1}^{m_{\widehat{T}}}$. We focus on the held-out risk minimizer in Definition 7.2 due to its superior empirical performance. However, the greedy tree learning algorithm also applies to the penalized empirical risk minimization form in Definition 7.1.

Given a small hyperrectangle $\mathcal{A}$ associated to a node of a DPT T, let $\mathcal{D}(\mathcal{A}) = \{i \in \{1, \ldots, n_1\} : x^{(i)} \in \mathcal{A}\}$ be indices of the training samples that fall into $\mathcal{A}$. We estimate all univariate and bivariate marginals $\widehat{p}_{\mathcal{A}}$ using the data $\{y^{(i)} : i \in \mathcal{D}(\mathcal{A})\}$. We then use the thresholded Chow-Liu algorithm as described in Section 2 to estimate the forest structure $\widehat{F}_{\mathcal{A}}$ with the density $\widehat{p}_{\widehat{F}_{\mathcal{A}}}$ and compute the held-out negative log-likelihood $\widehat{R}_{\text{out}}(\mathcal{A}, \widehat{F}_{\mathcal{A}}, \widehat{p}_{\widehat{F}_{\mathcal{A}}})$ using the validation data $\mathcal{D}_2 = \{(\widetilde{x}^{(1)}, \widetilde{y}^{(1)}), \ldots, (\widetilde{x}^{(n_2)}, \widetilde{y}^{(n_2)})\}$:

$$\widehat{R}_{\text{out}}(\mathcal{A}, \widehat{F}_{\mathcal{A}}, \widehat{p}_{\widehat{F}_{\mathcal{A}}}) = \frac{1}{n_2} \sum_{i:\widetilde{x}^{(i)} \in \mathcal{A}} \Big\{ \sum_{u=1}^{d} \log \widehat{p}_{\mathcal{A}}(\widetilde{y}_u^{(i)}) + \sum_{(u,v) \in E(\widehat{F}_{\mathcal{A}})} \log \frac{\widehat{p}_{\mathcal{A}}(\widetilde{y}_u^{(i)}, \widetilde{y}_v^{(i)})}{\widehat{p}_{\mathcal{A}}(\widetilde{y}_u^{(i)}) p_{\mathcal{A}}(\widetilde{y}_v^{(i)})} \Big\}. \tag{7.9}$$

Note that we tune the threshold ϵ, which gives different forest structures and choose the one that leads to the minimum held-out negative log-likelihood $\widehat{R}_{\text{out}}(\mathcal{A}, \widehat{F}_{\mathcal{A}}, \widehat{p}_{\widehat{F}_{\mathcal{A}}})$.

The greedy tree learning algorithm starts from the coarsest partition $\mathcal{X} = [0,1]^b$ and then computes the decrease in the held-out risk by dyadically splitting each hyperrectangle $\mathcal{A}$ along dimension $k \in \{1, \ldots b\}$. For each split, we pick the dimension k^* that leads to the largest decrease in the held-out risk:

$$k^* = \operatorname*{arg\,max}_{k \in \{1,\ldots,b\}} \widehat{R}_{\text{out}}(\mathcal{A}, \widehat{F}_{\mathcal{A}}, \widehat{p}_{\widehat{F}_{\mathcal{A}}}) - \widehat{R}_{\text{out}}(\mathcal{A}_L^{(k)}, \widehat{F}_{\mathcal{A}_L^{(k)}}, \widehat{p}_{\widehat{F}_{\mathcal{A}_L^{(k)}}}) - \widehat{R}_{\text{out}}(\mathcal{A}_R^{(k)}, \widehat{F}_{\mathcal{A}_R^{(k)}}, \widehat{p}_{\widehat{F}_{\mathcal{A}_R^{(k)}}}),$$

where $\mathcal{A}_L^{(k)}$ and $\mathcal{A}_R^{(k)}$ are the left and right children obtained by dyadically splitting $\mathcal{A}$ along the dimension k. If splitting any dimension k of $\mathcal{A}$ leads to an increase in the held-out risk, $\mathcal{A}$ should no longer be split and hence becomes a partition element of $\Pi(T)$.

7.5 EXPERIMENTAL RESULTS

We evaluate the performance of the greedy version of the Fo-CART estimator for discrete Y on both synthetic and real datasets. For all experiments, the threshold $\epsilon = n_1^{-\beta}$ is tuned on the validation data using the method described in Section 7.4 with $\beta \in \{0.05, 0.1, 0.15, \ldots, 0.95\}$; the dyadic integer N is set to 2^{10}. In addition, to guarantee a reasonable empirical estimate of the marginals, we always ensure that each leaf node contains at least 10 data points.

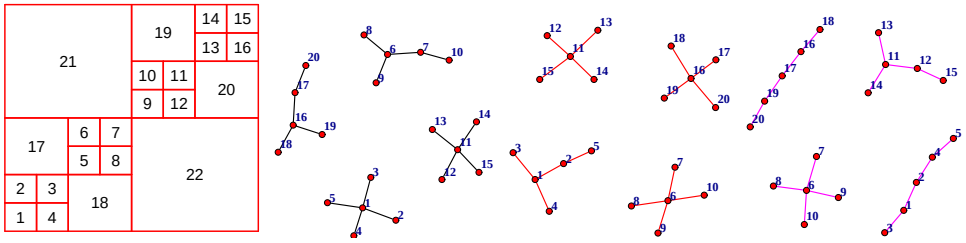


Figure 7.1: (a) The 22 subregions defined on $[0,1]^2$. The horizontal axis corresponds to the first dimension denoted as X_1 while the vertical axis corresponds to the second dimension denoted as X_2. The bottom left point corresponds to $[0,0]$ and the upper right point corresponds to $[1,1]$. (b) (c) (d) The ground true forest for the subregion 1, 18, and 22.

7.5.1 *Simulation Study*

We generate n data points $x^{(1)}, \ldots, x^{(n)}$ uniformly distributed on the unit hypercube $[0,1]^b$ with $b = 10$. We split the first two dimensions into 22 subregions as shown in Figure 7.1 (a). For the t-th subregion where $1 \leqslant t \leqslant 22$, we generate a forest F^t of $d = 20$ vertices as follows: F^t consists of 4 random disconnected subtrees, $F^t = \{T_1^t, T_2^t, T_3^t, T_4^t\}$, each of size 5. As an illustration, the randomly generated forest for subregions 1, 18 and 22 are shown in Figure 7.1 (b), (c) and (d) respectively. For each data point $x^{(i)}$ falling in to the t-th subregion, we associate a 20-dimensional binary response vector $y^{(i)} \in \{0,1\}^{20}$ as follows. For each subtree of F^t, we randomly choose a node u as the root and assign $y_u^{(i)} = 0$ and $y_u^{(i)} = 1$ with equal probability 0.5. Then we traverse the subtree to determine the parent node for each node via a breadth-first search (BFS). For any node v in this subtree with its parent $pa(v)$, we assign $y_v^{(i)} = 1 - y_{pa(v)}^{(i)}$ with probability 0.8 and $y_v^{(i)} = y_{pa(v)}^{(i)}$ with probability 0.2.

We compare Fo-CART with the Go-CART estimator of [92]. Since Y is binary, Go-CART cannot be directly applied due to the difficulty of computing the partition function in the Ising model. Here, we adopt the approximate log-likelihood proposed in [5] which gives an upper bound of the partition function. The tuning parameter for the ℓ_1-regularization in Go-CART is chosen from a very large value (resulting in empty graphs for all leaf nodes of the DPT) to a small value (resulting in full graphs) using the held-out negative log-likelihood criteria. We conduct 100 Monte-Carlo simulations for $n = 5000$ and $n = 10000$. Both Fo-CART and Go-CART never wrongly split on any of the irrelevant dimensions, i.e. X_3 to X_{10}. As a comparison between Fo-CART and Go-CART, we report the number of times that the algorithm correctly recovers the ground true partition as in Figure 7.1(a). For those simulations where we correctly identify the partitions, we list the graph estimation performance for subregions 1, 18, 22 which represents the small, middle and large region respectively. For each subregion, let $\widehat{E}$ be the estimated edge set while E is the true edge set.

Table 7.1: Comparison of Fo-CART and Go-CART with $n = 5000$ and $n = 10000$: for those simulation where the true partitions are correctly recovered, we report the mean value (standard deviation) for precision, recall and F1-score.

n=5000	Fo-CART			Go-CART		
# of correct partition recovering	64 / 100			54 / 100		
Subregion	1	18	22	1	18	22
Precision	0.9773 (0.05)	0.9978 (0.01)	0.9919 (0.03)	0.6366 (0.08)	0.7564 (0.10)	0.8417 (0.08)
Recall	0.9896 (0.03)	1.0000 (0.00)	1.0000 (0.00)	0.9971 (0.02)	1.0000 (0.00)	1.0000 (0.00)
F1-Score	0.9831 (0.04)	0.9989 (0.01)	0.9957 (0.01)	0.7743 (0.06)	0.8572 (0.06)	0.9110 (0.05)
n=10000	Fo-CART			Go-CART		
# of correct partition recovering	79 / 100			68 / 100		
Subregion	1	18	22	1	18	22
Precision	0.9913 (0.03)	0.9956 (0.02)	0.9958 (0.02)	0.7303 (0.08)	0.7993 (0.09)	0.8111 (0.10)
Recall	0.9984 (0.01)	1.0000 (0.00)	1.0000 (0.00)	1.0000 (0.00)	1.0000 (0.00)	1.0000 (0.00)
F1-Score	0.9947 (0.02)	0.9977 (0.01)	0.9978 (0.01)	0.8417 (0.05)	0.8849 (0.06)	0.8914 (0.06)

The graph estimation performance is measured in terms of precision, recall and F1-score defined as follows: precision $= |\widehat{E} \cap E|/|\widehat{E}|$, recall $= |\widehat{E} \cap E|/|E|$, F1-score $= (2 \cdot \text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$. The results are presented in Table 7.1.

As we see from the Table 7.1, in 100 runs, Fo-CART recovers the true partition with higher probability, and it outperforms Go-CART in terms of graph estimation. For Go-CART, we see that the recall is close to one, but the precision is low, which indicates that Go-CART tends to estimate an overly dense graph. This is expected since Go-CART does not utilize the fact that the underlying graph structure is a forest. For both methods, the graph estimation is better for larger subregions (e.g. subregion 22) than small ones (e.g. subregion 1); this is simply because that large region contains more data points. In addition, for $n = 5000$ where subregion 1 only contains about $5000/64 \approx 78$ data points, the graph estimation is almost perfect for Fo-CART, with an F1-Score of 0.9831.

7.5.2 *Stock Data Analysis*

We apply Fo-CART to analyze the relationship among stocks of different companies Y as a function of oil price X. For better visualization, we choose $d = 53$ well-known companies from the S&P 500 with at least 100,000 employees. Instead of considering the raw stock price, which could be at very different scales for different companies, we measure the *log-return*, defined as the logarithm ratio of the stock price at time t to its previous time $t - 1$. We take the sign of the log return as our output Y. We collect the data in a similar approach as in [76] from Jan 1, 2003 to Dec 31, 2005 with in total $n = 749$ data points

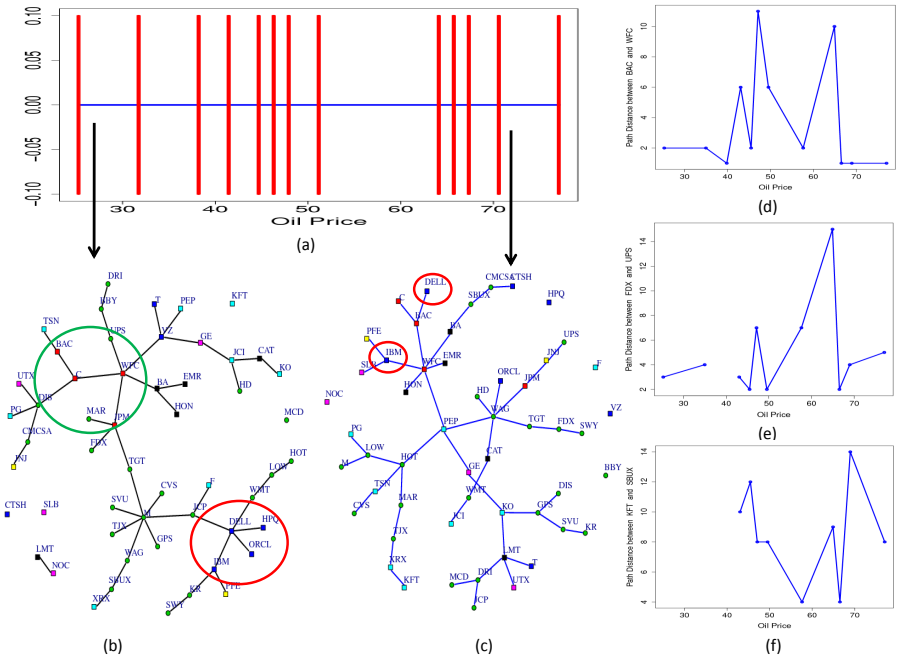


Figure 7.2: Results for the analysis of stock prices vs. oil price

[1]. In sum, our data consists of $\{x^{(i)}, y^{(i)}\}_{i=1}^{749}$, where each $x^{(i)} \in \mathbb{R}$ is the oil price and $y^{(i)} \in \mathbb{R}^{53}$ is the sign of the log-return of the stock prices. We split the data randomly in half for training and validation.

By learning a DPT, we split the oil price space $\mathcal{X}$ as shown in Figure 7.2 (a). As we can see, our DPT does not split $\mathcal{X}$ when the oil price is very high (above 70$), very low (below 30$) and between around 50$ and 65$. This indicates that the forest structure (and hence the dependency of stock between companies) is quite stable in these price ranges. In contrast, for other price ranges with many splits, the relationship on stock portfolio between companies is very sensitive to the oil price.

For each partition, we estimate a forest structure. As an illustrative example, we plot the forest structures corresponding to the lowest oil price range and highest oil price range in Figure 7.2 (b) and (c) respectively. We use different colors and shapes of vertices to represent the companies in different categories: for example, *IT technology* (DELL, IBM, HPQ (Hewlett-Packard), ORCL (Oracle), etc), *Financial* (BAC (Bank of America), C (Citi), JPM (JP Morgan Chase), WFC (Walls Fargo), etc), *Services* (e.g. FDX (FedEx), UPS, MCD (McDonald), etc). There are some interesting observations. For example, for IT companies, when the oil price is low, they form a small cluster with a strong dependency as shown in Figure 7.2 (b) by a red circle. A similar observation can also be made for the financial companies. When the oil price is high, the IT companies become more separated, as shown in Figure 7.2 (c). Interestingly, in Figure 7.2 (c), although DELL and IBM (in red circle) are not directly connected and hence conditionally independent, the shortest path between them passes BAC, WFC. This might suggest that

1 The stock data is collected from http://www.finance.yahoo.com and the oil price is from http://tonto.eia.doe.gov.

when the price of oil is high, DELL and IBM are related to each other through the financial companies.

We also investigate the relationship between pairs of stocks. For a pair of companies, we measure the shortest distance between them in the forest (counts of edges) at each partition and plot the distance as a function of oil price (see Figure 7.2 (d)–(f)). If there is no path between two companies, the distance is set to infinity. An interesting observation is that for companies that sell similar products or provide similar services, the distances between them as a function of oil price often shows a "bimodal" pattern. For example, as shown in Figure (d) for BAC and WFC and Figure (e) for FDX and UPS, the distance between them is small either when the oil price is low or high and the distance is large with two peaks around $40 to $50 and $60 to $65.

Part V

SPARSE LEARNING FOR TEXT MINING

8

LEARNING PREFERENCES WITH MILLIONS OF PARAMETERS BY ENFORCING SPARSITY

In the previous chapters of the thesis, we have demonstrated the applications of sparse learning to tumor classification, genome-wide association study, climate data analysis as well as stock analysis. In addition to these, another important application domain of sparse learning is text mining, since text data are usually ultra-high dimensional and large-scale. In this part, we study two applications of sparse learning to text mining tasks, learning preferences and latent semantic analysis.

In the first chapter, we study the retrieval task that ranks a set of objects for a given query in the pairwise preference learning framework. Recently researchers found out that raw features (e.g. words for text retrieval) and their pairwise features which describe relationships between two raw features (e.g. word synonymy or polysemy) could greatly improve the retrieval precision. However, most existing methods can not scale up to problems with many raw features (e.g. English vocabulary), due to the prohibitive computational cost on learning and the memory requirement to store a quadratic number of parameters. In this chapter, we propose to learn a sparse representation of the pairwise features under the preference learning framework using the L1 regularization. Based on stochastic gradient descent, an online algorithm is devised to enforce the sparsity using a mini-batch shrinkage strategy. On multiple benchmark datasets, we show that our method achieves better performance with fast convergence, and takes much less memory on models with millions of parameters.

8.1 INTRODUCTION AND MOTIVATION

Learning preferences among a set of objects (e.g. documents) given another object as query is a central task of information retrieval and text mining. One of the most natural frameworks for this task is the *pairwise preference learning*, expressing that one document is preferred over another given the query [50]. Most existing methods [141] learn the preference or relevance function by assigning a real valued score to a feature vector describing a (query, object) pair. This feature vector normally includes a small number of hand-crafted features, such as the BM25 scores for the title or the whole text, instead of the very natural raw features [98]. A drawback of using hand-crafted features is that they are often expensive and specific to datasets, requiring domain knowledge in preprocessing. In contrast, the raw features are easily available, and carry strong semantic information (such as word features in text mining).

In this chapter we study a basic model presented in [54, 4] which uses the raw word features under the supervised pairwise preference

learning framework and consider feature relationships in the model[1]. To be specific, let $\mathcal{D}$ be the dictionary size, i.e. the size of the query and document feature set[2], given a query $q \in \mathbb{R}^{\mathcal{D}}$ and a document $d \in \mathbb{R}^{\mathcal{D}}$, the relevance score between q and d is modeled as:

$$f(q, d) = q^\top W d = \sum_{i,j} W_{ij} \Phi(q_i, d_j), \tag{8.1}$$

where $\Phi(q_i, d_j) = q_i \cdot d_j$ and W_{ij} models the relationship/correlation between i^{th} query feature q_i and j^{th} document feature d_j. This is essentially a linear model with pairwise features $\Phi(\cdot, \cdot)$ and the parameter matrix $W \in \mathbb{R}^{\mathcal{D} \times \mathcal{D}}$ is learned from labeled data. Compared to most of the existing models, the capacity of this model is very large because of the $\mathcal{D}^2$ free parameters which can carefully model the relationship between each pair of words. From a semantic point of view, a notable superiority of this model is that it can capture synonymy and polysemy as it looks at all possible cross terms, and can be tuned directly for the task of interest.

Although it is very powerful, the basic model in (8.1) suffers from the following weakness which hinders its wide application:

1. Memory issue: Given the large dictionary size $\mathcal{D}$, it requires a huge amount of memory to store the W matrix with a size quadratic in $\mathcal{D}$. When $\mathcal{D} = 10,000$, storing W needs nearly 1Gb of RAM (assuming double); when $\mathcal{D} = 30,000$, it requires 8Gb of RAM.

2. Generalization ability: Given $\mathcal{D}^2$ free parameters (entries of W), when the number of training samples is limited, it can easily lead to overfitting. Considering the dictionary with the size $\mathcal{D} = 10,000$, we have $\mathcal{D}^2 = 10^8$ free parameters that need to be estimated which is far too many for small corpora.

To address the above weakness, we propose to constrain W to be a sparse matrix with many zero entries for the pairs of words which are irrelevant for the preference learning task. If W is a highly sparse matrix, then it consumes much less memory and has only a limited number of free parameters to be estimated. In other words, a sparse W matrix will allow us to greatly scale up the dictionary size to model those non-frequent words which are often essential for the preference ordering. In addition, we can have faster and more scalable learning algorithm since most entries of W are zeros so that those multiplications can be avoided. Another advantage of learning a sparse representation of W is its good interpretability. The zero W_{ij} indicate that i^{th} query feature and j^{th} document feature are not correlated to the specific task. A sparse W matrix will accurately capture correlation information between pairs of words and the learned correlated word

1 For the sake of clarity, we present the model in a text retrieval scenario. We use the term "words" for features, and "query" and "documents" for data instances, depending on their roles.

2 In our model and algorithm, there is no need to restrict that query q and document d have the same feature size $\mathcal{D}$; however we make this assumption for simplicity of explanation.

pairs could explain the semantic rationale behind the preference ordering.

In order to enforce the sparsity on W, inspired by the success of the sparse linear regression model—"lasso" [133], we impose the entry-wise ℓ_1 regularization on W. A practical challenge in using the ℓ_1 regularized model is to develop an efficient and scalable learning algorithm. Since in many preference learning related applications (e.g. search engine), the model needs to be trained in a timely fashion and new (query, document) pairs may be added to the repository at any time, stochastic gradient descent in the online learning framework is the most desirable learning method for our task [16]. To enforce the ℓ_1 regularization, based on [39], we propose to perform a mini-batch shrinking step for every T iterations in the stochastic gradient descent which can lead to the sparse solution. Moreover, to reduce the additional bias introduced by the ℓ_1 regularization, we further propose a *refitting* step which can improve the preference prediction while keeping the learned sparsity pattern.

The key idea of this work is that: learning on a large number of corpus-independent raw features, or even on combinations of such features, is not impossible. Although the number of involved parameters is intimidatingly large, using sparsity as a powerful tool, the model can be well controlled and efficiently learned. We believe these ideas form a fresh perspective and will benefit many other retrieval related tasks as well.

8.2 BASIC MODEL

Let us denote the set of documents in the corpus as $\{d^k\}_{k=1}^K \subset \mathbb{R}^{\mathcal{D}}$ and the query as $q \in \mathbb{R}^{\mathcal{D}}$, where $\mathcal{D}$ is the dictionary size, and the j^{th} dimension of a document/query vector indicates the frequency of occurrence of the j^{th} word, e.g. using the tf-idf weighting and then normalizing to unit length [3].

Given a query q and a document d, we wish to learn a scoring function $f(q, d)$ that measures the relevance of d to q. In this chapter, we assume that f is a linear function which takes the form of (8.1). Each entry of W represents a "relationship" between a pair of words.

8.2.1 *Margin Rank Loss*

Suppose we are given a set of tuples $\mathcal{R}$ (labeled data), where each tuple contains a query q, a preferred document d^+ and an unpreferred (or lower ranked) document d^-. We would like to learn a W such that $q^\top W d^+ > q^\top W d^-$, making the right preference prediction.

For that purpose, given tuple (q, d^+, d^-), we employ the widely adopted margin rank loss [57]:

$$\begin{aligned} L_W(q, d^+, d^-) &\equiv h(q^\top W d^+ - q^\top W d^-) \\ &= \max(0, 1 - q^\top W d^+ + q^\top W d^-), \end{aligned} \tag{8.2}$$

where $h(x) \equiv \max(0, 1-x)$ is the well-known hinge loss function as adopted in SVM.

Our goal is to learn the W matrix which minimize the loss in (8.2) summing over all tuples (q, d^+, d^-) in $\mathcal{R}$:

$$W^* = \arg\min_W \frac{1}{|\mathcal{R}|} \sum_{(q,d^+,d^-)\in\mathcal{R}} L_W(q, d^+, d^-). \tag{8.3}$$

8.2.2 *Stochastic Subgradient Descent*

In general, the size of $\mathcal{R}$ is very large and new tuples may be added to $\mathcal{R}$ in a streaming manner, which makes it difficult to directly train the objective in (8.3). To overcome this challenge, we adopt stochastic (sub)gradient descent (SGD) in an online learning framework [161],

Specifically, at each iteration, we randomly draw a sample (q, d^+, d^-) from $\mathcal{R}$, compute the subgradient[3] of $L_W(q, d^+, d^-)$ with respect to W as following:

$$\nabla L_W(q, d^+, d^-) = \begin{cases} -q(d^+ - d^-)^\top & \text{if} \quad q^\top W(d^+ - d^-) < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{8.4}$$

and then update the W matrix accordingly. It has been shown that SGD achieves fast learning on large scale datasets [16].

We suggest to initialize W^0 to the identity matrix as this initializes the model to the same solution as a cosine similarity score. The strategy introduces a prior expressing that the weight matrix should be close to the identity matrix. We consider term correlations only when it is necessary to increase the score of a relevant document, or conversely, decrease the score of a irrelevant document. As for the learning rate η_t, we suggest to adopt a decaying learning rate: $\eta_t = \frac{C}{\sqrt{t}}$, where C is a pre-defined constant as the initial learning rate. Intuitively, it should be better than the fixed learning rate since at the beginning, when W is far away from the optimal solution W^*, a larger learning rate is desirable since it leads to a significant decrease of the objective value. On the other hand, when W gets close to W^*, a smaller rate should be adopted to avoid missing the optimal solution. We will show the advantage of the decaying learning rate scheme over the fixed one in the experiment section.

8.3 PREFERENCE LEARNING WITH SPARSITY

As discussed in the introduction, the model and the learning algorithm in the previous section will lead to a dense W matrix which consumes a large amount of memory and has poor generalization ability for small corpora. To address these problems, we can learn a sparse model with a small number of nonzero entries of W. In order

3 Since L is a non-smooth function, it does not have the gradient but only has the subgradient.

to obtain a sparse W, inspired by [133], we add an entry-wise ℓ_1 norm to the W as a regularization term to the loss function. We propose to optimize the following objective function:

$$W^* = \arg\min_W \frac{1}{|\mathcal{R}|} \sum_{(q,d^+,d^-)\in\mathcal{R}} L_W(q, d^+, d^-) + \lambda\|W\|_1, \tag{8.5}$$

where $\|W\|_1 = \sum_{i,j=1}^{\mathcal{D}} |W_{ij}|$ is the entry-wise ℓ_1 norm of W and λ is the regularization parameter which controls the sparsity level (the number of nonzero entries) of W. In general, a larger λ leads to a more sparse W. On the other hand, a too sparse W will miss some useful relationship information among word pairs (considering diagonal W as an extreme case). Therefore, in practice, we need to tune λ to obtain a W with a suitable sparsity level.

8.3.1 *Training the Sparse Model*

To optimize (8.5), we adopt a variant of the general sparse online learning scheme in [39]. In [39], after updating W^t at each iteration in SGD, a shrinkage step is performed by solving the following optimization problem:

$$\widehat{W}^t = \arg\min_W \frac{1}{2}\|W - W^t\|_F^2 + \lambda\eta_t\|W\|_1, \tag{8.6}$$

and then use $\widehat{W}^t$ as the starting point for the next iteration. In 8.6, $\|\cdot\|_F$ denote the matrix Frobenius norm and η_t is the decaying learning rate for the t^{th} iteration. According to the proposition 3.3, we know that performing (8.6) will shrink those W_{ij}^t with an absolute value less than $\lambda\eta_t$ to zero and hence lead to a sparse W matrix. In particular, the solution $\widehat{W}^t$ to the optimization problem in (8.6) takes the following form:

$$\widehat{W}_{ij}^t = \begin{cases} W_{ij}^t + \lambda\eta_t & \text{if} \quad W_{ij}^t < -\lambda\eta_t \\ 0 & \text{if} \quad -\lambda\eta_t \leqslant W_{ij}^t \leqslant \lambda\eta_t \\ W_{ij}^t - \lambda\eta_t & \text{if} \quad W_{ij}^t > \lambda\eta_t \end{cases} \tag{8.7}$$

Although performing the shrinkage step leads a sparse W solution, it is very expensive for a large dictionary size $\mathcal{D}$. For example, when $\mathcal{D} = 10,000$, we need $\mathcal{D}^2 = 10^8$ operations. Therefore, we suggest to perform the shrinkage step for every T iteration cycles. In general, a smaller T guarantees that the shrinkage step can be done in a timely fashion so that the entries of W will not grow too large to produce inaccurate $\nabla L_W(q, d^+, d^-)$; on the other hand, a smaller T increases the computational cost of the training process. In practice, we suggest to set $T = 100$. The details of the algorithm are presented in Algorithm 8.1.

Note that when t is a multiple of T, we perform the shrinkage step with a cumulative regularization parameter for $\|W\|_1$ from $t - T + 1$

to t: $\lambda \sum_{k=t-T+1}^{t} \eta_t$. The reason why we adopt cumulative regularization parameter is due to the following simple fact that: W^T obtained by solving a sequence of successive optimization problems:

$$W^t = \arg\min_W \frac{1}{2}\|W - W_{t-1}\|_F^2 + \lambda\eta_t\|W\|_1, \quad t = 1, \dots T$$

is identical to the one by solving the following single optimization problem:

$$W^T = \arg\min_W \frac{1}{2}\|W - W_0\|_F^2 + \lambda \sum_{t=1}^{T} \eta_t\|W\|_1.$$

Although we take the gradient update so that Algorithm 8.1 is not exactly identical to the one taking the shrinkage step at every iteration, empirically, the learned sparse W from Algorithm 8.1 is a good approximation.

Algorithm 8.1 Sparse SGD

Initialization: $W^0 \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, T, learning rate sequence $\{\eta_t\}$.
Iterate for $t = 1, 2, \dots$ until convergence of W^t:

1. Randomly draw a tuple $(q, d^+, d^-) \in \mathcal{R}$
2. Compute the subgradient of $L_{W^{t-1}}(q, d^+, d^-)$ with respect to W: $\nabla L_{W^{t-1}}(q, d^+, d^-)$
3. Update $W^t = W^{t-1} - \eta_t \nabla L_{W^{t-1}}(q, d^+, d^-)$
4. If (t mod T =0)

$$W^t = \arg\min_W \frac{1}{2}\|W - W^t\|_F^2 + \lambda \sum_{k=t-T+1}^{t} \eta_t\|W\|_1$$

8.3.2 *Refitting the Sparse Model*

From (8.7), we see that ℓ_1 regularization will not only shrink the weights for uncorrelated word pairs to zero but also reduce the absolute value of the weights for correlated word pairs. This additional bias introduced by ℓ_1 regularization often harm the prediction performance. In order to reduce this bias, we propose to refit the model without ℓ_1 regularization, but enforcing the sparsity pattern of W obtained from Algorithm 8.1.

More precisely, after learning the sparse $\widehat{W}$ from Algorithm 8.1, let Ω be the indices of nonzero entries of $\widehat{W}$, i.e. $\Omega = \{(i,j)|\widehat{W}_{ij} \neq 0\}$. Given a matrix $W \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, let $P_\Omega(W) \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ be the matrix defined as following:

$$P_\Omega(W)_{ij} = \begin{cases} W_{ij} & \text{if} \quad (i,j) \in \Omega \\ 0 & \text{if} \quad (i,j) \notin \Omega \end{cases},$$

Table 8.1: The statistics of the experimental datasets

	20NG	RCV1	MNIST
No. of training samples	11,314	15,564	60,000
No. of testing samples	7,532	518,571	10,000
No. of Classes	20	53	10
Dictionary Size $\mathcal{D}$	10,000	10,000	784
No. of Free Parameters	10^8	10^8	614,656

where P_Ω is called the projection operator which projects W onto Ω.

In the refitting step, given Ω, we try to minimize the following objective function:

$$W^* = \arg\min_W \frac{1}{|\mathcal{R}|} \sum_{(q,d^+,d^-)\in\mathcal{R}} L_{P_\Omega(W)}(q, d^+, d^-), \tag{8.8}$$

We still adopt SGD to minimize (8.8), but replace $\nabla L_W(q, d^+, d^-)$ with $\nabla L_{P_\Omega(W)}(q, d^+, d^-)$. Using the chain rule for subgradient, we can show that $\nabla L_{P_\Omega(W)}(q, d^+, d^-)$ takes the following form:

$$\nabla L_{P_\Omega(W)}(q, d^+, d^-) = \begin{cases} -P_\Omega(q(d^+ - d^-)^\top) & \text{if } q^\top P_\Omega(W)(d^+ - d^-) < 1 \\ 0 & \text{otherwise} \end{cases}.$$

In the experiment section, we show that the prediction performance gets improved after the refitting step.

8.4 EXPERIMENT

8.4.1 *Experiment Setup*

Pairwise preference learning discussed in this chapter belongs to a more general framework "Learning to rank", which is a key topic in the research of information retrieval [141]. Learning to rank methods are usually evaluated using standard benchmark data like TREC[4] data or LETOR [98]. However, TREC has only a limited number of queries available, which makes the training of a large number of features very difficult. On the other hand, LETOR and most of the other learning to rank datasets (e.g. Microsoft Learning to Rank Datasets[5], Yahoo! Learning to Rank Challenge Datasets[6]) have only few hundred (or even less) features such as BM25 or pagerank scores instead of the actual word features, and are therefore not adequate for evaluating our methods. It would be ideal to test the proposed method on click-through data from web search logs, but such data are not publicly available.

4 http://trec.nist.gov/
5 http://research.microsoft.com/en-us/projects/mslr/
6 http://learningtorankchallenge.yahoo.com

As pointed out by a seminal paper [46], preference learning and multiclass classification can be modeled in a unified framework. It is natural to adopt the multiclass classification (with many different classes) datasets to evaluate the our proposed preference learning models. More precisely, we construct training samples $(q, d^+, d^-) \in \mathcal{R}$ where q and d^+ are in the same class while q and d^- belong to different classes. In our experiment, we use several benchmark multiclass classification datasets, including the text datasets 20 Newsgroups[7] (20NG), RCV1[8][88] and digital recognition dataset MNIST[9]. For 20NG and RCV1, we adopt the normalized tf-idf of the 10,000 most frequent words as the document features. For MINST, the normalized grey scale pixel values are used as features. Some statistics of these datasets are shown in Table 8.1.

For the experiments, we use the cosine similarity as the baseline, i.e. $W = I$ and mainly compare the following methods:

1. W is a diagonal matrix.
2. W is unconstrained and trained by SGD with the fixed learning rate $\eta = 0.01$.
3. W is unconstrained and trained by SGD with the decaying learning rates $\eta_t = \frac{C}{\sqrt{t}}$ where $C = 200$.
4. W is sparse and trained by Algorithm 8.1 with the decaying learning rates $\eta_t = \frac{C}{\sqrt{t}}$ where $C = 200$ and then perform the corresponding refitting step[10].

Note that for the decaying learning rate case, we try a wide range of starting rate C and find that $C = 200$ can provide us the most rapid decrease of the objective value. So C is set to be 200 through out the chapter. As for the regularization parameter λ, when the dictionary size is large, say $\mathcal{D} = 10,000$ for the text data, we choose λ which leads to the 5% to 10% density (percentage of nonzero entries) of W. When $\mathcal{D}$ is relatively small, say 748 for the MNIST dataset, we set λ so that the density of W is roughly 50%. This way of setting λ provides us a sparse enough W which has the advantage of the memory savings and being easy to interpret. In the meanwhile, we have enough nonzero entries of W to guarantee a reasonably good preference learning performance on the testing datasets.

We compare the performance of each method by the following metrics:

1. Test error rate: for a testing tuple (q, d^+, d^-), if $q^T W d^+ \leqslant q^T W d^-$, test error is increased by 1 and normalized by the test sample size.

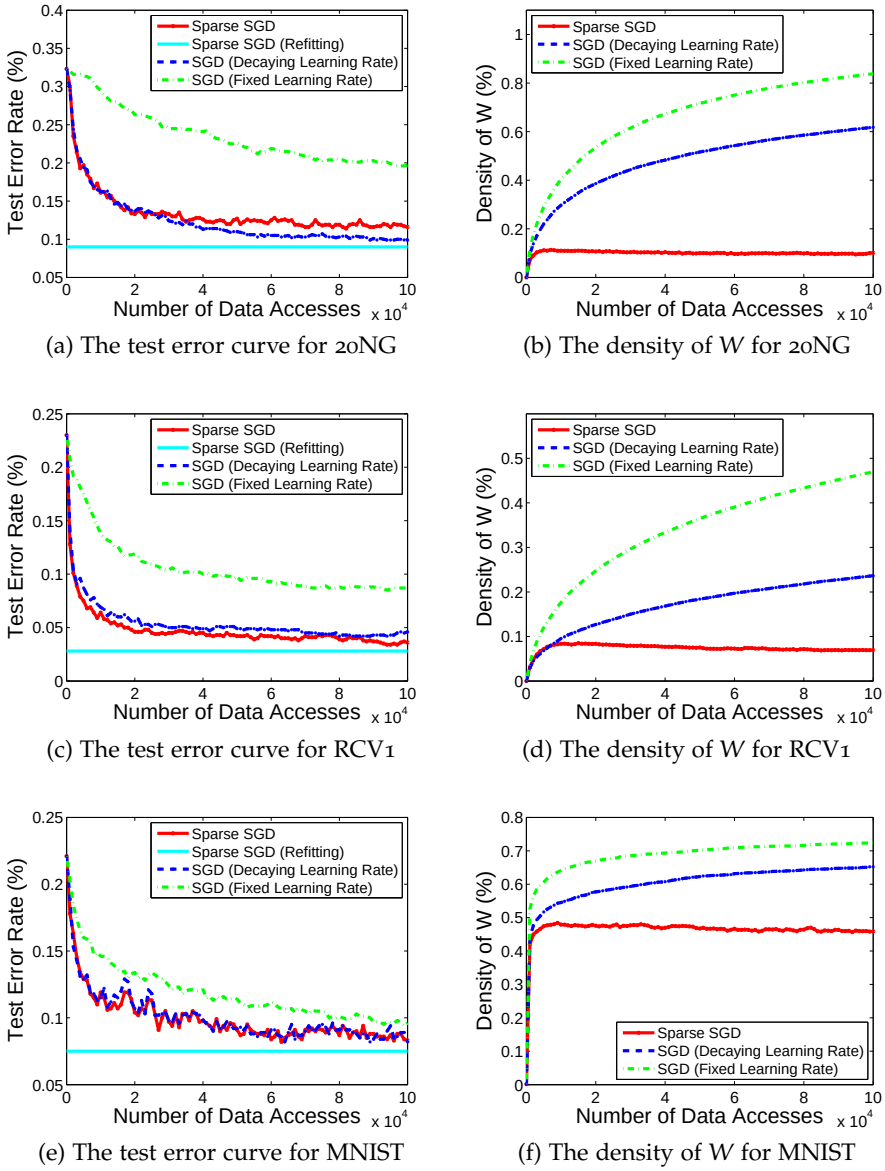
7 http://people.csail.mit.edu/jrennie/20Newsgroups

8 We adopt the preprocessing method in [7], remove the multi-labelled instances and result in 53 different classes.

9 http://yann.lecun.com/exdb/mnist

10 For the fair comparison, we use exactly the same training samples in the refitting step as in the sparse learning procedure without any additional samples.

Figure 8.1: The test error rate and the density W for 3 benchmark datasets



(a) The test error curve for 20NG

(b) The density of W for 20NG

(c) The test error curve for RCV1

(d) The density of W for RCV1

(e) The test error curve for MNIST

(f) The density of W for MNIST

2. Mean average precision (MAP) [34].

And we also provide the semantic information encoded in the W matrix for the text data.

8.4.2 *Results*

8.4.2.1 *Learning Curves*

It has been shown that test error rate is monotonic to area under ROC curve (AUC). In Figure 8.1, we show the curves of the test error rate and the corresponding density of W vs. the number of training iterations, i.e. the number of accesses of the training data. Each row in Figure 8.1 corresponds to one dataset.

We have the following observations:

Table 8.2: Retrieval Performance. Items in bold fonts are the best among methods tested.

	MAP	Test Error Rate	Density
Identity	0.185	0.323	1e-4
Diagonal	0.190	0.318	1e-4
SGD FLR	0.258	0.197	0.848
SGD DLR	0.399	0.099	0.618
Sparse	0.360	0.114	0.101
Sparse-R	0.426	0.090	0.101
W-Sparse	0.380	0.105	0.158
W-Sparse-R	**0.428**	**0.089**	0.158

(a) 20NG

	MAP	Test Error Rate	Density
Identity	0.380	0.230	1e-4
Diagonal	0.390	0.223	1e-4
SGD FLR	0.451	0.087	0.470
SGD DLR	0.453	0.046	0.236
Sparse	0.463	0.036	0.069
Sparse-R	0.501	**0.029**	0.069
W-Sparse	0.471	0.037	0.086
W-Sparse-R	**0.506**	**0.029**	0.086

(b) RCV1

	MAP	Test Error Rate	Density
Identity	0.453	0.221	1/784
Diagonal	0.460	0.318	1/784
SGD FLR	0.610	0.096	0.724
SGD DLR	0.654	0.082	0.652
Sparse	0.654	0.083	0.458
Sparse-R	**0.669**	**0.075**	0.458

(c) MNIST

1. For SGD with the fixed learning rate, the test error rate decreases very slowly, and is relatively flat compared to other methods.

2. The schemes with decaying learning rates (including two sparse models "Sparse SGD", and the dense model "SGD (Decaying Learning Rate)") have similarly good convergence rate.

3. Using the same regularization parameter, "Sparse SGD" achieves the most sparse model. For dense models, "SGD (Decaying Learning Rate)" reaches a more sparse model than using naive method "SGD (Fixed Learning Rate)". This is another evidence that decaying learning rates are superior to fixed learning rates.

Table 8.3: The examples of learned related word pairs in 20NG

Query word	Five most related document words
clinton	clinton government health people gay
cpu	mac drive scsi card jon
graphics	graphics tiff image color polygon
handgun	gun weapons handgun militia fbi
hockey	hockey game espn colorado team
motorcycle	bike brake turbo rpi cylinder
religions	god religions bible christian jesus

4. Refitting the sparse models clearly achieves the best test error rate while keeping the low density of the W matrix.

8.4.2.2 *Retrieval Performance and Memory*

In this section, we present mean average precision (MAP), test error rate and the memory space of W of each method after training 100,000 iterations. The results are presented in Table 8.2. We use the abbreviation for each method due to the space limitation of the table. "Identity" stands for $W = I$ and "Diagonal" means that W is a diagonal matrix where only the diagonal entries are learned. 'SGD FLR" means SGD with the fixed learning rate, while "SGD DLR" means SGD with the decaying learning rate. Both of them are trained on the basic model without the ℓ_1 regularization. "Sparse" and "Sparse-R" are sparse models, without refitting and with refitting, respectively.

The results on MAP are essentially consistent to those on the test error rate. The sparse method with decaying learning rate has similar performance compared to its dense counterpart but takes much less memory. Moreover, sparse models after refitting achieves the best performance

8.4.2.3 *Anecdotal Evidence*

The sparse model can also provide much useful semantic information. For example, we can easily infer the most related word pair between query word and document word from the sparse W matrix. More precisely, each row of W represents the strength of the correlation (either positive or negative) of document words to a specific query word. Given the i^{th} query word, we sort the absolute value of the i^{th} row of W in a descending order and pick the first few nonzero entries. Those selected entries of W represent the most correlated document words to the given i^{th} query word.

In Table 8.3, we present the query words from different categories and the five most correlated document words from the learned sparse W in (8.5). We can see that most correlated document words are clearly from the same topics as the query words. It also provide us some interesting semantic information. For example, in 20NG, "fbi" is closely related to "handgun"; "colorado" to "hockey" which indicates that there might be a popular hockey team in Colorado (in fact,

it is Colorado Avalanche team); "government" to "clinton" which indicates that Clinton might be a famous politician (The former US president Bill Clinton).

9

SPARSE LATENT SEMANTIC ANALYSIS

Latent semantic analysis (LSA), as one of the most popular unsupervised dimension reduction tools, has a wide range of applications in text mining and information retrieval. The key idea of LSA is to learn a projection matrix that maps the high dimensional vector space representations of documents to a lower dimensional latent space, i.e. so called latent *topic* space. In this chapter, we propose a new model called *Sparse LSA*, which produces a sparse projection matrix via the ℓ_1 regularization. Compared to the traditional LSA, Sparse LSA selects only a small number of relevant words for each topic and hence provides a compact representation of topic-word relationships. Moreover, Sparse LSA is computationally very efficient with much less memory usage for storing the projection matrix. Furthermore, we propose two important extensions of Sparse LSA: group structured Sparse LSA and non-negative Sparse LSA. We conduct experiments on several benchmark datasets and compare Sparse LSA and its extensions with several widely used methods, e.g. LSA, Sparse Coding and LDA. Empirical results suggest that Sparse LSA achieves similar performance gains to LSA, but is more efficient in projection computation, storage, and also well explain the topic-word relationships.

9.1 INTRODUCTION AND MOTIVATION

Latent Semantic Analysis (LSA) [36], as one of the most successful tools for learning the concepts or latent topics from text, has widely been used for the dimension reduction purpose in information retrieval. More precisely, given a document-term matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$, where N is the number of documents and M is the number of words, and assuming that the number of latent topics (the dimensionality of the latent space) is set as D ($D \leqslant \min\{N, M\}$), LSA applies singular value decomposition (SVD) to construct a low rank (with rank-D) approximation of $\mathbf{X}$: $\mathbf{X} \approx \mathbf{U}\mathbf{S}\mathbf{V}^T$, where the column orthogonal matrices $\mathbf{U} \in \mathbb{R}^{N \times D}$ ($\mathbf{U}^T\mathbf{U} = \mathbf{I}$) and $\mathbf{V} \in \mathbb{R}^{M \times D}$ ($\mathbf{V}^T\mathbf{V} = \mathbf{I}$) represent document and word embeddings into the latent space. $\mathbf{S}$ is a diagonal matrix with the D largest singular values of $\mathbf{X}$ on the diagonal [1]. Subsequently, the so-called *projection matrix* defined as $\mathbf{A} = \mathbf{S}^{-1}\mathbf{V}^T$ provides a transformation mapping of documents from the word space to the latent topic space, which is less noisy and considers word synonymy (i.e. different words describing the same idea). However, in LSA, each latent topic is represented by all word features which sometimes makes it difficult to precisely characterize the topic-word relationships.

1 Since it is easier to explain our Sparse LSA model in terms of document-term matrix, for the purpose of consistency, we introduce SVD based on the document-term matrix which is a different from standard notations using the term-document matrix.

1. It is intuitive that only a part of the vocabulary can be relevant to a certain topic. By enforcing sparsity of $\mathbf{A}$ such that each row (representing a latent topic) only has a small number nonzero entries (representing the most relevant words), Sparse LSA can provide us a compact representation for topic-word relationship that is easier to interpret.

2. With the adjustment of sparsity level in projection matrix, we could control the granularity ("level-of-details") of the topics we are trying to discover, e.g. more generic topics have more nonzero entries in rows of $\mathbf{A}$ than specific topics.

3. Due to the sparsity of $\mathbf{A}$, Sparse LSA provides an efficient strategy both in the time cost of the projection operation and in the storage cost of the projection matrix when the dimensionality of latent space D is large.

4. Sparse LSA could project a document q into a *sparse* vector representation $\widehat{q}$ where each entry of $\widehat{q}$ corresponds to a latent topic. In other words, we could know the topics that q belongs to directly form the position of nonzero entries of $\widehat{q}$. Moreover, sparse representation of projected documents will save a lot of computational cost for the subsequent retrieval tasks, e.g. ranking (considering computing cosine similarity), text categorization, etc.

Furthermore, we propose two important extensions based on Sparse LSA:

1. Group Structured Sparse LSA: we add group structured sparsity-inducing penalty as in [150] to select the most relevant groups of features relevant to the latent topic.

2. Non-negative Sparse LSA: we further enforce the non-negativity constraint on the projection matrix $\mathbf{A}$. It could provide us a pseudo probability distribution of each word given the topic, similar as in Latent Dirichlet Allocation (LDA) [14].

We conduct experiments on four benchmark data sets, with two on text categorization, one on breast cancer gene function identification, and the last one on topic-word relationship identification from NIPS proceeding papers. We compare Sparse LSA and its variants with several popular methods, e.g. LSA [36], Sparse Coding [85] and LDA [14]. Empirical results show clear advantages of our methods in terms of computational cost, storage and the ability to generate sensible topics and to select relevant words (or genes) for the latent topics.

9.2 SPARSE LSA

9.2.1 *Optimization Formulation of LSA*

We consider N documents, where each document lies in an M-dimensional feature space $\mathcal{X}$, e.g. tf-idf [3] weights of the vocabulary with the

normalization to unit length. We denote N documents by a matrix $\mathbf{X} = [X_1, \ldots, X_M] \in \mathbb{R}^{N \times M}$, where $X_j \in \mathbb{R}^N$ is the j-th feature vector for all the documents. For the dimension reduction purpose, we aim to derive a mapping that projects input feature space into a D-dimensional latent space where D is smaller than M. In the information retrieval content, each latent dimension is also called an hidden "topic".

Motivated by the latent factor analysis [55], we assume that we have D uncorrelated latent variables $U_1, \ldots, U_D$, where each $U_d \in \mathbb{R}^N$ has the unit length, i.e. $\|U_d\|_2 = 1$. Here $\|\cdot\|_2$ denotes the vector ℓ_2-norm. For the notation simplicity, we put latent variables $U_1, \ldots, U_D$ into a matrix: $\mathbf{U} = [U_1, \ldots, U_D] \in \mathbb{R}^{N \times D}$. Since latent variables are uncorrelated with the unit length, we have $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. We also assume that each feature vector X_j can be represented as a linear expansion in latent variables $U_1, \ldots, U_D$:

$$X_j = \sum_{d=1}^{D} a_{dj} U_d + \epsilon_j, \tag{9.1}$$

or simply $\mathbf{X} = \mathbf{U}\mathbf{A} + \boldsymbol{\epsilon}$, where $\mathbf{A} = [a_{dj}] \in \mathbb{R}^{D \times M}$ gives the mapping from the latent space to the input feature space and ϵ is the zero mean noise. Our goal is to compute the so-called projection matrix $\mathbf{A}$.

We can achieve this by solving the following optimization problem which minimizes the rank-D approximation error subject to the orthogonality constraint of $\mathbf{U}$:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{A}} \quad & \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{A}\|_F^2 \\ \text{subject to:} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}, \end{aligned} \tag{9.2}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm. The constraint $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ is according to the uncorrelated property among latent variables.

At the optimum of (9.2), $\mathbf{U}\mathbf{A}$ leads to the best rank-D approximation of the data $\mathbf{X}$. In general, larger the D is, the better the reconstruction performance. However, larger D requires more computational cost and large amount memory for storing $\mathbf{A}$. This is the issue that we will address in the next section.

After obtaining $\mathbf{A}$, given a new document $q \in \mathbb{R}^M$, its representation in the lower dimensional latent space can be computed as:

$$\widehat{q} = \mathbf{A}q. \tag{9.3}$$

9.2.2 *Sparse LSA*

As discussed in the introduction, one notable advantage of sparse LSA is due to its good interpretability in topic-word relationship. Sparse LSA automatically selects the most relevant words for each latent topic and hence provides us a clear and compact representation of the topic-word relationship. Moreover, for a new document q, if the words in q has no intersection with the relevant words of

d-th topic (nonzero entries in $\mathbf{A}^d$, the d-th row of $\mathbf{A}$), the d-th element of $\widehat{q}$, $\mathbf{A}^d q$, will become zero. In other words, the *sparse* latent representation of $\widehat{q}$ clearly indicates the topics that q belongs to.

Another benefit of learning sparse $\mathbf{A}$ is to save computational cost and storage requirements when D is large. In traditional LSA, the topics with larger singular values will cover a broader range of concepts than the ones with smaller singular values. For example, the first few topics with largest singular values are often too general to have specific meanings. As singular values decrease, the topics become more and more specific. Therefore, we might want to enlarge the number of latent topics D to have a reasonable coverage of the topics. However, given a large corpus with millions of documents, a larger D will greatly increase the computational cost of projection operations in traditional LSA. In contrary, for Sparse LSA, projecting documents via a highly sparse projection matrix will be computationally much more efficient; and it will take much less memory for storing $\mathbf{A}$ when D is large.

The illustration of Sparse LSA from a matrix factorization perspective is presented in Figure 9.1(a). An example of topic-word relationship is shown in Figure 9.1(b). Note that a latent topic ("law" in this case) is only related to a limited number of words.

In order to obtain a sparse $\mathbf{A}$, inspired by the lasso model in [133], we add an entry-wise ℓ_1-norm of $\mathbf{A}$ as the regularization term to the loss function and formulate the Sparse LSA model as:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{A}} \quad & \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{A}\|_F^2 + \lambda\|\mathbf{A}\|_1 \\ \text{subject to:} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}, \end{aligned} \tag{9.4}$$

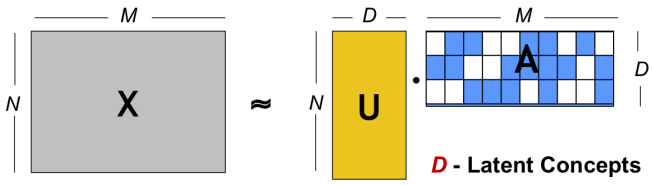
where $\|\mathbf{A}\|_1 = \sum_{d=1}^{D}\sum_{j=1}^{M}|a_{dj}|$ is the entry-wise ℓ_1-norm of $\mathbf{A}$ and λ is the positive regularization parameter which controls the density (the number of nonzero entries) of $\mathbf{A}$. In general, a larger λ leads to a sparser $\mathbf{A}$. On the other hand, a too sparse $\mathbf{A}$ will miss some useful topic-word relationships which harms the reconstruction performance. Therefore, in practice, we need to try to select larger λ to obtain a more sparse $\mathbf{A}$ while still achieving good reconstruction performance. We will show the effectiveness of λ in more details in Section 9.5.

9.2.3 *Optimization Algorithm*

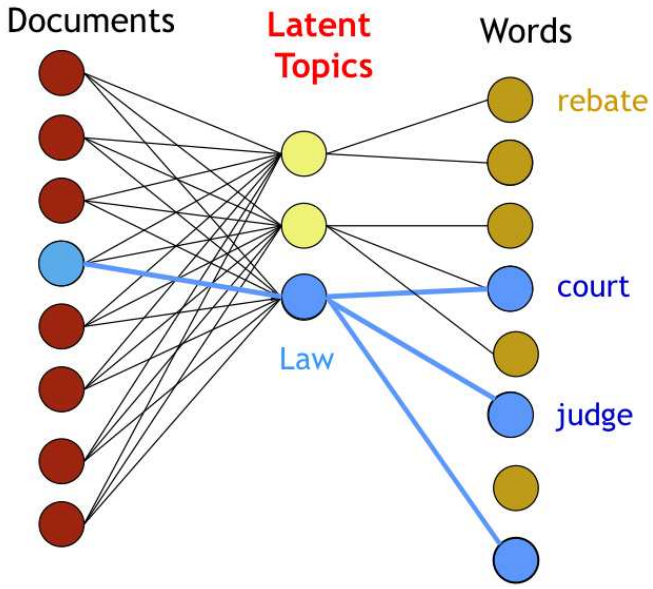
In this section, we propose an efficient optimization algorithm to solve (9.4). Although the optimization problem is non-convex, fixing one variable (either $\mathbf{U}$ or $\mathbf{A}$), the objective function with respect to the other is convex. Therefore, a natural approach to solve (9.4) is by the alternating approach:

1. When $\mathbf{U}$ is fixed, the optimization problem with respect to $\mathbf{A}$ takes the following form:

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{A}\|_F^2 + \lambda\|\mathbf{A}\|_1. \tag{9.5}$$



(a)



(b)

Figure 9.1: Illustration of Sparse LSA (a) View of Matrix Factorization, white cells in $\mathbf{A}$ indicates the zero entries (b) View of document-topic-term relationship.

The optimization in (9.5) has a closed-form solution due to the fact that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$. In particular, as shown in Proposition 3.3, the optimal $\mathbf{A}$ can be obtained via the soft-thresholding operation:

$$A_{ij} = \text{sign}\left((\mathbf{U}^T\mathbf{X})_{ij}\right) \max\left(0, |(\mathbf{U}^T\mathbf{X})_{ij}| - \lambda\right).$$

In fact, this is an extra computational benefit brought by the orthogonality constraint of the matrix $\mathbf{U}$.

2. When $\mathbf{A}$ is fixed, the optimization problem is equivalent to:

$$\begin{aligned} \min_{\mathbf{U}} \quad & \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{A}\|_F^2 \\ \text{subject to:} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}. \end{aligned} \tag{9.6}$$

The objective function in (9.6) can be further written as:

$$\begin{aligned} & \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{A}\|_F^2 \\ = \; & \frac{1}{2}\text{tr}((\mathbf{X} - \mathbf{U}\mathbf{A})^T(\mathbf{X} - \mathbf{U}\mathbf{A})) \\ = \; & -\text{tr}(\mathbf{A}^T\mathbf{U}^T\mathbf{X}) + \frac{1}{2}\text{tr}(\mathbf{X}^T\mathbf{X}) + \frac{1}{2}\text{tr}(\mathbf{A}^T\mathbf{U}^T\mathbf{U}\mathbf{A}) \\ = \; & -\text{tr}(\mathbf{A}^T\mathbf{U}^T\mathbf{X}) + \frac{1}{2}\text{tr}(\mathbf{X}^T\mathbf{X}) + \frac{1}{2}\text{tr}(\mathbf{A}^T\mathbf{A}), \end{aligned}$$

where the last equality is according to the constraint that $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}$. By the fact that $\mathrm{tr}(\mathbf{A}^\mathsf{T}\mathbf{U}^\mathsf{T}\mathbf{X}) \equiv \mathrm{tr}(\mathbf{U}^\mathsf{T}\mathbf{X}\mathbf{A}^\mathsf{T})$, the optimization problem in (9.6) is equivalent to

$$\begin{aligned} \max_{\mathbf{U}} \quad & \mathrm{tr}(\mathbf{U}^\mathsf{T}\mathbf{X}\mathbf{A}^\mathsf{T}) \\ \text{subject to:} \quad & \mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}. \end{aligned} \tag{9.7}$$

Let $\mathbf{V} = \mathbf{X}\mathbf{A}^\mathsf{T}$. In fact, $\mathbf{V}$ is the latent topic representations of the documents $\mathbf{X}$. Assuming that $\mathbf{V}$ is full column rank, i.e. with $\mathrm{rank}(\mathbf{V}) = D$, (9.7) has the closed form solution as shown in the next theorem [163]:

Theorem 9.1. *Suppose the singular value decomposition (SVD) of* $\mathbf{V}$ *is* $\mathbf{V} = \mathbf{P}\Delta\mathbf{Q}$, *the optimal solution to* (9.7) *is* $\mathbf{U} = \mathbf{P}\mathbf{Q}$.

Since N is much larger than D, in most cases, $\mathbf{V}$ is full column rank. If it is not, we may approximate $\mathbf{V}$ by a full column rank matrix $\widetilde{\mathbf{V}} = \mathbf{P}\widetilde{\Delta}\mathbf{Q}$. Here $\widetilde{\Delta}_{dd} = \Delta_{dd}$ if $\Delta_{dd} \neq 0$; otherwise $\widetilde{\Delta}_{dd} = \delta$, where δ is a very small positive number. In the later context, for the simplicity purpose, we assume that $\mathbf{V}$ is always full column rank.

It is worthy to note that since D is usually much smaller than the vocabulary size M, the computational cost of SVD of $\mathbf{V} \in \mathbb{R}^{N\times D}$ is much cheaper than SVD of $\mathbf{X} \in \mathbb{R}^{N\times M}$ in LSA.

As for the starting point, any $\mathbf{A}^0$ or $\mathbf{U}^0$ stratifying $(\mathbf{U}^0)^\mathsf{T}\mathbf{U}^0 = \mathbf{I}$ can be adopted. We suggest a very simple initialization strategy for $\mathbf{U}^0$ as following:

$$\mathbf{U}^0 = \begin{pmatrix} \mathbf{I}_D \\ \mathbf{0} \end{pmatrix}, \tag{9.8}$$

where $\mathbf{I}_D$ the D by D identity matrix. It is easy to verify that $(\mathbf{U}^0)^\mathsf{T}\mathbf{U}^0 = \mathbf{I}$.

The optimization procedure can be summarized in Algorithm 9.1.

Algorithm 9.1 Optimization Algorithm for Sparse LSA

Input: $\mathbf{X}$, the dimensionality of the latent space D, regularization parameter λ

Initialization:$\mathbf{U}^0 = \begin{pmatrix} \mathbf{I}_D \\ \mathbf{0} \end{pmatrix}$,

Iterate until convergence of $\mathbf{U}$ and $\mathbf{A}$:

1. Compute $\mathbf{A}$ directly by the soft-thresholding operation in (9.5).
2. Project $\mathbf{X}$ onto the latent space: $\mathbf{V} = \mathbf{X}\mathbf{A}^\mathsf{T}$.
3. Compute the SVD of $\mathbf{V}$: $\mathbf{V} = \mathbf{P}\Delta\mathbf{Q}$ and let $\mathbf{U} = \mathbf{P}\mathbf{Q}$.

Output: Sparse projection matrix $\mathbf{A}$.

As for the stopping criteria, let $\|\cdot\|_\infty$ denote the matrix entry-wise ℓ_∞-norm, for the two consecutive iterations t and $t+1$, we compute

the maximum change for all entries in $\mathbf{U}$ and $\mathbf{A}$: $\|\mathbf{U}^{(t+1)} - \mathbf{U}^{(t)}\|_\infty$ and $\|\mathbf{A}^{(t+1)} - \mathbf{A}^{(t)}\|_\infty$; and stop the optimization procedure when both quantities are less than the prefixed constant τ. In our experiments, we set $\tau = 0.01$.

9.3 EXTENSION OF SPARSE LSA

In this section, we propose two important extensions of Sparse LSA model.

9.3.1 *Group Structured Sparse LSA*

Although entry-wise ℓ_1-norm regularization leads to the sparse projection matrix $\mathbf{A}$, it does not take advantage of any prior knowledge on the structure of the input features (e.g. words). When the features are naturally clustered into groups, it is more meaningful to enforce the sparsity pattern at a group level instead of each individual feature; so that we can learn which groups of features are relevant to a latent topic. It has many potential applications in analyzing biological data. For example, in the latent gene function identification, it is more meaningful to determine which pathways (groups of genes with similar function or near locations) are relevant to a latent gene function (topic).

Inspired by the group lasso [150], we can encode the group structure via a ℓ_1/ℓ_2 mixed norm regularization of $\mathbf{A}$ in Sparse LSA. Formally, we assume that the set of groups of input features $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\}$ is defined as a subset of the power set of $\{1, \ldots, M\}$, and is available as prior knowledge. For the purpose of simplicity, we assume that groups are non-overlapped. The group structured Sparse LSA can be formulated as:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{A}} \quad & \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{A}\|_F^2 + \lambda \sum_{d=1}^{D} \sum_{g \in \mathcal{G}} w_g \|\mathbf{A}_{dg}\|_2 \qquad (9.9) \\ \text{subject to:} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}, \end{aligned}$$

where $\mathbf{A}_{dg} \in \mathbb{R}^{|g|}$ is the subvector of $\mathbf{A}$ for the latent dimension d and the input features in group g; w_g is the predefined regularization weight each group g, λ is the global regularization parameter; and $\|\cdot\|_2$ is the vector ℓ_2-norm which enforces all the features in group g for the d-th latent topic, $\mathbf{A}_{dg}$, to achieve zeros simultaneously. A simple strategy for setting w_g is $w_g = \sqrt{|g|}$ as in [150] so that the amount of penalization is adjusted by the size of each group.

9.3.2 *Non-negative Sparse LSA*

It is natural to assume that each word has a non-negative contribution to a specific topic, i.e. the projection matrix $\mathbf{A}$ should be non-negative. In such a case, we may normalize each row of $\mathbf{A}$ to 1:

$$\widetilde{a}_{dj} = \frac{a_{dj}}{\sum_{j=1}^{M} a_{dj}}.$$

Since a_{dj} measures the relevance of the j-th word, w_j, to the d-th topic t_d, from the probability perspective, $\widetilde{a}_{dj}$ can be viewed as a pseudo probability of the word w_j given the topic t_d, $\mathbb{P}(w_j|t_d)$. Similar to topic modeling in the Bayesian framework such as LDA [14], the non-negative Sparse LSA can also provide the most relevant/likely words to a specific topic.

More formally, the non-negative Sparse LSA can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{A}} \quad & \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{A}\|_F^2 + \lambda\|\mathbf{A}\|_1 \\ \text{subject to:} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}, \quad \mathbf{A} \geqslant 0. \end{aligned} \tag{9.10}$$

According to the non-negativity constraint of $\mathbf{A}$, $|a_{dj}| = a_{dj}$ and (9.10) is equivalent to:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{A}} \quad & \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{A}\|_F^2 + \lambda \sum_{d=1}^{D}\sum_{j=1}^{J} a_{dj} \\ \text{subject to:} \quad & \mathbf{U}^T\mathbf{U} = \mathbf{I}, \quad \mathbf{A} \geqslant 0. \end{aligned} \tag{9.11}$$

9.4 RELATED WORK

There are numerous related work in a larger context of the matrix factorization. Here, we briefly review those work mostly related to us and point out the difference from our model.

9.4.1 *PCA*

Principal component analysis (PCA) [55], which is closely related to LSA, has been widely applied for the dimension reduction purpose. In the content of information retrieval, PCA first center each document by subtracting the sample mean. The resulting document-term matrix is denoted as $\mathbf{Y}$. PCA computes the covariance matrix $\Sigma = \frac{1}{N}\mathbf{Y}^T\mathbf{Y}$ and performs SVD on Σ keeping only the first D eigenvalues: $\Sigma \approx \mathbf{P}\Delta_{D\times D}\mathbf{P}^T$. For each centered document $\mathbf{y}$, its projected image is $\mathbf{P}^T\mathbf{y}$. In recent years, many variants of PCA, including kernel PCA [122], sparse PCA [163], non-negative sparse PCA [153], robust PCA [78], have been developed and successfully applied in many areas.

However, it is worthy to point out that the PCA based dimension reduction techniques are not suitable for the large text corpus due to the following two reasons:

1. When using English words as features, the text corpus represented as $\mathbf{X}$ is a highly sparse matrix where each rows only has a small amount of nonzero entries corresponding to the words appeared in the document. However, by subtracting the sample mean, the centered documents will become a dense matrix which may not fit into memory since the number of documents and words are both very large.

2. PCA and its variants, e.g. sparse PCA, rely on the fact that $\mathbf{Y}^T\mathbf{Y}$ can be fit into memory and SVD can be performed on it. However, for large vocabulary size M, it is very expensive to store M by M matrix and computationally costly to perform SVD on $\mathbf{Y}^T\mathbf{Y}$.

In contrast with PCA and its variants (e.g. sparse PCA), our method directly works on the original sparse matrix without any standardization or utilizing the covariance matrix, hence is more suitable for the text learning task.

9.4.2 *Sparse Coding*

Sparse coding, as another unsupervised learning algorithm, learns basis functions which capture higher-level features in the data and has been successfully applied to image processing [114] and speech recognition [51]. Although the original form of sparse coding is formulated based on the term-document matrix, for the easy of comparison, in our notations, sparse coding [85] can be modeled as:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{A}} \quad & \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{A}\|_F^2 + \lambda\|\mathbf{U}\|_1 \\ \text{subject to:} \quad & \|A_j\|_2^2 \leqslant c, \quad j = 1,\dots M, \end{aligned} \tag{9.12}$$

where c is a predefined constant, $\mathbf{A}$ is called dictionary in sparse coding context; and $\mathbf{U}$ are the coefficients. Instead of projecting the data via $\mathbf{A}$ as our method, sparse coding directly use $\mathbf{U}$ as the projected image of $\mathbf{X}$. Given a new data q, its projected image in the latent space can be computed by solving the following lasso type of problem:

$$\min_{\hat{q}} \frac{1}{2}\|q - \mathbf{A}^T\hat{q}\| + \lambda\|\hat{q}\|_1. \tag{9.13}$$

In the text learning, one drawback is that since the dictionary $\mathbf{A}$ is dense, it is hard to characterize the topic-word relationships from $\mathbf{A}$. Another drawback is that for each new document, the projection operation in (9.13) is computationally very expensive.

9.4.3 *LDA*

Based on the LSA, *probabilistic LSA* [59] was proposed to provide the probabilistic modeling, and further *Latent Dirichlet Allocation* (LDA) [14] provides a Bayesian treatment of the generative process. One great advantage of LDA is that it can provide the distribution of words given a topic and hence rank the words for a topic. The non-negative Sparse LSA proposed in Section 9.3.2 can also provide the most relevant words to a topic and can be roughly viewed as a discriminative version of LDA. However, when the number of latent topics D is very large, LDA performs poorly and the posterior distribution is almost the same as prior. On the other hand, when using smaller D, the documents in the latent topic space generated by LDA are not discriminative for the classification or categorization task. In contrast, as we

show in experiments, our method greatly outperforms LDA in the classification task while providing reasonable ranking of the words for a given topic.

9.4.4 *Matrix Factorization*

Our basic model is also closely related to matrix factorization which finds the low-rank factor matrices $\mathbf{U}$, $\mathbf{A}$ for the given matrix $\mathbf{X}$ such that the approximation error $\|\mathbf{X} - \mathbf{U}\mathbf{A}\|_F^2$ is minimized. Important extensions of matrix factorization include non-negative matrix factorization [84], which enforces the non-negativity constraint to $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{A}$; probabilistic matrix factorization [121], which handles the missing values of $\mathbf{X}$ and becomes one of the most effective algorithms in collaborative filtering; sparse non-negative matrix factorization [60, 70], which enforces sparseness constraint on either $\mathbf{U}$ or $\mathbf{A}$; and orthogonal non-negative matrix factorization [38], which enforces the non-negativity and orthogonality constraints simultaneously on $\mathbf{U}$ and/or $\mathbf{A}$ and studies its relationship to clustering.

As compared to sparse non-negative matrix factorization [60, 70], we add orthogonality constraint to the $\mathbf{U}$ matrix, i.e. $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, which enforces the soft clustering effect of the documents. More specifically, each dimension of the latent space (columns of $\mathbf{U}$) can be viewed as a cluster and the value that a document has on that dimension as its fractional membership in the cluster. The orthogonality constraint tries to cluster the documents into different latent topics. As compared to orthogonal non-negative matrix factorization [38], instead of enforcing both non-negativity and orthogonality constraints, we only enforce orthogonality constraint on $\mathbf{U}$, which further leads to a closed-form solution for optimizing $\mathbf{U}$ as shown in Theorem 9.1. In summary, based on the basic matrix factorization, we combine the orthogonality and sparseness constraints into a unified framework and use it for the purpose of semantic analysis. Another difference between Sparse LSA and matrix factorization is that, instead of treating $\mathbf{A}$ as factor matrix, we use $\mathbf{A}$ as the projection matrix.

9.5 EXPERIMENTAL RESULTS

In this section, we conduct several experiments on real world datasets to test the effectiveness of Sparse LSA and its extensions.

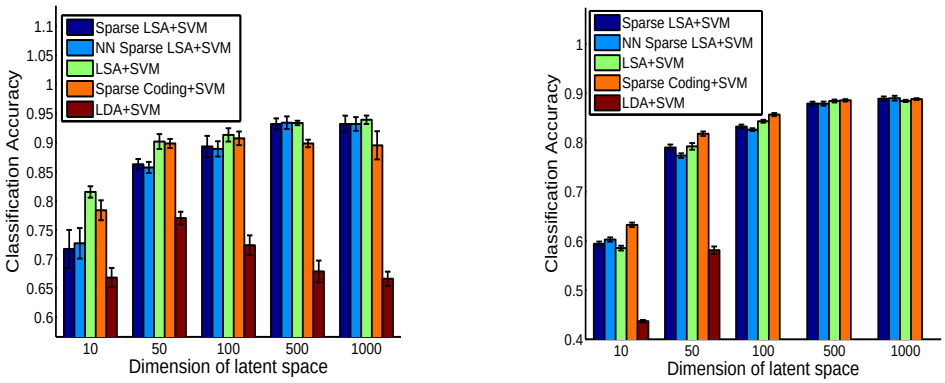
9.5.1 *Text Classification Performance*

In this subsection, we consider the text classification performance after we project the text data into the latent space. We use two widely adopted text classification corpora, 20 Newsgroups (20NG) dataset [2] and RCV1 [88]. For the 20NG, we classify the postings from two newsgroups *alt.atheism* and *talk.religion.misc* using the tf-idf of the vocabulary as features. For RCV1, we remove the words appearing fewer

2 See http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 9.1: The statistics of the experimental datasets

	20NG	RCV1
No. of Samples	1425	15,564
No. of Words	17,390	7,413
No. of Classes	2	53



(a) (b)

Figure 9.2: Classification accuracy vs the dimensionality of latent space for (a) 20NG; (b) RCV1.

than 10 times and standard stopwords; pre-process the data according to [8] [3]; and convert it into a 53 classes classification task. More statistics of the data are shown in Table 9.1.

We evaluate different dimension reduction techniques based on the classification performance of linear SVM classifier. Specifically, we consider

1. Traditional LSA;
2. Sparse Coding with the code from [85] and the regularization parameter is chosen by cross-validation on train set;
3. LDA with the code from [14];
4. Sparse LSA;
5. Non-negative Sparse LSA (NN Sparse LSA).

After projecting the documents to the latent space, we randomly split the documents into training/testing set with the ratio $2:1$ and perform the linear SVM using the package LIBSVM [22] with the regularization parameter $C_{svm} \in \{1e-4, \ldots, 1e+4\}$ selected by 5-fold cross-validation.

Firstly, following the traditional way of comparing different dimension reduction methods, we vary the dimensionality of the latent space and plot the classification accuracy in Figure 9.2. For Sparse LSA and NN Sparse LSA, the regularization parameter λ is fixed to

3 See http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

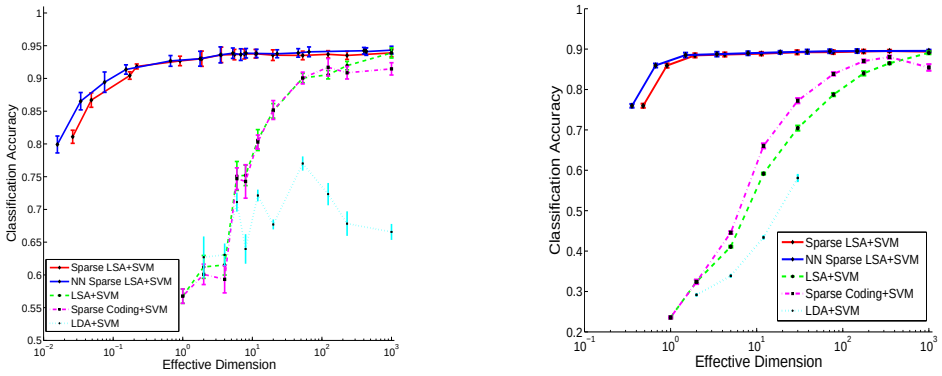
Table 9.2: Density of **A** (%)

Dimension	10	50	100	500	1000
Sparse LSA	1.48	0.80	0.74	0.32	0.18
NN Sparse LSA	1.44	0.72	0.55	0.31	0.17
Other Methods	100	100	100	100	100

(a) 20NG

Dimension	10	50	100	500	1000
Sparse LSA	13.52	7.46	7.40	2.71	1.13
NN Sparse LSA	11.65	4.97	0.40	1.91	0.79
Other Methods	100	100	100	100	100

(b) RCV1



(a) (b)

Figure 9.3: Classification Accuracy vs effective dimension for (a) 20NG (b) RCV1

be 0.05 and the corresponding densities (proportion of nonzero entries) of **A** are shown in Table 9.2.

It can be seen that the performances of LSA and Sparse Coding are comparable. Sparse Coding is slightly worse than LSA for 20NG while slightly better for RCV1. When the dimensionality of latent space is small, Sparse LSA is slightly worse than LSA. It is expectable since the number of parameters in the projection matrix is already very few, sparse model may miss important topic-word relationships. In contrast, for higher dimensional latent space, Sparse LSA shows its advantage in the sense that it can achieve similar classification performance with a highly sparse model (see Table 9.2). A sparse **A** will further save both computational and storage cost as shown in the next section. NN Sparse LSA achieves similar classification performance as Sparse LSA with a more sparse **A**. LDA performs not well for the classification task, which is also expectable since LDA is

a generative model designed for the better interpretability instead of dimension reduction performance [4].

Since Sparse LSA has fewer effective parameters (nonzero entries) in projection matrix, for more fair comparisons, we introduce a concept called *effective dimension* which has been widely adopted in sparse learning. We define the effective dimension of $\mathbf{A}$ to be $\frac{\#\mathrm{nz}(\mathbf{A})}{M}$, where $\#\mathrm{nz}(\mathbf{A})$ is the number of nonzero entries of $\mathbf{A}$ and M is the vocabulary size [5]. For other methods, including LSA, Sparse Coding, LDA, the effective dimension is just the dimensionality of the latent space since all the parameters affects the projection operation. In other words, we compare the classification performance of different methods based on the same number of learned nonzero parameters for the projection.

The result is shown in Figure 9.3. For Sparse LSA and NN Sparse LSA, we fix the number of latent topics to be $D = 1000$ and vary the value of regularization parameter λ from large number (0.5) to small one (0) to achieve different $\#\mathrm{nz}(\mathbf{A})$, i.e. different effective dimensions. As we can see, Sparse LSA and NN Sparse LSA greatly outperform other methods in the sense that they achieve good classification accuracy even for highly sparse models. In practice, we should try to find a λ which could lead to a sparser model while still achieving reasonably good dimension reduction performance.

In summary, Sparse LSA and NN Sparse LSA show their advantages when the dimensionality of latent space is large. They can achieve good classification performance with only a small amount of nonzero parameters in the projection matrix.

9.5.2 *Efficiency and Storage*

In this section, we fix the number of the latent topics to be 1000, regularization parameter $\lambda = 0.05$ and report the projection time, storage and the density of the projected documents for different methods in Table 9.3[6]. The Proj. time is computed as the CPU time for the projection operation and the density of projected documents is the proportion of nonzero entries of $\widehat{q} = \mathbf{A}q$ for a document q. Both quantities are computed for 1000 randomly selected documents in the corpus. Storage is the memory for storing the $\mathbf{A}$ matrix.

For Sparse LSA and NN Sparse LSA, although the classification accuracy is slightly worse (below 1%), the projection time and memory usage are smaller by orders of magnitude than LSA and Sparse Coding. In practice, if we may need to project millions of documents, e.g. web-scale data, into the latent space in a timely fashion (online setting), Sparse LSA and NN Sparse LSA will greatly cut computational cost. Moreover, given a new document q, using Sparse LSA or NN Sparse LSA, the projected document will also be a sparse vector.

4 For RCV1 using LDA, when the dimensionality of the latent space exceeds 100, the classification performance is very poor (nearly random guess). Therefore, we omit the result here.

5 Effective dimension might be less than 1 if $\#\mathrm{nz}(\mathbf{A}) < M$.

6 "Proj.", "Doc", "ACC." are abbreviations for "projection/projected", "document" and "classification accuracy", respectively.

Table 9.3: Computational Efficiency and Storage

	Proj. Time (ms)	Storage (MB)
Sparse LSA	0.25 (4.05E-2)	0.6314
NN Sparse LSA	0.22 (2.78E-2)	0.6041
LSA	31.6 (1.10)	132.68
Sparse Coding	1711.1 (323.9)	132.68
	Density of Proj. Doc. (%)	Acc. (%)
Sparse LSA	35.81 (15.39)	93.01 (1.17)
NN Sparse LSA	35.44 (15.17)	93.00 (1.14)
LSA	100 (0)	93.89 (0.58)
Sparse Coding	86.94 (3.63)	90.54 (1.55)

(a) 20NG

	Proj. Time (ms)	Storage (MB)
Sparse LSA	0.59 (7.36E-2)	1.3374
NN Sparse LSA	0.46 (6.66E-2)	0.9537
LSA	13.2 (0.78)	113.17
Sparse Coding	370.5 (23.3)	113.17
	Density of Proj. Doc. (%)	Acc. (%)
Sparse LSA	55.38 (11.77)	88.88 (0.43)
NN Sparse LSA	46.47 (11.90)	88.97 (0.49)
LSA	100 (0)	89.38 (0.58)
Sparse Coding	83.88 (2.11)	88.79 (1.55)

(b) RCV1

9.5.3 *Topic-word Relationship*

In this section, we qualitatively show that the topic-word relationship learned by NN Sparse LSA as compared to LDA. We use the benchmark data: NIPS proceeding papers[7] from 1988 through 1999 of 1714 articles, with a vocabulary 13,649 words. We vary the λ for NN Sparse LSA so that each topic has at least ten words. The top ten words for the top 7 topics [8] are listed in Table 9.4.

It is very clear that NN Sparse LSA captures different hot topics in machine learning community in 1990s, including neural network, reinforcement learning, mixture model, theory, signal processing and computer vision. For the ease of comparison, we also list the top 7 topics for LDA as in Table 9.5. Although LDA also gives the representative words, the topics learned by LDA are not very discriminative

7 Available at http://cs.nyu.edu/~roweis/data/
8 We use $D = 10$. However, due to the space limit, we report the top 7 topics.

Table 9.4: Topic-word learned by NN Sparse LSA

Topic 1	Topic 2	Topic 3	Topic 4
network	learning	network	model
neural	reinforcement	learning	data
networks	algorithm	data	models
system	function	neural	parameters
neurons	rule	training	mixture
neuron	control	set	likelihood
input	learn	function	distribution
output	weight	model	gaussian
time	action	input	em
systems	policy	networks	variables

Topic 5	Topic 6	Topic 7
function	input	image
functions	output	images
approximation	inputs	recognition
linear	chip	visual
basis	analog	object
threshold	circuit	system
theorem	signal	feature
loss	current	figure
time	action	input
systems	policy	networks

in the sense that all the topics seems to be closely related to neural network.

9.5.4 *Gene Function Identification with Gene Groups Information*

For text retrieval task, it is not obvious to identify the separated group structures among words. Instead, one important application for the group structured Sparse LSA is in gene-set identifications associated to hidden functional structures inside cells. Genes could be naturally separated into groups according to their functions or locations, known as pathways. We use a benchmark breast cancer dataset from [65], which includes a set of cancer tumor examples and each example is represented by a vector of real values, e.g. the quantities of different genes found in the data example. Essentially, group structured Sparse LSA analyzes relationships between the cancer ex-

Table 9.5: Topic-word learned by LDA

Topic 1	Topic 2	Topic 3	Topic 4
learning	figure	algorithm	single
data	model	method	general
model	output	networks	sets
training	neurons	process	time
information	vector	learning	maximum
number	networks	input	paper
algorithm	state	based	rates
performance	layer	function	features
linear	system	error	estimated
input	order	parameter	neural

Topic 5	Topic 6	Topic 7
rate	algorithms	function
unit	set	neural
data	problem	hidden
time	weight	networks
estimation	temporal	recognition
node	prior	output
set	obtain	visual
input	parameter	noise
neural	neural	parameters
properties	simulated	references

amples and genes they contain by discovering a set of "hidden gene functions" (i.e. topics in text case) related to the cancer and the gene groups. And it is of great interest for biologists to determine which sets of gene groups, instead of individual genes, associate to the same latent functions relevant to a certain disease.

Specifically, the benchmark cancer data consists of gene expression values from 3510 genes in 295 breast cancer examples (78 metastatic and 217 non-metastatic). Based each gene's associated "biological process" class in the standard "gene ontology" database [130], we split these 3510 genes into 1103 non-overlapped groups, which we use as group structures in applying group structured Sparse LSA.

We set the parameter $\lambda = 0.12$ and select the first five projected "functional" components which are relevant to 93 gene groups totally. The selected gene groups make a lot of sense with respect to their association with the breast cancer disease.

For instance, 12 gene groups are relevant to the second hidden "function". One selected pathway covering 6 gene variables involves with the so called "unsaturated fatty acid biosynthetic process". High fat diets are well-known to be associated with certain kinds of cancers, including breast cancer, in particular. The predominant usage of excessively high dietary unsaturated *omega-6* fatty acid is at the root of many modern health problems, some of which are concerned with the immune system. The association of this important process to cancer seems very reasonable. Other chosen groups involve functional enrichments of "mRNA metabolic process", "regulation of programmed cell death" and "B cell receptor signal" (B cells are an important component of adaptive immunity), etc. Clearly this hidden function (the 2nd projection) space involves the critical "metabolic" components relevant to important biochemical changes leading to characteristic cell change and death.

Alternatively, we also perform the dimension reduction on this cancer data using the basic Sparse LSA without considering the group structures among genes. The resulting functional components could not be analyzed as clear and as easy as the group structured Sparse LSA case. The reason of the difficulty is that the discovered gene functions are quite large gene groups (i.e. more than 100 genes involved). They represent relatively high level biological processes which are essential for cells in any case, but not necessarily limited to the certain cancer disease this data set is about. It is hard to argue the relationship between such a large amount of genes to the target "breast cancer" cause.

Part VI

CONCLUSIONS AND FUTURE DIRECTIONS

10 CONCLUSIONS AND FUTURE DIRECTIONS

10.1 CONCLUSIONS

In summary, this thesis makes some attempts to address the following challenges arising in big data analytics:

1. **High-dimensionality:** The modern scientific or web data are often ultra-high dimensional (e.g. text data, genetic data) but also extremely sparse in that only a small subset of features are relevant for specific learning tasks and these need to be identified. The ℓ_1-regularized sparse learning methods provide us a suite of powerful tools which strive to identify a small subset of variables maximally relevant for the learning task. This thesis adapts and extends the existing ℓ_1-regularized sparse learning methods in different aspects to address various real-world applications.

2. **Large-scale:** The web data are often large-scale. In particular, for many web applications, the data is collected in a streaming fashion and hence, the number of available instances could be unlimited. This thesis proposes an uniformly-optimal stochastic optimization method for general regularized expected risk minimization problems [30]. And the developed algorithm can be directly applied to perform categorization or ranking tasks for large-scale online information retrieval tasks.

3. **Complex Structures:** Many scientific data are often highly complex, which renders the prediction tasks very difficult. Examples include potentially implicit group structures, such as pathways in genetic and proteomic data; and graph structures, such as dependency and associative relationships in social network analysis, gene regulatory network in microarray data, etc.

    - **Known Prior Structures:** When the knowledge about the structure among variables is given as *a priori*, many structured sparsity-inducing penalties have been proposed to encode such knowledge. However, there is lack of a unified, efficient and scalable optimization method for solving objective functions with structured penalty functions. This thesis presents a general smoothing proximal gradient method to address different learning problems with a wide spectrum of penalty functions, including regression [27, 29], multi-task regression [29] and canonical correlation analysis [31].
    - **Unknown Structures:** When there is unknown hidden structure inside the data, it is desirable to extract those struc-

tures, which might lead to new scientific discoveries. In addition, the hidden structures (groups, networks) of the data are seldom static, as relations and dependencies change over time, location or task. In this thesis, we proposed partition based estimators for predicting dynamic network or forest structures and applied them to various real applications, ranging from climatological data analysis to stock analysis [92].

The aforementioned challenges are core challenges in understanding, extracting useful information and making predictions from high-dimensional large-scale complex data. This thesis conducts some preliminary research on addressing these challenges and opens up many opportunities for future works. In the next section, we will discuss a few promising future directions.

In addition to the proposed optimization and statistical methods, we also summarize the applications of the thesis as follows.

1. **Tumor Classification with Pathway Information:** Tumor classification from microarray data is an important issue in computational biology, which could potentially help detect cancer at an earlier stage. It remains a challenging problem to utilize rich structural information among genes (e.g., pathways) to facilitate the classification task. In Chapter 3, we incorporate pathways information into the overlapping group lasso penalty and solve the corresponding optimization problem using the proposed smoothing proximal gradient method. Our results on the breast cancer data show that we not only achieve a better classification accuracy as compared to the vanilla lasso approach, but also identify some important pathways.

2. **Genome-wide Association Study:** We propose to apply the canonical correlation analysis to an important genome-wide association problem—eQTL mapping, while incorporating rich structural information among genes. We obtain more significant functional enrichment results as compared to the ℓ_1-regularized sparse CCA (see Chapter 4 for more details). It will be great to collaborate with biologists in the future to conduct wet lab experiments to further verify the new observations from structured sparse CCA.

3. **Climate Data Analysis:** Global warming becomes one of the most critical environmental problems in the 21st century. One of the preliminary studies for this problem from the machine learning and statistical aspect is to better understand the correlation among different climatological factors. In Chapter 6, we apply Go-CART to estimate the graphical models among various important climatological factors at different locations of the US. Our results show that such a more refined analysis leads to more interpretable results than estimating a single graphical model by pooling all the data from different locations.

4. **Stock Data Analysis:** We apply the FoCART estimator proposed in Chapter 7 to an interesting stock data analysis problem. In contrast to the relationship between stock market and time which has been extensively studied, we study how the correlation among stocks changes with respect to the oil price and have some interesting observations (see Chapter 7).

5. **Ranking in Text Mining:** We propose to apply sparse learning technique for the ranking task with millions of raw features. Our method achieves fast convergence rate, better generalization ability and takes much less memory for web ranking with millions of features.

6. **Latent Semantic Analysis in Text Mining:** We propose a new topic modeling technique based on sparse learning technique. As compared to the Bayesian latent Dirichlet allocation approach [14], our method leads to more discriminative topic representations as well as better classification accuracy based on the latent representation of the documents.

In addition to the applications studied in this thesis, sparse learning techniques have a much wider spectrum of applications, e.g., computer vision, information retrieval, fMRI, economics, etc. In the next section, we will discuss about some other promising applications that we would like investigate using sparse learning techniques in the future.

10.2 FUTURE DIRECTIONS

The results in this thesis lead to several future directions as follows:

1. **Optimization:** Computation will always be one of the major bottlenecks for learning from massive data since the data grows at an unprecedented rate. Although we have discussed a distributed implementation of *optimal regularized dual averaging* in Chapter 5 using mini-batch strategy from [37], there is still much more room for improvement. It is desirable to develop parallel / distributed algorithms on multi-core machines or network-based clusters to process and learn from terabyte-scale to petabyte-scale data. To achieve that goal, one needs to carefully investigate different locking schemes in distributed optimization to accelerate the algorithm.

   Different optimization methods have their own advantages on different objective functions. In addition to first-order methods, it would be fruitful to investigate other optimization methods (e.g., coordinate descent, alternating direction method of multipliers) and propose their stochastic or distributed extensions to deal with large-scale or streaming data. Eventually, the goal is to provide user-friendly software for general large-scale distributed optimization to the public.

2. **Exploring and Learning Structures:** To understand complex data, it is critical to unveil the underlying structures among variables. How to automatically learn and utilize different hidden structures from data is one of the major challenges in modern data analysis.

   In Chapter 3, we study how to encode prior structural information via structured regularizers in regression settings. However, when the prior information is unavailable, it is important to automatically extract intrinsic structures (e.g., group structure, manifold structure, matrix block structure) from high-dimensional data and utilize the structures to facilitate the predication and data interpretation.

   In addition, learning the dependency relationship as an undirected network structure has been carefully studied in Part iv. There are many others structures, e.g., group structures, forest structures, directed graphical models, that need to be further explored.

3. **Applications:** We expect the results of this thesis could have more applications in various domains, e.g., functional magnetic resonance images (fMRI) study, computer vision, computational genomics and text mining. For example, based on the robust PCA [19], we further proposed a robust sparse matrix factorization approach for video background modeling and activity detection. It can be envisioned that sparse learning will become a powerful tool for many computer vision tasks and the scalable computational methods are increasingly important for addressing those tasks since there will be huge amount of images for the training purpose.

   Take fMRI study as another example, Liu et al. proposed multi-task lasso to address the problem of predicting human brain activity proposed in [102] and we further extended it to adaptive multi-task lasso to boost the performance. Since the fMRI images are usually ultra-high dimensional, sparse learning methods are particularly suitable for modeling fMRI images. In the future, it will be very fruitful to collaborate with domain experts to explore new challenges from real applications and propose new sparse learning methods along with scalable computational tools to address these challenges.

BIBLIOGRAPHY

[1] N. Abate, M. Chandalia, P. Satija, B. Adams-Huet, and et. al. Enpp1/pc-1 k121q polymorphism and genetic susceptibility to type 2 diabetes. *Diabetes*, 54(4):1027–1213, 2005.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73:243–272, 2008.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. Addison-Wesley Harlow, England, 1999.

[4] B. Bai, J. Weston, R. Collobert, and D. Grangier. Supervised semantic indexing. In *ECIR*, 2009.

[5] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9:485–516, March 2008.

[6] A. Beck and M. Teboulle. A fast iterative shrinkage thresholding algorithm for linear inverse problems. *SIAM Journal of Image Science*, 2(1):183–202, 2009.

[7] R. Bekkerman and M. Scholz. Data weaving: Scaling up the state-of-the-art in data clustering. In *CIKM*, 2008.

[8] R. Bekkerman and M. Scholz. Data weaving: Scaling up the state-of-the-art in data clustering. In *Proceedings of ACM International Confernece on Information and Knowledge Management*, 2008.

[9] G. Berriz, J. Beaver, C. Cenik, M. Tasan, and F. Roth. Next generation software for functional trend analysis. *Bioinformatics*, 25(22):3043–3044, 2009.

[10] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[11] P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *Annals of Statistics*, 37(4):1705–1732, 2009.

[12] G. Bindea and B. M. et. al. Cluego: a cytoscape plug-in to decipher functionally grouped gene ontology and pathway annotation networks. *Bioinformatics*, 25 (8):1091–1093, 2009.

[13] G. Blanchard, C. Schäfer, Y. Rozenholc, and K.-R. Müller. Optimal dyadic decision trees. *Mach. Learn.*, 66(2-3):209–241, 2007. ISSN 0885-6125. doi: http://dx.doi.org/10.1007/s10994-007-0717-6.

[14] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[15] J. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer, 2000.

[16] L. Bottou and Y. LeCun. Large-scale on-line learning. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2004.

[17] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and regression trees*. Wadsworth Publishing Co Inc, 1984.

[18] R. B. Brem and L. Krulyak. The landscape of genetic complexity across 5,700 gene expression traits in yeast. *PNAS*, 102 (5):1572–1577, 2005.

[19] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of ACM*, 58(1):1–37, 2009.

[20] R. Caruana. Multitask learning. *Machine Learning Journal*, 28: 41–75, 1997.

[21] A. Cayley. A theorem on trees. *Quarterly Journal of Mathematics*, 23:376–378, 1889.

[22] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[23] A. Chechetka and C. Gusterin. Efficient principled learning of thin junction trees. In *NIPS*, 2007.

[24] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific and Statistical Computing*, 20:33–61, 1998.

[25] X. Chen and H. Liu. An efficient optimization algorithm for structured sparse cca, with applications to eqtl mapping. *Statistics in Biosciences*, 4(1):3–26, 2012.

[26] X. Chen, B. Bai, Y. Qi, Q. Lin, and J. Carbonell. Learning preferences using millions of parameters by enforcing sparsity. In *International Conference on Data Mining (ICDM)*, 2010.

[27] X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. P. Xing. Smoothing proximal gradient method for general structured sparse learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2011.

[28] X. Chen, Y. Qi, B. Bai, Q. Lin, and J. Carbonell. Sparse latent semantic analysis. In *SIAM International Conference on Data Mining (SDM)*, 2011.

[29] X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. P. Xing. Smoothing proximal gradient method for general structured sparse learning. *Annals of Applied Statistics*, 6(2):719–752, 2012.

[30] X. Chen, Q. Lin, and J. Pena. Optimal regularized dual averaging methods for stochastic optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[31] X. Chen, H. Liu, and J. Carbonell. Structured sparse canonical correlation analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[32] M. Choi, V. Tan, A. Anandkumar, and A. Willsky. Learning latent tree graphical models. *Journal of Machine Learning Research*, 1501–1542:2011, 12.

[33] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[34] G. Cormack and T. Lynam. Statistical precision of information retrieval evaluation. In *SIGIR*, 2006.

[35] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[36] S. Deerwester, S. T. Dumais, G. W. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 1990.

[37] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

[38] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *ACM SIGKDD*, 2006.

[39] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

[40] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*, 2010.

[41] J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *Conference on Learning Theory (COLT)*, 2010.

[42] J. Duchi, P. L. Bartlett, and M. Wainwright. Randomized smoothing for stochastic optimization. arXiv:1103.4296v1, 2011.

[43] D. Edwards. *Introduction to graphical modelling*. Springer-Verlag Inc, 1995.

[44] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32:407–499, 2004.

[45] J. Z. et. al. Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature Genetics*, 40:854–861, 2008.

[46] F.Aiolli and A. Sperduti. Learning preferences for multiclass problems. In *Advances in Neural Information Processing Systems 18*, 2004.

[47] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1:302–332, 2007.

[48] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2008.

[49] J. H. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9 (3):432–441, 2007.

[50] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *ECML*, 2003.

[51] S. Garimella, S. Nemala, M. Elhilali, T.Tran, and H. Hermansky. Sparse coding for speech recognition. In *IEEE International Confernece on Acoustics, Speech and Signal Processing*, 2010.

[52] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, i: a generic algorithmic framework. *SIAM Journal on Optimization*, 22:1469–1492, 2012.

[53] A. Ghoting, P. Kambadur, E. Pednault, and R. Kannan. Nimble: A toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In *ACM SIGKDD Conference*, 2011.

[54] D. Grangier and S. Bengio. A discriminative kernel-based approach to rank images from text queries. *IEEE Trans. PAMI.*, 30 (8):1371–1384, 2008. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/TPAMI.2007.70791.

[55] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2 edition, 2009.

[56] E. Hazan and S. Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *Conference on Learning Theory (COLT)*, 2011.

[57] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[58] J.-B. Hiriart-Urruty and C. Lemarechal. *Fundamentals of Convex Analysis*. Springer, 2001.

[59] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 289–296, 1999.

[60] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[61] C. Hu, J. T. Kwok, and W. Pan. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[62] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Systematic and integrative analysis of large gene lists using david bioinformatics resources. *Nature Protoc*, 4(1):44–57, 2009.

[63] J. Huang and T. Zhang. The benefit of group sparsity. *Annals of Statistics*, 38(4):1978–2004, 2010.

[64] Climate Change 2007–The Physical Science Basis *IPCC Fourth Assessment Report*. IPCC, 2007.

[65] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *International Conference on Machine Learning*, 2009.

[66] R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, INRIA, 2009.

[67] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, 2010.

[68] A. Juditsky and Y. Nesterov. Primal-dual subgradient methods for minimizing uniformly convex functions. August 2010.

[69] M. Kanehisa and S. Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28:27–30, 2000.

[70] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23:1495–1502, 2007.

[71] S. Kim and E. P. Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS Genetics*, 5(8), 2009.

[72] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010.

[73] S. Kim, K.-A. Sohn, and E. P. Xing. A multivariate regression approach to association analysis of a quantitative trait network. *Bioinformatics*, 25(12):204–212, 2009.

[74] R. Kinderman and J. L. Snell. *Markov Random Fields and Their Applications*. American Math Society, 1980.

[75] M. Kolar, L. Song, and E. P. Xing. Sparsistent learning of varying-coefficient models with structural changes. In *NIPS*, 2009.

[76] M. Kolar, L. Song, A. Ahmed, and E. P. Xing. Estimating time-varying networks. *The Annals of Applied Statistics*, 4 (1):94–123, 2010.

[77] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[78] F. D. la Torre and M. Black. Robust principal component analysis for computer vision. In *Internation Conference on Computer Vision*, 2001.

[79] G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133 (1):365–397, 2012.

[80] G. Lan and S. Ghadimi. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, part ii: shrinking procedures and optimal algorithms. Technical report, University of Florida, 2010.

[81] G. Lan, Z. Lu, and R. D. C. Monteiro. Primal-dual first-order methods with $o(1/\epsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, 2009.

[82] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[83] S. L. Lauritzen. *Graphical Models*. Oxford: Clarendon Press, 1996.

[84] D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[85] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[86] S. Lee and S. J. Wright. Manifold identification of dual averaging methods for regularized stochastic online learning. In *International Conference on Machine Learning (ICML)*, 2011.

[87] S. Lee, J. Zhu, and E. P. Xing. Adatpive multi-task lasso: with applications to eqtl detection. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[88] D. Lewis, Y.Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization. *Journal of Machine Learning Research*, 5:361–397, 2004.

[89] H. Liu and X. Chen. Multivariate dyadic regression trees for sparse learning problems. In *NIPS*, 2010.

[90] H. Liu, J. Lafferty, and L. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *J. Mach. Learn. Res.*, 10:2295–2328, 2009.

[91] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *International Conference on Machine Learning (ICML)*, 2009.

[92] H. Liu, X. Chen, J. Lafferty, and L. Wasserman. Graph-valued regression. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[93] H. Liu, M. Xu, H. Gu, A. Gupta, J. Lafferty, and L. Wasserman. Forest density estimation. *Journal of Machine Learning Research*, 12:907–951, 2011.

[94] J. Liu and J. Ye. Fast overlapping group lasso. ArXiv:1009.0306v1 [cs.LG], 2010.

[95] J. Liu and J. Ye. Moreau-yosida regularization for grouped tree structure learning. In *NIPS*, 2010.

[96] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *UAI*, 2009.

[97] J. Liu, L. Yuan, and J. Ye. An efficient algorithm for a class of fused lasso problems. In *the 16th ACM SIGKDD*, 2010.

[98] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[99] A. C. Lozano, H. Li, A. Niculescu-Mizil, Y. Liu, C. Perlich, J. Hosking, and N. Abe. Spatial-temporal causal modeling for climate change attribution. In *ACM SIGKDD*, 2009.

[100] S. Ma and M. R. Kosorok. Detection of gene pathways with predictive power for breast cancer prognosis. *BMC Bioinformatics*, 11(1), 2010.

[101] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *NIPS*, 2010.

[102] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K. Chang, V. L. Malave, R. A. Mason, and M. A. Just. Predicting human brain activity associated with the meanings of nouns. *Science*, 320: 1191, 2008.

[103] D. Mol, D. Vito, and L. Rosasco. Elastic net regularization in learning theory. *Journal of Complexity*, 25:201–230, 2009.

[104] S. Negahban and M. J. Wainwright. Simultaneous support recovery in high dimensions: Benefits and perils of block ℓ_1/ℓ_∞-regularization. *IEEE Transactions on Information Theory*, 57 (6): 3841–3863, 2011.

[105] A. Nemirovski and D. Yudin. *Problem complexity and method efficiency in optimization*. John Wiley New York, 1983.

[106] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[107] Y. Nesterov. Excessive gap technique in non-smooth convex minimization. Technical report, UniversitÃ© catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2003.

[108] Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Pub, 2003.

[109] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[110] Y. Nesterov. Gradient methods for minimizing composite objective function. ECORE Discussion Paper 2007, 2007.

[111] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120:221–259, 2009.

[112] G. Obozinski, B. Taskar, and M. I. Jordan. High-dimensional union support recovery in multivariate regression. In *NIPS*, 2009.

[113] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252, 2010.

[114] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[115] A. Rakhlin, O. Shamir, and K. Sridharan. To average or not to average? making stochastic gradient descent optimal for strongly convex problems. In *International Conference on Machine Learning (ICML)*, 2012.

[116] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. Model selection in gaussian graphical models: High-dimensional consistency of l_1-regularized mle. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[117] P. Ravikumar, M. Wainwright, G. Raskutti, and B. Yu. Model selection in Gaussian graphical models: High-dimensional consistency of ℓ_1-regularized MLE. In *Advances in Neural Information Processing Systems 22*, Cambridge, MA, 2009. MIT Press.

[118] P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *Annals of Statistics*, 38 (3):1287–1319, 2010.

[119] R. Rockafellar. *Convex Analysis*. Princeton Univ. Press, 1996.

[120] A. J. Rothman, P. J. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

[121] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, 2007.

[122] B. Schölkopf, A. J. Smola, and K.-R. Müller. Kernel principal component analysis. In *Advances in Kernel Methods—Support Vector Learning*. MIT Press, 1999.

[123] C. Scott and R. Nowak. Minimax-optimal classification with dyadic decision trees. *Information Theory, IEEE Transactions on*, 52(4):1335–1353, 2006.

[124] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning (ICML)*, 2013.

[125] X. Shen and H.-C. Huang. Grouping pursuit through a regularization solution surface. *Journal of the American Statistical Association*, 105(490):727–739, 2010.

[126] A. Subramanian, P. Tamayo, V. Mootha, and et. al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

[127] V. Tan, A. Anandkumar, and A. Willsky. Learning Gaussian tree models: Analysis of error exponents and extremal structures. *IEEE Transactions on Signal Processing*, 58 (5):2701–2714, 2010.

[128] V. Tan, A. Anandkumar, and A. Willsky. A large-deviation analysis for the maximum-likelihood learning of markov tree structures. *IEEE Transactions on Information Theory*, 1714–1735:2011, 57 (3).

[129] V. F. Tan, A. Anandkumar, and A.S.Willsky. Learning high-dimensional markov forest distributions: Analysis of error rates. *Journal of Machine Learning Research*, 1:1–48, 2010.

[130] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–9, 2000.

[131] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437:1399–1320, 2005.

[132] S. Thrum and L. Pratt. *Learning to Learn*. Kluwer Academic Publishers, 1998.

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

[134] R. Tibshirani and M. Saunders. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B*, 67(1):91–108, 2005.

[135] R. Tibshirani and J. Taylor. The solution path of the generalized lasso. *Annals of Statistics*, 39 (3):1335–1371, 2010.

[136] R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics*, 0:1–12, 2007.

[137] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109 (3):475–494, 2001.

[138] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal on Optimization (Submitted)*, 2008.

[139] B. Turlach, W. Venables, and S. Wright. Simultaneous variable selection. *Technometrics*, 47:349–363, 2005.

[140] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical Programming*, 95:189–217, 2003.

[141] T.Y.Liu. *Learning to Rank for Information Retrieval*. Now Publishers Inc, 2009.

[142] M. J. van de Vijver et al. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347:1999–2009, 2002.

[143] M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programs. *IEEE Transactions on Information Theory*, 55:2183–2202, 2009.

[144] S. Webb, J. Caverlee, , and C. Pu. Introducing the webb spam corpus: Using email spam to identify web spam automaticall. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[145] D. Witten and R. Tibshirani. Extensions of sparse canonical correlation analysis with applications to genomic data. *Statistical Applications in Genetics and Molecular Biology*, 8 (1):1–27, 2009.

[146] D. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10:515–534, 2009.

[147] T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2:224–244, 2008.

[148] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

[149] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[150] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68:49–67, 2006.

[151] M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[152] M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[153] R. Zass and A. Shashua. Nonnegative sparse pca. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[154] J. Zhang, Z. Ghahramani, and Y. Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

[155] T. Zhang. Some sharp performance bounds for least squares regression with l1 regularization. *Annals of Statistics*, 37:2109–2114, 2009.

[156] P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

[157] P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497, 2009.

[158] W. Zhong and J. Kwok. Efficient sparse modeling with automatic feature grouping. In *ICML*, 2011.

[159] H. Zhou and K. Lange. A path algorithm for constrained estimation. Technical report, UCLA, 2011. arXiv:1103.3738v1 [stat.CO].

[160] S. Zhou, J. Lafferty, and L. Wasserman. Time varying undirected graphs. *Machine Learning*, 78(4), 2010.

[161] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.

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

[163] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15, 2004.