Spring 5-15-2018

Distributed Quantile Regression Analysis and a Group Variable Selection Method

Liqun Yu
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

Department of Mathematics

Dissertation Examination Committee:
Nan Lin, Chair
Yixin Chen
Jimin Ding
Jose Figueroa-Lopez
Todd Kuffner

Distributed Quantile Regression Analysis and a Group Variable Selection Method

by
Liqun Yu

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2018
St. Louis, Missouri

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful for my advisor, Professor Nan Lin, without whom the completion of this thesis would not have been possible. Prof. Lin not only offered me tremendous help on my Ph.D study and research with his immense knowledge and insight, but also affected me a great deal in life with his patience, passion, and attention to details. His encouragements and advices will continue to benefit me in my future career. I would like to especially thank Prof. Lin for his kind support of my career pursuit.

I would like to thank Professor Xiaogang Su from the University of Texas, El Paso and Professor Lan Wang from the University of Minnesota for the collaboration on my research. I would like to thank Professor Todd Kuffner for organizing the statistics seminars and reading groups, and for organizing the WHOA-PSI workshop, in which I had the opportunity to present my work. I would like to thank my committee members, Professor Yixin Chen, Professor Jimin Ding, and Professor Jose Figueroa-Lopez for taking their valuable time to evaluate my work and for providing advices from different perspectives.

I would also like to thank the Department of Mathematics for the generous support, without which I could not obtain my Ph.D degree. I am full of gratitude to all faculty in the department, from whom I learned so much beautiful mathematics. I would like to thank my fellow graduate students for their support and friendship over the years.

Last but not least, I would like to thank my family, especially my parents and my brother, for their unconditional love and support. I would like to especially thank my grandfather, who passed away in 2013, for his continuous encouragement and support of my education since my childhood, to which I am now forever in debt.

Liqun Yu

Washington University in St. Louis

May, 2018

Dedicated to my family.

ABSTRACT OF THE DISSERTATION

Distributed Quantile Regression Analysis and a Group Variable Selection Method

by

Liqun Yu

Doctor of Philosophy in Mathematics,

Washington University in St. Louis, May, 2018.

Professor Nan Lin, Chair

This dissertation develops novel methodologies for distributed quantile regression analysis for big data by utilizing a distributed optimization algorithm called the alternating direction method of multipliers (ADMM). Specifically, we first write the penalized quantile regression into a specific form that can be solved by the ADMM and propose numerical algorithms for solving the ADMM subproblems. This results in the distributed QR-ADMM algorithm. Then, to further reduce the computational time, we formulate the penalized quantile regression into another equivalent ADMM form in which all the subproblems have exact closed-form solutions and hence avoid iterative numerical methods. This results in the single-loop QPADM algorithm that further improve on the computational efficiency of the QR-ADMM. Both QR-ADMM and QPADM enjoy flexible parallelization by enabling data splitting across both sample space and feature space, which make them especially appealing for the case when both sample size n and feature dimension p are large.

Besides the QR-ADMM and QPADM algorithms for penalized quantile regression, we also develop a group variable selection method by approximating the Bayesian information criterion. Unlike existing penalization methods for feature selection, our proposed gMIC

algorithm is free of parameter tuning and hence enjoys greater computational efficiency. Although the current version of gMIC focuses on the generalized linear model, it can be naturally extended to the quantile regression for feature selection.

We provide theoretical analysis for our proposed methods. Specifically, we conduct numerical convergence analysis for the QR-ADMM and QPADM algorithms, and provide asymptotical theories and oracle property of feature selection for the gMIC method. All our methods are evaluated with simulation studies and real data analysis.

Chapter 1

Introduction

In this chapter, we briefly introduce the quantile regression and its penalized version, and discuss the challenges for quantile regression under the big data context.

1.1 Quantile Regression

Quantile regression, first proposed in the seminar paper [40], provides a useful approach to studying the relationship between a response variable and a set of covariates, particularly when the data are heterogeneous. Assume that the data are generated according to the following model,

$$y_i = \boldsymbol{x}_i \boldsymbol{\beta}(\tau) + \epsilon_i, \ i = 1, \ldots, n,$$

where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ are generated from a distribution $g(\boldsymbol{x})$, and $\boldsymbol{\beta}(\tau) = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ with τ being the quantile of interest. We assume that the conditional distributions of the error $\epsilon_i \mid \boldsymbol{x}_i \sim f_i(\cdot|\boldsymbol{x}_i)$ are independent and satisfy $F_i(0|\boldsymbol{x}_i) = \tau$. Then the conditional quantile of the response variable y_i given $\boldsymbol{x}_i$, denoted as $Q_\tau(y_i|\boldsymbol{x}_i)$, can be expressed as $Q_\tau(y_i|\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{\beta}(\tau)$.

We denote the sample response vector by $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$ and the design matrix

by $X = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})^T \in \mathbb{R}^{n\times p}$. In the classical setting where $n > p$, we can estimate $\boldsymbol{\beta}(\tau)$ by solving

$$\hat{\boldsymbol{\beta}}(\tau) = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \rho_\tau(\boldsymbol{y} - X\boldsymbol{\beta}), \tag{1.1}$$

where $\rho_\tau(u) = u[\tau - \mathbb{I}(u < 0)]$ for $u \in \mathbb{R}$ is the so-called check loss function with $\mathbb{I}(\cdot)$ being the indicator function, and $\rho_\tau(\boldsymbol{y} - X\boldsymbol{\beta}) = \sum_{i=1}^n \rho_\tau(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta})$. An extensive theoretical study of the quantile regression can be found in [39]. Specifically, the asymptotic normality of the estimate $\hat{\boldsymbol{\beta}}(\tau)$ in (1.1) is established as follows,

$$\sqrt{n}(\hat{\boldsymbol{\beta}}(\tau) - \boldsymbol{\beta}(\tau)) \xrightarrow{d} N(0, \Sigma) \tag{1.2}$$

where $\Sigma = \tau(1-\tau)D^{-1}\mathbb{E}(\boldsymbol{x}\boldsymbol{x}^T)D^{-1}$ with $D = \mathbb{E}(\boldsymbol{x}\boldsymbol{x}^T f(0|\boldsymbol{x}))$.

When dealing with high dimensional data, i.e., when p is large, penalization is often required for shrinkage and feature selection. Especially when $p \gg n$, the estimation problem (1.1) is ill-posed, in which case penalization becomes necessary to obtain reliable estimation. Specifically, the penalized quantile regression can be written in the following form,

$$\hat{\boldsymbol{\beta}}(\tau) = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \Big\{\rho_\tau(\boldsymbol{y} - X\boldsymbol{\beta}) + P_\lambda(\boldsymbol{\beta})\Big\}, \tag{1.3}$$

where $P_\lambda(\cdot)$ is a penalty function and the scalar $\lambda > 0$ is a tunable penalization parameter.

Common choices of $P_\lambda(\cdot)$ in (1.3) include the Lasso [69],

$$P_\lambda(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1, \tag{1.4}$$

the elastic net [96],

$$P_\lambda(\boldsymbol{\beta}) = \lambda(\lambda_2\|\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1),\ \lambda_1, \lambda_2 \geq 0, \tag{1.5}$$

the MCP [89],

$$P_\lambda(\boldsymbol{\beta}_{-0}) = \lambda \sum_{j=1}^{p} \left[\left(|\beta_j| - \frac{\beta_j}{2a\lambda} \right) I(0 \le |\beta_j| < a\lambda) + \frac{a\lambda^2}{2} I(|\beta_j| \ge a\lambda) \right], \ a > 1, \tag{1.6}$$

and the SCAD [19],

$$\begin{aligned} p_\lambda(\boldsymbol{\beta}) = \lambda \sum_{j=1}^{p} \Bigg[|\beta_j| I(0 \le |\beta_j| < \lambda) + \frac{a\lambda|\beta_j| - (\beta_j^2 + \lambda^2)/2}{a-1} I(\lambda \le |\beta_j| < a\lambda) \\ + \frac{(a+1)\lambda^2}{2} I(|\beta_j| > a\lambda) \Bigg], \ a > 2. \end{aligned} \tag{1.7}$$

For $p \gg n$, the theory for penalized linear quantile regression has been recently investigated by [3] for the Lasso penalty and by [74] for the non-convex penalties such as SCAD or MCP.

1.2 Challenges for Quantile Regression under the Big Data Context

Owing to the advances in information technologies, massive amount of data are being generated every day. While the abundance of "big data" has blessed us with unprecedented opportunities for knowledge discovery, it imposes fundamental challenges for quantile regression, both statistically and computationally [18, 33, 35, 84]. Especially when data size is too big to store and/or process in a single computer, distributed computational and statistical approaches become necessary.

From the computational perspective, the challenges for solving the quantile regression optimization problems was earlier discussed in [11]. But the discussion was not under the big data context, and was restricted to the unpenalized case (1.1). As mentioned in [11], the optimization problem (1.1) can be written as a linear programming (LP) problem and

solved by the simplex method [32] or the interior point method (IP) [37]. However, these LP methods cannot be generalized to penalized quantile regression with nonlinear penalties. Further, as noticed by [11], the simplex method is computationally infeasible for data beyond a few thousand examples and a few tens of features. The IP, on the other hand, is more scalable and can efficiently solve LP problems with moderate data size, but it still comes short when dealing with data beyond medium scale (typically a few million samples by a few hundred features) [82]. Big data (large n) can be too large for a single computer to process. Sometimes the data collection process itself is distributed. This often requires parallel algorithms that solve problems cooperatively across a group of computers, along with the modern distributed computing frameworks. Besides, to solve the penalized quantile regression optimization (1.3) (large p), algorithms other than the traditional LP algorithms become necessary.

From the statistical perspective, a potential solution to solving large-scale quantile regression is to resort to the *divide-and-combine* (DC) strategy. The DC strategy has been extensively exploited for statistical estimation and inference. The basic idea of DC is to conduct statistical analysis for different subsets/blocks of data and combine the subset results in a way that preserves the statistical efficiency. It is very useful when the data size is too large for a single computer to process and direct analysis on the entire data is infeasible, or when data transferring is prohibitive due to the large amount of communication overhead or security/privacy reasons. A DC procedure is expected to produce statistical results that are asymptotically equivalent to the results from the entire data. An early work in this area is the *aggregated estimating equation* (AEE) in [45, 62]. The AEE combines local estimating equation estimators by forming a certain weighted average. The AEE estimator is shown to be asymptotically equivalent to the global estimator as long as the number of subsets does not grow too fast with sample size. The effectiveness of DC for statistical analysis crucially relies on how the aggregation of local estimators is made. Typically, a

successful combination of local estimators involves aggregating local gradients and Hessians; see [13, 29, 36, 45, 62, 73] for examples. Another benefit of DC-based approaches is that they often enables data compression and online implementation for streaming data. The DC offers a unique opportunity for circumventing large-scale optimization in quantile regression by dividing the large problems into smaller ones. However, since existing DC-based statistical procedures rely on gradients and Hessian matrices of the loss functions, there are still technical difficulties to overcome before applying similar ideas to the non-smooth quantile regression problem.

In this thesis, we provide solutions to large-scale quantile regression from the computational perspective by deriving distributed and scalable optimizations algorithms for solving the penalized quantile regression optimization problem (1.3). Specifically, we write the penalized quantile regression problem (1.3) into specific forms that can be solved by an distributed optimization algorithms called the *alternating direction method of multipliers* (ADMM) and derive specific numerical algorithms to solve the ADMM subproblems. To reduce the time for solving the ADMM subproblems, we propose to introduce new variables to (1.3) that results in a new ADMM form in which all subproblems have closed-form solutions by utilizing the Majorization-Maximization. Then, as an independent but closely related topic, we develop a new group-type feature selection method termed *group minimum information criterion* (gMIC) by minimizing a smooth version of the Bayesian information criterion. Compared to existing penalization methods for feature selection, the gMIC is free of parameter tuning and is hence computationally efficient. Currently, the gMIC is developed under the context of generalized linear models. The same idea can however be generalized to quantile regression models for feature selection.

This thesis is organized as follows. In Chapter 2 we provide an extensive review of the ADMM algorithm and its application to large-scale model fitting. We show that most distributed statistical model fitting problems including the quantile regression can be reduced

to optimization on a connected network and efficiently solved by distributed numerical algorithms like the ADMM. In Chapter 3, we present our first algorithm, the distributed QR-ADMM algorithm, for large-scale penalized quantile regression. In Chapter 4, we combine the Majorization-Maximization and the ADMM and propose the single-loop QPADM algorithm for penalized quantile regression. The QPADM further improve on the computational efficiency of the QR-ADMM algorithm in that all subproblems of QPADM have closed-form solutions, and hence do not require iterative numerical methods to solve. In Chapter 5, we introduce the gMIC algorithm. Finally, Chapter 6 concludes the thesis.

Parts of the materials in this thesis are published in a book and peer-review journals. Please refer to [46, 85, 86] for more details.

Chapter 2

The ADMM and Its Application to Distributed Statistical Modeling

The alternating direction method of multiplier (ADMM) is a distributed convex optimization algorithm that solves a wide range of convex optimization problems. It was first introduced in the 1970s and has become popular in recent years due to its capability of solving large-scale optimization problems arising from the modern statistics and machine learning fields. In this chapter, we review the ADMM algorithm together with some of its variants. We show that many large-scale statistical model fitting problems can be expressed as distributed optimization problems and solved effectively and efficiently by the ADMM. Besides its wide range of applications, the ADMM is also scalable to modern-scale network big data by utilizing the computing power of modern distributed computing frameworks. We investigate different ways of parallelizing the ADMM for the distributed model fitting problems and present the consensus ADMM for distributed network analyses. Then, as a separate but closely related topic, we spare an independent section for a discussion on solving the ADMM subproblems, where a special technique for avoiding expensive numerical methods in the ADMM updates are shown.

2.1 Introduction to the ADMM Algorithm

The ADMM was first introduced by [22, 23] in the 1970s as a general convex optimization algorithm. It enjoys the ease of applicability and has proven to produce great empirical performances for a broad range of problems. It became popular recently due to its capability of solving large-scale optimizations that are now becoming more and more common in practice. In this section, we give a brief overview of the ADMM and also present some of its variants.

2.1.1 The ADMM Algorithm

The ADMM solves the following general optimization problem,

$$\min_{\boldsymbol{x},\boldsymbol{z}} \{f(\boldsymbol{x}) + g(\boldsymbol{z})\} \quad \text{s.t. } A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}, \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{z} \in \mathbb{R}^n$ are the parameters of interest, $A \in \mathbb{R}^{s\times n}, B \in \mathbb{R}^{s\times(p+1)}$ are constant matrices , $\boldsymbol{c} \in \mathbb{R}^s$ is a constant vector, and f and g are the convex objective functions. The ADMM solves (2.1) by iteratively minimizing its *augmented Lagrangian*

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}) := f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{u}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}||A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}||_2^2$$

in the primal variables $\boldsymbol{x}$ and $\boldsymbol{z}$ and updating the dual variable $\boldsymbol{u}$ via dual ascent, where ρ is the tunable augmentation parameter. Specifically, the ADMM carries out the following updates at iteration k,

$$\begin{aligned}
\boldsymbol{x}^{k+1} &:= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \tfrac{\rho}{2}||A\boldsymbol{x} + B\boldsymbol{z}^k - \boldsymbol{c} + \boldsymbol{u}^k/\rho||_2^2, \\
\boldsymbol{z}^{k+1} &:= \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \tfrac{\rho}{2}||A\boldsymbol{x}^{k+1} + B\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}^k/\rho||_2^2, \\
\boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \rho(A\boldsymbol{x}^{k+1} + B\boldsymbol{z}^{k+1} - \boldsymbol{c}).
\end{aligned} \tag{2.2}$$

In general, the updates in (2.2) are easily solvable compared to (2.1), although sometimes they may not have closed-form solutions and require approximation or iterative methods.

The formulation (2.1) is general enough to cover a wide range of problems in statistics, machine learning, engineering, finance, etc. As a matter of fact, the ADMM has been intensively used in these fields for a broad range of applications. A partial list of applications can be found in [4, 5, 10, 16, 21, 47, 64, 71, 83].

As a motivating example, let us consider the statistical model fitting problems. Under such a scenario, the function f is usually the loss function related to the data and g is the regularization on model parameters. One of the most commonly used models is the ℓ_1-penalized linear regression (Lasso, [69]),

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \ \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{2.3}$$

which can be written into the following equivalent ADMM form,

$$\min_{\boldsymbol{\beta},\boldsymbol{\gamma}\in\mathbb{R}^p} \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1 \text{ s.t. } \boldsymbol{\beta} = \boldsymbol{\gamma},$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is the response vector, $X \in \mathbb{R}^{n\times p}$ is the design matrix, $\boldsymbol{\beta} \in \mathbb{R}^p$ is the model coefficient vector, and $\|\cdot\|_p$ denotes the ℓ_p norm. Following (2.2), the problem (2.3) is then solved by iteratively carrying out the updates

$$\begin{aligned} \boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 + (\boldsymbol{u}^k)^T\boldsymbol{\beta} + \tfrac{\rho}{2}\|\boldsymbol{\beta} - \boldsymbol{\gamma}^k\|_2^2, \\ \boldsymbol{\gamma}^{k+1} &:= \arg\min_{\boldsymbol{\gamma}} \tfrac{\rho}{2}\|\boldsymbol{\gamma} - \boldsymbol{\beta}^{k+1} - \boldsymbol{u}^k\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1, \\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \rho(\boldsymbol{\beta}^{k+1} - \boldsymbol{\gamma}^{k+1}), \end{aligned} \tag{2.4}$$

where the $\boldsymbol{\beta}$-update has a closed-form solution and the $\boldsymbol{\gamma}$-update can be solved by soft-

thresholding,

$$\boldsymbol{\gamma}^{k+1} = \left(\boldsymbol{\beta}^{k+1} + \boldsymbol{u}^k - \frac{\lambda}{\rho}\mathbf{1}_p\right)_+ \left(-\boldsymbol{\beta}^{k+1} - \boldsymbol{u}^k - \frac{\lambda}{\rho}\mathbf{1}_p\right)_+. \tag{2.5}$$

The ADMM algorithm (2.2) has $O(1/k)$ convergence rate (k is the iteration number) for general convex problems. Faster convergence rate can be achieved with stronger assumptions, e.g., the strong convexity of the functions $f(\cdot)$ and $g(\cdot)$, and/or full-column rank conditions on the matrices A and B. It is worth mentioning that, the ADMM was originally designed only for convex problems, but was later extended to many noncovex problems, with the convergence established under more strict assumptions compared to the convex case. We refer the readers to [6] for a comprehensive review of the ADMM algorithm.

2.1.2 Some Variants: Stochastic and Online ADMM Algorithms

Several stochastic and online variants of ADMM has been proposed for improvement of computational efficiency and streaming data processing. In this section, we present some stochastic and online ADMM algorithms while refer readers to [55, 68, 72, 91, 92, 94] as a non-exclusive list for a deeper investigation.

Stochastic ADMM

In (2.2), the updates (usually the $\boldsymbol{x}$-update) sometimes have no closed-form solutions and require iterative methods. This results in a double-loop algorithm where the inner loop consists of the iterative method for the updates and the outer loop consists of the ADMM iterations. This can be computationally demanding. To address this issue, stochastic versions of ADMM were proposed. The idea is to linearize the $\boldsymbol{x}$-update in (2.2) so that it has a closed-form solution.

In [55], a basic version of stochastic ADMM was proposed. At iteration k, the $\boldsymbol{x}$-update

is linearized at $\boldsymbol{x}^k$ from the previous iteration,

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} \langle f'(\boldsymbol{x}^k), \boldsymbol{x}\rangle + \frac{\rho}{2}||A\boldsymbol{x} + B\boldsymbol{z}^k - \boldsymbol{c} + \boldsymbol{u}^k/\rho||_2^2 + \frac{\|\boldsymbol{x} - \boldsymbol{x}^k\|_2^2}{2(k+1)}, \tag{2.6}$$

where the last term penalizes the divergence between the solutions from two subsequent iterations. In [91], the $\boldsymbol{x}$-update (2.6) is further linearized as

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} \langle f'(\boldsymbol{x}^k), \boldsymbol{x}\rangle + \rho\langle A\boldsymbol{x}^k + B\boldsymbol{z}^k - \boldsymbol{c} + \boldsymbol{u}^k/\rho, \boldsymbol{x}\rangle + \frac{\|\boldsymbol{x} - \boldsymbol{x}^k\|_2^2}{2(k+1)}. \tag{2.7}$$

A more complicated stochastic version was proposed in [94]. Assume that we have n data samples and

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \ell_i(\boldsymbol{x}),$$

where each ℓ_i is the loss function corresponding to the i-th sample. At time t, an index $k(t) \in \{1, 2, \ldots, n\}$ is randomly selected. Then we define

$$\tau_i(t) = \begin{cases} t & i = k(t), \\ \tau_i(t-1) & \text{otherwise.} \end{cases}$$

The $\boldsymbol{x}$-update (2.7) is then replaced by

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} \left\{ \sum_{i=1}^{n} \langle f'(\boldsymbol{x}^{\tau_i(t)}, \boldsymbol{x}\rangle + L\|\boldsymbol{x} - \boldsymbol{x}^{\tau_i(t)}\|_2^2 \right\} + \rho\langle A\boldsymbol{x}^k + B\boldsymbol{z}^k - \boldsymbol{c} + \boldsymbol{u}^k/\rho, \boldsymbol{x}\rangle, \tag{2.8}$$

where L is a constant. Essentially equation (2.8) means that instead of linearizing all $\ell_i,\ i = 1, 2, \ldots, n$ at the same point $\boldsymbol{x}^k$, we linearize ℓ_i at the $\boldsymbol{x}$ value when the sample i was last visited. Compared to (2.6) and (2.7), equation (2.8) results in a faster convergence rate.

Online ADMM for Data Stream

The standard formulation of ADMM in (2.1) and (2.2) corresponds to the batch setting where we assume the function f is deterministic. For example, in statistical model fitting, e.g. (2.3), this means that the whole data $(X, \boldsymbol{y})$ are collected before fitting the model. Besides being computationally inefficient and challenging for storage when X is large, this assumption is also unrealistic in practice. Especially in network applications, data are collected in a streaming fashion, with new data coming in every day or even every second. An ideal scenario is to update results, e.g., $\boldsymbol{x}, \boldsymbol{z}$ in (2.1) or $\boldsymbol{\beta}$ in (2.3), on the fly and only store results instead of historical data.

Under data streaming scenarios, the problem (2.1) is reformulated as follows,

$$\min_{\boldsymbol{x},\boldsymbol{z}} \sum_{t=1}^{T} (f_t(\boldsymbol{x}) + g(\boldsymbol{z})) \quad \text{s.t. } A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}. \tag{2.9}$$

At time T, $f(\boldsymbol{x}) = \sum_{t=1}^{T} f_t(\boldsymbol{x})$ and f_t only corresponds to a single data sample or a small batch of samples. For example in (2.3), the problem can be reformulated as

$$\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \sum_{t=1}^{T} (\|\boldsymbol{y}_t - X_t\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{\gamma}\|_1) \quad \text{s.t. } \boldsymbol{\beta} = \boldsymbol{\gamma},$$

where $(X_t, \boldsymbol{y}_t)$ is a single sample or a batch of samples collected at time t.

In [72], the online ADMM (OADM) algorithm was proposed to solve (2.9). At time t, we consider solving

$$(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}) = \arg \min_{A\boldsymbol{x}+\boldsymbol{z}=\boldsymbol{c}} f_t(\boldsymbol{x}) + g(\boldsymbol{z}) + \eta B_\phi(\boldsymbol{x}, \boldsymbol{x}_t), \tag{2.10}$$

where $\eta \geq 0$ is the learning rate and $B_\phi(\boldsymbol{x}, \boldsymbol{x}_t) \geq \frac{\alpha}{2}\|\boldsymbol{x} - \boldsymbol{x}_t\|_2^2$ (for some constant α) is the Bregman divergence. The problem (2.10) itself can be solved by the ADMM iteratively so

that the constraint is exactly satisfied, but this results in a double-loop algorithm, which is computationally unappealing. The OADM solves (2.10) by only one pass through the ADMM iteration. The intuition is that, instead of requiring $(\boldsymbol{x}_t, \boldsymbol{z}_t)$ to satisfy the linear constraint for every t, OADM only requires the constraint to be satisfied in the long run. Specifically, the problem (2.9) is written into the following form,

$$\min_{\boldsymbol{x}_t, \boldsymbol{z}_t} \sum_{t=0}^{T} f_t(\boldsymbol{x}_t) + g(\boldsymbol{z}_t) \quad \text{s.t.} \ \sum_{t=1}^{T} \|A\boldsymbol{x}_t + B\boldsymbol{z}_t - \boldsymbol{c}\|_2^2 = o(T),$$

so that the cumulative constraint violation grows at a sub-linear rate, i.e., $o(T)$.

The augmented Lagrangian of (2.10) at time t is

$$L_{\rho t}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}) = f_t(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{u}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) + \eta B_\phi(\boldsymbol{x}, \boldsymbol{x}_t) + \frac{\rho}{2}\|A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}\|_2^2$$

and at time t the OADM consists of just one pass through the following updates,

$$\begin{aligned} \boldsymbol{x}_{t+1} &:= \arg\min_{\boldsymbol{x}} f_t(\boldsymbol{x}) + \boldsymbol{u}_t(A\boldsymbol{x} + B\boldsymbol{z}_t - \boldsymbol{c}) + \tfrac{\rho}{2}\|A\boldsymbol{x} + B\boldsymbol{z}_t - \boldsymbol{c}\|_2^2 + \eta B_\phi(\boldsymbol{x}, \boldsymbol{x}_t), \\ \boldsymbol{z}_{t+1} &:= \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \boldsymbol{u}_t(A\boldsymbol{x}_{t+1} + B\boldsymbol{z} - \boldsymbol{c}) + \tfrac{\rho}{2}\|A\boldsymbol{x}_{t+1} + B\boldsymbol{z} - \boldsymbol{c}\|_2^2, \\ \boldsymbol{u}_{t+1} &:= \boldsymbol{u}_t + \rho(A\boldsymbol{x}_{t+1} + B\boldsymbol{z}_{t+1} - \boldsymbol{c}). \end{aligned} \tag{2.11}$$

In [72], the authors only considered the case where each iteration only processes one sample. Since in (2.10) and (2.11) there is no limitation on how many samples f_t can depend on, the OADM can be naturally applied to the more general case where each iteration takes a batch of newly collected data.

2.2 Distributed Model Fitting with ADMM

Besides being easily applicable to a wide range of problems, another main advantage of ADMM is its ease of parallelization and hence the ability to solve large-scale problems. Many network applications involve large amount of data collected and stored distributedly across a network. In this section, we show how the ADMM is parallelized under the distributed model fitting context, which is commonly encountered in network applications. Two ways of parallelization, i.e., splitting across examples and splitting across features, are presented. Followed by this, we introduce the block-splitting ADMM recently proposed by [56]. The block-splitting ADMM offers a more flexible parallelization scheme by enabling splitting across both examples and features, which is appealing when both the sample size and feature dimension are large.

2.2.1 ADMM for Distributed Model Fitting

This part follows Chapter 8 of [6]. We start with the problem setup then discuss how the ADMM is parallelized.

Problem Setup

A general convex model fitting problem can be written as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} l(X\boldsymbol{\beta}-\boldsymbol{y})+r(\boldsymbol{\beta}), \tag{2.12}$$

where $X\in\mathbb{R}^{n\times p}$ is the design matrix, $\boldsymbol{b}\in\mathbb{R}^n$ is the response vector, $l(\cdot)$ is the loss function, and $r(\cdot)$ is the regularization. In practice, the loss function l is often additive, i.e.

$$l(X\boldsymbol{\beta}-\boldsymbol{y})=\sum_{i=1}^{n} l_i(\boldsymbol{x}_i^T\boldsymbol{\beta}-y_i),$$

where $\boldsymbol{x}_i$ denotes the i-th row of X (i-th sample). The regularization $r(\cdot)$ is also assumed to be additive. For example, $r(\cdot)$ can be the ℓ_2-norm or Lasso [69]. Specifically,

$$r(\boldsymbol{\beta}) = \sum_{j=1}^{p} r(\beta_j).$$

Splitting across Examples

Here we discuss how to solve problem (2.12) in parallel with a large number of samples (large n) and a moderate number of features (relatively small p). First, we partition X and $\boldsymbol{y}$ by rows,

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix},$$

where each $X_i \in \mathbb{R}^{n_i \times p}$ ($\sum_{i=1}^{N} n_i = n$) denotes a block (subset) of data. Next, we write problem (2.12) into the following equivalent form,

$$\min_{\boldsymbol{\beta}_i, \boldsymbol{z} \in \mathbb{R}^p} \sum_{i=1}^{N} l_i(X_i\boldsymbol{\beta}_i - \boldsymbol{y}_i) + r(\boldsymbol{z}) \quad \text{s.t. } \boldsymbol{\beta}_i = \boldsymbol{z},\ i = 1, 2, \ldots, N. \tag{2.13}$$

Problems (2.12) and (2.13) are equivalent since we force each local $\boldsymbol{\beta}_i$ equal to a global $\boldsymbol{z}$. A direct application of (2.2) to (2.13) results in the following update,

$$\begin{aligned} \boldsymbol{\beta}_i^{k+1} &:= \arg\min_{\boldsymbol{\beta}_i \in \mathbb{R}^p} l_i(X_i\boldsymbol{\beta}_i - \boldsymbol{y}_i) + \rho/2 \|\boldsymbol{\beta}_i - \boldsymbol{z}^k + \boldsymbol{u}_i^k/\rho\|_2^2, \\ \boldsymbol{z}^{k+1} &:= \arg\min_{\boldsymbol{z}} r(\boldsymbol{z}) + (N\rho/2)\|\boldsymbol{z} - \bar{\boldsymbol{\beta}}^{k+1} - \bar{\boldsymbol{u}}^k/\rho\|_2^2, \\ \boldsymbol{u}_i^{k+1} &:= \boldsymbol{u}_i^k + \boldsymbol{\beta}_i^{k+1} - \boldsymbol{z}^{k+1}, \end{aligned} \tag{2.14}$$

where $\bar{\boldsymbol{\beta}}^{k+1} = (1/N)\sum_{i=1}^{N} \boldsymbol{\beta}_i^{k+1}$ and $\bar{\boldsymbol{u}}^{k+1} = (1/N)\sum_{i=1}^{N} \boldsymbol{u}_i^{k+1}$, and $\boldsymbol{x}$, $\boldsymbol{z}$, A and B in (2.1) corresponds to $(\boldsymbol{\beta}_1 T, \ldots, \boldsymbol{\beta}_N^T)^T$, $\boldsymbol{z}$, I_{Np}, and $-[I_p \ldots I_p]^T \in \mathbb{R}^{Np \times p}$, respectively.

The first update in (2.14) can be carried out in parallel for each data block. In practice,

the data blocks $X_i,\ i = 1, \ldots, N$ are distributed across N computing nodes with each computing node i storing data block i and carrying out updates with subscript i. The second update requires gathering variables to form the average. Notice that (2.14) does not require $r(\cdot)$ to be separable.

Splitting across Features

In (2.12), when sample size n is not too large but the dimension p is high, the ADMM can be parallelized in a different way. First, the data matrix is partitioned across columns, $X = [X_1 \ldots X_N]$ with $X_i \in \mathbb{R}^{n \times p_i}$ ($\sum_{i=1}^N p_i = p$). And the features are also split correspondingly, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_N^T)^T$. Problem (2.12) is then written as

$$\min_{\boldsymbol{\beta}} l\left(\sum_{i=1}^N X_i\boldsymbol{\beta}_i - \boldsymbol{y}\right) + \sum_{i=1}^N r(\boldsymbol{\beta}_i),$$

or equivalently,

$$\min_{\boldsymbol{\beta}, \boldsymbol{z_i}} l\left(\sum_{i=1}^n \boldsymbol{z}_i - \boldsymbol{y}\right) + \sum_{i=1}^N r(\boldsymbol{\beta}_i) \quad \text{s.t. } X_i\boldsymbol{\beta}_i - \boldsymbol{z}_i = 0,\ i = 1, \ldots, N. \tag{2.15}$$

Applying (2.2) to (2.15) with some algebraic manipulation gives the following updates,

$$\begin{aligned}
\boldsymbol{\beta}_i^{k+1} &:= \arg\min_{\boldsymbol{\beta}_i} r(\boldsymbol{\beta}_i) + (\rho/2)\|X_i\boldsymbol{\beta}_i - X_i\boldsymbol{\beta}_i^k - \bar{\boldsymbol{z}}^k + \overline{X\boldsymbol{\beta}}^k + \boldsymbol{u}^k/\rho\|_2^2, \\
\bar{\boldsymbol{z}}^{k+1} &:= \arg\min_{\bar{\boldsymbol{z}}} l(N\bar{\boldsymbol{z}} - \boldsymbol{\beta}) + (N\rho/2)\|\bar{\boldsymbol{z}} - \overline{X\boldsymbol{\beta}}^{k+1} - \bar{\boldsymbol{u}}^k/\rho\|_2^2, \\
\boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \overline{X\boldsymbol{\beta}}^{k+1} - \boldsymbol{z}^{k+1},
\end{aligned} \tag{2.16}$$

where $\bar{\boldsymbol{z}} = (1/N)\sum_{i=1}^N \boldsymbol{z}_i$, $\bar{\boldsymbol{u}} = (1/N)\sum_{i=1}^N \boldsymbol{u}_i$, and $\overline{X\boldsymbol{\beta}} = (1/N)\sum_{i=1}^N X_i\boldsymbol{\beta}_i$.

The $\boldsymbol{\beta}_i$-update in (2.16) can be carried out in parallel where the update with subscript i is conducted on a local machine i that stores the i-th block of features. The second and third updates involves aggregation across different blocks of features. Notice that the formulation

(2.16) does not require the loss function $l(\cdot)$ to be separable.

2.2.2 Flexible Parallelization via Block Splitting

Section 2.2.1 showed how the ADMM is parallelized across examples or features. In [56], a block splitting formulation of ADMM that can be parallelized across both examples and features was proposed. In modern big data applications, the data sometimes are not only big in size (large n), but also high-dimensional (large p). The block splitting ADMM is especially appealing for such problems. The rest of this section follows [56].

The block splitting aims to solve the following problem,

$$\min_{\boldsymbol{x}\in\mathbb{R}^m,\boldsymbol{z}\in\mathbb{R}^n}\{f(\boldsymbol{x})+g(\boldsymbol{z})\} \quad \text{s.t. } \boldsymbol{x}=A\boldsymbol{z}, \tag{2.17}$$

where $A\in\mathbb{R}^{m\times n}$, and functions f and g are assumed to be block separable, i.e.,

$$f(\boldsymbol{x})=\sum_{i=1}^{M} f_i(\boldsymbol{x}_i), \quad g(\boldsymbol{z})=\sum_{j=1}^{N} g_j(\boldsymbol{z}_j),$$

where $\boldsymbol{x}=(\boldsymbol{x}_1^T,\ldots,\boldsymbol{x}_M^T)^T$ and $\boldsymbol{z}=(\boldsymbol{z}_1^T,\ldots,\boldsymbol{z}_M^T)^T$. And $\boldsymbol{x}_i\in\mathbb{R}^{m_i}$, $\boldsymbol{z}_j\in\mathbb{R}^{n_j}$ with $\sum_{i=1}^{M} m_i=m$ and $\sum_{j=1}^{N} n_i=n$. Correspondingly,

$$A=\begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \ldots & A_{MN} \end{bmatrix} \text{ with } A_{ij}\in\mathbb{R}^{m_i\times n_j}.$$

As a result of the block splitting above, problem (2.17) is written as

$$\min_{\boldsymbol{x}\in\mathbb{R}^m,\boldsymbol{z}\in\mathbb{R}^n} \sum_{i=1}^{M} f_i(\boldsymbol{x}_i) + \sum_{j=1}^{N} g_j(\boldsymbol{z}_j) \quad \text{s.t. } \boldsymbol{x}_i = \sum_{j=1}^{N} A_{ij}\boldsymbol{z}_j,\ i = 1,\ldots,M, \tag{2.18}$$

Setting $\boldsymbol{r} = X\boldsymbol{\beta} - \boldsymbol{y}$, the distributed model fitting problem (2.12) is a special case of (2.18) with $\boldsymbol{x} = \boldsymbol{r} + \boldsymbol{y}$, $\boldsymbol{z} = \boldsymbol{\beta}$, $f(\boldsymbol{x}) = l(\boldsymbol{x} - \boldsymbol{y}) = l(\boldsymbol{r})$, $f_i(\boldsymbol{x}_i) = l(\boldsymbol{x}_i - \boldsymbol{y}_i) = l(\boldsymbol{r}_i)$, $g(\cdot) = g_j(\cdot) = r(\cdot)$, and $A = X$.

Problem (2.18) can be further written as

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^m,\boldsymbol{z}\in\mathbb{R}^n} \quad & \textstyle\sum_{i=1}^{M} f_i(\boldsymbol{x}_i) + \sum_{j=1}^{N} g_j(\boldsymbol{z}_j) \\ \text{s.t.} \quad & \boldsymbol{z}_j = \boldsymbol{z}_{ij},\ \boldsymbol{x}_i = \textstyle\sum_{i=1}^{N} \boldsymbol{x}_{ij}, \quad i = 1,\ldots,M \\ & \boldsymbol{x}_{ij} = A_{ij}\boldsymbol{z}_{ij}, \quad i = 1,\ldots,M, \quad j = 1,\ldots,N, \end{aligned} \tag{2.19}$$

or equivalently,

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^m,\boldsymbol{z}\in\mathbb{R}^n} \quad & \textstyle\sum_{i=1}^{M} f_i(\boldsymbol{x}_i) + \sum_{j=1}^{N} g_j(\boldsymbol{z}_j) + \sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(\boldsymbol{x}_{ij}, \boldsymbol{z}_{ij}) \\ \text{s.t.} \quad & \boldsymbol{z}_j = \boldsymbol{z}_{ij},\ \boldsymbol{x}_i = \textstyle\sum_{i=1}^{N} \boldsymbol{x}_{ij}, \quad i = 1,\ldots,M, \end{aligned} \tag{2.20}$$

where I_{ij} is the indicator function of the graph of A_{ij}, i.e.,

$$I_{ij}(\boldsymbol{x}_{ij}, \boldsymbol{z}_{ij}) = \begin{cases} 0 & \text{if } \boldsymbol{x}_{ij} = A_{ij}\boldsymbol{z}_{ij}, \\ \infty & \text{otherwise.} \end{cases}$$

Problem (2.20) can be solved by the formulation of ADMM for the generic convex constrained optimization problem

$$\min_{w} \varphi(w) \quad \text{s.t. } w \in \mathcal{C}, \tag{2.21}$$

where $\varphi(\cdot)$ is a convex function and $\mathcal{C}$ is a closed convex set. Moving the constraint to the

objective function, we have the following equivalent problem,

$$\min_{w} \varphi(w^{1/2}) + I_{\mathcal{C}}(w^1) \quad \text{s.t. } w^{1/2} = w^1, \tag{2.22}$$

where $I_{\mathcal{C}}(\cdot)$ is the indicator function of $\mathcal{C}$. Denoting the value of $w^{1/2}$ at k-th iteration as $w^{k+1/2}$ and the value of w^1 at k-th iteration as w^{k+1}, from (2.2), problem (2.22) is solved by

$$\begin{aligned} w^{k+1/2} &:= \mathbf{prox}_{\varphi}(w^k - \tilde{w}^k), \\ w^{k+1} &:= \arg\min_{w^1} I_{\mathcal{C}}(w^1) + (\rho/2)\|w^1 - (w^{k+1/2} + \tilde{w}^k)\|_2^2 \\ &= \Pi_{\mathcal{C}}(w^{k+1/2} + \tilde{w}^k), \\ \tilde{w}^{k+1} &:= \tilde{w}^k + w^{k+1/2} - w^{k+1}, \end{aligned} \tag{2.23}$$

where $\mathbf{prox}_{\varphi}(\nu) = \arg\min_w \left(\varphi(w) + (\rho/2)\|w - \nu\|_2^2\right)$, $\Pi_{\mathcal{C}}$ is the projection onto $\mathcal{C}$, and $\tilde{w}$ is the dual variable equivalent to $\boldsymbol{u}$ in (2.2). Applying (2.23) to (2.20) with w consisting of $\boldsymbol{x}$, $\boldsymbol{z}$, $(\boldsymbol{x}_{ij},\ \boldsymbol{z}_{ij})$, $i = 1, \ldots, M$, $j = 1, \ldots, N$, and $\varphi(w) = \sum_{i=1}^{M} f_i(\boldsymbol{x}_i) + \sum_{j=1}^{N} g_j(\boldsymbol{z}_j) + \sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(\boldsymbol{x}_{ij}, \boldsymbol{z}_{ij})$, we have the following updates,

$$\begin{aligned} \boldsymbol{x}_i^{k+1/2} &:= \mathbf{prox}_{f_i}(\boldsymbol{x}_i^k - \tilde{\boldsymbol{x}_i}^k), \\ \boldsymbol{z}_j^{k+1/2} &:= \mathbf{prox}_{g_j}(\boldsymbol{z}_j^k - \tilde{\boldsymbol{z}_j}^k), \\ (\boldsymbol{z}_{ij}^{k+1/2}, \boldsymbol{x}_{ij}^{k+1/2}) &:= \Pi_{ij}(\boldsymbol{z}_{ij}^k - \tilde{\boldsymbol{z}_{ij}}^k, \boldsymbol{x}_{ij}^k - \tilde{\boldsymbol{x}_{ij}}^k) \\ \boldsymbol{z}_j^{k+1} &:= \left(\boldsymbol{z}_j^{k+1/2} + \textstyle\sum_{i=1}^{M} \boldsymbol{z}_{ij}^{k+1/2}\right)/(M+1) \\ \boldsymbol{x}_i^{k+1} &:= \boldsymbol{x}_i^{k+1/2} - \left(\boldsymbol{x}_i^{k+1/2} - \textstyle\sum_{j=1}^{N} \boldsymbol{x}_{ij}^{k+1/2}\right)/(N+1) \\ \boldsymbol{x}_{ij}^{k+1} &:= \boldsymbol{x}_{ij}^{k+1/2} + \left(\boldsymbol{x}_i^{k+1/2} - \textstyle\sum_{j=1}^{N} \boldsymbol{x}_{ij}^{k+1/2}\right)/(N+1) \\ \tilde{\boldsymbol{z}}_j^{k+1} &:= \tilde{\boldsymbol{z}}_j^k + \boldsymbol{z}_j^{k+1/2} - \boldsymbol{z}_j^{k+1}, \\ \tilde{\boldsymbol{x}}_i^{k+1} &:= \tilde{\boldsymbol{x}}_i^k + \boldsymbol{x}_i^{k+1/2} - \boldsymbol{x}_i^{k+1}, \\ \tilde{\boldsymbol{z}}_{ij}^{k+1} &:= \tilde{\boldsymbol{z}}_{ij}^k + \boldsymbol{z}_{ij}^{k+1/2} - \boldsymbol{z}_j^{k+1}, \end{aligned} \tag{2.24}$$

where Π_{ij} denotes the projection onto $\{(\boldsymbol{c}, \boldsymbol{d}) \in \mathbb{R}^{m+n} | \boldsymbol{d} = A_{ij}\boldsymbol{c}\}$. The formulation (2.24)

involves some simplification process, we refer the readers to [56] for technical details.

In (2.24), each of the M $\boldsymbol{x}_i^{k+1/2}$-updates, N $\boldsymbol{z}_j^{k+1/2}$-updates, and the MN $(\boldsymbol{z}_{ij}^{k+1/2}, \boldsymbol{x}_{ij}^{k+1/2})$-updates can be carried out in parallel on different machines, so do the updates in the last three equations of (2.24). The fourth to sixth lines of (2.24) involves aggregation and hence communication between different machines. In [56], the communication details are described with graphs, and parallel implementation of (2.24) on Amazon EC2 is also presented in the simulation part. We refer interested readers to [56] for details.

2.3 Distributed ADMM for Big Data Optimization

In many applications, data are often collected and stored across a distributed network consisting of computing nodes from different locations. For many of these applications, the tasks reduce to distributed model fitting across a connected cluster of computing nodes. This results in optimizing a global object function which is a combination of local objective functions known by the local computing nodes only. For example, in spam filtering, emails are distributed across user computers or over the cloud. The goal is to build a spam filter that detects spams. This involves building a classifier by minimizing some global loss function (e.g., number of misclassification for all users), which is a sum of local loss function (number of misclassification for each user).

Due to the distributed nature of such applications, it is often unrealistic to collect and process all data in a single computer. On one hand, transferring local data to a center results in huge communication overhead. On the other hand, in most real applications, the data are almost surely too large for a single computer to store or process. Hence, algorithms that are capable of solving problems collectively over the network are required for such applications.

The ADMM is one such algorithm. In the previous section, we presented the parallelization of ADMM for distributed model fitting problems. The distributed model fitting

problem is a special case of a generic problem in network applications called the *consensus* problem. In this section, we present several versions of the distributed ADMM algorithms for optimization over network big data that solve the consensus problem in different ways. The distributed ADMM is communication efficient and flexible to the network topology.

2.3.1 The Consensus Problem

Consider the optimization

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}), \tag{2.25}$$

where the goal is to find a global variable $\boldsymbol{x}$ that minimizes the global object function f that can be split into N objective functions $f_1, \ldots, f_N$. Equivalently, we solve

$$\min_{\boldsymbol{x}} \sum_{i=1}^{N} f_i(\boldsymbol{x}_i), \quad \text{s.t. } \boldsymbol{x}_i = \boldsymbol{z},\ i = 1, 2, \ldots, N. \tag{2.26}$$

Problem (2.26) is called the *global consensus* by the fact that all local variables $\boldsymbol{x}_i,\ i = 1, \ldots, N$ are forced to agree with the global variable $\boldsymbol{z}$. Sometimes, we consider adding certain regularization on the global variable $\boldsymbol{z}$, which results in the regularized consensus problem

$$\min_{\boldsymbol{x}} \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + g(z), \quad \text{s.t. } \boldsymbol{x}_i = \boldsymbol{z},\ i = 1, 2, \ldots, N, \tag{2.27}$$

where g is the regularization function.

A direct application of (2.2) to (2.26) and (2.27) gives us the following updates

$$\begin{aligned} \boldsymbol{x}_i^{k+1} &:= \arg\min_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \tfrac{\rho}{2}||\boldsymbol{x}_i - \boldsymbol{z} + \boldsymbol{u}^k/\rho||_2^2, \\ \boldsymbol{z}^{k+1} &:= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \left(\boldsymbol{x}_i^{k+1} + \boldsymbol{u}_i^k/\rho\right), \\ \boldsymbol{u}_i^{k+1} &:= \boldsymbol{u}_i^k + \rho(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{k+1}), \end{aligned} \tag{2.28}$$

and

$$\begin{aligned}
\boldsymbol{x}_i^{k+1} &:= \arg\min_{\boldsymbol{x}} f_i(\boldsymbol{x}_i) + \tfrac{\rho}{2}||\boldsymbol{x}_i - \boldsymbol{z} + \boldsymbol{u}^k/\rho||_2^2, \\
\boldsymbol{z}^{k+1} &:= \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \tfrac{N\rho}{2}||\boldsymbol{z} - \bar{\boldsymbol{x}}^{k+1} - \bar{\boldsymbol{u}}^k/\rho||_2^2, \\
\boldsymbol{u}_i^{k+1} &:= \boldsymbol{u}_i^k + \rho(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{k+1}),
\end{aligned} \tag{2.29}$$

respectively, where the upper bar denotes the average over $i = 1, 2, \ldots, N$.

In practice, the functions f_i's are often only known to a local agent i (e.g., local processors or computing nodes), and solving the consensus problem involves minimizing the global objective cooperatively across local agents. For example, the distributed model fitting problem (2.13) in Section 2.2 is a special case of consensus problem with f_i being the local loss function l_i, which depends on the i-th block of data, and is hence only known to agent i that stores the local data X_i and $\boldsymbol{y}_i$.

The consensus problem finds its application in many fields, e.g., in signal processing and wireless communication, see Chapter 7 of [6] and the references therein.

An Extension: the Asynchronous Consensus ADMM

In (2.28) and (2.29), the updates are synchronized. Each agent i conducts the $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$-updates and sends the result to the *master*. After receiving updates from all agents, the master updates $\boldsymbol{z}$ with the aggregated $\boldsymbol{x}$ and $\boldsymbol{u}$ values from across agents $1, 2, \ldots, N$. The master then sends the updated $\boldsymbol{z}$ to each agent for the next iteration. This can be problematic in practice. Since the master cannot proceed without receiving updates from all agents, the overall performance is decided by the slowest agent in the network. The master and all other agents have to wait until the slowest agent to finish its updates before they can proceed to the next iteration.

To address this issue, an asynchronous version of the consensus ADMM (2.26) and (2.27) was proposed in [90]. In the asynchronous consensus ADMM, the master and each agent keep their own timeline. The master keeps the master clock k which starts at 1 and increases

by 1 after each $\boldsymbol{z}$-update; agent i keeps its worker clock k_i which also starts at 1 and increases by 1 after each $\boldsymbol{u}_i$-update. For the master node, it does not have to wait until all agents finish. Instead, it can proceed after receiving updates from at least S ($1 \leq S \leq N$) agents. In practice, S can be much smaller than N. The master then carries out the $\boldsymbol{z}$-update with the S updated $\boldsymbol{x}_i,\ \boldsymbol{u}_i$ values and $N-S$ outdated $\boldsymbol{x}_i,\ \boldsymbol{u}_i$ values. The updated $\boldsymbol{z}$ value is then sent back only to the S agents that sent their updates to the master in the latest iteration, and each of the remaining $N-S$ agents that did not send updates to the master uses the latest $\boldsymbol{z}$ value it received. For agent i, denoting the latest $\boldsymbol{z}$-value it received as $\tilde{\boldsymbol{z}}_i$, then each agent i updates $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ with the outdates $\tilde{\boldsymbol{z}}_i$. To make sure that the asynchronization works, a constraint called *bounded delay condition* is imposed. The bounded delay condition requires that updates from each agent has to be served at least once every τ iterations, where $\tau \geq 1$ is a user-defined parameter. A counter τ_i is kept by the master for each agent i. Each τ_i increases as the master clock k increases. But once the master receives the updates from agent i, the corresponding τ_i is reset to 1. The bounded delay condition guarantees the freshness of the updates from each agents.

The convergence of the asynchronous consensus ADMM was analyzed in [90]. The convergence rate of $O(\frac{N\tau}{TS})$ is established, where T denotes the iteration number. Decreasing S and increasing τ result in slower convergence rate but may benefit from faster speed when there are slow agents in the network, as indicated by the simulations in [90].

2.3.2 Distributed ADMM for the Consensus Problem

In the previous section, we showed how the ADMM is applied to solving the consensus problem. The problem formulation (2.26) and (2.28) or (2.27) and (2.29) results in a centralized network topology where all agents communicate with a center (master). The centralized network topology may incur communication inefficiency and instability. On the one hand, the center is overloaded with communication with the whole network. And establishing a

direct connection between all agents and the center can be expensive or unrealistic in practice. On the other hand, the performance of such centralized network depends crucially on the center. The success of the entire network is at stake if anything goes wrong inside the center. To increase robustness of the network and improve communication efficiency, several distributed consensus ADMM formulations were proposed. The word "distributed" means decentralization, i.e., all computing nodes in the network are treated equally and there is no master role in the network. The topic of the limitation of centralization and the prospect of decentralization is far beyond the scope of this chapter, we strongly recommend readers to the amazing book of Kevin Kelly [38], which provides profound insights about the connection between decentralization and the emergence of machine intelligence. In the following, we present the distributed consensus ADMM algorithm.

Problem Formulation

We follow the distributed ADMM formulation of [76]. Consider a network represented by an undirected connected simple graph with N nodes and M edges, $G = \{V, E\}$, where V and E denote the sets of nodes (agents) and edges (connection between agents) of the network, respectively. We assume nodes are ordered from 1 to N and denote the edge between nodes i and j as e_{ij} $(i < j)$. To simplify the notation, we assume the functions f_i in (2.25) are univariate and replace the notation of variable $\boldsymbol{x}$ by x for the remainder of this section.

To concisely represent the distributed ADMM formulation, we define the *edge-node incidence matrix* of the network G as a matrix $A \in \mathbb{R}^{M \times N}$, with each row corresponding to an edge and each column corresponding to a node. The row corresponding to the edge e_{ij}, denoted by $[A]^{e_{ij}}$, has 1 in its i-th coordinate and -1 in its j-th coordinate and 0 for other

coordinates. Specifically, the elements of A is given by

$$[A]_k^{e_{ij}} = \begin{cases} 1 & \text{if } k = i, \\ -1 & \text{if } k = j, \\ 0 & \text{otherwise.} \end{cases}$$

Now instead of requiring all local x_i's in (2.25) to be equal to a global z, we only require local variables x_i's to be equal to variables of its neighboring nodes. This requirement is represented concisely as

$$\min_{\boldsymbol{x}=(x_1,\ldots,x_N)^T} \sum_{i=1}^{N} f_i(x_i) \quad \text{s.t. } A\boldsymbol{x} = 0. \tag{2.30}$$

Since the graph G is connected, problem (2.30) is equivalent to (2.25) and (2.26).

The Distributed Consensus ADMM

For each node i in network G, we partition its neighbors into the *predecessors* and the *successors*, defined by $P(i) = \{j \mid e_{ij} \in E,\ i > j\}$ and $S(i) = \{j \mid e_{ij} \in E,\ i < j\}$. In [76], the authors applied the ADMM to (2.30) with the variables updated in a sequential order from x_1 to x_N. This results in the distributed ADMM algorithm as follows,

$$\begin{aligned} x_i^{k+1} &:= \arg\min_{x_i} f_i(x_i) + \tfrac{\rho}{2} \textstyle\sum_{j\in P(i)} \|x_j^{k+1} - x_i - \frac{1}{\rho} u_{ji}^k\|_2^2 + \frac{\rho}{2} \sum_{j\in S(i)} \|x_j^k - x_i - \frac{1}{\rho} u_{ij}^k\|_2^2, \\ u_{ji}^{k+1} &:= u_{ji}^k - \rho(x_j^{k+1} - x_i^{k+1}), \end{aligned} \tag{2.31}$$

where each agent i updates x_i and the u_{ji}'s it owns, i.e., u_{ji}'s for all $j \in P(i)$.

For the formulation (2.31), the x_i-update and u_{ji} update only rely on the x_j values from its neighboring nodes in the network, and hence the communication is only between neighboring nodes, i.e., the communication is decentralized.

Decentralization on Colored Networks

One drawback of the formulation (2.31) is that, the variables are updated sequentially. Each agent finishes its own update and sends the results to its successors, and an agent cannot proceed before receiving updates from all its predecessor. This node-by-node updating mechanism may require significant amount of time for each iteration of ADMM. To allow for certain amount of parallelization, a different version of distributed ADMM was proposed in [53].

The distributed ADMM formulation of [53] relies on the existence of *coloring scheme* for the network. To be specific, a coloring scheme is an assignment of colors to the nodes in the network such that no adjacent nodes have the same color. In practice, we want to use as few colors as possible for a network.

Assume that there are C colors in the coloring scheme. Without loss of generality, we assume that the nodes are ordered such that the first C_1 nodes have color 1, and the next C_2 nodes have color 2, and so on. The variable's $\boldsymbol{x} = (x_1, \ldots, x_N)^T$ and the edge-node matrix A are split accordingly,

$$\boldsymbol{x} = (\underbrace{x_1, \ldots, x_{C_1}}_{\bar{\boldsymbol{x}}_1}, \ldots, \underbrace{x_{N-C_N+1}, \ldots, x_N}_{\bar{\boldsymbol{x}}_C})^T,$$

and

$$A = [A_1, \ldots, A_C].$$

Problem (2.30) is then written as

$$\min_{\boldsymbol{x}} \sum_{c=1}^{C} \sum_{i \in \mathcal{C}_c} f_i(x_i) \quad \text{s.t. } A_1 \bar{\boldsymbol{x}}_1 + \ldots + A_C \bar{\boldsymbol{x}}_C = 0, \tag{2.32}$$

where $\mathcal{C}_c$ is the set consisting of the indices of nodes with color c.

Compared to the standard ADMM in (2.1), the problem (2.32) is called the *extended ADMM* form since there are more than two sets of variables when $C > 2$. The extended ADMM algorithm solves (2.32) by the following updates,

$$\begin{aligned} \bar{\boldsymbol{x}}_1^{k+1} &:= \arg\min_{\bar{\boldsymbol{x}}_1} L_\rho(\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2^k, \ldots, \bar{\boldsymbol{x}}_C^k; \boldsymbol{u}^k), \\ &\vdots \\ \bar{\boldsymbol{x}}_C^{k+1} &:= \arg\min_{\bar{\boldsymbol{x}}_C} L_\rho(\bar{\boldsymbol{x}}_1^{k+1}, \ldots, \bar{\boldsymbol{x}}_{C-1}^{k+1}, \bar{\boldsymbol{x}}_C; \boldsymbol{u}^k), \\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \rho \textstyle\sum_{c=1}^{C} A_c \bar{\boldsymbol{x}}_c^{k+1}, \end{aligned} \tag{2.33}$$

where L_ρ is the Lagrangian

$$L_\rho(\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_C; \boldsymbol{u}) = \sum_{i=1}^{N} f_i(x_i) + \boldsymbol{u}^T(A_1\bar{\boldsymbol{x}}_1 + \ldots + A_C\bar{\boldsymbol{x}}_C) + \frac{\rho}{2}\|A_1\bar{\boldsymbol{x}}_1 + \ldots + A_C\bar{\boldsymbol{x}}_C\|_2^2.$$

Explicitly, we have

$$\bar{\boldsymbol{x}}_c^{k+1} = \arg\min_{\bar{\boldsymbol{x}}_c} \sum_{i\in\mathcal{C}_c} f_i(x_i) + \boldsymbol{u}^{k^T} A_c\bar{\boldsymbol{x}}_c + \frac{\rho}{2}\|A_c\bar{\boldsymbol{x}}_c + \sum_{d<c} A_d\bar{\boldsymbol{x}}_d^{k+1} + \sum_{d>c} A_d\bar{\boldsymbol{x}}_d^k\|_2^2. \tag{2.34}$$

Denoting the set of nodes neighboring i as $\mathcal{N}_i$ and $D_i = |\mathcal{N}_i|$ and utilizing the fact that nodes with the same color are not adjacent to each other, the $\bar{\boldsymbol{x}}_c$-update ($c = 1, 2, \ldots, C$) (2.34) can be simplified as

$$\bar{\boldsymbol{x}}_c^{k+1} = \arg\min_{\bar{\boldsymbol{x}}_c} \sum_{i\in\mathcal{C}_c} \left(f_i(x_i) + v_i^k x_i + \frac{D_i\rho}{2} x_i^2 \right), \tag{2.35}$$

where

$$v_i^k := \gamma_i^k - \rho \sum_{j\in\mathcal{N}_i, j<i} x_j^{k+1} - \rho \sum_{j\in\mathcal{N}_i, j>i} x_j^k, \text{ with } \gamma_i^k = \sum_{j\in\mathcal{N}_i, j<i} u_{ji}^k - \sum_{j\in\mathcal{N}_i, j>i} u_{ji}^k.$$

Notice that the update (2.35) can be parallelized for nodes with color c,

$$x_i = \arg\min_{x_1} f_i(x_i) + v_i^k x_i + \frac{D_i \rho}{2} x_i^2 \quad \text{for } i \in \mathcal{C}_c.$$

Since the x_i-updates does not depend directly on $\boldsymbol{u}$, the $\boldsymbol{u}$-update can also be simplified to

$$\gamma_i^{k+1} = \gamma_i^k + \rho \sum_{j \in \mathcal{N}_i} (x_i^{k+1} - x_j^{k+1}),\ i = 1, 2, \ldots, N,$$

which is carried out in parallel for $i = 1, 2, \ldots, N$.

If we take a close look at the update (2.35), we can find that the communication is still only between neighboring nodes. The difference between (2.34) and (2.31) is that, instead of updating variables sequentially node-by-node, (2.35) updates variables color-by-color, with the updates inside each color being completely parallel. As a result, if we can find a coloring scheme that contains only a few colors (compared to the number of nodes), the computational speed can be significantly improved.

In this section, two versions of distributed ADMM algorithms are presented. We finish this section by mentioning that there are other versions of distributed ADMM algorithms, and encourage readers to make the exploration. For example, [48] applies a simple distributed ADMM for optimization in modern communication networks, [28] proposed an online version of distributed ADMM for network applications.

2.4 Solving the ADMM Updates

An iteration of the ADMM consists of several updates, each of which is a subproblem that solves a small optimization problem. In practice, how efficiently the subproblems are solved is crucial to the performance of ADMM. Until now, we have put aside this very important part of the ADMM, i.e., how to solve the ADMM updates. Intentionally or unintentionally,

we have treated the updates of ADMM as a black box, with the belief that these subproblems are guaranteed to be solved accurately and efficiently. In many real applications, however, the ADMM updates can be highly nontrivial and expensive to solve. This is typical when no closed-form solutions are available for the subproblems. Meanwhile, the performance of the ADMM heavily depends on the accuracy and efficiency of the solutions of the subproblems. Especially in the distributed case, e.g., (2.14) and (2.16), where each computing node in the network may only have very limited computing power, efficient algorithms for the ADMM subproblems becomes crucial.

When closed-form solutions do not exist, one usually resort to iterative numerical methods for the ADMM updates. Numerical methods, including the Newton's method (see Section 3.3 of [54]) and the coordinate descent (CD) (see Section 9.3 of [54]), can be applied to a wide range of optimization problems, but may entail significant computational cost. In this section, we present an alternative approach for efficiently solving the ADMM updates. Following the "Unwrapping ADMM" proposed in [24], we show that by writing problems into specific ADMM forms, the solutions of the resulted updates can be obtained without numerical methods. To be concrete, we use the model fitting problem (2.12) for illustration throughout this section. Similar ideas may be applied to a broader range of problems.

2.4.1 Solving the Updates with Iterative Methods

Naively, problem (2.12) can be formulated into the ADMM form

$$\min_{\boldsymbol{\beta},\boldsymbol{z}} \{l(X\boldsymbol{\beta}-\boldsymbol{y})+r(\boldsymbol{z})\} \quad \text{s.t. } \boldsymbol{\beta}=\boldsymbol{z}. \tag{2.36}$$

Applying (2.2) to (2.36) gives us the following updates,

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} l(X\boldsymbol{\beta}-\boldsymbol{y}) + (\rho/2)\|\boldsymbol{\beta}-\boldsymbol{z}^k+\boldsymbol{u}^k/\rho\|_2^2, \\
\boldsymbol{z}^{k+1} &:= \arg\min_{\boldsymbol{z}} r(\boldsymbol{z}) + (\rho/2)\|\boldsymbol{z}-\boldsymbol{\beta}^{k+1}-\boldsymbol{u}^k/\rho\|_2^2, \\
\boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \boldsymbol{\beta}^{k+1} - \boldsymbol{z}^{k+1}.
\end{aligned} \tag{2.37}$$

In (2.37), the solution of $\boldsymbol{z}$-update is usually presented as the *proximal* mapping,

$$\boldsymbol{\beta}^{k+1} := prox_r(\boldsymbol{\beta}^{k+1} - \boldsymbol{u}^k/\rho, \rho),$$

where the proximal mapping for a function f is defined as

$$prox_f(\boldsymbol{v}, \rho) := \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + (\rho/2)\|\boldsymbol{x}-\boldsymbol{v}\|_2^2.$$

The proxy can usually be efficiently evaluated with explicit solutions for a wide range of penalty functions $r(\cdot)$. On the other hand, in the $\boldsymbol{\beta}$-update the coordinates of $\boldsymbol{\beta}$ are coupled and hence closed-form solutions are not available except when the loss function $l(\cdot)$ is quadratic. Depending on the form of the loss function, either Newton's method or the CD can be applied to solve the β-update. This results in a double-loop algorithm that could be time consuming.

2.4.2 Reformulating the Problem to Avoid Iterative Methods

The Unwrapping ADMM [24] suggests that, if a problem can be written into the following form,

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) \quad \text{s.t. } \boldsymbol{y} = D\boldsymbol{x}, \tag{2.38}$$

with $f(\cdot)$ being a decomposible function, i.e., $f(\boldsymbol{y}) = \sum_{i=1}^{s} f_i(y_i)$ for $\boldsymbol{y} \in R^s$, then it can be efficiently solved by the ADMM. To be specific, the ADMM solution for (2.38) is

$$\begin{aligned} \boldsymbol{x}^{k+1} &:= \arg\min_{\boldsymbol{x}} \|D\boldsymbol{x} - \boldsymbol{y}^k + \boldsymbol{u}^k\|_2^2 = (D^T D)^{-1} D^T (\boldsymbol{y}^k - \boldsymbol{u}^k), \\ \boldsymbol{y}^{k+1} &:= \arg\min_{\boldsymbol{y}} f(\boldsymbol{y}) + (\rho/2)\|\boldsymbol{y} - D\boldsymbol{x}^{k+1} - \boldsymbol{u}^k/\rho\|_2^2, \\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + D\boldsymbol{x}^{k+1} - \boldsymbol{y}^{k+1}, \end{aligned} \tag{2.39}$$

and when $f(\boldsymbol{y})$ is decomposable, the $\boldsymbol{y}$-update is coordinate-wise decoupled and simple solutions are readily available.

As an application of the unwrapping ADMM, problem (2.12) can be reformulated into the following "Unwrapped form",

$$\min_{\boldsymbol{\beta}, \boldsymbol{z}} g(\boldsymbol{z}) \quad \text{s.t. } D\boldsymbol{\beta} - \boldsymbol{z} = (0, \ldots, 0, \boldsymbol{y}^T)^T \in \mathbb{R}^{p+n}, \tag{2.40}$$

where $D = \begin{pmatrix} I_p \\ X \end{pmatrix}$ and $g(\boldsymbol{z}) = \sum_{i=1}^{p+n} g_i(z_i)$ with

$$g_i(z_i) = \begin{cases} r(z_i) & \text{for } i \le p, \\ l(z_i) & \text{for } i > p. \end{cases}$$

Then (2.39) results in

$$\begin{aligned} \boldsymbol{\beta}^{k+1} &:= (D^T D)^{-1} D^T \left\{ \boldsymbol{z}^k + (0, \ldots, 0, \boldsymbol{y}^T)^T - \boldsymbol{u}^k/\rho \right\}, \\ z_i^{k+1} &:= \arg\min_{z_i} r(z_i) + (\rho/2)(z_i - \boldsymbol{d_i} \cdot \boldsymbol{\beta}^{k+1} - u_i)^2, \ i = 1, 2, \ldots, p, \\ z_j^{k+1} &:= \arg\min_{z_j} l(z_j) + (\rho/2)(z_j - \boldsymbol{d_j} \cdot \boldsymbol{\beta}^{k+1} - u_j)^2, \ j = p+1, , \ldots, p+n, \\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + D\boldsymbol{\beta}^{k+1} - \boldsymbol{z}^{k+1} - (0, \ldots, 0, \boldsymbol{y}^T)^T, \end{aligned} \tag{2.41}$$

where $\boldsymbol{d_j}$ denotes the j-th row of matrix D.

Compared to the (2.37), the updates in (2.41) are decomposed into coordinate-wise optimizations, where the solutions are easily achievable without iterative methods. Also, the Unwrapping ADMM facilitates a natural parallel implementation, where the $\boldsymbol{z}$- and $\boldsymbol{u}$-updates can be completely parallelized, and the matrix evaluation $D^T D = I_p + \sum_{i=1}^{N} X_i^T X_i$ only need to be computed separately across the network and aggregated once before the iteration starts (the data are split by rows with X_i being the i-th subsample).

The Unwrapping ADMM enjoys easy implementation and low computational cost, but may be restrictive because not all problems can be written into the unwrapped form (2.38). Furthermore, as commented by [61], a more difficult subproblem of ADMM at each iteration helps make more progress towards the global minimum and may hence converge faster. Compared to (2.37) where time consuming numerical methods are applied to solve the subproblems, the Unwrapping ADMM may result in a slower convergence in practice.

2.5 Conclusion

In this chapter, we presented the ADMM algorithm and its applications to large-scale statistical optimizations. Compared to traditional convex optimization algorithms, the ADMM enjoys two major advantages. First, it is a general purpose optimization tool that are found useful for a broad scope of research fields and applications, with comparable and often better performances to domain-specific algorithms. Second, under general context, the updates of ADMM often lend themselves to parallel implementations, which facilitates the parallel implementations of the ADMM and grant it the strength to solve large-scale problems. Throughout this chapter, we have seen that the flexible formulation of the ADMM enables it to represent a variety of problems. This flexibility, combined with its easy parallelization, places the ADMM among the most powerful tools for large-scale optimization. We did not dive deep into the detailed implementation of the ADMM in this chapter. Here we only

point out that the ADMM computation naturally falls into the MapReduce computation paradigm [17] and can be implemented in modern distributed computing frameworks like the Hadoop [78] and Spark [88]. We refer the readers to Chapter 10 of [6] for a detailed discussion of implementing ADMM under distributed computing environments.

Chapter 3

ADMM for Penalized Quantile Regression in Big Data

Traditional linear programming algorithms for quantile regression, e.g., the simplex method and the interior point method, work well for data of small to moderate sizes. However, these methods are difficult to generalize to high-dimensional big data for which penalization is usually necessary. Further, the massive size of contemporary big data calls for the development of large-scale algorithms on distributed computing platforms. The traditional linear programming algorithms are intrinsically sequential and not suitable for such frameworks. In this chapter, we discuss how to use the popular ADMM algorithm to solve large-scale penalized quantile regression problems. The ADMM algorithm can be easily parallelized and implemented in modern distributed frameworks. Simulation results demonstrate that the ADMM is as accurate as traditional LP algorithms while faster even in the nonparallel case.

3.1 The QR-ADMM Algorithm

In this section we propose to solve the penalized quantile regression problem (1.3) based on ADMM.

Problem (1.3) can be formulated into the following equivalent ADMM form,

$$\min_{\boldsymbol{\beta},\boldsymbol{r}} \{\rho_\tau(\boldsymbol{r}) + P_\lambda(\boldsymbol{\beta})\} \quad \text{s.t. } \boldsymbol{y} - X\boldsymbol{\beta} = \boldsymbol{r}, \tag{3.1}$$

with Lagrangian

$$L_\gamma(\boldsymbol{r}, \boldsymbol{u}, \boldsymbol{\beta}) = \rho_\tau(\boldsymbol{r}) + \boldsymbol{u}^T(\boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{r}) + \frac{\gamma}{2}||\boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{r}||_2^2 + P_\lambda(\boldsymbol{\beta}).$$

Following (2.2), we have the updating rules for (3.1) as follows,

$$\begin{aligned} \boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}} \tfrac{\gamma}{2} \left\|\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{r}^k\right\|_2^2 + P_\lambda(\boldsymbol{\beta}), \\ \boldsymbol{r}^{k+1} &:= \arg\min_{\boldsymbol{r}} \rho_\tau(\boldsymbol{r}) + \tfrac{\gamma}{2} \left\|\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \boldsymbol{r}\right\|_2^2, \\ \boldsymbol{u}^{k+1} &:= \boldsymbol{u}^k + \gamma(\boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \boldsymbol{r}^{k+1}). \end{aligned} \tag{3.2}$$

The $\boldsymbol{r}$-update in (3.2) has a closed form solution,

$$\boldsymbol{r}^{k+1} := \left[\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \tau\gamma^{-1}\mathbf{1}_n\right]_+ - \left[-\gamma^{-1}\boldsymbol{u}^k - \boldsymbol{y} + X\boldsymbol{\beta}^{k+1} + (\tau-1)\gamma^{-1}\mathbf{1}_n\right]_+. \tag{3.3}$$

The computational difficulty mainly lies in the $\boldsymbol{\beta}$-update, especially when we use non-convex penalties. The $\boldsymbol{\beta}$-update can be viewed as a penalized least square problem with pseudo response $\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - \boldsymbol{r}^k$ and penalty P_λ. In the following, we derive in details how to solve the $\boldsymbol{\beta}$-update in (3.2) with the Lasso (1.4), elastic net (1.5), MCP (1.6) and SCAD (1.7) penalties. Methods for solving the $\boldsymbol{\beta}$-update with other penalties may be similarly derived.

$\boldsymbol{\beta}$-update for the Lasso Penalty

When $P_\lambda(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$, the $\boldsymbol{\beta}$-update in (3.2) solves the following Lasso penalized least square problem with response $\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - \boldsymbol{r}^k$ at the kth iteration of the ADMM,

$$\boldsymbol{\beta}^{k+1} := \arg\min_{\boldsymbol{\beta}} \left\| \left(\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - \boldsymbol{r}^k\right) - X\boldsymbol{\beta} \right\|_2^2 + 2\lambda\gamma^{-1}\|\boldsymbol{\beta}\|_1. \tag{3.4}$$

The problem (3.4) can be iteratively solved by the coordinate descent method (CD). The idea is to alternatively optimize the object function in β_j , $j = 1, 2, \ldots, p$, by treating all other β_i's, $i \neq j$, as fixed. Denote the $\boldsymbol{\beta}$ value at the kth iteration of the ADMM and after the tth inner CD iteration as $\boldsymbol{\beta}^{k(t)}$. Set

$$\boldsymbol{s_j}^{k(t)} = \left(\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - \boldsymbol{r}^k\right) - \left(\sum_{i<j} \beta_i^{k(t+1)} \boldsymbol{x_i} + \sum_{i>j} \beta_i^{k(t)} \boldsymbol{x_i} \right), \ j = 1, \ldots, p, \tag{3.5}$$

then

$$\beta_j^{k(t)} := \arg\min_{\beta_j} \|\boldsymbol{s_j}^{k(t)} - \boldsymbol{x_j}\beta_j\|_2^2 + 2\lambda\gamma^{-1}|\beta_j|. \tag{3.6}$$

It follows that

$$\beta_j^{k(t)} := \frac{\text{sign}(\boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)})(|\boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)}| - \lambda\gamma^{-1})_+}{\|\boldsymbol{x_j}\|_2^2}, \tag{3.7}$$

where $\bar{\boldsymbol{s_0}}^{k(t)}$ denotes the elementwise mean of the vector $\boldsymbol{s_0}^{k(t)}$. The inner CD iterations terminate until two consecutive $\boldsymbol{\beta}^{k(t+1)}$ and $\boldsymbol{\beta}^{k(t)}$ are sufficiently close.

An alternative approach to CD is to use approximation. For example, (3.4) can be approximated by

$$\boldsymbol{\beta^{k+1}} := \arg\min_{\boldsymbol{\beta}} \gamma \left\{ \left[X^T \left(X\boldsymbol{\beta}^k - \gamma^{-1}\boldsymbol{u}^k - \boldsymbol{y} + \boldsymbol{r}^{k+1} \right) \right]^T (\boldsymbol{\beta} - \boldsymbol{\beta}^k) + b\|\beta - \beta^k\|_2^2 \right\} + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{3.8}$$

where the part of the objective function in the brace serves as a majorization to the quadratic

function in (3.4) when constant $b > 0$ is large enough. Let $\boldsymbol{v}^k = \boldsymbol{\beta}^k - bX^T\left(X\boldsymbol{\beta}^k - \gamma^{-1}\boldsymbol{u}^k - \boldsymbol{y} + \boldsymbol{r}^{k+1}\right)$, problem (3.8) then has the following closed-form solution,

$$\boldsymbol{\beta}^{k+1} = \left[\boldsymbol{v}^k - \lambda b\gamma^{-1}\mathbf{1}_{\boldsymbol{p}}\right]_+ - \left[-\boldsymbol{v}^k - \lambda b\gamma^{-1}\mathbf{1}_{\boldsymbol{p}}\right]_+. \tag{3.9}$$

This avoids iterations to solve the $\boldsymbol{\beta}$-update but may result in more iterations for the ADMM to converge.

$\boldsymbol{\beta}$-update for the Elastic Net Penalty

When $P_\lambda(\boldsymbol{\beta}) = \lambda(\lambda_2\|\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1)$, the $\boldsymbol{\beta}$-update in (3.2) is,

$$\boldsymbol{\beta}^{k+1} := \arg\min_{\boldsymbol{\beta}} \left\|\left(\gamma^{-1}\boldsymbol{u}^k + \boldsymbol{y} - \boldsymbol{r}^k\right) - X\boldsymbol{\beta}\right\|_2^2 + 2\lambda\gamma^{-1}(\lambda_2\|\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1) \tag{3.10}$$

Similar to (3.7), the tth CD update of (3.10) is

$$\beta_j^{(t)} := \frac{\text{sign}(\boldsymbol{x_j}^T\boldsymbol{s_j}^{k(t)})(|\boldsymbol{x_j}^T\boldsymbol{s_j}^{k(t)}| - \lambda\lambda_1\gamma^{-1})_+}{\|\boldsymbol{x_j}\|_2^2 + 2\lambda\lambda_2\gamma^{-1}}. \tag{3.11}$$

$\boldsymbol{\beta}$-update for the MCP or SCAD Penalty

When the non-convex MCP or the SCAD penalty is applied, the $\boldsymbol{\beta}$-update in (3.2) becomes non-convex. For non-convex problems, the CD may get stuck in a local minima. Fortunately, for the $\boldsymbol{\beta}$-update, CD is still guaranteed to converge to a global minimum, see Theorem 4 in [50]. In the following, we derive the CD iterations for the $\boldsymbol{\beta}$-update with MCP or SCAD penalty.

Similar to (3.6), at the t-th update of the CD, we update $\beta_j,\ j = 1, \ldots, p$, by

$$\beta_j^{(t)} := \arg\min_{\beta_j} \ell_j(\beta_j) = \|\boldsymbol{s_j}^{k(t)} - \boldsymbol{x_j}\beta_j\|_2^2 + 2\gamma^{-1}P_\lambda(\beta_j), \tag{3.12}$$

where $P_\lambda(\cdot)$ is the MCP or SCAD penalty. The function $\ell_j(\beta_j)$ is a continuous and piecewise quadratic function in β_j and approaches $+\infty$ when $\beta_j \to \pm\infty$, for either the SCAD or the MCP penalty. So the global minimum for (3.12) exists and is among the critical points of ℓ_j.

For the MCP penalty,

$$\ell_j(\beta_j) \propto \begin{cases} \|\boldsymbol{x_j}\|_2^2 \cdot \beta_j^2 - 2(\boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)})\beta_j, & \text{if } |\beta_j| \geq a\lambda, \\ (\gamma\|\boldsymbol{x_j}\|_2^2 - a^{-1})\,\beta_j^2 - 2\left[\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} + \lambda\right]\beta_j, & \text{if } \beta_j \in (-a\lambda, 0], \\ (\gamma\|\boldsymbol{x_j}\|_2^2 - a^{-1})\,\beta_j^2 - 2\left[\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} - \lambda\right]\beta_j, & \text{if } \beta_j \in (0, a\lambda). \end{cases}$$

The solution to (3.12) is the one that gives the minimum ℓ_j value among the following candidate critical points,

$$\beta_j = \left\{0, \frac{\boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)}}{\|\boldsymbol{x_j}\|_2^2}, \frac{\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} \pm \lambda}{\gamma\|\boldsymbol{x_j}\|_2^2 - a^{-1}}\right\}. \tag{3.13}$$

The last three points in (3.13) are only counted as critical points when they are in $(-\infty, -a\lambda) \cup (a\lambda, +\infty)$, $(-a\lambda, 0)$ and $(0, a\lambda)$, respectively.

Similarly, for the SCAD penalty, we have

$$\ell_j(\beta_j) \propto \begin{cases} \|\boldsymbol{x_j}\|_2^2 \beta_j^2 - 2\boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)}\beta_j, & \text{if } |\beta_j| \geq a\lambda, \\ \left(\gamma\|\boldsymbol{x_j}\|_2^2 - \frac{1}{a-1}\right)\beta_j^2 - 2\left(\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} + \frac{a\lambda}{a-1}\right)\beta_j & \text{if } \beta_j \in (-a\lambda, -\lambda], \\ \left(\gamma\|\boldsymbol{x_j}\|_2^2 - \frac{1}{a-1}\right)\beta_j^2 - 2\left(\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} - \frac{a\lambda}{a-1}\right)\beta_j, & \text{if } \beta_j \in (-a\lambda, -\lambda], \\ \gamma\|\boldsymbol{x_j}\|_2^2 \beta_j^2 - 2\left[\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} + \lambda\right]\beta_j, & \text{if } \beta_j \in (-\lambda, 0], \\ \gamma\|\boldsymbol{x_j}\|_2^2 \beta_j^2 - 2\left[\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} - \lambda\right]\beta_j, & \text{if } \beta_j \in (0, \lambda]. \end{cases}$$

And the solution to (3.12) is then among the critical points,

$$\beta_j = \left\{ 0, \frac{\boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)}}{\|\boldsymbol{x_j}\|_2^2}, \frac{\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} \pm \frac{a\lambda}{a-1}}{\gamma \|\boldsymbol{x_j}\|_2^2 - \frac{1}{a-1}}, \frac{\gamma \boldsymbol{x_j}^T \boldsymbol{s_j}^{k(t)} \pm \lambda}{\gamma \|\boldsymbol{x_j}\|_2^2} \right\}. \tag{3.14}$$

And the last five points in (3.14) are only counted as critical points when they are in $(-\infty, -a\lambda) \cup (a\lambda, +\infty)$, $(-a\lambda, -\lambda)$, $(\lambda, a\lambda)$, $(-\lambda, 0)$ and $(0, \lambda)$, respectively.

Compared to LP algorithms, the QR-ADMM is significantly faster especially when the dimension p is large. The time complexity of IP for unpenalized quantile regression was derived in [58]. Each iteration of the IP has complexity $\mathcal{O}(np^2)$ and the final purification step requires $\mathcal{O}(np^3)$ operations. On the other hand, in each iteration of the ADMM update for the unpenalized quantile regression, the $\boldsymbol{\beta}$-update, which takes $\mathcal{O}(np^2 + p^3)$ for the first iteration and $\mathcal{O}(np^2)$ for each of later iterations, dominates the computation time in (3.2).Considering that the fact that ADMM can have slow convergence sometimes, the QR-ADMM seems not to necessarily dominate the IP in terms of time. But as we can see in the simulation part, the QR-ADMM does converge fast enough for the penalized quantile regression problems we consider.

The QR-ADMM is very similar to the QICD algorithm in [57]. The QICD is also a double-loop algorithm for solving the MCP and SCAD penalized quantile regression. The outer loop of QICD uses Majorization-Minimization (MM) that approximates the objective function (1.3) by replacing the non-convex penalty with a linear approximation. The inner loop uses CD to solve (1.3) with the linearized penalty. The QICD has shown to perform well in terms of both computation speed and accuracy for solving penalized quantile regression with SCAD or MCP penalty, but cannot be easily parallelized. A comparison of the QR-ADMM and the QICD is given in Section 3.3.

3.1.1 Convergence of the QR-ADMM

The convergence of ADMM for convex problems has been well established in the literature. The penalized quantile regression with convex penalties, e.g., the Lasso or elastic net, belongs to this case.

Due to the lack of convexity, the convergence of ADMM for non-convex problems requires strong assumptions on the objective functions. To the best of our knowledge, all current work in the non-convex ADMM literature requires f or g to have a Lipschitz continuous first derivative in order to guarantee the convergence. Without such assumptions, the Fejér monotonicity of the sequences generated by ADMM cannot be established and as a consequence the convergence of ADMM remains unknown, see [26, 43, 93] for technical details. For MCP or SCAD penalized quantile regression, neither the loss function $\rho_\tau(\cdot)$ nor the penalty has a Lipschitz continuous first derivatives. Consequently, convergence is unrealistic to establish without further assumptions.

In the following, we first establish the convergence of the QR-ADMM for convex penalties. The proof follows standard arguments in the ADMM literature and is provided in the Appendix. Then we explore the convergence of the QR-ADMM for non-convex penalties. Specifically, we provide an assumption under which the QR-ADMM converges to a stationary solution of the PQR with non-convex penalties.

First, we state the following assumptions.

Assumption 3.1.1 *The unaugmented Lagrangian* $L_0 = \rho_\tau(\boldsymbol{r}) + \boldsymbol{u}^T(\boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{r}) + P_\lambda(\boldsymbol{\beta})$ *has a saddle point* $(\boldsymbol{r}^*, \boldsymbol{\beta}^*, \boldsymbol{u}^*)$.

Assumption 3.1.2 *The* $\boldsymbol{r}^k$*'s generated by (3.2) are bounded.*

Assumption 3.1.3 *The matrix* X *has full column rank.*

Assumption 3.1.4 *The $\boldsymbol{r}^k$'s generated by (3.2) satisfies the following condition: for any $i \in \{1, 2, \ldots, n\}$, if the signs of r_i^k and r_i^{k+1} are different, then $|r_i^k - r_i^{k+1}| > 2\delta$ for some constant $\delta > 0$.*

We have the following theorems,

Theorem 3.1.1 *For convex penalties, e.g., the (group) Lasso and elastic net, when Assumption 3.1.1 holds, $\rho_\tau(\boldsymbol{r}^k) + P_\lambda(\boldsymbol{\beta}^{\boldsymbol{k}})$ converges to the minimum of problem (3.1). If Assumption 3.1.3 also holds, then $(\boldsymbol{r}^k, \boldsymbol{\beta}^k)$ converges to a solution of (3.1).*

Proof The proof of this theorem is in the Appendix. ■

Theorem 3.1.2 *For non-convex penalties, e.g., the MCP and SCAD, when Assumption 3.1.4 holds and $\gamma > 1/(\sqrt{2}\delta)$, then any cluster point (a point is a cluster point of a sequence if there is a subsequence that converges to this point) of $(\boldsymbol{r}^k, \boldsymbol{\beta}^k, \boldsymbol{u}^k)$ is a stationary point of (3.1). If further Assumptions 3.1.2 and 3.1.3 hold, then $(\boldsymbol{r}^k, \boldsymbol{\beta}^k, \boldsymbol{u}^k)$ converges to a stationary point of (3.1).*

Proof The proof of this theorem is in the Appendix. ■

Remark Assumptions 3.1.1 and 3.1.3 are standard assumptions for convergence of the ADMM. Assumption 3.1.2 is by no means restrictive since in practice, we restrict ourselves to finding a finite solution $\boldsymbol{\beta}$, and hence $\boldsymbol{r}$. ■

Remark Assumption 3.1.4 is a technical assumption needed for the proof and is subject to further relaxation. We emphasize that without Lipschitz continuity condition on the objective functions, such assumptions on the sequences generated by ADMM are necessary to establish the convergence. For example, the convergence analyses of non-convex ADMM in [34, 65, 81] all make uncheckable assumptions on the sequences generated by the ADMM. Although Assumption 3.1.4 may need further theoretical backup, it can be justified empirically as we observe that the signs of r_i^k's all remain unchanged after several iterations for all

the simulations we consider in this chapter. As a matter of fact, although the convergence of ADMM for non-convex problems remains largely open, it is observed to work well for a wide range of non-convex applications, see [26] and the references therein. ■

3.1.2 Extension to Multivariate Quantile Regression

The multivariate quantile regression requires a definition of quantiles for multivariate distributions. There are controversies on the definition of quantiles for multivariate distributions. However, this topic is beyond the scope of this paper. Our goal here is not to address these controversies, but to demonstrate the potential of using QR-ADMM to solve multivariate quantile regression problems. Specifically, we show that the QR-ADMM can be used to parallelize the computation for a multivariate quantile regression procedure. In the following, we use the quantile contour definition and the resulted multivariate quantile regression procedure in [77] to illustrate our point.

Suppose that a m-dimensional random variable $\boldsymbol{Y}$ has a c.d.f. $F(\cdot)$. The vector $\boldsymbol{Y}$ is centralized such that the marginal sample median on all components are at the origin. [77] defined a central τ-interval as $[F_{\boldsymbol{u}}(\frac{1-\tau}{2}), F_{\boldsymbol{u}}(\frac{1+\tau}{2})]$, where $F_{\boldsymbol{u}}$ is the conditional c.d.f. of $\boldsymbol{Y}$ in direction $\boldsymbol{u}$. The reference quantile contour of $\boldsymbol{Y}$ is defined as the surface

$$\left\{ \boldsymbol{x} \in \bigcup_{\boldsymbol{u} \in \mathbb{S}^{m-1}} \left[F_{\boldsymbol{u}}(\frac{1-\tau}{2}), F_{\boldsymbol{u}}(\frac{1+\tau}{2}) \right] \right\},$$

which can be parameterized by a function $g_\tau(\boldsymbol{\gamma})$ in the polar system. The sample reference quantile contour g_τ can then be estimated by minimizing

$$\sum_{k=1}^{2} \sum_{i=1}^{n} \rho_{\frac{1+\tau}{2}} \left(r_i^k - g_\tau(\boldsymbol{\gamma_i}^k) \right), \tag{3.15}$$

over a family of smooth function g_τ, where $(r_i^1, \boldsymbol{\gamma_i}^1)$ and $(-r_i^2, \boldsymbol{\gamma_i}^2)$ are the coordinates of $\boldsymbol{Y_i}$

and $-\boldsymbol{Y_i}$ respectively in the p-dimensional polar system.

Having this definition of quantile contour, the multivariate quantile regression problem is formulated in [77] as a two-step procedure. Denoting the response matrix as $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) \in \mathbb{R}^{n \times m}$ and the design matrix as X, the first step is to solve the following conditional quantile estimation problems for $\tau \in (0, 1)$,

$$Q_\tau(\boldsymbol{y}_i | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-1}, X) = \sum_{k=1}^{i-1} \boldsymbol{y}'_{k,i} \boldsymbol{\alpha}_{k,i}(\tau) + X\boldsymbol{\beta}_i(\tau), \ i = 1, \ldots, m. \tag{3.16}$$

In (3.16), the conditional quantile functions are in linear form. This is exact when (Y, X) are jointly Gaussian. For non-Gaussian distributions, e.g., the general elliptical distributions, the linear form (3.16) is considered as an approximation. The coefficients $\boldsymbol{\beta}_i(\tau)$'s and $\boldsymbol{\alpha}_{k,i}$'s are estimated for several quantile levels and are then extended to $\tau \in (0, 1)$ by splines. This gives the conditional distributions of $Y_i | Y_1, \ldots, Y_{i-1}, X$. In the second step, to get the quantile contour of Y at $\tilde{X}$, N samples are drawn from the conditional distribution $Y|\tilde{X}$ as follows: First, generate $U_i \overset{i.i.d.}{\sim} U(0, 1)$ for $i = 1, \ldots, m$. Then sequentially for $i = 1, 2, \ldots, m$, draw sample y_i as the U_i-th quantile of the conditional distribution $Y_i | Y_1, \ldots, Y_{i-1}, \tilde{X}$. Then $(y_i, \ldots, y_m)$ form a random sample of Y. After taking N samples, (3.15) is used to estimate the quantile contour of $Y|\tilde{X}$.

The QR-ADMM algorithm can be applied to each step of the procedure. In the first step, the function $g_\tau(\boldsymbol{\gamma})$ for $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_{m-1})^T$ can be estimated by the additive model,

$$g_\tau(\boldsymbol{\gamma}) = \sum_{j=1}^{m-1} f_j(\gamma_j) \text{ with } f_j = \sum_{l=1}^{q} a_{jl} h_{jl}(\gamma_j), \tag{3.17}$$

where h_{jl}'s are the base functions. After some algebraic manipulation, problem (3.15) is

formulated as follows,

$$\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m-1} = \underset{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_{m-1}}{\arg\min} \rho_{\frac{1+\tau}{2}} \left(\boldsymbol{r} - \sum_{j=1}^{m-1} H_j \boldsymbol{a}_j \right), \tag{3.18}$$

where $\boldsymbol{r} = (r_1, \ldots, r_n, -r_1, \ldots, -r_n) \in \mathbb{R}^{2n}$, $H_j \in \mathbb{R}^{2n \times q}$ with $(H_j)_{i,l} = (H_j)_{i+n,l} = h_{il}(\gamma_{i,j})$, and $\boldsymbol{a}_j = (a_{j1}, \ldots, a_{jq})^T \in \mathbb{R}^q$. In high-dimensional settings, a penalty can also be added to (3.18) for feature selection. Problem (3.18) can be solved by combining the block descent and QR-ADMM. First, $\boldsymbol{a}_j$'s are sequentially updated by treating all other $\boldsymbol{a}_h$'s ($h \neq j$) as fixed. Then the minimization problem (3.18) for $\boldsymbol{a}_j$ is in the standard quantile regression form (1.1) or (1.3) if penalization is applied, and can hence be solved by the QR-ADMM. The second step assumes the standard quantile regression model assumption (3.16), so the parameters can also be estimated by the QR-ADMM.

In terms of scalability of QR-ADMM, the first step involves solving $m - 1$ quantile regression with size $2n \times q$ alternatively. For multivariate quantile regression applications, the dimensionality of the response m is typically a small number and may not raise scalability concerns. For large n and/or q, the QR-ADMM can be parallelized in both n and q directions and hence scales well, as discussed in Section 3.2. For the second step, QR-ADMM solves m quantile regression problems with size no larger than $n \times (p + m)$. Again, when n or p is large, the QR-ADMM can be parallelized to scale to the data size.

3.2 Parallelization of QR-ADMM

The main advantages of the ADMM is its capability of parallelized implementation in modern distributed computing frameworks. When the data are too large for a single computer to store or process, computing frameworks that store and analyze the data distributedly become necessary. In this section, we demonstrate that the ADMM can be easily carried out in a

distributed way for (penalized) quantile regression. The resulted parallel algorithms can be efficiently implemented in large-scale computation frameworks like the Hadoop [78] and Spark [88].

3.2.1 Parallelization of QR-ADMM

By splitting the data (y, X) into different blocks and apply the ADMM, we can achieve a parallelized version of the QR-ADMM algorithm. In the following, we present three ways to parallelize the QR-ADMM.

Splitting along n

First, assume that the data are split into M blocks as follows, each block being a subset of the whole data,

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \\ \vdots \\ \boldsymbol{y_M} \end{pmatrix}, \text{ and correspondingly, } X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{bmatrix}.$$

The problem (1.3) is then written as

$$\min_{\boldsymbol{r_b}, \boldsymbol{\beta}} \left\{ \sum_{b=1}^{M} \rho_\tau(\boldsymbol{r_b}) + P_\lambda(\boldsymbol{\beta}) \right\} \quad \text{s.t. } \boldsymbol{y_b} - X_b \boldsymbol{\beta_b} = \boldsymbol{r_b}, \boldsymbol{\beta_b} = \boldsymbol{\beta},\ b = 1, 2, \ldots, M. \tag{3.19}$$

This results in the following updating,

$$
\begin{aligned}
\boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}} \frac{M\gamma}{2} \left\|\boldsymbol{\beta} - \bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k\right\|_2^2 + P_\lambda(\boldsymbol{\beta}), \\
\boldsymbol{r_b}^{k+1} &:= \arg\min_{\boldsymbol{r_b}} \rho_\tau(\boldsymbol{r_b}) + \frac{\gamma}{2}||\boldsymbol{y_b} - \boldsymbol{X_b}\boldsymbol{\beta_b}^{k+1} + \boldsymbol{u_b}^k - \boldsymbol{r_b}||_2^2, \\
\boldsymbol{\beta_b}^{k+1} &:= (X_b^T X_b + I)^{-1} \left(X_b^T(\boldsymbol{y_b} - \boldsymbol{r_b}^{k+1} + \boldsymbol{u_b^k}) - \boldsymbol{\eta_b^k} + \boldsymbol{\beta}^{k+1}\right), \\
\boldsymbol{u_b}^{k+1} &:= \boldsymbol{u_b}^k + \boldsymbol{y_b} - X_b\boldsymbol{\beta_b}^{k+1} - \boldsymbol{r_b}^{k+1}, \\
\boldsymbol{\eta_b}^{k+1} &:= \boldsymbol{\eta_b}^k + \boldsymbol{\beta_b}^{k+1} - \boldsymbol{\beta}^{k+1},
\end{aligned}
\tag{3.20}
$$

where $\bar{\boldsymbol{\beta}}^k = M^{-1}\sum_{b=1}^M \boldsymbol{\beta_b}^k$ and $\bar{\boldsymbol{\eta}}^k = M^{-1}\sum_{b=1}^M \boldsymbol{\eta_b}^k$. The last four updates in (3.20) with subscript b depend only on the b-th block of data, so they can be solved distributedly in parallel. But there does exist some communication cost between different blocks of data in each iteration. All $(\boldsymbol{\beta_b}, \boldsymbol{\eta_b})_{b=1}^M$ values need to be aggregated by a center to conduct the $\boldsymbol{\beta}$-update, and the center broadcasts the updated $\boldsymbol{\beta}$ back to each block updating.

Splitting along p

When data are of ultra-high dimensionality, a necessary choice for parallelization is to split along the p direction. First, the data matrix X is partitioned along its columns,

$$X = [X_1,\ X_2,\ \ldots,\ X_N],$$

and conformably,

$$\boldsymbol{\beta} = [\boldsymbol{\beta_1},\ \boldsymbol{\beta_2},\ \ldots,\ \boldsymbol{\beta_N}].$$

Then we get the following optimization problem,

$$\min_{\boldsymbol{z_b}, \boldsymbol{\beta_b}} \left\{ \rho_\tau(\boldsymbol{y} - \sum_{b=1}^N \boldsymbol{z_b}) + \sum_{b=2}^N P_\lambda(\boldsymbol{\beta_b}) + P_\lambda((\boldsymbol{\beta_1})) \right\} \quad \text{s.t. } X_b\boldsymbol{\beta_b} = \boldsymbol{b_i},\ b = 1, 2, \ldots, N, \tag{3.21}$$

with ADMM updates,

$$\begin{aligned}
\bar{\boldsymbol{z}}^{k+1} :=&\ \arg\min_{\bar{\boldsymbol{z}}} \rho_\tau(\boldsymbol{y} - N\bar{\boldsymbol{z}}) + \frac{N\gamma}{2}\|\bar{\boldsymbol{z}} - N^{-1}\sum_{b=1}^{N} X_b{\boldsymbol{\beta_b}}^k - \boldsymbol{u}^k\|_2^2, \\
{\boldsymbol{\beta_b}}^{k+1} :=&\ \arg\min_{\boldsymbol{\beta_b}} P_\lambda(\boldsymbol{\beta_b})\mathbb{I}(b \neq 1) + P_\lambda((\boldsymbol{\beta_1}))\mathbb{I}(b = 1) \\
&\qquad + \frac{\gamma}{2}\|X_b\boldsymbol{\beta_b} - X_b{\boldsymbol{\beta_b}}^k - \bar{\boldsymbol{z}}^{k+1} + N^{-1}\sum_{i=1}^{N} X_b{\boldsymbol{\beta_b}}^k + \boldsymbol{u}^k\|_2^2, \\
\boldsymbol{u}^{k+1} :=&\ \boldsymbol{u}^k + N^{-1}\sum_{b=1}^{N} X_b{\boldsymbol{\beta_b}}^{k+1} - \bar{\boldsymbol{z}}^{k+1}.
\end{aligned} \tag{3.22}$$

In (3.22), each $\boldsymbol{\beta_b}$-update only depends on the b-th subset of the features, and hence can be parallelized.

Splitting along both n and p

More generally, we can patition the data matrix X into $M \times N$ blocks and parallelize the ADMM algorithm in both the n and p directions, as discussed in [56],

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1N} \\ X_{21} & X_{22} & \dots & X_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ X_{M1} & X_{M2} & \dots & X_{MN} \end{bmatrix},$$

where $X_{ij} \in \mathbb{R}^{m_i \times n_j}$. Correspondingly, the $\boldsymbol{y}$, $\boldsymbol{r}$ and $\boldsymbol{\beta}$ vectors in (2.8) can be splitted into M and N subvectors,

$$\boldsymbol{y} = (\boldsymbol{y_1}, \dots, \boldsymbol{y_M}), \quad \boldsymbol{r} = (\boldsymbol{r_1}, \dots, \boldsymbol{r_M}), \quad \boldsymbol{\beta} = (\boldsymbol{\beta_1}, \dots, \boldsymbol{\beta_N}),$$

where $\boldsymbol{r_i} \in \mathbb{R}^{m_i}$, and $\boldsymbol{\beta_j} \in \mathbb{R}^{n_j}$. Problem (1.3) is now reformulated as

$$\begin{aligned} \min_{\boldsymbol{r_i},\boldsymbol{\beta_j}} \quad & \left\{\sum_{i=1}^{M} \rho_\tau(\boldsymbol{r_i}) + \sum_{j=1}^{N} p_\lambda(\boldsymbol{\beta_j})\right\}, \\ \text{s.t.} \quad & \boldsymbol{\beta_j} = \boldsymbol{\beta_{ij}}, \quad i = 1, \ldots, M, \\ & \boldsymbol{y_i} - \boldsymbol{r_i} = \sum_{j=1}^{N} \boldsymbol{r_{ij}}, \quad i = 1, \ldots, M, \\ & \boldsymbol{r_{ij}} = X_{ij}\boldsymbol{\beta_{ij}}, \quad i = 1, \ldots, M,\ j = 1, \ldots, N. \end{aligned} \tag{3.23}$$

Following [56], we have

$$\begin{aligned} \boldsymbol{r_i}^{k+1/2} &:= \arg\min_{\boldsymbol{r}} \ \rho_\tau(\boldsymbol{r}) + \frac{\gamma}{2}\|\boldsymbol{r} - (\boldsymbol{r_i}^k - \tilde{\boldsymbol{r}}_{\boldsymbol{i}}^k)\|_2^2, \\ \boldsymbol{\beta_j}^{k+1/2} &:= \arg\min_{\boldsymbol{\beta}} \ P_\lambda(\boldsymbol{\beta}) + \frac{\gamma}{2}\|\boldsymbol{\beta} - (\boldsymbol{\beta_j}^k - \tilde{\boldsymbol{\beta}}_{\boldsymbol{j}}^{\,k})\|_2^2, \\ (\boldsymbol{\beta_{ij}}^{k+1/2}, \boldsymbol{r_{ij}}^{k+1/2}) &:= \prod_{ij}(\boldsymbol{\beta_j}^k - \tilde{\boldsymbol{\beta}}_{\boldsymbol{ij}}^{\,k}, \boldsymbol{r_{ij}}^k + \tilde{\boldsymbol{r}}_{\boldsymbol{i}}^k), \\ \boldsymbol{\beta_j}^{k+1} &:= \left(\boldsymbol{\beta_j}^{k+1/2} + \sum_{i=1}^{M} \boldsymbol{\beta_{ij}}^{k+1/2}\right)/(M+1), \\ \boldsymbol{r_i}^{k+1} &:= \boldsymbol{r_i}^{k+1/2} - \left(\boldsymbol{r_i}^{k+1/2} - \boldsymbol{y_i} + \sum_{j=1}^{N} \boldsymbol{r_{ij}}^{k+1/2}\right)/(N+1), \\ \boldsymbol{r_{ij}}^{k+1} &:= \boldsymbol{r_{ij}}^{k+1/2} - \left(\boldsymbol{r_i}^{k+1/2} - \boldsymbol{y_i} + \sum_{j=1}^{N} \boldsymbol{r_{ij}}^{k+1/2}\right)/(N+1), \\ \tilde{\boldsymbol{\beta}}_{\boldsymbol{j}}^{\,k+1} &:= \tilde{\boldsymbol{\beta}}_{\boldsymbol{j}}^{\,k} + \boldsymbol{\beta_j}^{k+1/2} - \boldsymbol{\beta_j}^{k+1}, \\ \tilde{\boldsymbol{r}}_{\boldsymbol{i}}^{k+1} &:= \tilde{\boldsymbol{r}}_{\boldsymbol{i}}^k + \boldsymbol{r_i}^{k+1/2} - \boldsymbol{r_i}^{k+1}, \\ \tilde{\boldsymbol{\beta}}_{\boldsymbol{ij}}^{\,k+1} &:= \tilde{\boldsymbol{\beta}}_{\boldsymbol{ij}}^{\,k} + \boldsymbol{\beta_{ij}}^{k+1/2} - \boldsymbol{\beta_{ij}}^{k+1}, \end{aligned} \tag{3.24}$$

where $\prod_{ij}(\boldsymbol{a}, \boldsymbol{b})$ is defined as the projection of $(\boldsymbol{a}, \boldsymbol{b})$ onto the plane $\boldsymbol{b} = X_{ij}\boldsymbol{a}$.

3.3 Simulation Studies

In this section, we conduct simulations to investigate the performance of the QR-ADMM algorithm. Specifically, we show how the QR-ADMM scales to the data size and how the way we divide the data and parallelize the QR-ADMM affects the performance. A comparison is made between the QR-ADMM and the IP, and the QR-ADMM and the QICD. All the simulations are implemented in R on a PC with the Intel Core i7 2.2 GHz CPU and a 8GB RAM.

3.3.1 The QR-ADMM VS the IP

We start with a simple model similar to that of Section 3.10 in [39] to illustrate how the QR-ADMM scales to n and p,

$$Y = X\boldsymbol{\beta}^* + \boldsymbol{\epsilon}, \tag{3.25}$$

where elements of $X \in \mathbb{R}^{n\times p}$ and the error term $\boldsymbol{\epsilon}$ are both i.i.d from the t-distribution with p degrees of freedom, and $\boldsymbol{\beta}^*$ is generated from $\mathcal{N}(\mathbf{1}_p, 2I_p)$. To test the scalability of the QR-ADMM, we consider three steps. First, we fix $p = 5$ and let n increase from 5×10^5 to 5×10^6 with step-size 5×10^5. Then we fix $n = 5,000$ and increase p from 200 to 1,800 with step-size 200. Finally, we let n increase from 10,000 to 100,000 with step-size 10,000 and set $p = \lfloor\sqrt{n}\rfloor$.

Both the IP and the QR-ADMM are applied to estimate $\boldsymbol{\beta}^*$, with $\tau = 0.5,\ 0.7,\ 0.9$. Each simulation is repeated 100 times. We define the ℓ_1 relative error of an estimator $\hat{\boldsymbol{\beta}}$ as

$$\frac{\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_1}{\|\boldsymbol{\beta}^*\|_1},$$

and define the adjusted check loss as

$$\rho_\tau(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}) - \rho_\tau(\boldsymbol{y} - X\boldsymbol{\beta}^{\boldsymbol{*}}).$$

Figures 3.1–3.3 show the comparison for $\tau = 0.9$. Plots for other quantiles show similar results and are hence omitted. We can see in Figure 3.2 that at about the same level of accuracy, the QR-ADMM is significantly faster than the IP, especially when p is large. We notice a significant gap between the check losses in Figure 3.3. But the difference is negligible compared to the magnitude of the check losses.

We also compare the performance on the following heterogeneous model,

$$Y = X_1\boldsymbol{\beta}_1 + (X_2\boldsymbol{\beta}_2) \cdot \boldsymbol{\epsilon}. \tag{3.26}$$

To generate $X = [X_1, X_2]$ in (3.26), we first simulate $\tilde{X} = [\tilde{X}_1 \;\; \tilde{X}_2]$ i.i.d. from $\mathcal{N}(0, 3^2)$, with X_1 consisting of the first $\lfloor 0.9p \rfloor$ columns of X and X_2 consisting of the rest of the columns, and then set $X_1 = \tilde{X}_1$ and $X_2 = \Phi(\tilde{X}_2)$, where Φ is the c.d.f. of the standard normal distribution. Elements of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are all set to 1. And we generate the error $\boldsymbol{\epsilon}$ from $\mathcal{N}(0, \boldsymbol{I}_n)$. So $\boldsymbol{\beta}^* = (0, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \cdot \epsilon_\tau)$, where ϵ_τ is the τ-th sample quantile of $\boldsymbol{\epsilon}$. For this model, the QR-ADMM reaches the same level of accuracy in a slightly shorter time compared to the IP.

As we can see in the simulations, the QR-ADMM is at least comparable to (often better than) the IP in the nonparallel case. This matches the common observation in practice that the ADMM is at least comparable to very specialized algorithms [6].

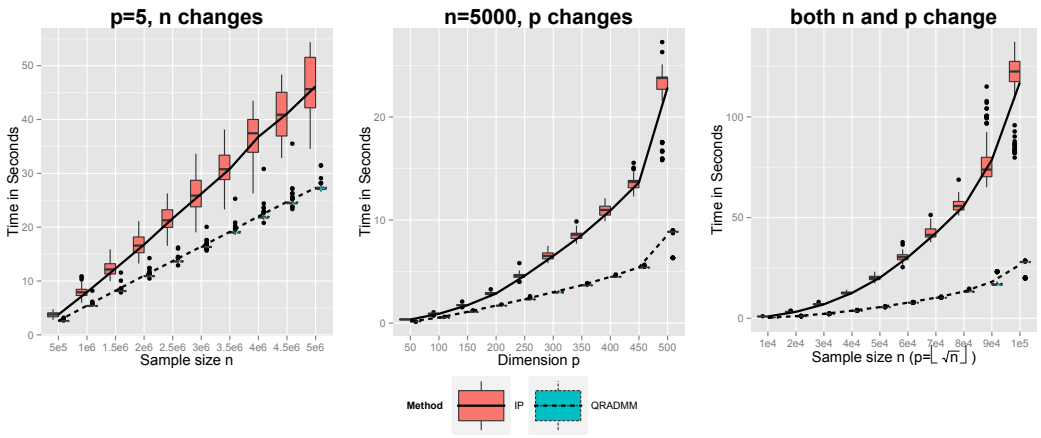


Figure 3.1: Time performance of IP and the QR-ADMM algorithm for (3.25) with $\tau = 0.9$.

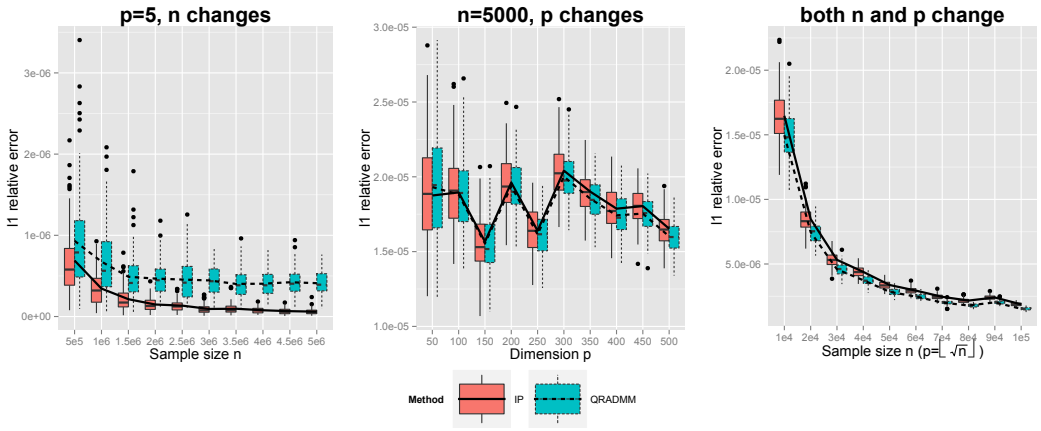


Figure 3.2: A comparison of accuracy between the IP and the QR-ADMM for $\tau = 0.9$.

3.3.2 The QR-ADMM VS the QICD

The QICD [57] is a non-parallelized algorithm for the MCP and SCAD penalized quantile regression that was shown to be both fast and accurate compared to previous approaches, e.g., the LLA [57]. We compare the time, estimation and feature selection accuracy of the QR-ADMM and the QICD in the nonparallel case based on our own implementations. Following [57], we generate our data in the following way. First, we generate $(\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_p)^T \sim \mathcal{N}(0, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is the covariance matrix with elements $\sigma_{ij} = 0.5^{|i-j|}$. Then we set $X_1 = \Phi(\tilde{X}_1)$ and $X_k = \tilde{X}_k$ for $k = 2, 3, \ldots, p$. Then we generate the response according to the

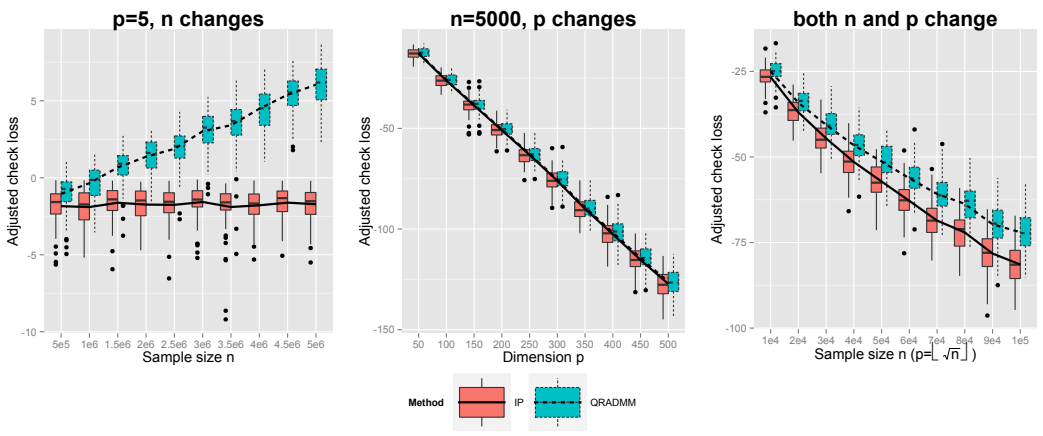


Figure 3.3: A comparison of accuracy between the IP and the QR-ADMM in terms of check loss for $\tau = 0.9$.

following model,

$$Y = X_6 + X_{12} + X_{15} + X_{20} + 0.7X_1\epsilon, \tag{3.27}$$

where ϵ are the $i.i.d$ standard normal errors. For the 0.5 quantile, X_1 has no effect; but for other quantiles, X_1 does have an effect. Here we set $n = 300$ and $p = 1,000$. Then we fit penalized quantile regression with the MCP or SCAD penalty at quantile $\tau = 0.3, 0.5,$ and 0.7. We repeat the simulation for 100 times. The result is summarized in Table 3.1. In Table 3.1, Size is the average number of nonzero elements in the estimation; P1 is the percentage that $X_6, X_{12}, X_{15}, X_{20}$ are selected; P2 is the percentage that X_1 is selected; AE is the ℓ_1 distance between the estimate and the true $\boldsymbol{\beta}$; SD is the standard deviation of AE; Time means the running time of the algorithms.

As we can see from Table 3.1, in the nonparallel case, the accuracy and feature selection performance of the QR-ADMM and the QICD are close, with the QICD being a little faster. But they both are very fast compared to the other computing algorithms for the MCP/SCAD penalized quantile regression, e.g., the LLA, see [57].

Method	Size	P1	P2	AE	SD	Time (Sec)
QR-ADMM-SCAD ($\tau = 0.5$)	8.62	100%	0%	0.063	0.018	1.46
QR-ADMM-SCAD ($\tau = 0.3$)	11.15	100%	80%	0.068	0.018	1.73
QR-ADMM-SCAD ($\tau = 0.7$)	10.97	100%	83%	0.069	0.019	1.70
QICD-SCAD ($\tau = 0.5$)	8.23	100%	0%	0.067	0.018	0.53
QICD-SCAD ($\tau = 0.3$)	11.63	100%	78%	0.070	0.018	0.90
QICD-SCAD ($\tau = 0.7$)	11.19	100%	76%	0.071	0.019	0.96
QR-ADMM-MCP ($\tau = 0.5$)	8.60	100%	0%	0.062	0.017	1.22
QR-ADMM-MCP ($\tau = 0.3$)	10.96	100%	80%	0.072	0.018	1.50
QR-ADMM-MCP ($\tau = 0.7$)	10.97	100%	84%	0.076	0.019	1.50
QICD-MCP ($\tau = 0.5$)	8.37	100%	0%	0.067	0.018	0.63
QICD-MCP ($\tau = 0.3$)	11.54	100%	79%	0.070	0.018	0.86
QICD-MCP ($\tau = 0.7$)	11.23	100%	79%	0.071	0.019	0.93

Table 3.1: Comparison of QR-ADMM and QICD.

3.3.3 Parallelizing the QR-ADMM

In this subsection, we investigate how the way we split the data affects the performance when we parallelize the QR-ADMM. We consider two cases where the data X are split along n and along p separately. We emphasize that the parallel implementation of the QR-ADMM algorithm is actually pseudo-parallel. We implement the parallel versions of the QR-ADMM algorithm in a sequential way by stacking up the supposedly parallel subproblems of the ADMM. As part of the future work, we plan to have a truly parallel implementation of the QR-ADMM algorithm in distributed computing frameworks like the Spark [88].

Splitting along n

We simulate data according to model (3.27) but now with $n = 1,000,000$ and $p = 100$, and uses the Lasso penalty for feature selection. The data are randomly and evenly split into $N = 10, 10^2, 10^3$ and 10^4 blocks separately and we monitor the convergence of the algorithm. The experiment is repeated 100 times and the average convergence is recorded. The result for $\tau = 0.9$ is shown in Figure 3.4. Simulation results of $\tau = 0.5$ and $\tau = 0.7$ are omitted here.

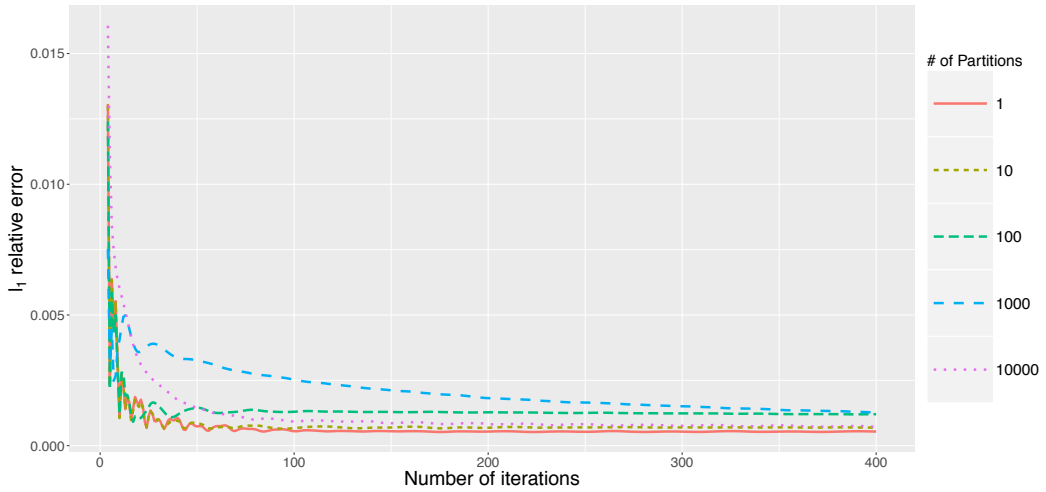


Figure 3.4: Convergence of QR-ADMM for model (3.27) with $n = 1,000,000$, $p = 10$ and $\tau = 0.9$ starting from the 20th iteration. The split is along n.

As we can see in Figures 3.4, for a wide range of partition numbers, the QR-ADMM converges fast with high accuracy in a few tens of iterations. More partitions generally result in slower convergence, but the trend is not always monotonic.

Splitting along p

We still consider model (3.27), with $n = 300$ and $p = 1000$. The SCAD penalty is used for sparsity. The data X is evenly split into $N = 1, 10, 50, 100,$ and 500 blocks along p and we keep track of the convergence. The result for $\tau = 0.9$ is summarized in Figure 3.5.

In Figure 3.5, all other cases converges fast to high accuracy within 100 iterations except for $N = 500$. Again, more partitions generally yields slower convergence. The partition number does not have a great impact on the convergence rate.

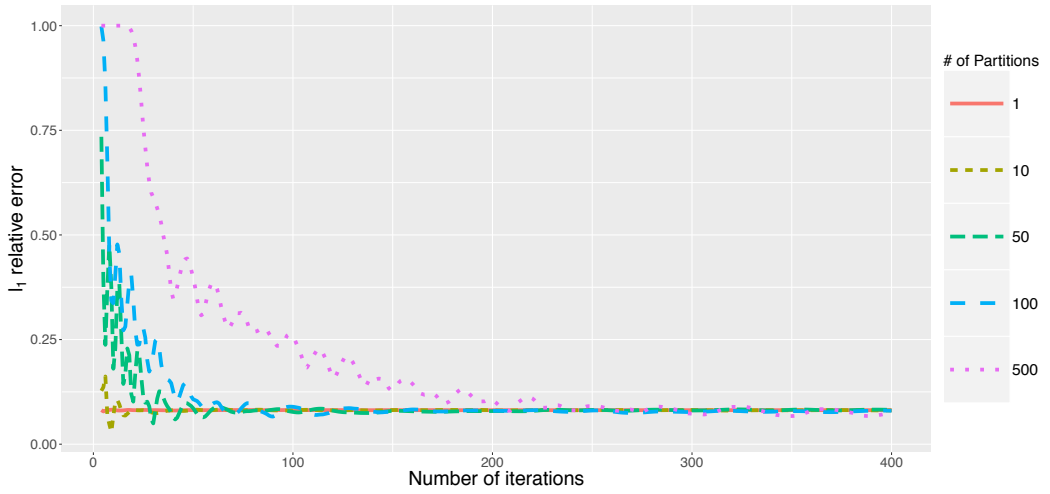


Figure 3.5: Convergence of QR-ADMM for model (3.27) with $n = 300$, $p = 1,000$ and $\tau = 0.9$ starting from the 20th iteration. The splitting is along p.

3.4 Conclusions and Discussions

In this chapter, we discuss using the ADMM to solve large-scale penalized quantile regression problems. We derive the QR-ADMM algorithm for solving penalized quantile regression with different penalties and present possible ways to parallelize the QR-ADMM algorithm. The convergence results are established for all the penalties we consider. The simulation studies show that the QR-ADMM is comparable to the IP and the QICD in terms of accuracy and faster than the IP even in the nonparallel case. The capability of parallelized implementation and the potential to solve large-scale problems distinguishes the QR-ADMM from algorithms like the IP and the QICD.

It is also worth mentioning that, in real implementations of the ADMM, more complicated strategies than (3.20), (3.22), and (3.24) can be involved. Some of these strategies are already discussed in Chapter 2. For example, in (3.20), (3.22), and (3.24), the centralized communication may not be efficient in communication, e.g., when some computing nodes (which process some local blocks of data) are distant from the center. To solve the communication efficiency problem, several decentralized ADMM were proposed, see [53, 76], etc.

Synchronization is also a problem in (3.20), (3.22), and (3.24). The iteration cannot proceed before all blocks finish their updates. So the overall computing speed is limited by the slowest block. The asynchronized ADMM proposed in [90] addresses the problem. In practice, one should always consider using such strategies to improve the implementation efficiency of the ADMM algorithm for large-scale problems.

Chapter 4

A Single-loop Algorithm for Distributed Non-convex Penalized Quantile Regression

Compared to traditional LP methods, the QR-ADMM algorithm we proposed in Chapter 3 can be easily parallelized. And we have shown that, the QR-ADMM is significantly faster than traditional LP solvers, even when parallelization is not applied. The QR-ADMM algorithm is a double-loop algorithm where the outer loop is the ADMM iterations and the inner loop is the coordinate descent for solving the $\boldsymbol{\beta}$-update. Motivated by the parallel implementation in (3.20), we further introduce an auxiliary parameter and write (1.3) into a different ADMM form other than (3.1). We show that this new formulation of the problem, together with a convex approximation for the $\boldsymbol{\beta}$-update when non-convex penalty is applied, will result in a single-loop algorithm where all the updates of the ADMM have closed-form expressions. This can further improve on the computational efficiency of the QR-ADMM.

4.1 The Single-loop QPADM Algorithm

As in Chapter 3, by formulating the penalized quantile regression (1.3) into the equivalent form (2.1), a direct application of ADMM will result in the updates in (3.2). However, the $\boldsymbol{\beta}$-update in (3.2) has no closed-form solution and hence iterative numerical methods like the coordinate descent is required. This results in a double-loop algorithm. We comment that the reason $\boldsymbol{\beta}$-update in (3.2) has no closed-form solution is due to the fact that the coordinates of $\boldsymbol{\beta}$ are entangled with each other as $\boldsymbol{\beta}$ is multiplied with X. We show in the following that, by introducing a new variable, we can disentangle X and $\boldsymbol{\beta}$ and hence simplify the $\boldsymbol{\beta}$-update.

we first split the data into M blocks as follows,

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1^T & \boldsymbol{y}_2^T & \dots & \boldsymbol{y}_M^T \end{pmatrix}^T, \text{ and correspondingly, } X = \begin{bmatrix} X_1^T & X_2^T & \dots & X_M^T \end{bmatrix}^T,$$

where $\boldsymbol{y}_b \in \mathbb{R}^{n_b}$, $X_b \in \mathbb{R}^{n_b \times p}$, and $\sum_{b=1}^M n_b = n$. Then we rewrite problem (1.3) into the following equivalent problem by introducing the new variables $\boldsymbol{\beta}_b$'s,

$$\min_{\boldsymbol{r}_b, \boldsymbol{\beta}_b, \boldsymbol{\beta}} \left\{ \sum_{b=1}^M \rho_\tau(\boldsymbol{r}_b) + P_\lambda(\boldsymbol{\beta}) \right\} \quad \text{s.t. } \boldsymbol{y}_b - X_b \boldsymbol{\beta}_b = \boldsymbol{r_b}, \boldsymbol{\beta}_b = \boldsymbol{\beta},\ b = 1, 2, \dots, M. \tag{4.1}$$

To see why (4.1) follows the standard ADMM form (2.1), we define

$$A = [A_1\ A_2] \text{ with } A_1 = -\begin{bmatrix} I_p & \dots & I_p & 0_{p \times n} \end{bmatrix}^T \in \mathbb{R}^{(Mp+n) \times p}, A_2 = \begin{bmatrix} 0_{n \times Mp} & I_n \end{bmatrix}^T \in \mathbb{R}^{(Mp+n) \times n},$$

and

$$B = \begin{bmatrix} I_p & & & X_1^T & & \\ & \ddots & & & \ddots & \\ & & I_p & & & X_M^T \end{bmatrix}^T \in \mathbb{R}^{(Mp+n) \times Mp},\ \boldsymbol{c} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{y} \end{bmatrix} \in \mathbb{R}^{Mp+n},\ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{r} \end{pmatrix},\ \boldsymbol{z} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M \end{pmatrix}.$$

Then (4.1) can be written exactly as (2.1) with $f(\boldsymbol{x}) = \rho_\tau(\boldsymbol{r}) + P_\lambda(\boldsymbol{\beta})$ and $g(\boldsymbol{z}) = 0$. The resulted updates (2.2) can be explicitly written as

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &:= \arg\min_{\boldsymbol{\beta}} \frac{M\gamma}{2} \left\|\boldsymbol{\beta} - \bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma\right\|_2^2 + P_\lambda(\boldsymbol{\beta}), \\
\boldsymbol{r}_b^{k+1} &:= \arg\min_{\boldsymbol{r}_b} \rho_\tau(\boldsymbol{r}_b) + \frac{\gamma}{2}||\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b^k + \boldsymbol{u}_b^k/\gamma - \boldsymbol{r}_b||_2^2, \\
\boldsymbol{\beta}_b^{k+1} &:= (X_b^T X_b + I)^{-1} \left(X_b^T(\boldsymbol{y}_b - \boldsymbol{r}_b^{k+1} + \boldsymbol{u}_b^k/\gamma) - \boldsymbol{\eta}_b^k/\gamma + \boldsymbol{\beta}^{k+1}\right), \\
\boldsymbol{u}_b^{k+1} &:= \boldsymbol{u}_b^k + \gamma(\boldsymbol{y}_b - X_b\boldsymbol{\beta}_b^{k+1} - \boldsymbol{r}_b^{k+1}), \\
\boldsymbol{\eta}_b^{k+1} &:= \boldsymbol{\eta}_b^k + \gamma(\boldsymbol{\beta}_b^{k+1} - \boldsymbol{\beta}^{k+1}),
\end{aligned} \tag{4.2}$$

where $\bar{\boldsymbol{\beta}}^k = M^{-1}\sum_{b=1}^M \boldsymbol{\beta}_b^k$ and $\bar{\boldsymbol{\eta}}^k = M^{-1}\sum_{b=1}^M \boldsymbol{\eta}_b^k$. The $\boldsymbol{x}$-update in (2.2) is separated into the $\boldsymbol{\beta}$-update and $\boldsymbol{r}$-update in (4.1) since $A_1^T A_2 = 0$. We call the Quantile regression with Parallel ADMm (QPADM) algorithm.

Compared to (3.2), the formulation (4.1) avoids CD by introducing new variables $\boldsymbol{\beta}_b$. All updates in (4.1) including the $\boldsymbol{\beta}$-updates, now can be solved without iterative methods, as shown at the end of this section.

The updates in (4.1) with subscript b depend only on the bth block of the data. When $M > 2$, data blocks $(X_b, \boldsymbol{y}_b)$ can be processed in different computers and hence parallelization can be easily achieved. The parallelization of ADMM was discussed in [6] where the implementation of ADMM on a distributed computing framework called Hadoop [17, 78] was presented. We point out that a new generation of distributed computation framework called Spark [88] is faster for iterative computation and hence more suitable for QPADM. We leave the parallel implementation of QPADM to future work.

The matrix inversion in the $\boldsymbol{\beta}_b$-update in (4.1) takes considerable amount of time when p is large. We suggest to use the Woodbury matrix identity [79] $(X_b^T X_b + I)^{-1} = I - X_b^T(I + X_b X_b^T)^{-1}X_b$, when p is larger than n_b. This makes the QPADM suitable for the case when both n and p are large. We can choose a split M such that for each b we have $n_b \ll p$, so

the $\boldsymbol{\beta}_b$-updates can be implemented efficiently with Woodbury's identity.

Now we show that the updates in (4.1) can be solved without iterative numerical methods. This is clear except for the $\boldsymbol{\beta}$-update. In the following, we derive the solution of the $\boldsymbol{\beta}$-update with the Lasso, SCAD, and MCP penalties. The $\boldsymbol{\beta}$-update for other penalties may be derived in a similar manner.

For the Lasso penalty $P_\lambda(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$, the $\boldsymbol{\beta}$-update of QPADM is solved by

$$\boldsymbol{\beta}^{k+1} = (\bar{\boldsymbol{\beta}}^k + \bar{\boldsymbol{\eta}}^k/\gamma - \lambda/(M\gamma)\mathbf{1}_p)_+ - (-\bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma - \lambda/(M\gamma)\mathbf{1}_p)_-. \tag{4.3}$$

For the SCAD and MCP penalties, the $\boldsymbol{\beta}$-update is nonconvex. Motivated by the majorization step in QICD, at iteration $k+1$, we linearize the penalty $P_\lambda(\boldsymbol{\beta})$ as

$$P_\lambda(\boldsymbol{\beta}) = \sum_{j=1}^{p} P_\lambda(|\beta_j|) \approx \sum_{j=1}^{p} P_\lambda(|\beta_j^k|) + P'_\lambda(|\beta_j^k|_+)(|\beta_j| - |\beta_j^k|). \tag{4.4}$$

Replacing $P_\lambda(\boldsymbol{\beta})$ with the RHS of (4.4) results in

$$\boldsymbol{\beta}^{k+1} := \arg\min_{\boldsymbol{\beta}} \Big\{ \frac{M\gamma}{2} \left\|\boldsymbol{\beta} - \bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma\right\|_2^2 + \sum_{j=1}^{p} P'_\lambda(|\beta_j^k|_+)|\beta_j| \Big\}. \tag{4.5}$$

Denoting $\boldsymbol{v}_\lambda^k := \left[P'_\lambda(|\beta_1^k|_+), \ldots, P'_\lambda(|\beta_p^k|_+)\right]^T$, then (4.5) has a closed-form solution

$$\boldsymbol{\beta}^{k+1} = (\bar{\boldsymbol{\beta}}^k + \bar{\boldsymbol{\eta}}^k/\gamma - \lambda\boldsymbol{v}_\lambda^k/(M\gamma))_+ - (-\bar{\boldsymbol{\beta}}^k - \bar{\boldsymbol{\eta}}^k/\gamma - \lambda\boldsymbol{v}_\lambda^k/(M\gamma))_-. \tag{4.6}$$

Neither (4.3) nor (4.6) requires iterative computation.

4.2 Discussion on the Convergence of QPADM

The convergence of ADMM for convex problems is well studied in the literature. For example, [52] showed the convergence of ADMM under three assumptions: first, f and g are both convex; second, the global minimum exists for problem (2.1); third, A and B have full column ranks. It is easy to check that these assumptions hold for (4.1) with convex penalties, so the convergence of QPADM is guaranteed for convex PQR. We summarize this as the following theorem,

Theorem 4.2.1 *For convex penalties P_λ, the QPADM converges to the solution of (1.3), i.e., $\boldsymbol{\beta}^k$ generated by the QPADM converges to a point $\boldsymbol{\beta}^*$ that solves (1.3).*

Proof The proof is similar to that of Theorem 3.1.1 and we omit it here. ■

As commented in Chapter 3, while the convergence behavior of ADMM for convex problems is well understood, the convergence of ADMM for nonconvex problems remains unknown. Some recent works, including [26, 43, 75], analyzed the convergence of nonconvex ADMM. Their convergence results crucially rely on the Lipschitz-continuity of the subderivative of the objective function, which is not satisfied in our case. Hence the convergence analysis techniques in the literature cannot be directly applied. Although the behavior of nonconvex ADMM remains largely open, the convergence has been widely observed in practice, see the discussions in [26, 75]. We acknowledge that the convergence of nonconvex QPADM needs further theoretical backup as the convergence properties of the ADMM for general nonconvex problems are now still under development, but we emphasis that the QPADM works well for nonconvex PQR in all our simulation studies, as shown in the next section.

4.3 Simulation Study

We evaluate the computational efficiency, estimation accuracy and feature selection accuracy of the QPADM, and compare it with the QICD. We implemented the QPADM in R and used the R package "QICD" for the QICD simulations. All simulations were conducted on a PC with Core i7 4-core processor and 8GB RAM.

The simulation setup is similar to that in Chapter 3, and we repeat it hear for clarity. First, we generate $(\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_p)^T \sim \mathcal{N}(0, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is the covariance matrix with elements $\sigma_{ij} = 0.5^{|i-j|}$, $1 \leq i, j \leq p$. Then we set $X_1 = \Phi(\tilde{X}_1)$ and $X_k = \tilde{X}_k$ for $k = 2, 3, \ldots, p$. We consider the following heteroscedastic regression model,

$$Y = X_6 + X_{12} + X_{15} + X_{20} + 0.7X_1\epsilon, \tag{4.7}$$

where $\epsilon \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$. Three quantile levels were considered: $\tau = 0.3,\ 0.5$ and 0.7. Notice that the effect of X_1 is only present for $\tau = 0.5$. We chose $(n, p) = (300, 1000)$, $(30000, 1000)$ and $(30000, 100)$ respectively, all with $M = 1$. The simulations were repeated 100 times. Tables 4.1–4.3 summarize the results for the SCAD penalty. Results for the MCP penalty are left to Appendix B.1. In the tables, size is the number of nonzero coefficients; P1 is the percentage that $X_6, X_{12}, X_{15}, X_{20}$ were selected; P2 is the percentage that X_1 was selected; AE is the ℓ_1 estimation error; Time measures the running time of the algorithms. Numbers in the parenthesis represent standard deviations.

Following the recent work of [41], we choose the λ that minimizes

$$\text{HBIC}(\lambda) = \log\Big(\sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x_i}^T\hat{\boldsymbol{\beta}}(\lambda)\Big) + |\mathcal{S}_\lambda|\frac{\log(\log n)}{n}C_n, \tag{4.8}$$

where $\hat{\boldsymbol{\beta}}(\lambda)$ is the PQR estimator with the tuning parameter λ, $\mathcal{S}_\lambda \equiv \{j : \hat{\beta}_{\lambda,j} \neq 0, 1 \leq j \leq p\}$and $|\mathcal{S}_\lambda|$ is its cardinality, and C_n is a sequence of positive constants diverging to infinity

as n increases such that $C_n = O(\log(p))$.

The QPADM performs similarly to QICD in terms of model selection accuracy and estimation accuracy. When n is relatively small (Table 1), QPADM is slightly slower than the QICD. In fact, QPADM spends most of the time on the matrix inversion while the iterations of QPADM is cheap compared to the iterations of QICD. The computational advantage of QPADM becomes more evident when n gets larger (Tables 4.2 & 4.3). This is because, as n increases, the amount of time spent on the matrix inversion ($O(p^3)$) becomes less significant and the time required for the iterations dominates. This is further supported by the results in Table 4.3.

Setting $M > 1$ is sometimes necessary when data are too large for a single computer to store and process. We illustrate the advantage of parallelization of QPADM using the same simulation setup as above for $(n, p) = (30000, 100)$. The block number M is set to 1, 10, and 100 respectively. Denoting the time cost of the matrix inversion $(X_b^T X_b + I)^{-1}$ as T_b^0 and the time at iteration k for the $\boldsymbol{\beta}$-update and updates with subscript b as $T_{\boldsymbol{\beta}}^k$ and T_b^k, respectively, the total time cost after iteration K is calculated as $T^K = \max\{T_1^0, \ldots, T_M^0\} + \sum_{k=1}^K T_{\boldsymbol{\beta}}^k + \sum_{k=1}^K \max\left\{T_1^k, \ldots, T_M^k\right\}$. This mimics a real parallel computing framework where the master which conducts the $\boldsymbol{\beta}$-update waits until all workers which conduct updates with subscript b to finish their work before it proceed to the next iteration. The time and estimation accuracy in terms of ℓ_1 estimation error were compared for different M's. The results for $\tau = 0.3$ with SCAD penalty are shown in Figure 4.1. All other cases follow the same pattern, see Appendix B.1 for more details.

As can be seen from Figure 4.1, increasing the block number M does not have significant impact on the convergence but reduces the computational time. One factor that is not considered in this simulation is the communication time. When implementing QPADM in a real distributed framework, increasing M will increase the communication overhead. As a result, the computational gain of increasing M will be exceeded by the increase in

communication cost at some point.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	6.17(1.97)	100%	87%	0.051(0.024)	1.57(0.29)
	$\tau = 0.5$	4.42(0.61)	100%	0%	0.040(0.021)	1.65(0.33)
	$\tau = 0.7$	6.21(2.54)	100%	91%	0.049(0.024)	1.68(0.33)
QICD	$\tau = 0.3$	7.33(3.68)	100%	86%	0.049(0.025)	0.91(0.64)
	$\tau = 0.5$	4.19(0.49)	100%	0%	0.039(0.020)	1.57(1.52)
	$\tau = 0.7$	7.17(3.94)	100%	90%	0.051(0.026)	1.33(1.36)

Table 4.1: Comparison of QPADM and QICD for $(n, p) = (300, 1000)$, SCAD.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0036(0.0016)	44.97(1.66)
	$\tau = 0.5$	4.01(0.00)	100%	0%	0.0037(0.0019)	46.02(1.71)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0039(0.0018)	45.43(1.75)
QICD	$\tau = 0.3$	5.04(0.17)	100%	100%	0.0032(0.0015)	99.83(13.29)
	$\tau = 0.5$	4.10(0.33)	100%	0%	0.0039(0.0014)	125.39(16.35)
	$\tau = 0.7$	5.14(0.27)	100%	100%	0.0030(0.0014)	129.98(17.72)

Table 4.2: Comparison of QPADM and QICD with $(n, p) = (30000, 1000)$, SCAD.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0030(0.0011)	3.05(0.63)
	$\tau = 0.5$	4.00(0.00)	100%	0%	0.0029(0.0011)	3.59(0.58)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0030(0.0012)	3.98(0.64)
QICD	$\tau = 0.3$	5.04(0.17)	100%	100%	0.0027(0.0011)	11.98(3.64)
	$\tau = 0.5$	4.16(0.37)	100%	0%	0.0026(0.0011)	21.73(11.25)
	$\tau = 0.7$	5.04(0.16)	100%	100%	0.0026(0.0009)	24.99(14.12)

Table 4.3: Comparison of QPADM and QICD with $(n, p) = (30000, 100)$, SCAD.

We point out that our implementation of the QPADM is kept as simple as possible with no special implementation-level optimization or tuning. The computational advantage of QPADM may not be fully illustrated by the simulations results above.

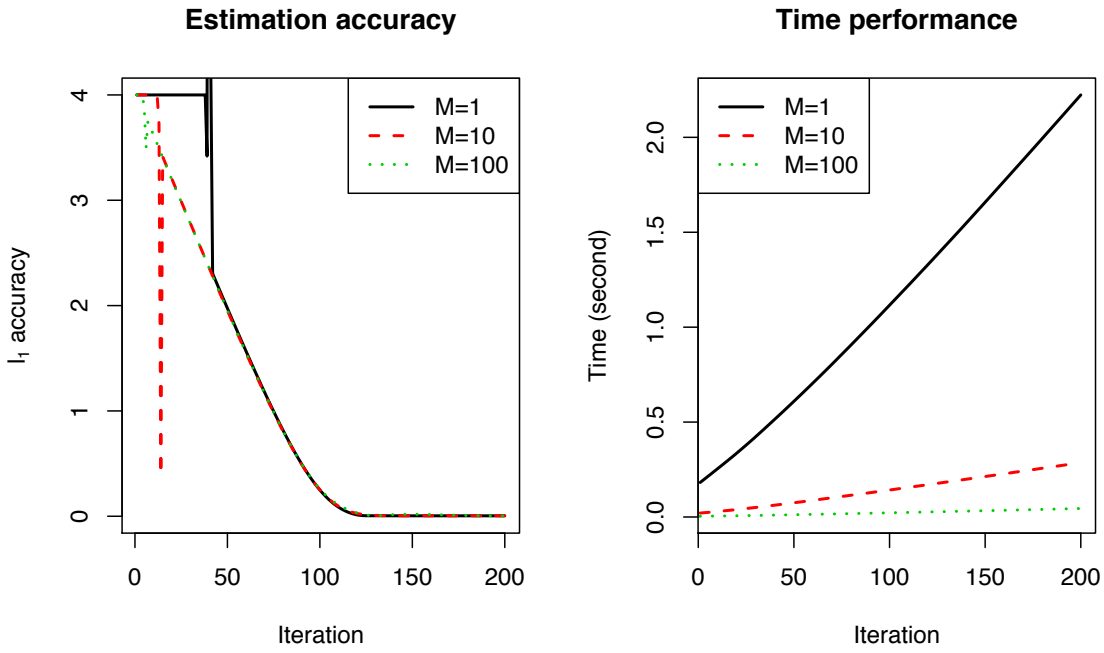


Figure 4.1: Comparison of QPADM with SCAD penalty for different M values.

4.4 Conclusion and Discussion

In this chapter, we propose the single-loop QPADM algorithm for the penalized quantile regression problem (1.3). It is computationally advantageous compared to QICD since each of its iteration can be solved without iterative methods. The simulation study showed that the QPADM performs similarly as QICD in terms of statistical accuracy and can be significantly faster when n is large. More importantly, unlike the QICD, the QPADM is a distributed algorithm that can be implemented in distributed framework like Spark. This gives QPADM the ability to solve large scale problems. As can be seen from the simulation studies in Section 4.3, our current "parallel" implementation of the QPADM is actually pseudo-parallel. We implement the parallel QPADM in a sequential way be stacking up the supposedly parallel ADMM updates. As a future work, we intend to implement the parallel QPADM algorithm in Spark.

To the best of our knowledge, the approach of introducing new variables to the ADMM to avoid iterative methods for the updates is novel in the literature. It can potentially be generalized to other statistical model fitting problems, when the inner-loop of ADMM requires time-consuming iterative algorithms.

Chapter 5

Group Sparsity via Approximated Information Criteria

We propose a new group variable selection and estimation method, and illustrate its application to the generalized linear model (GLM). This new method, termed "gMIC", is derived as a smooth approximation the information criterion. The gMIC is formulated in two steps. First, a smooth unit dent function is applied for the approximation of the information criterion. Then, the approximated information criterion is further reparameterized in a way that yields sparse estimation from a smooth programming problemCompared to existing group variable selection and estimation methods, the gMIC is free of parameter tuning and hence computationally advantageous. We establish the oracle property of the proposed method that is supported by both simulation studies and real examples. We emphasize that, although our current work on gMIC focuses on GLM, it may be naturally extended to variable selection for quantile regression.

5.1 Introduction

Consider data that consist of $\{(y_i, \mathbf{x}_i) : i = 1, \ldots, n\}$, where y_i is the response and $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$ is the p-dimensional predictor vector. A regression model tries to find the relationship between the responses y_i's and the linear combinations of the covariates $\boldsymbol{x}_i$'s. For example, a generalized linear model [51] links the mean response $E(y_i)$ to the covariates $\boldsymbol{x}_i$ through the linear predictor $\boldsymbol{x}_i^T\boldsymbol{\beta}$ via certain link function, where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ is the vector of regression coefficients. The linear model is a special case of GLM where the link function is the identity function. Concerning variable selection, the true $\boldsymbol{\beta}$ is often sparse in the sense that some of its components are zeros.

Standard approaches to variable selection and estimation include the best subset selection (BSS) and penalization methods. In BSS, an information criterion, for example, the *Akaike Information criterion* [1] (AIC) or the *Bayesian information criterion* [63] (BIC), is used to evaluate and compare all possible models, and the "best" model is selected accordingly. To this end, we assume that either there is no nuisance parameter involved or the nuisance parameters and $\boldsymbol{\beta}$ are orthogonal [15]. Hence we simply denote the log-likelihood function as $L(\boldsymbol{\beta})$. Specifically, the BSS solves

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{-2L(\boldsymbol{\beta}) + \lambda_0 \|\boldsymbol{\beta}\|_0\right\}, \tag{5.1}$$

where the ℓ_0-norm of a vector $\boldsymbol{\beta} \in \mathbb{R}^p$, denoted as $\|\boldsymbol{\beta}\|_0$, is defined as the number of non-zero elements in $\boldsymbol{\beta}$. The penalization parameter λ_0 depends on the information criterion used, but is fixed a priori. For example, the parameter λ_0 is set to 2 or $\ln(n)$ for AIC or BIC, respectively. The computation of (5.1) is NP-hard [9] as we need to go through all possible combinations of the variables. This makes it infeasible even for moderately large p. Penalization methods can be thought of as the (convex or non-convex) relaxation of the AIC or BIC that leads to tractable computation. It enforces sparsity to the model by

approximating the ℓ_0-norm $\|\boldsymbol{\beta}\|_0$ with a nonnegative continuous penalty function. That is, penalization methods solve

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ -2L(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} P(|\beta_j|) \right\}, \tag{5.2}$$

where $P(\cdot)$ is the penalty function. Common choices of the penalty function include the convex Lasso [69], adaptive Lasso [95], elastic net [96], and the non-convex SCAD [19] and MCP [89], among others. These penalty functions have a common feature that they are symmetric around 0 and "sharp" at 0, that is, $P(x) = P(-x)$ and $\lim_{x\to 0^+} P'(x) \geq \delta$ for some $\delta > 0$. For example, in Lasso, the penalty function is $P(|\beta|) = |\beta|$ with $\delta = 1$. From the optimization perspective, such penalty functions provide consistent momentum to push small coefficients towards 0 when solving (5.2), e.g., with gradient descent, resulting in a solution $\hat{\boldsymbol{\beta}}$ with some coordinates being exactly 0. A geometric interpretation of why such penalties will result in sparse estimation of $\boldsymbol{\beta}$ can be found in [69]. A key difference between (5.1) and (5.2) is that the penalization parameter $\lambda > 0$ in (5.2) is not fixed a priori, but needs to be tuned, typically by cross-validation. We refer readers to [20] and [49] for an overview of penalization methods.

The BSS method and penalization methods as formulated in (5.1) and (5.2) are for individual variable selections, that is, each individual variable comes into or leaves the model as a unit. However in practice, the components in $\boldsymbol{\beta}$ appear in groups under many scenarios. For example, dummy variables introduced by multi-level categorical variables, basis functions based on one variable in basis expansion, and genes that share a common biological function or participate in the same metabolic pathway [80], all form natural groups of variables. In such applications, group variable selection where the entire group of variables comes into or leaves the model as a unit is essential. On one hand, for interpretability reasons, we may want the entire group of variables (for example, when interpreted as a factor) to be

included in or excluded from the model. On the other hand, group variable selection results in better estimation and variable selection performance in the presence of group structures, as indicated by [31].

To incorporate the group sparsity structure, we assume that the columns of the data matrix $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T \in \mathbb{R}^{n \times p}$ can be partitioned into K subsequent groups as $X = (X_1, \ldots, X_K)$. Each $X_k \in \mathbb{R}^{n \times m_k}$ consists of a subset of columns of X corresponding to the k-th group of variables, where m_k is the number of variables in the k-th group for $k = 1, \ldots, K$, satisfying $\sum_{k=1}^{K} m_k = p$. Accordingly, the components of $\boldsymbol{\beta}$ are also partitioned into K groups as $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T, \ldots, \boldsymbol{\beta}_K^T)^T$, where $\boldsymbol{\beta}_k = (\beta_{k1}, \ldots, \beta_{km_k})^T \in \mathbb{R}^{m_k}$ contains the coordinates of the k-th group of variables. Throughout this chapter, we consider the dimension p and the partition of variables as fixed. And for any vector $\boldsymbol{u} \in \mathbb{R}^p$, we denote its k-th group subcomponent according to this partition as $\boldsymbol{u}_k$.

Several extensions of penalization methods have been proposed to facilitate group-level variable selection. The group Lasso [87] replaces the ℓ_1-penalty on single variables in the Lasso by the ℓ_2 penalty on the groups. It solves the following optimization,

$$\min_{\boldsymbol{\beta}} \left\{ -2L(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{K} m_k \|\boldsymbol{\beta}_k\|_{\mathbf{M}_k} \right\}, \tag{5.3}$$

where $\|\beta\|_{\mathbf{M}}$ is defined as

$$\|\boldsymbol{\beta}\|_{\mathbf{M}} = \sqrt{\boldsymbol{\beta}^T \mathbf{M} \boldsymbol{\beta}},$$

with a positive-definite matrix $\mathbf{M}$, and m_k's adjust for the group sizes. The positive-definite matrices M_k's are usually set to the identity matrices I_{m_k}'s for $k = 1, 2, \ldots, K$. As shown in [87], the penalty $\|\cdot\|_{\mathbf{M}}$ imposes sparsity at group level, i.e., the grouped variables $\boldsymbol{\beta}_k$ will be exactly $\mathbf{0}$ for some k's. The group versions of SCAD and MCP can be similarly derived by replacing the ℓ_0-norm in these penalties with ℓ_2 penalties on the groups of variables. We refer the readers to [30] for a comprehensive review of group-type selection with penalization

methods.

While the group extensions of penalization methods has been widely adopted and studied in the literature, the BSS method has not yet been considered for group variable selection to the best of our knowledge. The BSS method enjoys two advantages over penalization methods. First, it does not involve parameter tuning. In penalization method, the penalization parameter λ needs to be tuned. This means $\hat{\boldsymbol{\beta}}$ is computed for each λ over a grid of tuning parameters, which is computationally expensive. In BSS (5.1), the parameter λ_0 is fixed and hence requires no tuning. With a proper approximation, an approximated solution of problem (5.1) can potentially be efficiently solved using existing optimization algorithms. This could compare favorably to penalization methods in terms of computation as the approximated BSS solution only needs to be computed once with a fixed λ_0. Second, penalization methods often use cross-validation to select the tuning parameter λ which may result in different models selected with different training/validation partitioning of the same data, but the BSS does not suffer such problems.

In this chapter, we propose an alternative to penalization methods for group variable selection by minimizing an approximated "group version" of the Bayesian information criterion. The proposed method, named *"group minimum information criterion"* (gMIC), is derived as follows. First, we propose a group version of the information criterion for group variable selection. Then, a smooth approximation of the information criterion is applied. Finally, we apply a specific form of reparameterization to the approximated information criterion that results in a sparsity-generating yet smooth programming problem that can be efficiently solved by existing optimization tools. Previous attempts of this idea on individual variable selection can be found in [66] and [67].

The remainder of this chapter is organized as follows. Section 5.2 presents our proposed method in details. In Section 5.3, we derive the oracle properties of the gMIC estimator. Section 5.4 is a discussion on the inference of $\boldsymbol{\beta}$. In Section 5.5, we conduct simulations and

present real data examples to illustrate the performance of gMIC. Section 5.6 concludes the chapter with some discussion.

5.2 The Group MIC Formulation

First, we modify the information criterion (5.1) to facilitate group variable selection. Specifically, we replace the ℓ_0 penalty on the cardinality of $\boldsymbol{\beta}$ by a penalty on the cardinality at the group level,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ -2L(\boldsymbol{\beta}) + \lambda_0 \sum_{k=1}^{K} m_k \mathbb{I}(\boldsymbol{\beta}_k \neq \mathbf{0}) \right\}. \tag{5.4}$$

The formulation (5.4) serves as a generalization of the information criterion in (5.1). With the practical assumption that an entire group of variables enter (all non-zero) or leave (all zero) the model, (5.4) is equivalent to (5.1). However, compared to (5.1), (5.4) enforces variable selection at the group level.

To solve the discrete optimization problem (5.4) one needs to fit a generalized linear model for each combination of the groups of the variables, which can be computationally demanding. As a solution to avoiding the computational complexity, we consider applying smooth approximation to the discrete problem (5.4). This results in a single smooth optimization problem that can be efficiently solved by existing smooth programming algorithms. In this chapter, we consider using a specific type of functions, called *unit dent functions*, for this approximation. A unit dent function $w(x)$ is a continuous and even function that equals 0 at $x = 0$ and monotonically increases with $\lim_{x\to\infty} w(x) = 1$. The approximation is done in two steps as we detail below.

First, the information criterion (5.4) is approximated by the smooth unit dent function

$tanh(\cdot)$ as follows,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ -2L(\boldsymbol{\beta}) + \lambda_0 \sum_{k=1}^{K} m_k \tanh(a_n \|\boldsymbol{\beta}_k\|^2_{\mathbf{M}_k}) \right\}, \tag{5.5}$$

where $\tanh(a_n \|\boldsymbol{\beta}_k\|^2_{\mathbf{M}_k})$ approximates the indicator function $\mathbb{1}(\boldsymbol{\beta}_k \neq \mathbf{0})$, as shown in Figure 5.1. The parameter a_n adjusts for the approximation accuracy and may be dependent on n. Generally, any smooth unit dent function can be used as an approximation to the problem (5.4). Here we select the unit dent function $\tanh(\cdot)$ because it has a simple form and its derivative can also be easily calculated.

As indicated in Figure 5.1 below, the direct approximation (5.5) results in a smooth optimization problem. However, noticing that the $\tanh(\cdot)$ function is "flat" at 0, this approximation will not result in sparse estimation of $\boldsymbol{\beta}$.

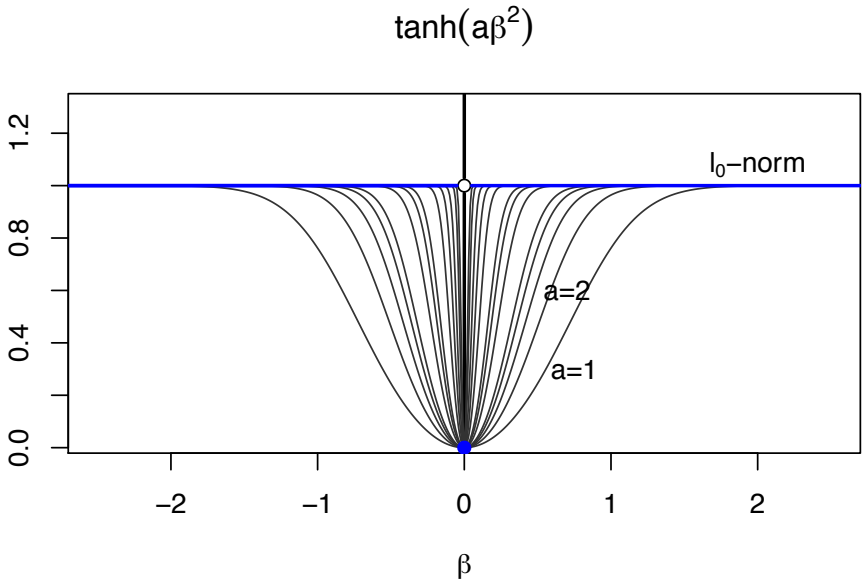


Figure 5.1: The $\tanh(\cdot)$ function as an approximation to the ℓ_0-norm.

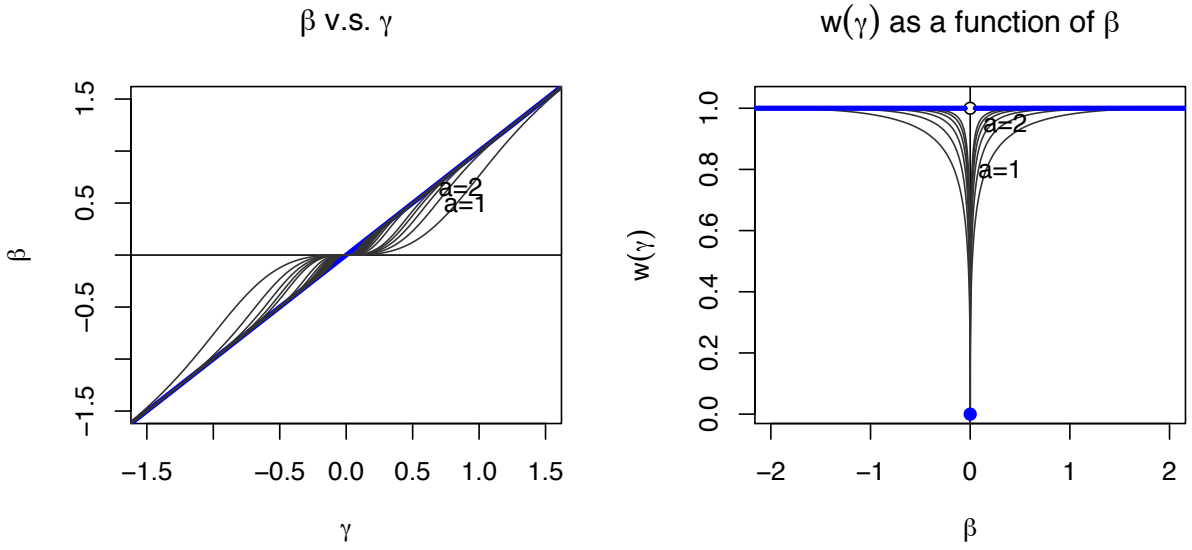
To enforce sparsity, we apply the following one-to-one reparameterization as the second step,

$$\boldsymbol{\beta}_k = \boldsymbol{\gamma}_k w(\boldsymbol{\gamma}_k),\ k = 1, 2, \ldots, K, \tag{5.6}$$

where $w(\boldsymbol{\gamma}_k) := \tanh(a_n \|\boldsymbol{\gamma}_k\|_2^2)$, and $\boldsymbol{\beta}_k$ and $\boldsymbol{\gamma}_k$ both correspond to the k-th group of variables. Then we reformulate the problem (5.5) as

$$\hat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}} \left\{ -2L(\mathbf{W}\boldsymbol{\gamma}) + \lambda_0 \sum_{k=1}^{K} m_k w(\boldsymbol{\gamma}_k) \right\}, \tag{5.7}$$

where $\mathbf{W}$ is a $K \times K$ block-diagonal matrix with the k-th block being $w(\boldsymbol{\gamma}_k)\mathbf{I}_{m_k}$. After solving (5.7), the solution for $\hat{\boldsymbol{\beta}}$ can be computed as $\hat{\boldsymbol{\beta}} = \mathbf{W}(\hat{\boldsymbol{\gamma}})\hat{\boldsymbol{\gamma}}$.



(a) Relationship between β and γ (b) $w(\gamma)$ as a function of β

Figure 5.2: The reparameterization.

The reparameterization (5.6) is shown in Figure 5.2. The intuition why the smooth reparameterization (5.6) generates sparse estimation is explained as follows. Although the $\tanh(\cdot)$ part in (5.7) as a function of $\boldsymbol{\gamma}$ is still smooth at 0 and does not generate sparse

estimation of $\boldsymbol{\gamma}$, it will shrink the estimation of $\boldsymbol{\gamma}$ towards 0. Then, notice in Figure 5.2a that the derivative of $\boldsymbol{\gamma}$ w.r.t. $\boldsymbol{\beta}$ goes to infinity when $\boldsymbol{\gamma}$ goes to zero. This will drastically "squeeze" the estimation of $\boldsymbol{\beta}$ towards 0 when $\boldsymbol{\gamma}$ is near 0. As a result, this reparameterization will generate an estimate of $\boldsymbol{\beta}$ with some very small coordinates (e.g, less than 10^{-6}) that can be virtually regarded as 0. Another point of view is to treat $w(\boldsymbol{\gamma}_k)$ in (5.7) as a penalty of $\boldsymbol{\beta}$. This penalty, although being smooth at 0 as a function of $\boldsymbol{\gamma}$, is "sharp" as a function of $\boldsymbol{\beta}$, as show in Figure 5.2b. In theory, this sharp penalty of $\boldsymbol{\beta}$ should generate sparse estimate of $\boldsymbol{\beta}$ if we plug in $\boldsymbol{\beta}$ into (5.7) and solve the optimization w.r.t. $\boldsymbol{\beta}$. In practice, since $\boldsymbol{\gamma}$ does not have a well defined expression in terms of $\boldsymbol{\beta}$, we solve (5.7) w.r.t. $\boldsymbol{\gamma}$ instead for computational convenience and recover $\boldsymbol{\beta}$ after solving for $\boldsymbol{\gamma}$. Due to smoothness, this practice does not directly recover the sparsity for the estimate of $\boldsymbol{\gamma}$, but rather generates some small coordinates that are close to zero. These small coordinates of $\boldsymbol{\gamma}$, are further shrunk towards 0 when converting the estimate of $\boldsymbol{\gamma}$ to the estimate of $\boldsymbol{\beta}$. In practice, sparse estimation of $\boldsymbol{\beta}$ is achieved after applying some very small threshold, e.g., 10^{-6}.

We emphasize that the parameter a_n can be fixed without tuning. It plays the same role as the additional shape parameter in SCAD and MCP, both being fixed. Unlike penalization methods whose performance crucially relies on the tuning of the penalization parameter λ, the performance of gMIC is insensitive to the choice of a_n. Empirically, we show in the simulation study in Section 5.5 that the performance of gMIC stabilizes for a wide range of a_n's. Theoretically, we prove that as long as $a_n = O(n)$, the performance of the gMIC is guaranteed, see Section 5.3 for details.

In principle, the gMIC approximation can be applied to any information criterion. For simplicity of discussion, we focus on approximating the BIC as an illustration and henceforth set the parameter λ_0 to $\ln(n)$ in this chapter. For notational simplicity, we also set the positive definite matrices M_k's to the identity matrices $I_{m_k},\ k = 1, 2, \ldots, K$ in all theoretical

derivations. That is, we formulate the gMIC estimator as the solution

$$\hat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}} \left\{ -2L(\mathbf{W}\boldsymbol{\gamma}) + \ln(n) \sum_{k=1}^{K} m_k \tanh(a_n \|\boldsymbol{\gamma}_k\|_2) \right\}. \tag{5.8}$$

The simulated annealing in [2] followed by the modified BFGS in [42] is applied to solve the non-convex optimization problem (5.8). Due to the non-convexity of the objective function, the uniqueness of solution to (5.8) is not guaranteed. For the theoretical analysis in the next section, we focus on any local minimum of problem (5.8).

5.3 Asymptotic Properties

Assume that $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$ are i.i.d. copies from a density $f(\boldsymbol{x}, y; \boldsymbol{\beta}_0)$, where $\boldsymbol{\beta}_0$ denotes the true regression vector. Without loss of generality, the groups in $\boldsymbol{\beta}_0$ have been arranged such that $\boldsymbol{\beta}_0 = (\boldsymbol{\beta}_{0(1)}^T, \boldsymbol{\beta}_{0(0)}^T)^T$, where $\boldsymbol{\beta}_{0(1)} = (\boldsymbol{\beta}_{01}^T, \ldots, \boldsymbol{\beta}_{0K_1}^T)^T$ consists of all K_1 nonzero components and $\boldsymbol{\beta}_{0(0)} = \mathbf{0}$ consists of$(K - K_1)$ groups of all the zero components. Denote $p_1 = \sum_{k=1}^{K_1} m_k$ and $p_0 = p - p_1$ so that $\boldsymbol{\beta}_{0(1)} \in \mathbb{R}^{p_1}$ and $\boldsymbol{\beta}_{0(0)} \in \mathbb{R}^{p_0}$. Let $\mathbf{I} = \mathbf{I}(\boldsymbol{\beta}_0)$ be the Fisher information for the full model (that is, the model with all variables included), and let $\mathbf{I}_1$ be the Fisher information corresponding to the reduced true model setting $\boldsymbol{\beta}_{0(0)} = \mathbf{0}$. It is well known that $\mathbf{I}_1$ equals the p_1-th principal submatrix of $\mathbf{I}$.

For theoretical exploration, we consider the gMIC estimator $\widetilde{\boldsymbol{\beta}}$ obtained from minimizing the following objective function

$$Q_n(\boldsymbol{\beta}) = -2\, L(\boldsymbol{\beta}) + \ln(n) \sum_{j=1}^{K} m_k \tanh(a_n \|\boldsymbol{\gamma}_k\|_2^2), \tag{5.9}$$

where $\boldsymbol{\beta}_k = w_k \boldsymbol{\gamma}_k$ with $w_k = \tanh(a_n \|\boldsymbol{\gamma}_k\|_2^2)$ for $k = 1, \ldots, K$. As a generic notation, we use $\widetilde{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{\beta}}$ to denote the gMIC and MLE estimators, respectively. The MLE estimator $\widehat{\boldsymbol{\beta}}$ will be used in the proof of Theorems 5.3.1. Throughout the chapter, we shall use $\|\cdot\|$ for the

Euclidean norm $\|\cdot\|_2$ as default.

We first establish a lemma concerning the reparameterization step, which will later be used in the proof of Theorems 5.3.1 and 5.4.1.

Lemma 5.3.1 *Given $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^m$ that satisfies $\boldsymbol{\beta} = w\boldsymbol{\gamma}$ with $w = w(\boldsymbol{\gamma}) = \tanh(a\|\boldsymbol{\gamma}\|^2)$, we have*

(i) $\|\boldsymbol{\beta} - \boldsymbol{\gamma}\| = \dfrac{2\|\boldsymbol{\gamma}\|}{\exp(-2a\|\boldsymbol{\gamma}\|^2) + 1}$.

(ii) As a function of $\boldsymbol{\beta}$, $w(\cdot)$ is not differentiable at $\boldsymbol{\beta} = \mathbf{0}$. For $\boldsymbol{\beta} = (\beta_j)_{j=1}^m$ with $\beta_j \neq 0$,

$$\frac{dw}{d\boldsymbol{\beta}} = 2a(1 - w^2)\left\{w\,\boldsymbol{I}_m + 2a(1 - w^2)\,\boldsymbol{\gamma}\boldsymbol{\gamma}^T\right\}^{-1}\boldsymbol{\gamma}.$$

Proof The proof can be found in Appendix A.3. ■

We then introduce some regularity conditions that are required to establish our main results.

Assumption 5.3.1 (The regularity conditions)

(A) The first and second logarithmic derivative of f satisfies

$$E_{\boldsymbol{\beta}}\left[\frac{\partial \log f(\boldsymbol{x}, y; \boldsymbol{\beta})}{\partial \beta_j}\right] = 0 \quad \textit{for } j = 1, \ldots, d$$

and

$$I_{jk}(\boldsymbol{\beta}) = E_{\boldsymbol{\beta}}\left[\frac{\partial}{\partial \beta_j}\log f(\boldsymbol{x}, y; \boldsymbol{\beta})\frac{\partial}{\partial \beta_k}\log f(\boldsymbol{x}, y; \boldsymbol{\beta})\right] = E_{\boldsymbol{\beta}}\left[\frac{\partial^2}{\partial \beta_j \partial \beta_k}\log f(\boldsymbol{x}, y; \boldsymbol{\beta})\right].$$

(B) The Fisher information matrix

$$I(\boldsymbol{\beta}) = E\left\{\left[\frac{\partial}{\partial \boldsymbol{\beta}}\log f(\boldsymbol{x}, y; \boldsymbol{\beta})\right]\left[\frac{\partial}{\partial \boldsymbol{\beta}}\log f(\boldsymbol{x}, y; \boldsymbol{\beta})\right]^T\right\}$$

is finite and positive definite at $\boldsymbol{\beta} = \boldsymbol{\beta}_0$.

(C) There exists an open subset ω *that contains the true parameter point* $\boldsymbol{\beta}_0$ *such that for almost all* $(\boldsymbol{x}, y)$ *the density* $f(\boldsymbol{x}, y; \boldsymbol{\beta})$ *admits all third derivatives* $(\partial^3 f(\boldsymbol{x}, y; \boldsymbol{\beta}))/(\partial\beta_j \partial\beta_k \partial\beta_l)$ *for all* $\boldsymbol{\beta} \in \omega$. *Furthermore, there exist functions* M_{jkl} *such that*

$$\left| \frac{\partial^3}{\partial\beta_j \partial\beta_k \partial\beta_l} \log f(\boldsymbol{x}, y; \boldsymbol{\beta}) \right| \leq M_{jkl}(\boldsymbol{x}, y) \quad \textit{for all } \boldsymbol{\beta} \in \omega,$$

where $m_{jkl} = E_{\boldsymbol{\beta}_0} [M_{jkl}(\boldsymbol{x}, y)] < \infty$ *for* $j,\ k,\ l$.

Assumptions (A)–(C) are the same as conditions (A)–(C) in [19] and are standard in related literature. The following theorem shows that there exists a local minimizer $\widetilde{\boldsymbol{\beta}}$ of $Q_n(\boldsymbol{\beta})$ that is $\sqrt{n}$-consistent to $\boldsymbol{\beta}_0$ and this $\sqrt{n}$-consistent $\widetilde{\boldsymbol{\beta}}$ enjoys the 'oracle' property under some standard assumptions.

Theorem 5.3.1 *Let* $\{(\boldsymbol{x}_i, Y_i) : i = 1, \ldots, n\}$ *be* n *i.i.d. copies from a density* $f(\boldsymbol{x}, Y; \boldsymbol{\beta}_0)$. *Concerning group sparsity, we assume that none of the components of* $\boldsymbol{\beta}_{0k}$ *is zero for* $k = 1, \ldots, K_1$ *while* $\boldsymbol{\beta}_{0k} = \mathbf{0}$ *for* $k = K_1+1, \ldots, K$. *Following [19], under the regularity conditions (A)–(C), we have the following conclusions when* $a_n = O(n)$,

(i). ($\sqrt{n}$*-Consistency) there exists a local minimizer* $\widetilde{\boldsymbol{\beta}}$ *of* $Q_n(\boldsymbol{\beta})$ *that is* $\sqrt{n}$*-consistent for* $\boldsymbol{\beta}_0$ *in the sense that* $\|\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}_0\| = O_p(n^{-1/2})$.

(ii). (Sparsity and Asymptotic Normality) Partition $\widetilde{\boldsymbol{\beta}}$ *in (i) as* $(\widetilde{\boldsymbol{\beta}}_{(1)}^T, \widetilde{\boldsymbol{\beta}}_{(0)}^T)^T$ *in a similar manner to* $\boldsymbol{\beta}_0$. *With probability tending to 1 as* $n \to \infty$, $\widetilde{\boldsymbol{\beta}}$ *must satisfy that*

$$\widetilde{\boldsymbol{\beta}}_{(0)} = \mathbf{0}$$

and

$$\sqrt{n}(\widetilde{\boldsymbol{\beta}}_{(1)} - \boldsymbol{\beta}_{0(1)}) \ \to\ N\left(\mathbf{0},\, \mathbf{I}_1^{-1}\right).$$

Proof The proof can be found in Appendix A.4. ■

The regularity conditions (A)–(C) essentially assure asymptotic normality of the ordinary maximum likelihood estimators. The additional assumption that $\boldsymbol{\beta}_k$ has no zero component for $k = 1, \ldots, K_1$ meets the need in the common applications of group sparsity such as dummy variables introduced for a categorical predictor or basis functions based on one variable in basis expansion.

Theorem 5.3.1 essentially established the "oracle" property of the gMIC estimator in the usual sense. The feature selection consistency of gMIC is also established in the second conclusion of Theorem 5.3.1. The asymptotic distribution of the "zero" part of the model parameter, i.e., $\boldsymbol{\beta}_{(0)}$, on the other hand, is not available here. This is very common under regularized estimation scenarios where the asymptotic properties of the "zero" part of the estimators are typically missing.

Another observation is that, from the theoretical perspective, the gMIC is insensitive to the choice of a_n. As long as a_n goes to infinity at the same rate with n, the above asymptotic results are guaranteed. This observation is further supported by the numerical examples in Section 5.5, where we found that the performance of gMIC is insensitive to the choice of a_n. As a result, the parameter a can be fixed a priori without tuning.

5.4 Inference of $\boldsymbol{\beta}$ via $\boldsymbol{\gamma}$

We aim to make inferences about the gMIC estimate at the group level. Specifically, we want to test if $\boldsymbol{\beta}_k = \mathbf{0}$ for a specific group k. This goal is not directly achievable from Theorem 5.3.1, where the asymptotic distribution is only available for selected variables, i.e., for $\boldsymbol{\beta}_k \neq \mathbf{0}$. Observe that the objective function (5.7) is smooth in $\boldsymbol{\gamma}_k$, so the statistical properties, or more specifically, the asymptotic normality, of $\tilde{\boldsymbol{\gamma}}$ is readily available following standard M-estimator arguments. Also, notice that the reparameterization (5.6) is a bijection

between $\boldsymbol{\beta}_k$ and $\boldsymbol{\gamma}_k$, and $\boldsymbol{\beta}_k = \mathbf{0}$ if and only if $\boldsymbol{\gamma}_k = \mathbf{0}$. Consequently, testing $\boldsymbol{\beta}_k = \mathbf{0}$ is equivalent to testing $\boldsymbol{\gamma}_k = \mathbf{0}$, which can be done based on the asymptotic distribution of $\tilde{\boldsymbol{\gamma}}$. In particular, we establish the asymptotic normality of $\tilde{\boldsymbol{\gamma}}$ in the following theorem,

Theorem 5.4.1 *Let $\boldsymbol{\gamma}_0$ be the reparameterized parameter vector associated with $\boldsymbol{\beta}_0$. Under the regularity conditions (A)–(C) in [19], and that $a_n = O(n)$, we have*

$$\sqrt{n}[\mathbf{D}(\boldsymbol{\gamma}_\mathbf{0})(\tilde{\boldsymbol{\gamma}} - \boldsymbol{\gamma}_\mathbf{0}) + \mathbf{b_n}] \xrightarrow{\mathbf{d}} \mathbf{N}(\mathbf{0}, \mathbf{I^{-1}}(\boldsymbol{\gamma}_\mathbf{0})), \tag{5.10}$$

where $D(\boldsymbol{\gamma}_0)$ is a $K \times K$ block diagonal matrix with the k-th block being

$$D_k(\boldsymbol{\gamma}_0) = \{w_k I_{m_k} + 2a_n(1 - w_k^2)\boldsymbol{\gamma}_{0k}\boldsymbol{\gamma}_{0k}^T\}\big|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}_0},$$

and the asymptotic bias

$$\mathbf{b}_n = \{-\ddot{L}(\boldsymbol{\beta}_0)\}^{-1}\frac{\ln(n)}{2}\mathbf{diag}\left\{2a_n(1 - w_k^2)D_k^{-1}(\widetilde{\boldsymbol{\gamma}}_k)\widetilde{\boldsymbol{\gamma}}_k\right\}_{k=1}^K. \tag{5.11}$$

And D and $\mathbf{b}_n$ satisfy (i) $\lim_{n\to\infty}[D(\boldsymbol{\gamma}_0)]_{ii} = I\{\gamma_{0i} \neq 0\}$, $\lim_{n\to\infty}[D(\boldsymbol{\gamma}_0)]_{ij} = 0$ $(i \neq j)$ and (ii) $\mathbf{b_n} = \mathbf{o_p}(\mathbf{1})$.

Proof The proof can be found in Appendix A.5. ■

5.4.1 Testing $H_0: \boldsymbol{\beta}_k = 0$ v.s. $H_1: \boldsymbol{\beta}_k \neq 0$

Based on Theorem 5.4.1, a Wald-type test statistic for the k-th group of variable can be defined by

$$\chi^2_{W,k} := n(\tilde{\mathbf{D}}_k\tilde{\boldsymbol{\gamma}}_k + \tilde{\mathbf{b}}_{nk})^T I(\tilde{\boldsymbol{\gamma}}_k)(\tilde{\mathbf{D}}_k\boldsymbol{\gamma}_k + \tilde{\mathbf{b}}_{nk}),$$

where $\tilde{\mathbf{D}}_k$ and $\tilde{\mathbf{b}}_n$ are the estimate of $\mathbf{D_k}$ and $\mathbf{b_n}$, by replacing $\boldsymbol{\gamma}_0$ with $\tilde{\boldsymbol{\gamma}}$. And the matrix $I(\tilde{\boldsymbol{\gamma}}_k)$ is taken from the diagonal blocks of $I(\tilde{\boldsymbol{\gamma}})$ corresponding to the k-th group of variables,

i.e.,

$$I(\tilde{\boldsymbol{\gamma}}) = \begin{bmatrix} I(\tilde{\boldsymbol{\gamma}}_1) & * & * \\ * & \ddots & * \\ * & * & I(\tilde{\boldsymbol{\gamma}}_K) \end{bmatrix}, \text{ with } I(\tilde{\boldsymbol{\gamma}}_k) \in \mathbb{R}^{m_k \times m_k}$$

Further simplification of $\chi^2_{W,k}$ can be made by ignoring $\tilde{\mathbf{D}}_k$, $\tilde{D}_{jj}$, $\tilde{\mathbf{b}}_{nk}$, and $\tilde{b}_{nj}$, and as $\tilde{\mathbf{D}}_k$ approaches the identity for nonzero $\boldsymbol{\gamma}_k$'s and 0 for zero $\boldsymbol{\gamma}_k$'s, and $\tilde{\mathbf{b}} \to \mathbf{0}$. This results in

$$\chi^2_{W,k} := n\tilde{\boldsymbol{\gamma}}_k^T I(\tilde{\boldsymbol{\gamma}}_k)\boldsymbol{\gamma}_k. \tag{5.12}$$

Under H_0, the test statistic $\chi^2_{W,k}$ in (5.12) has an asymptotic $\chi^2_{m_k}$-distribution. Then at significance level α, we reject H_0 when $\chi^2_{W,k} > \chi^2_{m_k,1-\alpha}$. The p-value of the test for the k-th group of variables can also be calculated from $\chi^2_{W,k}$ as

$$\text{p-value} \;=\; P(X > \chi^2_{W,k} \text{ where } X \sim \chi^2_{m_k}), \tag{5.13}$$

where $\chi^2_{W,k}$ is computed from (5.12).

5.4.2 Confidence Region of $\boldsymbol{\gamma}_k$

Theorem 5.4.1 can also be used to construct confidence intervals for individual β_j's ($j = 1, \ldots, p$) and confidence regions for the grouped variables $\boldsymbol{\gamma}_k$'s ($k = 1, \ldots, K$). Specifically, a $100(1-\alpha)\%$ confidence interval of γ_j is given by

$$(\tilde{D}_{jj}\tilde{\gamma}_j + \tilde{b}_{nj}) \pm z_{1-\alpha/2}\sqrt{I^{-1}(\tilde{\boldsymbol{\gamma}})_{jj}/n},$$

which can be simplified as

$$\tilde{\gamma_j} \pm z_{1-\alpha/2}\sqrt{I^{-1}(\tilde{\boldsymbol{\gamma}})_{jj}/n}. \tag{5.14}$$

by ignoring both $\tilde{D}_{jj}$ and $\tilde{b}_{nj}$ since $\tilde{b}_{nj} = o_p(1)$. A $100(1-\alpha)\%$ confidence region for $\boldsymbol{\gamma}_k$ is given by

$$\left\{\boldsymbol{\gamma} \in \mathbb{R}^{m_k} : \; n\big[\tilde{\mathbf{D}}_k(\boldsymbol{\gamma} - \tilde{\boldsymbol{\gamma}}_k) + \tilde{\mathbf{b}}_{nk}\big]^T I(\tilde{\boldsymbol{\gamma}}_k)\big[\tilde{\mathbf{D}}_k(\boldsymbol{\gamma} - \tilde{\boldsymbol{\gamma}}_k) + \tilde{\mathbf{b}}_{nk}\big] \leq \chi^2_{m_k,1-\alpha}\right\},$$

which can be simplified as

$$\left\{\boldsymbol{\gamma} \in \mathbb{R}^{m_k} : \; n\big(\boldsymbol{\gamma} - \tilde{\boldsymbol{\gamma}}_k\big)^T I(\tilde{\boldsymbol{\gamma}}_k)\big(\boldsymbol{\gamma} - \tilde{\boldsymbol{\gamma}}_k\big) \leq \chi^2_{m_k,1-\alpha}\right\}. \tag{5.15}$$

5.5 Empirical Study

In this section, we conduct numerical simulations to evaluate the performance of gMIC and compare it with available group variable selection methods, including the gLasso, the gSCAD, and the gMCP. Two real data examples are also included to demonstrate the applications of gMIC. We implement the gMIC algorithm in R [60] and use the R package **grpreg** [7] for the computation of gLasso, gSCAD and gMCP. The penalization parameter λ of gLasso, gMCP, and gSCAD was selected by 10-fold cross-validation based on residual sum of squares for linear models and deviance for logistic and log-linear (Poisson) models. All simulations and real data analyses were done on a MAC laptop with a 1.6GHZ core i5 processor and a 4GB RAM. The R code for this chapter can be find in `https://github.com/liqun730/gMIC`.

5.5.1 Numerical Simulations

This section presents simulation studies in different linear models and generalized linear models designed to assess gMIC and compare it with other methods.

Experiment 1

We first consider the linear regression model $Y = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma^2 = 1$. The design matrix $X \in \mathbb{R}^{n \times p}$ contains $p = 12$ predictors that are evenly divided into 4 groups. X is generated from $\mathcal{N}(0, \Sigma)$ with $\Sigma = \Sigma_1 + \Sigma_2$. Matrix Σ_1 has diagonal elements equal to 1 and off-diagonal elements equal to $\rho_1 = 0.1$ or 0.3 indicating weak or moderate levels of dependencies between groups, and matrix Σ_2 is a block diagonal matrix with 4 blocks, each block being a 3×3 matrix with diagonal elements 1 and off-diagonal elements $\rho_2 = 0.7$ indicating strong within group dependencies. Two sample sizes $n = 100,\ 300$ were considered. For each $n,\ \sigma,\ \rho_1$ and ρ_2 combination, we fit three different models:

$$\textbf{Model A: } \boldsymbol{\beta} = (\underline{2, 2, 2},\ \underline{2, 2, 2},\ \underline{0, 0, 0},\ \underline{0, 0, 0})^T;$$

$$\textbf{Model B: } \boldsymbol{\beta} = (\underline{0.67, 0.67, 0.67},\ \underline{0.67, 0.67, 0.67},\ \underline{0, 0, 0},\ \underline{0, 0, 0})^T;$$

$$\textbf{Model C: } \boldsymbol{\beta} = (\underline{5, 5, 5},\ \underline{0.5, 0.5, 0.5},\ \underline{0, 0, 0},\ \underline{0, 0, 0})^T.$$

Model A corresponds to strong and balanced signals across groups of variables; Model B corresponds to balanced but weak signals; Model C corresponds to imbalanced signals across groups of variables, which increases the difficulty of detecting the group of variables with weaker signal (the second group). The parameter a was fixed at $a = 100$. The simulation was repeated 200 times and the average performance was recorded. We consider six metrics of performance, including **mean squared error (MSE)** defined as $(\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)^T \mathbb{E}(X^T X)(\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ where $\boldsymbol{\beta}_0$ is the true model parameter, **Size** defined as the number of non-zero groups, **False Positive (FP)**, **False Negative (FN)**, **Correct %** representing the percentage of correct model selection, and **Time** measuring the average time in seconds for a single run under each setting.

The performance for $\sigma = 1$ is shown in Table 5.1 below for $\rho_1 = 0.1$. The simulation results for $\rho_1 = 0.3$ does not vary much. Except for the gLasso which has comparably inferior

			Performance					
Method	Model	n	MSE	Size	FP	FN	Correct %	Time
gMIC	Model A	$n = 100$	0.083	2.015	0.015	0.000	0.985	0.052
		$n = 300$	0.061	2.000	0.000	0.000	1.000	0.053
	Model B	$n = 100$	0.088	2.010	0.010	0.000	0.990	0.051
		$n = 300$	0.059	2.025	0.025	0.000	0.975	0.049
	Model C	$n = 100$	0.088	2.030	0.030	0.000	0.970	0.048
		$n = 300$	0.061	2.000	0.000	0.000	1.000	0.050
gLasso	Model A	$n = 100$	0.230	2.270	0.270	0.000	0.740	0.063
		$n = 300$	0.106	2.105	0.105	0.000	0.900	0.079
	Model B	$n = 100$	0.240	2.205	0.205	0.000	0.810	0.064
		$n = 300$	0.103	2.045	0.045	0.000	0.955	0.084
	Model C	$n = 100$	0.224	2.280	0.280	0.000	0.750	0.064
		$n = 300$	0.105	2.075	0.075	0.000	0.930	0.078
gSCAD	Model A	$n = 100$	0.170	2.000	0.000	0.000	1.000	0.063
		$n = 300$	0.124	2.000	0.000	0.000	1.000	0.078
	Model B	$n = 100$	0.187	2.000	0.000	0.000	1.000	0.062
		$n = 300$	0.122	2.000	0.000	0.000	1.000	0.077
	Model C	$n = 100$	0.165	2.020	0.020	0.000	0.985	0.060
		$n = 300$	0.108	2.000	0.000	0.000	1.000	0.075
gMCP	Model A	$n = 100$	0.134	2.000	0.000	0.000	1.000	0.063
		$n = 300$	0.106	2.000	0.000	0.000	1.000	0.077
	Model B	$n = 100$	0.168	2.000	0.000	0.000	1.000	0.063
		$n = 300$	0.117	2.000	0.000	0.000	1.000	0.081
	Model C	$n = 100$	0.187	2.015	0.015	0.000	0.990	0.060
		$n = 300$	0.126	2.000	0.000	0.000	1.000	0.074

Table 5.1: Experiment 1: Comparison of gMIC with other methods with $\sigma = 1$, $a = 100$, and the cross-group correlation $\rho_1 = 0.1$.

performance when the sample size is small, all other methods do very well in selecting variable groups, with similar performances. Note that the gMIC seems to win out in terms of MSE. This indicates that gSCAD and gMCP apply extra or unnecessary shrinkage to nonzero estimates. Also, the gMIC is faster than other methods, a result from the fact that no selection of the tuning parameter is involved in the computation of gMIC.

To check the stability of the gMIC with respect to a_n, we simulated data from each model with $\rho_1 = 0.1$, $n = 200,\ 300,\ 400,\ 500$ and $\sigma^2 = 1,\ 4$. For each simulated data set, the gMIC is applied with $a = (25,\ 50,\ 75,\ 100, \ldots,\ 500)$. We plot the percentage of correct model selection against a. Figure 5.3 below, , showing such plots for Model C, indicates that the gMIC performance is insensitive to the choice of a. We omit the results for model A and B here for simplicity as they lead to the same conclusion.

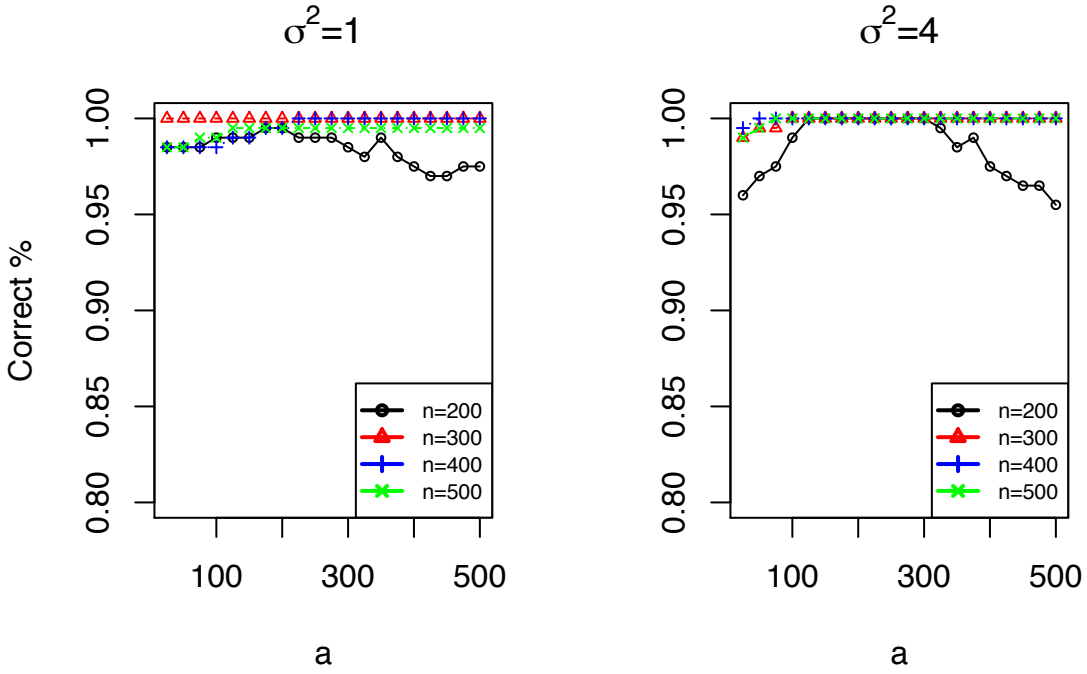


Figure 5.3: Experiment 1: Performance of the gMIC for model A with different choices of a on the percentage of correct model selection.

Experiment 2

To test the performance of gMIC when strong cross-group correlations are present, we follow Experiment 1 from [87]. First, the latent variables $Z_1, Z_2, \ldots, Z_{15}$ were generated from the multivariate normal distribution $N(0, \Sigma)$ where $\Sigma_{ij} = 0.5^{|i-j|}$. Then define X_i to be 0 if $z_i < \Phi^{-1}(1/3)$; 2 if $\Phi^{-1}(1/3) < z_i < \Phi^{-1}(2/3)$; and 1 otherwise. The response variable Y is generated from the following model,

$$Y = 1.8I(X_1 = 1) - 1.2I(X_1 = 0) + I(X_3 = 1) + 0.5I(X_3 = 0) + I(X_5 = 1) + I(X_5 = 0) + \epsilon, \tag{5.16}$$

where $\epsilon \sim N(0, \sigma^2)$. The discretization of Z_i's imposes a natural group structure as the dummy variables $I(X_i = 0)$, $I(X_i = 1)$, and $I(X_i = 2)$ form a group of variables.

The sample sizes n were set at 200, 300, and 500. The variance σ^2 was set to the values that gave the signal-to-noise ratio (snr) of 2, and the approximation parameter a was set to 100. The simulation was repeated 200 times. Table 5.2 below is a summary of simulation results of gMIC and other methods.

From Table 5.2, the performance of gMIC is comparable to gSCAD and gMCP in terms of group variable selection, but consistently superior in terms of MSE and computing time. The gLasso performs poorly for this model compared to other methods. There may be two reasons for this. First, the gLasso introduces too much biases to the estimation, as can be seen from the MSE column. Besides this, another reason for the deteriorated performance of gLasso may be a result of the presence of strong correlations among groups of variables here. It is well-known that when two variables are strongly correlated, the Lasso tends to select one of them at random, which results in false negative selections, see [96]. To counter the effect of false negative selections that increases the residual sum of squares, the cross-validation is in favor of a small λ during parameter tuning, an observation that was also made in [25].

		Performance					
Method	n	MSE	Size	FP	FN	Correct %	Time
gMIC	n=200	0.217	2.885	0.005	0.120	0.880	0.093
	n=300	0.180	2.940	0.000	0.060	0.940	0.099
	n=500	0.127	2.995	0.000	0.005	0.995	0.108
gLasso	n=200	0.541	3.380	0.380	0.000	0.735	0.118
	n=300	0.414	3.255	0.255	0.000	0.805	0.128
	n=500	0.273	3.175	0.175	0.000	0.870	0.151
gSCAD	n=200	0.524	3.125	0.155	0.030	0.875	0.119
	n=300	0.385	3.040	0.040	0.000	0.965	0.129
	n=500	0.238	3.025	0.025	0.000	0.975	0.156
gMCP	n=200	0.454	2.960	0.010	0.050	0.945	0.118
	n=300	0.341	2.990	0.005	0.015	0.980	0.129
	n=500	0.224	2.995	0.000	0.005	0.995	0.153

Table 5.2: Experiment 2: A comparison of group MIC and other methods.

This in turn results in an increase in the false positive rate, as can be seen from the FP column of Table 5.2. The comparison of gMIC and other methods for other signal-to-noise ratios does not vary much, so we omit these results here.

Again, we check the stability of gMIC for model (5.16) to the choice of parameter a. As shown in Figure 5.4 below, the performance of gMIC does not vary much across a wide range of a.

Experiment 3

In this subsection, we conduct simulations for the logistic regression model,

$$y|\boldsymbol{x}, \boldsymbol{\beta} \sim \text{Binomial}(p|\boldsymbol{x}, \boldsymbol{\beta}), \text{ where } \text{logit}(p|\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{x}\boldsymbol{\beta},$$

and the Poisson model,

$$y|\boldsymbol{x}, \boldsymbol{\beta} \sim \text{Poisson}(\exp(\boldsymbol{x}\boldsymbol{\beta})),$$

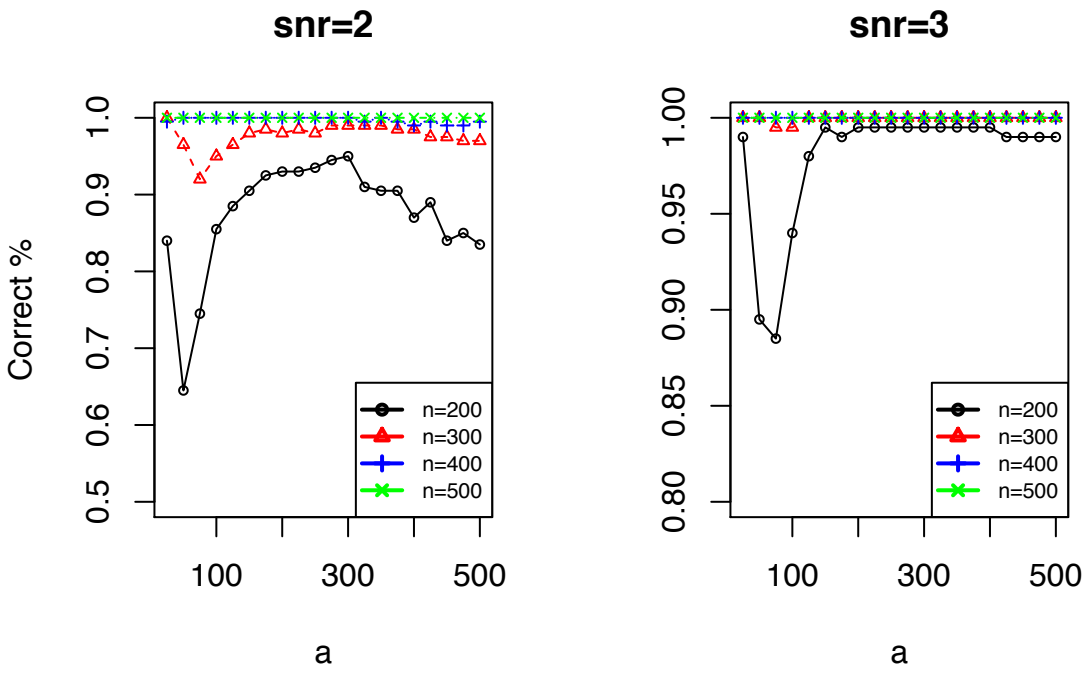


Figure 5.4: Experiment 2: performance of the gMIC with different choices of a on the percentage of correct model selection with different signal-to-noise ratios (snr).

where $\beta = (\underline{1,1,1}, \underline{0.5,0.5,0.5}, \underline{0,0,0}, \underline{0,0,0}, \underline{0,0,0})^T \in \mathbb{R}^{15}$. We split the parameter vector $\boldsymbol{\beta}$ evenly into 5 groups. The design matrix X is generated from

$$X \sim N(0, \Sigma) \text{ with } \Sigma = \Sigma_1 + \Sigma_2$$

where Σ_1 is the matrix with diagonal elements equal to 1 and off-diagonal elements equal to $\rho_1 = 0.1$, and Σ_2 is a block diagonal matrix with ten blocks, where each block is 3×3 matrix with diagonal elements 1 and off-diagonal elements $\rho_2 = 0.6$. The covariance matrix Σ indicates that there are weak correlations ($\rho_1 = 0.1$) between covariates from different groups and relatively strong correlations ($\rho_2 = 0.6$) among covariates in the same group. The sample size is chosen at $n = 300,\ 500$. The performance of gMIC is evaluated at $a = 100$ for both logistic regression and Poisson regression models for each n. The simulation was repeated

200 times and the results are summarized in Table 5.3 and Table 5.4 below.

		Performance					
Method	n	MSE	Size	FP	FN	Correct %	Time
gMIC	n=300	0.146	2.000	0.000	0.000	1.000	0.164
	n=500	0.133	2.000	0.000	0.000	1.000	0.165
gLasso	n=300	2.936	2.505	0.505	0.000	0.645	0.300
	n=500	2.408	2.330	0.330	0 .000	0.740	0.388
gSCAD	n=300	0.852	2.000	0.000	0.000	1.000	0.182
	n=500	0.441	2.000	0.000	0.000	1.000	0.247
gMCP	n=300	0.936	1.995	0.000	0.005	0.995	0.184
	n=500	0.467	2.000	0.000	0.000	1.000	0.250

Table 5.3: Experiment 3: Comparison of gMIC and other methods for logistic regression model.

		Performance					
Method	n	MSE	Size	FP	FN	Correct %	Time
gMIC	n=300	0.091	2.000	0.000	0.000	1.000	0.135
	n=500	0.081	2.000	0.000	0.000	1.000	0.150
gLasso	n=300	3.104	2.705	0.705	0.000	0.520	0.203
	n=500	2.132	2.525	0.525	0.000	0.595	0.278
gSCAD	n=300	0.344	2.000	0.000	0.000	1.000	0.196
	n=500	0.205	2.000	0.000	0.000	1.000	0.237
gMCP	n=300	0.357	2.000	0.000	0.000	1.000	0.205
	n=500	0.255	2.000	0.000	0.000	1.000	0.221

Table 5.4: Experiment 3: Comparison of gMIC and other methods for Poisson regression model.

The gMIC, gSCAD, and gMCP work exceptionally well for the logistic and Poisson models, with correct models selected for almost all of the 200 repetitions. However, gMIC still wins out in both ME and computing time. The notably inferior performance of gLasso may still be the result of introducing too much bias to the estimation and the presence of cross-group correlation. Among all the methods, the gMIC has significantly smaller MSE

and faster computational time. We comment that, similar to Experiment 1 and Experiment 2, the performance of gMIC is still stable across a wide range of a for the logistic regression models and Poisson regression models.

Inference of $\boldsymbol{\beta}$ via $\boldsymbol{\gamma}$

In this subsection, we conduct inference of $\boldsymbol{\beta}$ via $\boldsymbol{\gamma}$. Since we are considering group variable selection, we report the p-value for testing an entire group of variables instead of a single variable, i.e., the test is in the form of $H_0: \boldsymbol{\beta}_k = 0$ v.s. $H_1: \boldsymbol{\beta}_k \neq 0$, where the p-value is computed according to (5.13). Further, we compute the confidence region for each group of variables according to (5.15) and report its empirical coverage probability. We present the inference results for the linear model in Experiment 2 and the logistic regression model in Experiment 3. The size of the tests were fixed at $\alpha = 0.05$ and the 95% confidence regions were computed. For Experiment 2, we consider $n = 100,\ 200,\ 300$ and signal-to-noise ratios $snr = 0.5,\ 1$. For the logistic regression model in Experiment 3, we consider $n = 200,\ 300,\ 400$, and the "strong" model $\boldsymbol{\beta} = (\underline{1, 1, 1}, \underline{0.5, 0.5, 0.5}, \underline{0, 0, 0}, \underline{0, 0, 0}, \underline{0, 0, 0})^T \in \mathbb{R}^{15}$ and the "weak" model $\boldsymbol{\beta} = (\underline{0.33, 0.33, 0.33}, \underline{0.17, 0.17, 0.17}, \underline{0, 0, 0}, \underline{0, 0, 0}, \underline{0, 0, 0})^T \in \mathbb{R}^{15}$. For both models, the parameter a were fixed at $a = 100$. The inference results based on $1,000$ repetitions are summarized in Tables 5.5 and 5.6. To save space, we omit X_6 to X_{15} and only include the empirical coverages for X_1 to X_5 in Table 5.5. The empirical coverages of X_6 to X_{15} are all around the nominal 95% coverage, similar to those of X_1 to X_5. Empirical power is based on the p-value (5.13) and empirical coverage is based on the confidence region (5.15).

		Empirical Power			Empirical Coverage				
n	SNR	X_1	X_3	X_5	X_1	X_2	X_3	X_4	X_5
n=100	0.5	0.595	0.110	0.154	0.937	0.956	0.951	0.930	0.940
	1	0.994	0.312	0.465	0.943	0.937	0.942	0.951	0.940
n=200	0.5	0.922	0.223	0.287	0.941	0.949	0.946	0.932	0.939
	1	1.000	0.646	0.828	0.948	0.929	0.926	0.943	0.932
n=300	0.5	0.988	0.290	0.437	0.937	0.941	0.953	0.945	0.956
	1	1.000	0.847	0.959	0.948	0.938	0.929	0.936	0.960

Table 5.5: Experiment 2: Inference of $\boldsymbol{\beta}$ via $\boldsymbol{\gamma}$.

		Empirical Power		Empirical Coverage				
n	Model	$\boldsymbol{\gamma}_1$	$\boldsymbol{\gamma}_2$	$\boldsymbol{\gamma}_1$	$\boldsymbol{\gamma}_2$	$\boldsymbol{\gamma}_3$	$\boldsymbol{\gamma}_4$	$\boldsymbol{\gamma}_5$
n=200	strong	1.000	1.000	0.969	0.964	0.953	0.954	0.955
	weak	1.000	0.690	0.962	0.726	0.966	0.961	0.961
n=300	strong	1.000	1.000	0.958	0.962	0.956	0.966	0.954
	weak	1.000	0.814	0.958	0.779	0.981	0.983	0.973
n=400	strong	1.000	1.000	0.956	0.956	0.961	0.960	0.968
	weak	1.000	0.889	0.963	0.860	0.992	0.994	0.993

Table 5.6: Experiment 3: Inference of $\boldsymbol{\beta}$ via $\boldsymbol{\gamma}$ for the logistic regression model.

Summary of the Simulation Results

From Experiments 1–3 we can see that the gMIC produces comparable performances compared to the gMCP and gSCAD in terms of model selection accuracy. Compared to other methods, the gMIC generates smaller ME's, an indication that the gMIC penalty introduces less bias to the estimation compared to the MCP or SCAD penalty. The gMIC is not sensitive to the choice of parameter a, as its performance stays stable for a wide range of a. In practice, we find that setting $a = \min(n, 100)$ typically yields good performance.

The gMIC is also faster than other methods. The computational advantage of gMIC comes from the fact that it does not require parameter tuning for λ. We emphasize that the gMIC code was written in R with no specific implementation level optimization, while the gLasso, gSCAD, and gMCP were implemented in the R package **grpreg** with intensive usage of C++. So the computational advantage of gMIC over other methods may not be

fully uncovered here. The *grpreg* package also used a fast group descent algorithm proposed in [8] to accelerate the computation. We point out that this group descent algorithm could also potentially be used for gMIC, which could be a future research avenue.

5.5.2 Real Data Examples

We analyze the `mpg` dataset from the UCI Machine Learning Repository [44] and the `birthwt` dataset from the R package **MASS** [70]. Either dataset has categorical features which induce naturally grouped dummy variables. A linear model and a logistic regression model were used for the mpg dataset and the birthweight dataset separately. The gLasso, gSCAD, gMCP, and gMIC were used to conduct model selection for both datasets. For model interpretability concern, an intercept term is always included and unpenalized in all these approaches.

The mpg Data

The `mpg` dataset was first used in the 1983 American Statistical Association Exposition for testing several graphical analysis packages, and was later used in [59] for the study of city-cycle fuel consumption in miles per gallon. The dataset consists of 395 samples and 7 covariates, including 6 numerical variables (`cylinders, displacement, horsepower, weight, acceleration, model year`) and 1 categorical variable (`origin`, 3 categories). The categorical variable origin introduces two dummy variables `origin2` and `origin3`, which form a group of variables. The response is the `mpg`, a continuous variable indicating the fuel efficiency of a car in term of miles-per-gallon. We generate dummy variables for the categorical variable `origin` and fit a group penalized linear regression with gLasso, gSCAD, gMCP, and gMIC. For gMIC, the standard error of the estimation $\hat{\boldsymbol{\beta}}$ was obtained from Theorem 5.3.1 (ii) and a single p-value is computed for each group of variables according to (5.13) and is hence reported only once for origin2 and origin3. Table 5.7 summarizes the results. We put NA's in the `Intercept` row since the intercept is not a part of the variable selection process

	Full model		gMIC					
	$\hat{\boldsymbol{\beta}}$	SE	$\hat{\boldsymbol{\beta}}$	SE	p-value	gLasso	gSCAD	gMCP
`Intercept`	23.450	0.166	23.450	0.166	0.000	NA	NA	NA
`cylinders`	−0.347	0.569	0.000		0.990		✓	✓
`displacement`	2.036	0.771	0.000		1.000			
`horsepower`	0.309	0.202	0.000		1.000			
`weight`	−6.084	0.496	−5.649	0.495	0.000	✓	✓	
`acceleration`	0.433	0.213	0.000		1.000			
`year`	2.936	0.186	2.771	0.182	0.000	✓	✓	✓
`origin2`	1.019	0.212	0.776	0.170	0.000	✓		
`origin3`	1.063	0.214	0.879	0.210				

Table 5.7: A comparison of model selection between gMIC, gLasso, gSCAD, and gMCP for the mpg data.

as we always include an intercept for the model.

The gMIC and gLasso have the same selected model while the gSCAD and gMCP failed to select the `origin`. Two sub-models with `cylinders+weight+year+origin` or `cylinder+year+origin` were fit, and in both of the submodels the origin appeared to be highly significant. A submodels with `year+weight+cylinder` was also considered, in which the cylinder has a p-value over 0.7 while `weight` is highly significant. These observations provided evidence that `weight` and `origin`, instead of `cylinders`, should have been selected by the gSCAD and gMCP. The p-values obtained from (5.13) also support the selection result of gMIC.

The Birthweight Data

The birthweight data were collected at Baystate Medical Center, Springfield, Mass during 1986. The dataset used here was obtained from the R package **grpreg** and is a reparameterized version of the birthwt data in the R package **MASS**. The birthweight dataset contains 189 observations and 16 predictors. The response is the binary indicator of low birthweight (1-low and 0-normal) and the predictors describe features of mothers who were giving birth

	Full model		gMIC					
	$\hat{\boldsymbol{\beta}}$	SE	$\hat{\boldsymbol{\beta}}$	SE	p-value	gLasso	gSCAD	gMCP
`Intercept`	−0.953	0.182	−0.924	0.181	0.000	NA	NA	NA
`age1`	0.089	0.234	0.000	0.000	1.000			
`age2`	0.122	0.226	0.000	0.000				
`age3`	−0.296	0.252	0.000	0.000				
`weight1`	−0.656	0.276	−0.682	0.280	0.007	✓	✓	✓
`weight2`	−0.160	0.291	−0.196	−0.209				
`weight3`	−0.315	0.237	−0.294	0.205				
`race`	0.325	0.172	0.366	0.174	0.053	✓	✓	✓
`smoke`	0.326	0.172	0.328	0.166	0.065	✓	✓	✓
`ptl`	−0.131	0.176	0.000		1.000			
`height`	0.477	0.182	0.494	0.178	0.006	✓	✓	✓
`ui`	0.342	0.164	0.318	0.120	0.049	✓	✓	✓
`ftv2`	−0.078	0.186	0.000		1.000			
`ftv3`	0.214	0.178	0.000					

Table 5.8: A comparison of model selection between gMIC, gLasso, gSCAD, and gMCP for the birthwt data.

to the babies. The `age` variable is further categorized into four levels (0: under 18; 1: 18 to 25; 2: 25 to 30; 3: over 30). The 16 predictors include the categorical `age` with four levels, `weight` (polynomial terms with degree up to 3), `race` (black or white), `smoke`, `ptl` (previous premature labors), `height`, `ui` (presence of uterine irritability), and `ftv` (`ftv2`, `ftv3` indicating number of physician visits). A penalized logistic regression was fit to the data with gLasso, gSCAD, gMCP and gMIC. The model selection results are summarized in Table 5.8. Again, the selection and p-value calculation were conducted at a group level and hence we only present them for the first variable of each group. All methods selected the same model.

5.6 Conclusion and Discussion

We proposed the gMIC method for group variable selection. Derived as an approximation to BIC, the gMIC does not involve parameter tuning compared to existing group variable selection methods. We established the oracle property of the gMIC estimator. The simulation results indicate that the gMIC has comparable performance compared to other group variable selection methods while being computationally more efficient. The gMIC is insensitive to the approximation parameter a_n in theory, since the oracle property holds as long as a_n goes to infinity with n. This is further supported by the empirical study, as we found that the gMIC performance is stable for a wide range of a_n values in all the simulations. On these bases, we recommend fixing the approximation parameter a_n at $\min(100, n)$, in gMIC.

The gMIC can be thought of as a bridge between the penalization methods and information criteria as it inherits the benefits of both sides. On one hand, like the penalization methods, the gMIC has a continuous objective function that can be solved efficiently with existing optimization tools. On the other hand, as an approximation of the information criterion, the gMIC involves no parameter tuning, and is hence computationally advantageous compared to penalization methods where the penalization parameter needs to be tuned. With the fixed parameter a_n, The computational complexity involved in gMIC is essentially equivalent to that of a penalization method with a fixed penalty parameter. In this chapter, we considered variable selection with a fixed dimension p. Under high-dimensional settings, dimensionality grows with sample size. The gMIC can potentially be extended to high-dimensional model selection by approximating the Extended BIC in [12]. We leave this to our future work.

Finally, we would like to emphasize that, although we illustrate the application of gMIC in the generalized linear model, it shall be considered as a general approach for group variable selection. The gMIC can be naturally extended to other models, for example, the Cox PH

model [14] and especially, the quantile regression model. By replacing the penalty term in (1.3) with the gMIC penalty, we can conduct group variable selection for quantile regression too. This, combined with distributed optimization algorithms similar to Chapters 3 and 4 (we do emphasize that the ADMM is not able to solve the non-linear gMIC)can further reduce the computational time for large-scale quantile regression. This can be very interesting future work.

Chapter 6

Conclusion and Future Work

In this thesis, we provide solutions to distributed quantile regression analysis for big data. Specifically, we propose distributed and scalable numerical optimization algorithms for solving the quantile regression optimization based on the distributed ADMM algorithm. The QR-ADMM and QPADM algorithms proposed in Chapters 3 and 4 provide efficient numerical tools for penalized quantile regression optimization. They enjoy the favorable property of great scalability by enabling easy and flexible parallelization, and have shown to significantly outperform traditional linear programming methods even in non-parallel case. As an independent but closely related topic, Chapter 5 proposed a computationally efficient group variable selection method that is free of parameter tuning. The gMIC method, although proposed under the generalized linear model context, can be directly applied to quantile regression for feature selection.

The current thesis aims to address large-scale penalized quantile regression via the numerical approaches. An interesting future work is to derive divide-and-combine strategies for quantile regression aggregation. As current DC-based methods, including the aggregated estimating equation [45] and the distributed one-step estimator [29] focus on smooth problems, they can not be directly applied to the non-smooth quantile regression. An idea to extend

DC for the non-smooth quantile regression is through kernel smoothing. For example, [27] applied a specific form of kernel smoothing to the ℓ_1 loss function in median regression. We can apply similar kernel smoothing for the quantile check loss too. For high-dimensional cases, a type of high-dimensional distributed learning algorithm of special interest to us is the approximate Hessian methods in [36, 73]. In [36, 73], the global loss function is approximated with Taylor expansion where the global Hessian matrix is approximated by the local Hession. Based on this approximation, a distributed method can be derived. Essentially with smooth approximations, all these methods can be applied to quantile regression. We have done some preliminary theoretical analysis and simulation studies on this topic, and the results look very promising. Another important future work is to implement our algorithms in distributed computing frameworks. As can be seen in the simulation studies in Chapters 3 and 4, we currently implement our parallel algorithms in a non-parallel way by stacking up the parallel components of the computation. As a next step, we plan to have truly parallel implementations of these algorithms on distributed computing frameworks like the Spark [88].

Some other future work is related to the gMIC method. First, one may want to extend the gMIC to the varying p case when the dimension p grows with n. One may borrow ideas from the extended BIC in [12]. Second, as the current theoretical analysis of the gMIC method focuses on GLM, it would be beneficial to extend the theoretical framework to penalized quantile regression too. Finally, combining the gMIC with methods proposed in previous chapters can be a challenging while highly rewarding task. Since it is difficult to write the gMIC into the ADMM form, approaches proposed in Chapters 3 and 4 may be directly applied to solving the gMIC optimization. Other than the ADMM, one may need to resort to other distributed optimization tools for distributed gMIC optimization.

Chapter 7

Appendix

A.1. Proof of Theorem 3.1.1

Proof The proof of Theorem 3.1.1 mainly follows from the proof in [6]. We reproduce it here for completeness.

Without loss of generality, we assume that the columns of X are normalized. Define $\boldsymbol{e}^k := \boldsymbol{y} - X\boldsymbol{\beta}^k - \boldsymbol{r}^k$ and $V^k := \gamma^{-1}\|\boldsymbol{u}^k - \boldsymbol{u}^*\|_2^2 + \gamma\|\boldsymbol{r}^k - \boldsymbol{r}^*\|_2^2$. Our goal is to show that V^k is decreasing w.r.t. the iteration number k. First, since $(\boldsymbol{r}^*, \boldsymbol{\beta}^*, \boldsymbol{u}^*)$ is the saddle point of L_0, we have

$$L_0(\boldsymbol{r}^*, \boldsymbol{\beta}^*, \boldsymbol{u}^*) \leq L_0(\boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}, \boldsymbol{u}^*) \tag{7.1}$$

for any k. Rearranging (7.1) we have

$$p^* = \rho_\tau(\boldsymbol{r}^*) + P_\lambda(\boldsymbol{\beta}^*_{-\mathbf{0}}) \leq \rho_\tau(\boldsymbol{r}^{k+1}) + P_\lambda(\boldsymbol{\beta}^{\boldsymbol{k+1}}_{-\mathbf{0}}) + (\boldsymbol{u}^*)^T\boldsymbol{e}^{k+1} = p^{k+1} + (\boldsymbol{u}^*)^T\boldsymbol{e}^{k+1}. \tag{7.2}$$

Also, since $\boldsymbol{\beta}^{k+1}$ is the minimum at the $(k+1)$-th update, we have $0 \in \partial_{\boldsymbol{\beta}} L_\gamma(\boldsymbol{r}^k, \boldsymbol{\beta}^{k+1}, \boldsymbol{u}^k)$,

i.e. $0 \in \partial_\beta P_\lambda(\boldsymbol{\beta}_{\boldsymbol{-0}}^{\boldsymbol{k+1}}) - X^T\boldsymbol{u}^k + \gamma X^T(X\boldsymbol{\beta}^{k+1} + \boldsymbol{r}^k - \boldsymbol{y})$. Plugging in $\boldsymbol{u}^k = \boldsymbol{u}^{k+1} - \gamma\boldsymbol{e}^{k+1}$ and rearranging, we have $0 \in \partial_\beta P_\lambda(\boldsymbol{\beta_{-0}}^{k+1}) - X^T\left(\boldsymbol{u}^{k+1} + \gamma(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k)\right)$, from which we have the conclusion that

$$\boldsymbol{\beta}^{k+1} = \arg\min_\beta P_\lambda(\boldsymbol{\beta_{-0}}) - \left(\boldsymbol{u}^{k+1} + \gamma(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k)\right)^T X\boldsymbol{\beta}.$$

Remark. The above aguements cannot hold for a non-convex penalties since a non-convex function does not have a subdifferencital. So for non-convex penalized quantile regression we need different techniques to prove. ■

Similarly for $\boldsymbol{r}$, we have

$$\boldsymbol{r}^{k+1} = \arg\min_{\boldsymbol{r}} \rho_\tau(\boldsymbol{r}) - (\boldsymbol{u}^{k+1})^T\boldsymbol{r}. \tag{7.3}$$

Then we have

$$P_\lambda(\boldsymbol{\beta}_{\boldsymbol{-0}}^{\boldsymbol{k+1}}) - \left(\boldsymbol{u}^{k+1} + \gamma(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k)\right)^T X\boldsymbol{\beta}^{k+1} \leq P_\lambda(\boldsymbol{\beta}_{\boldsymbol{-0}}^{\boldsymbol{\star}}) - \left(\boldsymbol{u}^{k+1} + \gamma(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k)\right)^T X\boldsymbol{\beta}^*, \tag{7.4}$$

and also

$$\rho_\tau(\boldsymbol{r}^{k+1}) - (\boldsymbol{u}^{k+1})^T\boldsymbol{r}^{k+1} \leq \rho_\tau(\boldsymbol{r}^*) - (\boldsymbol{u}^{k+1})^T\boldsymbol{r}^*. \tag{7.5}$$

Adding (7.4) and (7.5) and using the fact that $\boldsymbol{y} - X\boldsymbol{\beta}^* - \boldsymbol{r}^* = 0$, we have

$$p^{k+1} - p^* \leq -(\boldsymbol{u}^{k+1})^T\boldsymbol{e}^{k+1} - \gamma(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k)^T(\boldsymbol{e}^{k+1} + \boldsymbol{r}^{k+1} - \boldsymbol{r}^*) \tag{7.6}$$

Adding up (7.2) and (7.6) and replacing $\boldsymbol{u}^{k+1}$ and $\boldsymbol{u}^{k+1} - \boldsymbol{u}^k$ by $\boldsymbol{u}^k$ and $\boldsymbol{u}^{k+1} - \boldsymbol{u}^* - (\boldsymbol{u}^k - \boldsymbol{u}^*)$, respectively, and after some simplification, we have

$$V^{k+1} - V^k \leq -\gamma\|\boldsymbol{e}^{\boldsymbol{k+1}}\|_2^2 - \gamma\|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|_2^2 + 2\gamma\boldsymbol{e}^{k+1}(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k). \tag{7.7}$$

Finally from (7.3), we have

$$\rho_\tau(\boldsymbol{r}^{k+1}) + (\boldsymbol{u}^{k+1})^T\boldsymbol{r}^{k+1} \leq \rho_\tau(\boldsymbol{r}^k) + (\boldsymbol{u}^{k+1})^T\boldsymbol{r}^k, \tag{7.8}$$

and

$$\rho_\tau(\boldsymbol{r}^k) + (\boldsymbol{u}^k)^T\boldsymbol{r}^k \leq \rho_\tau(\boldsymbol{r}^{k+1}) + (\boldsymbol{u}^k)^T\boldsymbol{r}^{k+1}. \tag{7.9}$$

Adding up (7.8) and (7.9), we have

$$(\boldsymbol{u}^{k+1} - \boldsymbol{u}^k)(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k) = \gamma\boldsymbol{e}^{k+1}(\boldsymbol{r}^{k+1} - \boldsymbol{r}^k) \leq 0.$$

So in (7.7), we actually have

$$V^{k+1} - V^k \leq -\gamma\|\boldsymbol{e}^{\boldsymbol{k+1}}\|_2^2 - \gamma\|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|_2^2. \tag{7.10}$$

Since $V^k > 0$, we have $\boldsymbol{e}^{k+1} = \boldsymbol{y} - X\boldsymbol{\beta}^{k+1} - \boldsymbol{r}^{k+1} \to 0$ and $\boldsymbol{r}^{k+1} - \boldsymbol{r}^k \to 0$.And since $V^k \leq V^0$, the sequences $\boldsymbol{u}^k$ and $\boldsymbol{r}^k$, $k = 1, 2, \ldots,$ are bounded. So from (7.6), we have the convergence of the objective funciton: $p^{k+1} \to p^*$. And also $\boldsymbol{r}^{k+1} \to \tilde{\boldsymbol{r}}^*$, $\boldsymbol{u}^{k+1} \to \tilde{\boldsymbol{u}}^*$, and $X\boldsymbol{\beta}^{k+1} \to \boldsymbol{y} - \tilde{\boldsymbol{r}}^*$ for some $\tilde{\boldsymbol{r}}^*$ and $\tilde{\boldsymbol{u}}^*$. If Assumption 3.1.3 holds, then the convergence of $X\boldsymbol{\beta}^{k+1}$ implies the convergence of $\boldsymbol{\beta}^{k+1}$: $\boldsymbol{\beta}^{k+1} \to \tilde{\boldsymbol{\beta}}^*$. And we have

$$\lim_{k\to\infty} p^{k+1} = \lim_{k\to\infty} \rho_\tau(\boldsymbol{r}^{k+1}) + P_\lambda(\boldsymbol{\beta}^{\boldsymbol{k+1}}_{\boldsymbol{-0}}) = \rho_\tau(\tilde{\boldsymbol{r}}^*) + P_\lambda(\tilde{\boldsymbol{\beta}}^{\boldsymbol{*}}_{\boldsymbol{-0}}) = p^*$$

and $\boldsymbol{y} - X\tilde{\boldsymbol{\beta}}^* = \tilde{\boldsymbol{r}}^*$. That is, $\tilde{\boldsymbol{r}}^*$ and $\tilde{\boldsymbol{\beta}}^*$ is a solution of (3.1). ■

A.2. Proof of Theorem 3.1.2

To prove Theorem 3.1.2, we need the following lemmas,

Lemma 1 *The Lagrangian $L(\boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}, \boldsymbol{u}^{k+1})$ is decreasing w.r.t. k, and we have*

$$L_\gamma(\boldsymbol{u}^{k+1}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^{k}, \boldsymbol{r}^{k}, \boldsymbol{\beta}^{k}) \leq (\frac{1}{4\delta^2\gamma} - \frac{\gamma}{2})\|\boldsymbol{r}^{k+1} - \boldsymbol{r}^{k}\|^2 \leq 0.$$

Proof From the optimality condition of the $\boldsymbol{r}$-update in (3.2), we have,

$$0 \in \partial\rho_\tau(\boldsymbol{r}^{k+1}) + \gamma\left(\boldsymbol{r}^{k+1} - \gamma^{-1}\boldsymbol{u}^k - \boldsymbol{y} + X\boldsymbol{\beta}^{k+1}\right),$$

where ∂ is the subdifferential of a nonsmooth function. Plugging in the $\boldsymbol{u}$-update, we have $\boldsymbol{u}^{k+1} \in \partial\rho_\tau(\boldsymbol{r}^{k+1})$.

Since Assumption 3.1.4 holds, when $|u_i^{k+1} - u_i^k| = 1$, r_i^{k+1} and r_i^k have different signs and so $|r_i^{k+1} - r_i^k| \geq 2\delta$; otherwise, $|u_i^{k+1} - u_i^k| = 0$. So we always have

$$\|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_2^2 \leq \frac{1}{4\delta^2}\|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|_2^2. \tag{7.11}$$

We then split the successive difference of augmented Lagrangian by

$$\begin{aligned} &L_\gamma(\boldsymbol{u}^{k+1}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^{k}, \boldsymbol{r}^{k}, \boldsymbol{\beta}^{k}) \\ =&L_\gamma(\boldsymbol{u}^{k+1}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^{k}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) + L_\gamma(\boldsymbol{u}^{k}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^{k}, \boldsymbol{r}^{k}, \boldsymbol{\beta}^{k}) \end{aligned} \tag{7.12}$$

The first term on the RHS of (7.12)

$$L_\gamma(\boldsymbol{u}^{k+1}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^{k}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) = (\boldsymbol{u}^{k+1} - \boldsymbol{u}^k)^T(\boldsymbol{y} - X\boldsymbol{\beta}^{k+1} + \boldsymbol{r}^{k+1}) = \gamma^{-1}\|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_2^2.$$

The second term on the RHS of (7.12) can be bounded by

$$\begin{aligned}
&L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^k) \\
=&L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^{k+1}) + L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^k) \\
\leq&L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^{k+1}) + 0 \\
\leq&\left\{ \langle \partial_{\boldsymbol{r}} L(\boldsymbol{u}^k, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^k), \boldsymbol{r}^{k+1} - \boldsymbol{r}^k \rangle - \frac{\gamma}{2} \|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|^2 \right\} = -\frac{\gamma}{2} \|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|^2.
\end{aligned}$$

where the last inequality comes from the fact that L is $\frac{\gamma}{2}$-strongly convex w.r.t $\boldsymbol{r}$. Combining the above two parts, when $\gamma > \frac{1}{\sqrt{2}\delta}$,

$$\begin{aligned}
&L_\gamma(\boldsymbol{u}^{k+1}, \boldsymbol{r}^{k+1}, \boldsymbol{\beta}^{k+1}) - L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^k) \leq \gamma^{-1} \|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_2^2 - \frac{\gamma}{2} \|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|^2 \\
&\leq (\frac{1}{4\delta^2 \gamma} - \frac{\gamma}{2}) \|\boldsymbol{r}^{k+1} - \boldsymbol{r}^k\|^2 \leq 0,
\end{aligned}$$

where the second last inequality comes from (7.11). Our conclusion then follows.

Remark. The relationship between $\boldsymbol{u}^{k+1} - \boldsymbol{u}^k$ and $\boldsymbol{r}^{k+1} - \boldsymbol{r}^k$ is crucial to establish the monotonicity of the Lagrangian. When function ρ_τ is Lipschitz continuous, this relationship can be achieved from the fact that $\boldsymbol{u} = \partial \rho_\tau(\boldsymbol{r})$. Since here $\partial \rho_\tau$ does not have Lipschitz continuity, Assumption 3.1.4 becomes necessary to establish (7.11). ■ ■

Lemma 2 $\lim_{k \to \infty} L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^k) = L^*$ *for some constant* L^*.

Proof Based on **Lemma 1**, we only need to prove that $L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^k)$ is bounded below.

$$\begin{aligned}
L_\gamma(\boldsymbol{u}^k, \boldsymbol{r}^k, \boldsymbol{\beta}^k) &= \rho_\tau(\boldsymbol{r}^k) + \gamma \boldsymbol{u}^k (\boldsymbol{y} - X\boldsymbol{\beta}^k - \boldsymbol{r}^k) + \frac{\gamma}{2} \|\boldsymbol{y} - X\boldsymbol{\beta}^k - \boldsymbol{r}^k\|^2 + P_\lambda(\boldsymbol{\beta_{-0}}) \\
&= \rho_\tau(\boldsymbol{r}^k) + P_\lambda(\boldsymbol{\beta_{-0}}) + (\boldsymbol{u}^k)^T (\boldsymbol{u}^k - \boldsymbol{u}^{k-1}) + \frac{\gamma}{2} \|\boldsymbol{u}^k - \boldsymbol{u}^{k-1}\|^2,
\end{aligned}$$

which is bounded below since $\boldsymbol{u}^k$ is uniformly bounded by $max(\tau, 1-\tau)$. ■

Now we are ready to prove Theorem 3.1.2.

Combining the conclusions of **Lemma 1** and **Lemma 2** we have $\boldsymbol{r}^{k+1} - \boldsymbol{r}^k \to 0$, and hence $X(\boldsymbol{\beta^k} - \boldsymbol{\beta^{k+1}}) = \boldsymbol{r}^{k+1} - \boldsymbol{r}^k \to 0$.

Let $(\boldsymbol{r^*}, \boldsymbol{\beta^*}, \boldsymbol{u}^*)$ be any cluster point of $(\boldsymbol{r^k}, \boldsymbol{\beta^k}, \boldsymbol{u}^k)$, then by the definition of $\boldsymbol{\beta}^k, \boldsymbol{r}^k, \boldsymbol{u}^k$, and taking limit w.r.t. k, we have

$$\begin{aligned} &\boldsymbol{y} - X\boldsymbol{\beta}^* = \boldsymbol{r}^*, \\ &0 \in \partial\rho_\tau(\boldsymbol{r}^*) + \boldsymbol{u}^*, \\ &\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \frac{\gamma}{2} \left\| \boldsymbol{u}^k + \boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{r}^{k+1} \right\|_2^2 + P_\lambda(\boldsymbol{\beta_{-0}}). \end{aligned} \tag{7.13}$$

That is, $(\boldsymbol{\beta}^*, \boldsymbol{r}^*, \boldsymbol{u}^*)$ is a stationary point of (3.1).

If Assumptions 3.1.2 and 3.1.3 also hold, then $\boldsymbol{r}^k \to \tilde{\boldsymbol{r}}^*$ and $\boldsymbol{\beta}^k \to \tilde{\boldsymbol{\beta}}^*$ for some $\tilde{\boldsymbol{r}}^*$ and $\tilde{\boldsymbol{\beta}}^*$, and $\boldsymbol{u}^k = \partial\rho_\tau(\boldsymbol{r}^k) \to \partial\rho_\tau(\tilde{\boldsymbol{r}}^k) := \tilde{\boldsymbol{u}}^k$. And again by the definition of $\boldsymbol{\beta}^k, \boldsymbol{r}^k, \boldsymbol{u}^k$, and taking limit w.r.t. k, we have (7.13) holds for $\tilde{\boldsymbol{r}}^*$, $\tilde{\boldsymbol{\beta}}^*$ and $\tilde{\boldsymbol{u}}^*$. That is, $(\tilde{\boldsymbol{\beta}}^*, \tilde{\boldsymbol{r}}^*, \tilde{\boldsymbol{u}}^*)$ is a stationary point of (3.1).

A.3. Proof of Lemma 5.3.1

We will use the following two matrix inequalities in Lemma 6.3.1.0, whose proofs are omitted.

Lemma 6.3.1.0 *Suppose that* $\mathbf{A} = (a_{ii'}) \in \mathbb{R}^{m\times m}$ *is a symmetric matrix and* $\mathbf{x} = (x_i) \in \mathbb{R}^m$ *is an* m*-dimensional vector.*

(i) If $|a_{ii'}| \le M_0$ *for* $i, i' = 1, \ldots, m$*, then* $\mathbf{x}^T\mathbf{A}\mathbf{x} \le mM_0\|\mathbf{x}\|^2$.

(ii) If $\mathbf{A} \succ 0$ *with eigenvalues* $d_1 < d_2 < \cdots < d_m$*, then* $\|\mathbf{A}\mathbf{x}\| \le d_m\|\mathbf{x}\|$.

We now prove Lemma 5.3.1.

Proof We omit (i) and prove (ii) by the inverse function theorem. This is because

$$\begin{aligned}\frac{dw}{d\boldsymbol{\beta}} &= \frac{d\boldsymbol{\gamma}}{d\boldsymbol{\beta}}\cdot\frac{dw}{d\boldsymbol{\gamma}} = \left(\frac{d\boldsymbol{\beta}}{d\boldsymbol{\gamma}}\right)^{-1}\cdot\frac{dw}{d\boldsymbol{\gamma}} \\ &= 2a(1-w^2)\left\{w\,Ivv_m + 2a(1-w^2)\,\boldsymbol{\gamma}\boldsymbol{\gamma}^T\right\}^{-1}\boldsymbol{\gamma}.\end{aligned}$$

The matrix $d\boldsymbol{\beta}/d\boldsymbol{\gamma}$ is not invertible at $\boldsymbol{\beta}=\boldsymbol{\gamma}=\mathbf{0}$. ■

A.4. Proof of Theorem 5.3.1

Proof *(i)* It suffices to show that, $\forall\varepsilon>0$, $\exists$ a large constant $C>0$ s.t.

$$\Pr\left\{\inf_{\|\mathbf{u}\|=C} Q_n(\boldsymbol{\beta}_0+\frac{\mathbf{u}}{\sqrt{n}}) > Q_n(\boldsymbol{\beta}_0)\right\} \geq 1-\varepsilon.$$

This implies the existence of a local minimum $\widetilde{\boldsymbol{\beta}}$ of $Q_n(\boldsymbol{\beta})$ within the ball $\{\boldsymbol{\beta}_0+\mathbf{u}/\sqrt{n}:\|\mathbf{u}\|\leq C\}$ and hence $\widetilde{\boldsymbol{\beta}}$ is $\sqrt{n}$-consistent for $\boldsymbol{\beta}_0$.

For notational convenience, we simply denote it as $w(\boldsymbol{\beta}_k)$ when $w(\boldsymbol{\gamma}_k)=\tanh(a\|\boldsymbol{\gamma}_k\|_2^2)$ is treated as a function of $\boldsymbol{\beta}_k$. Consider

$$\begin{aligned}D_n &= Q_n(\boldsymbol{\beta}_0+\frac{\mathbf{u}}{\sqrt{n}}) - Q_n(\boldsymbol{\beta}_0) \\ &= -2\left\{L_n(\boldsymbol{\beta}_0+\frac{\mathbf{u}}{\sqrt{n}}) - L_n(\boldsymbol{\beta}_0)\right\} + \ln(n)\sum_{k=1}^{K} m_k\left\{w(\boldsymbol{\beta}_{0k}+\frac{\mathbf{u}_k}{\sqrt{n}}) - w(\boldsymbol{\beta}_{0k})\right\} \\ &\geq -2\left\{L_n(\boldsymbol{\beta}_0+\frac{\mathbf{u}}{\sqrt{n}}) - L_n(\boldsymbol{\beta}_0)\right\} + \ln(n)\sum_{k=1}^{K_1} m_k\left\{w(\boldsymbol{\beta}_{0k}+\frac{\mathbf{u}_k}{\sqrt{n}}) - w(\boldsymbol{\beta}_{0k})\right\} \\ &= -\frac{2}{\sqrt{n}}\mathbf{u}^T\nabla L(\boldsymbol{\beta}_0) - \frac{1}{n}\mathbf{u}^T\nabla^2 L(\boldsymbol{\beta}_0)\mathbf{u}\{1+o_p(1)\} + \frac{\ln(n)}{\sqrt{n}}\sum_{k=1}^{K_1} m_k\mathbf{u}_k^T\frac{dw(\boldsymbol{\beta}_{0k})}{d\boldsymbol{\beta}_k}\{1+o(1)\} \\ &= \text{I} + \text{II} + \text{III} \qquad (7.14)\end{aligned}$$

The third step is obtained since $\sum_{k=K_1+1}^{K} w(\boldsymbol{\beta}_{0k})=0$ and $\sum_{k=K_1+1}^{K} w(\boldsymbol{\beta}_{0k}+\mathbf{u_k}/\sqrt{n})\geq 0$.

Under regularity conditions (A)–(C) in Lemma 5.3.1, $\nabla L(\boldsymbol{\beta}_0)$ is $O_p(\sqrt{n})$. Thus the term I is $O_p(\|\mathbf{u}\|)$. Since $\nabla^2 L(\boldsymbol{\beta}_0) = -n\mathbf{I}$, the term II equals

$$-\frac{1}{n}\mathbf{u}^T(-n\mathbf{I})\mathbf{u}\{1+o_p(1)\} = O_p(\mathbf{u}^T\mathbf{I}\mathbf{u}),$$

which dominates the term I if $\|\mathbf{u}\| = C$ is sufficiently large. We next bound term III and show that it is also dominated by term II. By Lemma 5.3.1(ii),

$$\text{III} = \frac{\ln(n)}{\sqrt{n}} \sum_{k=1}^{K_1} m_k, \mathbf{u}_k^T \left\{2a_n(1-w_k^2)\mathbf{A_k}^{-1}\boldsymbol{\gamma}_{0k}\right\}\{1+o(1)\}$$

where matrix $\mathbf{A}_k = w_k\, I v_m + 2a_n(1-w_k^2)\,\boldsymbol{\gamma}_{0k}\boldsymbol{\gamma}_{0k}^T \succeq 0$. Let $\sigma_{\min}(\mathbf{A})$ denote the minimum eigenvalue of a symmetric matrix $\mathbf{A}$. By the Cauchy-Schwarz inequality, III is bounded by

$$\begin{aligned}
& \frac{2a_n\ln(n)}{\sqrt{n}} \sum_{k=1}^{K_1} m_k(1-w_k^2)\|\mathbf{u}_k\| \cdot \|\mathbf{A_k}^{-1}\boldsymbol{\gamma}_{0k}\| \\
\leq\ & \frac{2a_n\ln(n)}{\sqrt{n}} \sum_{k=1}^{K_1} \frac{m_k(1-w_k^2)}{\sigma_{\min}(\mathbf{A_k})}\|\mathbf{u}_k\| \cdot \|\boldsymbol{\gamma}_{0k}\| \text{ (by Lemma 0)}, \\
\leq\ & \frac{2a_n\ln(n)}{\sqrt{n}} \sum_{k=1}^{K_1} \frac{m_k(1-w_k^2)}{\sigma_{\min}(w_k\, I v_m)}\|\mathbf{u}_k\| \cdot \|\boldsymbol{\gamma}_{0k}\| \\
=\ & \frac{2\ln(n)}{\sqrt{n}} \sum_{k=1}^{K_1} \frac{m_k a_n(1-w_k^2)}{w_k}\|\mathbf{u}_k\| \cdot \|\boldsymbol{\gamma}_{0k}\|.
\end{aligned}$$

The third inequality holds since $\sigma_{\min}(\mathbf{A_k}) \geq \sigma_{\min}(w_k\, I v_m)$ as an application of the variational theorem. For for $1 \leq k \leq K_1$, $\|\|\boldsymbol{\gamma}_{0k}\|$ is bounded from below. Hence

$$\frac{a_n(1-w_k^2)}{w_k} = \frac{a_n\,\mathrm{sech}(a_n\,\|\boldsymbol{\gamma}_{0k}\|^2)}{\tanh(a_n\|\boldsymbol{\gamma}_{0k}\|^2)} = \frac{2a_n}{\exp(a_n\|\boldsymbol{\gamma}_{0k}\|^2) - \exp(-a_n\|\boldsymbol{\gamma}_{0k}\|^2)},$$

which is bounded if either a_n is set as a constant or $\lim_{n\to\infty} a_n = \infty$. As a result, term III is of order $O(\ln(n)\|\mathbf{u}\|/\sqrt{n})$ and dominated by term II as well. In view of $\mathbf{I} \succ 0$ under regularity

conditions, term II > 0 and hence $D_n > 0$ with probability tending to one as $n \to \infty$. This completes the proof of (i) of Theorem 5.3.1.

(ii) It suffices to show that, for any $\sqrt{n}$-consistent $\boldsymbol{\beta} = (\boldsymbol{\beta}_{(1)}^T, \boldsymbol{\beta}_{(0)}^T)^T$ such that $\|\boldsymbol{\beta}_{(1)} - \boldsymbol{\beta}_{0(1)}\| = O_p(1/\sqrt{n})$ and $\|\boldsymbol{\beta}_{(0)}\| = O_p(1/\sqrt{n})$, $\partial Q_n(\boldsymbol{\beta})/\partial\beta_j$ has the sign as β_j for any component β_j of $\boldsymbol{\beta}_{(0)}$ with probability tending to 1 as $n \to \infty$. In this case, Q_n decreases when $\beta_j < 0$ and increases when $\beta_j > 0$, hence reaches a local minimum at $\beta_j = 0$.

Suppose that β_j is in the k-th group, for $k \in \{K_1 + 1, \ldots, K\}$. The quantity $\partial Q_n(\boldsymbol{\beta})/\partial\beta_j$ is

$$\frac{\partial Q_n(\boldsymbol{\beta})}{\partial \beta_j} \;=\; -\,2\,\frac{\partial L_n(\boldsymbol{\beta})}{\partial \beta_j} + \ln(n)\, m_k\, \frac{\partial w_k}{\partial \beta_j} \;=\; \mathrm{I} + \mathrm{II}$$

for $j = (p_1{+}1), \ldots, p$ when evaluated at $\boldsymbol{\beta}$. Note that $\beta_j = O_p(1/\sqrt{n})$ yet $\beta_j \neq 0$ for $\beta_j \in \boldsymbol{\beta}_{(0)}$. By standard arguments (see [19]) and using the fact that $\|\boldsymbol{\beta} - \boldsymbol{\beta}_0\| = O_p(1/\sqrt{n})$, the first term I is of order $O_p(\sqrt{n})$ under the regularity conditions. It remains to show that term II is of higher order than $O_p(\sqrt{n})$ and has the same sign as $\mathrm{sign}(\beta_j)$. To this end, consider

$$\frac{\partial w_k}{\partial \beta_j} \;=\; \frac{2a_n\gamma_j(1 - w_k^2)}{w_k + 2a_n\gamma_j^2(1 - w_k^2)} \;=\; \frac{4a_n\gamma_j}{\exp(a_n\|\boldsymbol{\gamma}_k\|^2) + \exp(-a_n\|\boldsymbol{\gamma}_k\|^2) + 4a_n\gamma_j^2} \tag{7.15}$$

by Lemma 5.3.1*(ii)*. Since $a_n = O(n)$ and $\gamma_j w_k = \beta_j = O_p(1/\sqrt{n})$, it follows that $\gamma_j = O_p(1/\sqrt{n})$ and $a_n\gamma_j^2 = O_p(1)$. Thus $\partial w_k/\partial\beta_j$ in (7.15) is $O_p(1/\gamma_j) = O_p(\sqrt{n})$ and hence term II is $O_p\{\ln(n)\sqrt{n}\}$, dominating term I. Besides, the sign of term II is the same as $\mathrm{sign}(\gamma_j) = \mathrm{sign}(\beta_j)$ as seen in (7.15).

With slight abuse of notations, define

$$\widehat{\boldsymbol{\beta}}_{(1)} = \underset{\{\boldsymbol{\beta}:\, \boldsymbol{\beta}_{(0)}=\mathbf{0}\}}{\arg\max}\; L_n(\boldsymbol{\beta})$$

to be the oracle estimator. We prove the desired asymptotic normality of $\widetilde{\boldsymbol{\beta}}_{(1)}$ by showing

that

$$\|\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)}\| = o_p(1/\sqrt{n}) \tag{7.16}$$

and then making an appeal to Slutsky's theorem.

First of all, since both $\widetilde{\boldsymbol{\beta}}_{(1)}$ and $\widehat{\boldsymbol{\beta}}_{(1)}$ are $\sqrt{n}$-consistent to $\boldsymbol{\beta}_{0(1)}$, it follows immediately by the triangular inequality that $\|\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)}\| = O_p(1/\sqrt{n})$. By the definition of $\widetilde{\boldsymbol{\beta}}_{(1)}$, it must satisfy

$$\frac{\ln(n)}{2} \frac{\partial \sum_{k=1}^{K} m_k w_k(\tilde{\boldsymbol{\beta}}_k)}{\partial \beta_j} = \nabla_j L_n(\widetilde{\boldsymbol{\beta}}_{(1)}) \tag{7.17}$$

for $j = 1, \ldots, p_1$, where $\nabla_j L_n(\widetilde{\boldsymbol{\beta}}_{(1)}) = \partial L_n / \partial \beta_j$ evaluated at $\widetilde{\boldsymbol{\beta}}_{(1)}$. Expanding the RHS of (7.17) at $\widehat{\boldsymbol{\beta}}_{(1)}$ yields

$$\begin{aligned} \frac{\ln(n)}{2} \frac{\partial \sum_{k=1}^{K_1} m_k w_k(\widetilde{\boldsymbol{\beta}}_k)}{\partial \beta_j} &= \nabla_j L_n(\widehat{\boldsymbol{\beta}}_{(1)}) + (\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)})^T \boldsymbol{\nabla}_j^2 L_n(\widehat{\boldsymbol{\beta}}_{(1)}) \\ &\quad + (\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)})^T \boldsymbol{\nabla}_j^3 L_n(\boldsymbol{x_i})(\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)}), \end{aligned} \tag{7.18}$$

where $\nabla_j L_n(\widehat{\boldsymbol{\beta}}_{(1)}) = 0$ by the definition of $\widehat{\boldsymbol{\beta}}_{(1)}$; $\boldsymbol{\nabla}_j^2 L_n(\widehat{\boldsymbol{\beta}}_{(1)}) \in \mathbb{R}^{p_1}$ is the j-th column vector of the Hessian matrix (or the negative observed total Fisher information matrix $\mathbf{I}_{1n}$) evaluated at $\widehat{\boldsymbol{\beta}}_{(1)}$; and

$$\boldsymbol{\nabla}_j^3 L_n(\boldsymbol{x_i}) = \left(\frac{\partial^3 L_n(\boldsymbol{x_i})}{\partial \beta_j \partial \beta_{j'} \partial \beta_{j''}} \right)_{j', j''=1,\ldots,p_1} \in \mathbf{R}^{p_1 \times p_1}.$$

for some $\boldsymbol{x_i}$ falling between $\widetilde{\boldsymbol{\beta}}_{(1)}$ and $\widehat{\boldsymbol{\beta}}_{(1)}$. Note that $\mathbf{I}_{1n}/n \xrightarrow{p} \mathbf{I}_1$.

Define $\mathbf{v} = (v_j)_{j=1}^{p_1} \in \mathbf{R}^{p_1}$ with $v_j = \sum_{k=1}^{K} m_k \partial w_k / \partial \beta_j$ for the LHS of (7.18). Define vector $\mathbf{r} = (r_j) \in \mathbf{R}^{p_1}$ such that r_j equals the remainder term on the RHS of (7.18). One regularity condition states that every third derivative element in $\boldsymbol{\nabla}^3 L_n / n$ is bounded in probability. By Lemma 0, we have

$$|r_j|/n \leq C \cdot \|\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)}\|^2, \tag{7.19}$$

for some constant C w.p.t.1 as $n \to \infty$. In matrix form, (7.18) now becomes

$$\begin{aligned} \widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)} &= \frac{1}{n}\left(\frac{\mathbf{I}_{1n}}{n}\right)^{-1}\mathbf{r} - \frac{\ln(n)}{2n}\left(\frac{\mathbf{I}_{1n}}{n}\right)^{-1}\mathbf{v} \\ \Rightarrow \quad \|\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)}\| &\leq \left\|\left(\frac{\mathbf{I}_{1n}}{n}\right)^{-1}\frac{\mathbf{r}}{n}\right\| + \frac{\ln(n)}{2n}\left\|\left(\frac{\mathbf{I}_{1n}}{n}\right)^{-1}\mathbf{v}\right\| = \mathrm{I} + \mathrm{II}. \end{aligned}$$

With some algebra, it can be shown that the first term I is $O_p(\|\widetilde{\boldsymbol{\beta}}_{(1)} - \widehat{\boldsymbol{\beta}}_{(1)}\|^2) = O_p(1/n)$ and dominates the second term II, which is of order $O_p\{\ln(n)/\exp(n\max_k \|\boldsymbol{\beta}_{0k}^2\|^2)\}$. Thus, the proof is completed ■

A.5. Proof of Theorem 5.4.1

Proof First, note that $\boldsymbol{\gamma} - \boldsymbol{\gamma}w(\boldsymbol{\gamma}) = \boldsymbol{\gamma}\{1 - \tanh(a_n\|\boldsymbol{\gamma}\|^2)\} = 2\boldsymbol{\gamma}/\{\exp(2a_n\|\boldsymbol{\gamma}\|^2) + 1\}$. It follows that $|\|\boldsymbol{\gamma}_{0k}\| - \|\boldsymbol{\beta}_{0k}\|| = O\{\exp(-2a_n\|\boldsymbol{\gamma}_{0k}\|)\}$ for $\boldsymbol{\gamma}_{0k} \neq \mathbf{0}$ and 0 otherwise. Since the function $\boldsymbol{\beta} = \boldsymbol{\gamma}w(\boldsymbol{\gamma})$ is continuous and so is its inverse, it follows by the continuous mapping theorem that $\tilde{\boldsymbol{\gamma}} \xrightarrow{p} \boldsymbol{\gamma}_0$.

To explore the asymptotic normality, we consider $\tilde{\boldsymbol{\gamma}}$ as a local minimizer of $Q_n(\cdot)$. Since the objective function $Q_n(\boldsymbol{\gamma})$ is smooth in $\boldsymbol{\gamma}$, $\tilde{\boldsymbol{\gamma}}$ satisfies the first-order necessary condition $\partial Q_n(\tilde{\boldsymbol{\gamma}})/\partial\boldsymbol{\gamma} = \mathbf{0}$, which leads to

$$\begin{aligned} & -2\frac{\partial L(\tilde{\boldsymbol{\beta}})}{\partial\boldsymbol{\beta}}\frac{\partial\boldsymbol{\beta}(\tilde{\boldsymbol{\gamma}})}{\partial\boldsymbol{\gamma}} + \ln(n)\frac{\partial\sum_k w(\tilde{\boldsymbol{\gamma}}_k)}{\partial\boldsymbol{\gamma}} = \mathbf{0} \\ \Longleftrightarrow \quad & \dot{L}(\tilde{\boldsymbol{\beta}})\mathbf{diag}\{w_k I_{m_k} + 2a_n(1 - w_k^2)\tilde{\boldsymbol{\gamma}}_k\tilde{\boldsymbol{\gamma}}_k^T\} = \frac{\ln(n)}{2}\left(\frac{\partial w(\tilde{\boldsymbol{\gamma}}_k)}{\partial\boldsymbol{\gamma}_k}\right)_{k=1}^{K} \\ \Longleftrightarrow \quad & \dot{L}(\tilde{\boldsymbol{\beta}}) = \frac{\ln(n)}{2}\{2a_n(1 - w_k^2)D_k^{-1}(\tilde{\boldsymbol{\gamma}}_k)\boldsymbol{\gamma}_k\}_{k=1}^{K}, \end{aligned}$$

where the matrix D_k is defined previously in Theorem 5.4.1. Next, applying Taylor's expan-

sion of $\dot{L}(\tilde{\boldsymbol{\beta}})$ at $\boldsymbol{\gamma}_0$ gives

$$\frac{\ln(n)}{2}\{2a_n(1-w_k^2)D_k^{-1}(\tilde{\boldsymbol{\gamma}}_k)\boldsymbol{\gamma}_k\}_{k=1}^K = L(\dot{\boldsymbol{\beta}}_0) + \ddot{L}(\boldsymbol{\beta}_0)\left(\frac{\partial\boldsymbol{\beta}}{\partial\boldsymbol{\gamma}}\bigg|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}_0}\right)(\tilde{\boldsymbol{\gamma}}-\boldsymbol{\gamma}_0) + \mathbf{r}_n,$$

where $\mathbf{r}_n$ denotes the remaider term. We notice that $\left(\frac{\partial\boldsymbol{\beta}}{\partial\boldsymbol{\gamma}}\big|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}_0}\right)$ is equal to the block diagonal matrix $D_k(\boldsymbol{\gamma}_0)$ defined in Theorem 5.4.1. It follows that

$$D(\boldsymbol{\gamma}_0)(\tilde{\boldsymbol{\gamma}}-\boldsymbol{\gamma}_0) = \left\{-\ddot{L}(\boldsymbol{\beta}_0)\right\}^{-1}\left[\dot{L}(\boldsymbol{\beta}_0) - \frac{\ln(n)}{2}\{2a_n(1-w_k^2)D_k^{-1}(\tilde{\boldsymbol{\gamma}}_k)\boldsymbol{\gamma}_k\}_{k=1}^K + \mathbf{r}_n\right].$$

Therefore,

$$\sqrt{n}[D(\boldsymbol{\gamma}_0)(\tilde{\boldsymbol{\gamma}}-\boldsymbol{\gamma}_0)+\mathbf{b}_n] = \left\{-\frac{\ddot{L}(\boldsymbol{\beta}_0)}{n}\right\}^{-1}\frac{\dot{L}(\boldsymbol{\beta}_0)}{\sqrt{n}} + \mathbf{r}'_n, \tag{7.20}$$

where $\mathbf{b}_n$ is defined in (refbias), and the remainder term is

$$\mathbf{r}'_n = \left\{-\frac{\ddot{L}(\boldsymbol{\beta}_0)}{n}\right\}^{-1}\frac{\mathbf{r}_n}{\sqrt{n}}.$$

Under the regularity assumptions, standard arguments yield $\left\{-\ddot{L}(\boldsymbol{\beta}_0)/n\right\}^{-1} \xrightarrow{p} I^{-1}(\boldsymbol{\gamma}_0)$; $\dot{L}(\boldsymbol{\beta}_0)/\sqrt{n} \xrightarrow{d} \mathbf{N}(0, I(\boldsymbol{\gamma}_0))$; and $\mathbf{r}'_n = o_p(1)$ as $n\to\infty$. Bringing these results into (7.20) and an appeal to Slutsky's Theorem give the desired asymptotic normality in (5.10).

Note that the elements D_{ij} of the block diagonal matrix $D(\boldsymbol{\gamma}_0)$ in Theorem 5.4.1 are evaluated at $\boldsymbol{\gamma}_0$. Then for any $k\in\{1,2,\ldots,K\}$, since $a_n\to\infty$ as $n\to\infty$ and variables in the same group are either all zero or all nonzero, we have

$$[D_k(\boldsymbol{\gamma}_0)]_{ii} = w_k + 2a_n(1-w_k^2)\gamma_{0k,i}^2 \to \begin{cases} 1 & \text{if } \gamma_{0k,i}\neq 0, \\ 0 & \text{otherwise,}\end{cases} \text{ as } n\to\infty,$$

and for $i \neq j$,

$$[D_k(\boldsymbol{\gamma}_0)]_{ij} = 2a_n(1-w_k^2)\gamma_{0k,i}\cdot\gamma_{0k,j} \to 0 \text{ as } n \to \infty.$$

To study the limit of bias $\mathbf{b}_n$, we rewrite (5.11) as

$$\mathbf{b}_n = \left\{\frac{-\ddot{L}(\boldsymbol{\beta}_0)}{n}\right\}^{-1} \frac{\ln(n)}{2\sqrt{n}}\left[\frac{1}{\sqrt{n}}\cdot 2a_n(1-w_k^2)D_k^{-1}(\widetilde{\boldsymbol{\gamma}}_k)\widetilde{\boldsymbol{\gamma}}_k\right]_{k=1}^{K}. \tag{7.21}$$

To prove that $\mathbf{b}_n = o_p(1)$, it is sufficient to show that

$$\mathbf{b}_{n,k} := \frac{1}{\sqrt{n}}\cdot 2a_n(1-w_k^2)D_k^{-1}(\widetilde{\boldsymbol{\gamma}}_k)\widetilde{\boldsymbol{\gamma}}_k = O_p(1)$$

for all $k = 1, 2, \ldots, K$. From Lemma 0 (ii), we have

$$\|\mathbf{b}_{n,k}\| \leq \frac{1}{\sqrt{n}}\cdot 2a_n\frac{1-w_k^2}{w_k}\|\widetilde{\boldsymbol{\gamma}}_k\| = \frac{1}{\sqrt{n}}\cdot\frac{4a_n\|\widetilde{\gamma}_k\|}{\exp(a_n\|\widetilde{\gamma}_k\|^2)-\exp(-a_n\|\widetilde{\gamma}_k\|^2)}.$$

When $\boldsymbol{\gamma}_{0,k} \neq \mathbf{0}$, we have $\widetilde{\boldsymbol{\gamma}}_k = \boldsymbol{\gamma}_{0,k} + O_p(n^{-1/2})$. Then $a_n\|\widetilde{\boldsymbol{\gamma}}_k\| \to \infty$ and hence $\mathbf{b}_{n,k} \to \mathbf{0}$ at an exponential rate. When $\boldsymbol{\gamma}_{0,k} = \mathbf{0}$, we have $\|\widetilde{\boldsymbol{\beta}}_k\| = O_p(n^{-1/2}) = \|\widetilde{\boldsymbol{\gamma}}_k\|\tanh(a_n\|\widetilde{\boldsymbol{\gamma}}_k\|^2) \approx a_n\|\widetilde{\boldsymbol{\gamma}}_k\|^3$. So $\|\widetilde{\boldsymbol{\gamma}}_k\| = O_p(n^{-1/2})$ with $a_n = O(n)$. In this case,

$$\frac{1}{\sqrt{n}}\cdot 2a_n\frac{1-w_k^2}{w_k}\|\widetilde{\boldsymbol{\gamma}}_k\| \approx \frac{2a_n\|\widetilde{\boldsymbol{\gamma}}_k\|}{a_n\|\widetilde{\boldsymbol{\gamma}}_k\|^2}\cdot\frac{1}{\sqrt{n}} = \frac{2}{\sqrt{n}\|\widetilde{\boldsymbol{\gamma}}_k\|} = O_p(1).$$

So in this case, the bias goes to zero with rate $\ln(n)/\sqrt{n}$. This completes the proof. ■

B.1. Additional Simulation Results for Chapter 4

Simulation results for the MCP penalty with $M = 1$

In Chapter 4, the performance of the QPADM fwith SCAD penalty were shown. In the following, we show the performance of the QPADM with the MCP penalty.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.80(1.56)	100%	94%	0.048(0.023)	1.65(0.31)
	$\tau = 0.5$	4.31(0.64)	100%	0%	0.036(0.022)	1.54(0.29)
	$\tau = 0.7$	6.80(1.42)	100%	93%	0.043(0.024)	1.67(0.33)
QICD	$\tau = 0.3$	7.56(3.82)	100%	92%	0.050(0.026)	0.99(1.13)
	$\tau = 0.5$	4.24(0.59)	100%	0%	0.040(0.020)	1.51(1.30)
	$\tau = 0.7$	6.80(3.62)	100%	93%	0.049(0.026)	1.46(1.59)

Table 7.1: Comparison of QPADM and QICD with $n = 300,\ p = 1,000$.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0040(0.0016)	45.09(1.55)
	$\tau = 0.5$	4.00(0.00)	100%	0%	0.0042(0.0019)	47.16(1.68)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0037(0.0017)	44.81(1.57)
QICD	$\tau = 0.3$	5.02(0.14)	100%	100%	0.0031(0.0016)	99.37(11.46)
	$\tau = 0.5$	4.16(0.37)	100%	0%	0.0033(0.0015)	121.47(16.35)
	$\tau = 0.7$	5.08(0.25)	100%	100%	0.0032(0.0014)	118.35(16.17)

Table 7.2: Comparison of QPADM and QICD with $n = 30,000,\ p = 1,000$.

Method	Quantile	Size	P1	P2	AE	Time (Sec)
QPADM	$\tau = 0.3$	5.00(0.00)	100%	100%	0.0032(0.0011)	3.43(0.56)
	$\tau = 0.5$	4.00(0.00)	100%	0%	0.0031(0.0011)	3.54(0.67)
	$\tau = 0.7$	5.00(0.00)	100%	100%	0.0030(0.0015)	3.42(0.58)
QICD	$\tau = 0.3$	5.06(0.24)	100%	100%	0.0027(0.0011)	12.21(3.33)
	$\tau = 0.5$	4.08(0.27)	100%	0%	0.0026(0.0011)	22.33(11.71)
	$\tau = 0.7$	5.02(0.15)	100%	100%	0.0026(0.0009)	25.45(19.00)

Table 7.3: Comparison of QPADM and QICD with $n = 30000,\ p = 100$.

Parallel QPADM: More Results

In Chapter 4 only showed the performance of parallel QPADM for the SCAD penalty with quantile level $\tau = 0.3$, while the simulation were done with $\tau = 0.3,\ 0.5,\ 0.7$ for both the SCAD and MCP penalties. We include the remaining results in the following.

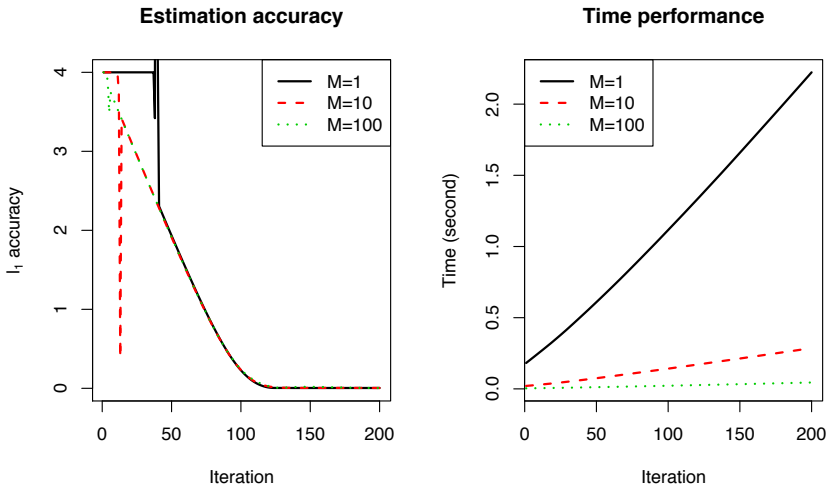


Figure 7.1: Comparison of QPADM with SCAD penalty for different M values at τ=0.5.

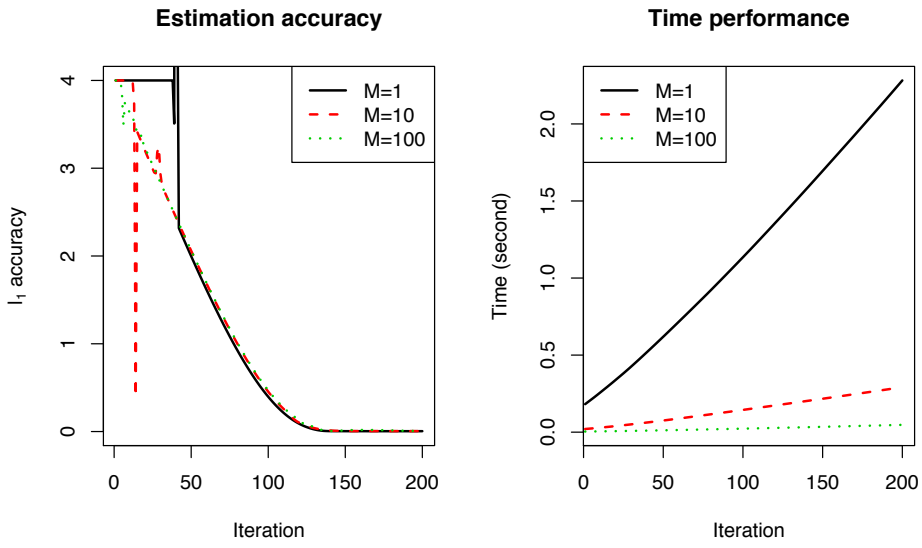


Figure 7.2: Comparison of QPADM with SCAD penalty for different M values at τ=0.7.

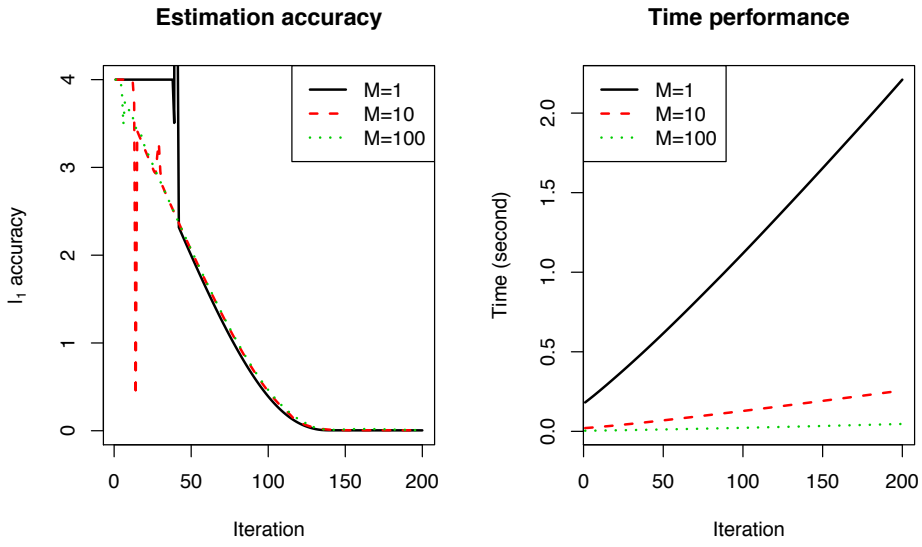


Figure 7.3: Comparison of QPADM with MCP penalty for different M values at τ=0.3.

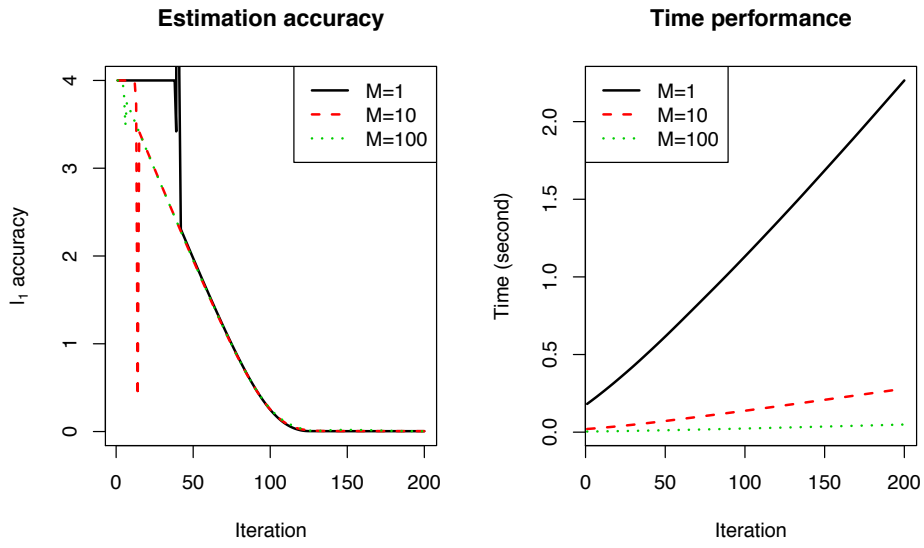


Figure 7.4: Comparison of QPADM with MCP penalty for different M values at τ=0.5.

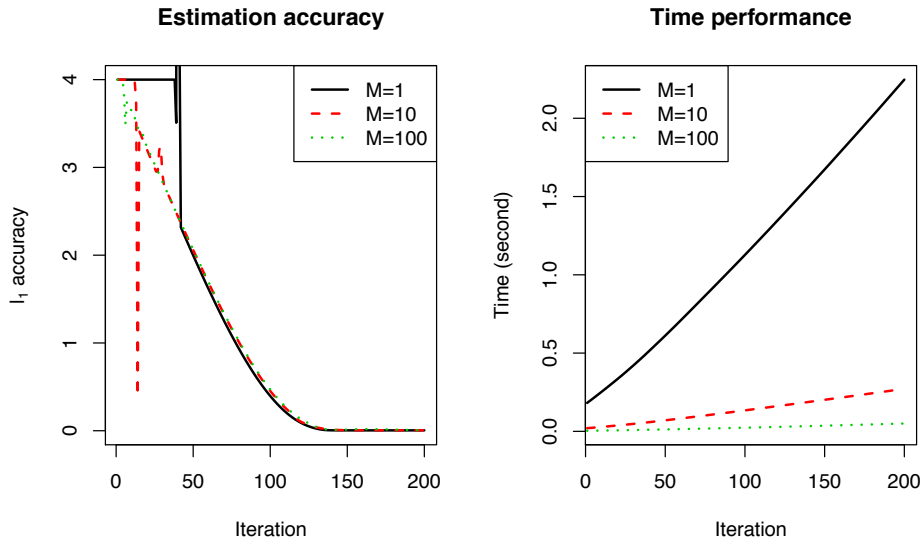


Figure 7.5: Comparison of QPADM with MCP penalty for different M values at τ=0.7.

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] Claude JP Bélisle. Convergence theorems for a class of simulated annealing algorithms on r^d. *Journal of Applied Probability*, 29(04):885–895, 1992.

[3] Alexandre Belloni and Victor Chernozhukov. ℓ_1-penalized quantile regression in high-dimensional sparse models. *The Annals of Statistics*, 39(1):82–130, 2011.

[4] Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

[5] Jacob Bien and Robert J Tibshirani. Sparse estimation of a covariance matrix. *Biometrika*, 98(4):807–820, 2011.

[6] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[7] Patrick Breheny. grpreg: Regularization paths for regression models with grouped covariates. r package version 3.0-2, 2016.

[8] Patrick Breheny and Jian Huang. Group descent algorithms for nonconvex penalized

linear and logistic regression models with grouped predictors. *Statistics and computing*, 25(2):173–187, 2015.

[9] Leo Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995.

[10] Rick Chartrand and Brendt Wohlberg. A nonconvex ADMM algorithm for group sparsity with sparse groups. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6009–6013. IEEE, 2013.

[11] Colin Chen and Ying Wei. Computational issues for quantile regression. *Sankhyā: The Indian Journal of Statistics*, 67(2):399–417, 2005.

[12] Jiahua Chen and Zehua Chen. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

[13] Xueying Chen and Min-ge Xie. A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica*, 24(4):1655–1684, 2014.

[14] David R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society, Series B (Methodological)*, 34(2):187–220, 1972.

[15] David R. Cox and Nancy Reid. Parameter orthogonality and approximate conditional inference. *Journal of the Royal Statistical Society, Series B (Methodological)*, 49(1):1–39, 1987.

[16] Puja Das, Nicholas Johnson, and Arindam Banerjee. Online lazy updates for portfolio selection with transaction costs. In *AAAI*, 2013.

[17] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[18] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National Science Review*, 1(2):293–314, 2014.

[19] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

[20] Jianqing Fan, Richard Samworth, and Yichao Wu. Ultrahigh dimensional feature selection: Beyond the linear model. *Journal of Machine Learning Research*, 10(9), 2009.

[21] Sholeh Forouzan and Alexander Ihler. Linear approximation to ADMM for MAP inference. In *Asian Conference on Machine Learning*, pages 48–61, 2013.

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

[23] Roland Glowinski and A Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(2):41–76, 1975.

[24] Tom Goldstein, Gavin Taylor, Kawika Barabin, and Kent Sayre. Unwrapping ADMM: Efficient distributed computing via transpose reduction. In *Artificial Intelligence and Statistics*, pages 1151–1158, 2016.

[25] Mohamed Hebiri and Johannes Lederer. How correlations influence lasso prediction. *IEEE Transactions on Information Theory*, 59(3):1846–1854, 2013.

[26] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. Convergence analysis of al-

ternating direction method of multipliers for a family of nonconvex problems. *arXiv preprint arXiv:1410.1390*, 2014.

[27] Joel L. Horowitz. Bootstrap methods for median regression models. *Econometrica*, 66(6):1327–1351, 1998.

[28] Saghar Hosseini, Airlie Chapman, and Mehran Mesbahi. Online distributed ADMM on networks. *arXiv preprint arXiv:1412.7116*, 2014.

[29] Cheng Huang and Xiaoming Huo. A distributed one-step estimator. *arXiv preprint arXiv:1511.01443*, 2015.

[30] Jian Huang, Patrick Breheny, and Shuangge Ma. A selective review of group selection in high-dimensional models. *Statistical Science*, 27(4):481–499, 2012.

[31] Junzhou Huang, Tong Zhang, et al. The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978–2004, 2010.

[32] James P Ignizio. *Goal Programming and Extensions*. Lexington Books, 1976.

[33] Lilli Japec, Frauke Kreuter, Marcus Berg, Paul Biemer, Paul Decker, Cliff Lampe, Julia Lane, Cathy ONeil, and Abe Usher. *AAPOR* Report on Big Data. 2015.

[34] Bo Jiang, Shiqian Ma, and Shuzhong Zhang. Alternating direction method of multipliers for real and complex polynomial optimization models. *Optimization*, 63(6):883–898, 2014.

[35] Michael I Jordan et al. On statistics, computation and scalability. *Bernoulli*, 19(4):1378–1390, 2013.

[36] Michael I Jordan, Jason D Lee, and Yun Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, just-accepted, 2018.

[37] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[38] Kevin Kelly. *Out of Control: The New Biology of Machines, Social Systems, and the Economic World*. Hachette UK, 2009.

[39] R. Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005.

[40] Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

[41] Eun Ryung Lee, Hohsuk Noh, and Byeong U Park. Model selection via Bayesian information criterion for quantile regression models. *Journal of the American Statistical Association*, 109(505):216–229, 2014.

[42] Dong-Hui Li and Masao Fukushima. A modified BFGS method and its global convergence in nonconvex minimization. *Journal of Computational and Applied Mathematics*, 129(1):15–35, 2001.

[43] Guoyin Li and Ting Kei Pong. Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization*, 25(4):2434–2460, 2015.

[44] Moche Lichman. UCI machine learning repository, 2013.

[45] Nan Lin and Ruibin Xi. Aggregated estimating equation estimation. *Statistics and Its Interface*, 4(1):73–83, 2011.

[46] Nan Lin and Liqun Yu. The ADMM and its application to network big data. In Yulei Wu, Fei Hu, Geyong Min, and Albert Y. Zomaya, editors, *Big Data and Computational*

Intelligence in Networking, chapter 8, pages 151–176. Taylor & Francis LLC, CRC Press, 2018.

[47] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2013.

[48] Lanchao Liu and Zhu Han. Multi-block ADMM for big data optimization in modern communication networks. *Journal of Communications*, 10(9), 2015.

[49] Po-Ling Loh and Martin J Wainwright. Regularized M-estimators with nonconvexity: Statistical and algorithmic theory for local optima. In *Advances in Neural Information Processing Systems*, pages 476–484, 2013.

[50] Rahul Mazumder, Jerome H Friedman, and Trevor Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

[51] Peter McCullagh and John A Nelder. *Generalized Linear Models*. CRC press, 1989.

[52] João FC Mota, João MF Xavier, Pedro MQ Aguiar, and Markus Püschel. A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. *arXiv preprint arXiv:1112.2295*, 2011.

[53] João FC Mota, João MF Xavier, Pedro MQ Aguiar, and Markus Puschel. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *Signal Processing, IEEE Transactions on*, 61(10):2718–2723, 2013.

[54] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[55] Hua Ouyang, Niao He, and Alexander Gray. Stochastic ADMM for nonsmooth optimization. *arXiv preprint arXiv:1211.0632*, 2012.

[56] Neal Parikh and Stephen Boyd. Block splitting for distributed optimization. *Mathematical Programming Computation*, 6(1):77–102, 2014.

[57] Bo Peng and Lan Wang. An iterative coordinate descent algorithm for high-dimensional nonconvex penalized quantile regression. *Journal of Computational and Graphical Statistics*, 64(3):676–694, 2015.

[58] Stephen Portnoy and Roger Koenker. The Gaussian hare and the Laplacian tortoise: Computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 1997.

[59] John R. Quinlan. Combining instance-based and model-based learning. *In Proceedings on the Tenth International Conference of Machine Learning*, pages 236–243, 1993.

[60] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[61] Aaditya Ramdas and Ryan J Tibshirani. Fast and flexible ADMM algorithms for trend filtering. *Journal of Computational and Graphical Statistics*, 25(3):839–858, 2016.

[62] Elizabeth D Schifano, Jing Wu, Chun Wang, Jun Yan, and Ming-Hui Chen. Online updating of statistical inference in the big data setting. *Technometrics*, 58(3):393–403, 2016.

[63] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[64] Chao Shen, Tsung-Hui Chang, Kun-Yu Wang, Zhengding Qiu, and Chong-Yung Chi. Distributed robust multicell coordinated beamforming with imperfect CSI: An ADMM approach. *IEEE Transactions on signal processing*, 60(6):2988–3003, 2012.

[65] Yuan Shen, Zaiwen Wen, and Yin Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29(2):239–263, 2014.

[66] Xiaogang Su. Variable selection via subtle uprooting. *Journal of Computational and Graphical Statistics*, 24(4):1092–1113, 2015.

[67] Xiaogang Su, Chalani S Wijayasinghe, Juanjuan Fan, and Ying Zhang. Sparse estimation of Cox proportional hazards models via approximated information criteria. *Biometrics*, 72(3):751–759, 2016.

[68] Taiji Suzuki. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *International Conference on Machine Learning*, pages 392–400, 2013.

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

[70] William N Venables and Brian D Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002.

[71] Bo Wahlberg, Stephen Boyd, Mariette Annergren, and Yang Wang. An ADMM algorithm for a class of total variation regularized estimation problems. *IFAC Proceedings Volumes*, 45(16):83–88, 2012.

[72] Huahua Wang and Arindam Banerjee. Online alternating direction method. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1119–1126, 2012.

[73] Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Efficient distributed

learning with sparsity. In *International Conference on Machine Learning*, pages 3636–3645, 2017.

[74] Lan Wang, Yichao Wu, and Runze Li. Quantile regression for analyzing heterogeneity in ultra-high dimension. *Journal of the American Statistical Association*, 107(497):214–222, 2012.

[75] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*, 2015.

[76] Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5445–5450, 2012.

[77] Ying Wei. An approach to multivariate covariate-dependent quantile contours with application to bivariate conditional growth charts. *Journal of the American Statistical Association*, 103(481):397–409, 2008.

[78] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2012.

[79] Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42:106, 1950.

[80] Jun Xie and Lingmin Zeng. Group variable selection methods and their applications in analysis of genomic data. In *Frontiers in Computational and Systems Biology*, pages 231–248. Springer, 2010.

[81] Yangyang Xu, Wotao Yin, Zaiwen Wen, and Yin Zhang. An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, 7(2):365–384, 2012.

[82] Jiyan Yang, Xiangrui Meng, and Michael W Mahoney. Quantile regression for large-scale applications. *SIAM Journal on Scientific Computing*, 36(5):78–110, 2014.

[83] Junfeng Yang and Yin Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.

[84] Bin Yu. Let us own data science. *Institute of Mathematical Statistics (IMS) Presidental Address, ASC-IMS Joint Conference*, 2014.

[85] Liqun Yu and Nan Lin. ADMM for penalized quantile regression in big data. *International Statistical Review*, 85(3):494–518, 2017.

[86] Liqun Yu, Nan Lin, and Lan Wang. A parallel algorithm for large-scale nonconvex penalized quantile regression. *Journal of Computational and Graphical Statistics*, 26(4):935–939, 2017.

[87] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[88] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *in Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2010.

[89] Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

[90] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1701–1709, 2014.

[91] Xiaoqun Zhang, Martin Burger, and Stanley Osher. A unified primal-dual algorithm framework based on Bregman iteration. *Journal of Scientific Computing*, 46(1):20–46, 2011.

[92] Peilin Zhao, Jinwei Yang, Tong Zhang, and Ping Li. Adaptive stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, pages 69–77, 2015.

[93] Leon Wenliang Zhong and James T Kwok. Gradient descent with proximal average for nonconvex and composite regularization. In *Proceedings of the National Conference on Artificial Intelligence*, pages 2206–2212, 2014.

[94] Wenliang Zhong and James Kwok. Fast stochastic alternating direction method of multipliers. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 46–54, 2014.

[95] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

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