Generalized Lasso Problem With Equality And Inequality Constraints Using ADMM

by

Chi Xu

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 14, 2019

Keywords: convex optimization, ADMM, generalized LASSO, parallel computing, distributed ADMM, equality and inequality constraints

Approved by

Peng Zeng, Chair, Associate Professor of Mathematics and Statistics
Ash Abebe, Professor of Mathematics and Statistics
Nedret Billor, Professor of Mathematics and Statistics
Guanqun Cao, Associate Professor of Mathematics and Statistics

Abstract

With the development of technology, the volume of the data has gradually become much larger and attracting much more attention in data science. Statisticians nowadays need to consider the case when the dataset is extremely huge, which can be referred to as ”Big Data Problem”. In the applications of big data analysis, many problems can be formulated into the family of convex optimization problem. In my dissertation, I will mainly discuss the algorithm of alternating direction of multipliers method (ADMM), which is an efficient algorithm in distributed convex optimization and attracting more and more attention in recent years because of its superior performance in big data analysis and machine learning. This algorithm is generally useful in splitting the global problem into subproblems and solving the parallel computing of those subproblems instead of working on the global problem directly. Generalized LASSO problem is one of the most commonly used convex optimization problems. In this dissertation, I will focus on the generalized LASSO problem with equality and inequality constraints. Several ADMM and distributed ADMM algorithms will be proposed to solve the problem with high efficiency and low computational cost.

Acknowledgments

I would like to express my sincere gratitude to Dr. Peng Zeng for his expert guidance, support and persistent encouragement during my graduate studies at Auburn University. I would like to emphasize that his influence on me was not only in acquiring scientific knowledge but also as a person.

The author would also like to thank his parents for their love, constant emotional support and encouragement during his studies.

Thanks also go to my colleagues and friends, Minghong Jian, Zhaohui Jin, Le Cai, Rui Wang, Boyao Zhao, Wei Huang, Dr. Wei Sun, Yuan Yuan, Luyining Gan, Yu Wang, Lin Lu, Xiangqian Meng, Lingxiao Wang, Jingjing Li, Mingyang Chi, Dang Li, Meng Niu for all their help and suggestions.

I would like to give my appreciation and sincere thanks to Dr. Ash Abebe, Dr. Nedret Billor, Dr. Guanqun Cao for serving on my committee, and Dr. Jianjun Dong for serving as my university reader, and for their generous guidance and kind support, to Dr. Ulrich Albrecht, Dr. Mark Carpenter, Dr. Xiaoyu Li and Dr. Philippe Gaillard for their helpful discussions, and to Dr. Paul Schmidt and Dr. Xiaoying Han for their insightful suggestions.

Table of Contents

Abstract ii

Acknowledgments iii

1 Introduction 1

1.1 Big Data Problem 1

1.2 Parallel and Distributed Computing 2

1.3 Model 5

1.4 Contribution 7

2 ADMM 10

2.1 Notation 10

2.2 Convergence 12

2.3 Stopping Criteria 13

2.4 Literature Review 13

3 ADMM Algorithms 17

3.1 Extended ADMM With Slack Variables 17

3.2 Extended ADMM With Large p 22

3.3 Distributed ADMM Splitting Along n 29

3.4 Distributed ADMM Splitting Along p 35

3.5 Distributed ADMM Splitting Along n and p 41

4 Proof of Convergence 51

4.1 Extended ADMM With Slack Variables 54
4.2 Extended ADMM With Large p 54
4.3 Distributed ADMM Splitting Along n 55
4.4 Distributed ADMM Splitting Along p 59
4.5 Distributed ADMM Splitting Along n and p 61

5 Implementation 68
5.1 Selection of Tuning Parameters 68
5.2 Sparse Matrices 69
5.3 Distributed Computing 69

6 Simulation 70
6.1 Influence of ρ 70
6.2 Influence of λ 71
6.3 Influence of number of subgroups 73
6.4 Influence of n and p 75
6.5 Comparison of ADMM and QP 80

7 Real Example 83
7.1 Cooperative spectrum sensing for cognitive radio networks 83
7.2 National health and nutrition examination survey 87

8 Future Work 91

References 93

List of Figures

6.1 Influence of ρ 71

6.2 Influence of λ 73

6.3 Influence of λ 74

6.4 Performance when n increases 77

6.5 Performance when n increases 78

6.6 Performance when n increases 79

6.7 Convergence rate of extended ADMM algorithms 82

7.1 Tuning parameter selection with CR data 87

7.2 Tuning parameter selection with NHNES data 89

List of Tables

6.1 Mean and Standard Deviation of Performance Metrics 80

6.2 Mean and Standard Deviation of Linear Constraints 81

Chapter 1

Introduction

1.1 Big Data Problem

With the development of technology, the volume of the data has gradually became much larger and attracting much more attention in data science. There are 2.5 quintillion bytes of data created in every single day and around 90 percent of the data on earth are generated in the most recent two years. For each event that occured in the world, like Nobel Prize, tournament game, one video uploaded from a Youtuber, there will be millions of comments following those events, which are too huge to collect. In stock market, the New York Stock Exchange generates about one terabyte of new trade data per day. In social media, the statistic shows that more than 500 terabytes of new data get ingested into the databases of Facebook every day. It is common in contemporary society to use airports to travel and attend meetings. However, a single jet engine can generate 10 terabytes of data in 30 minutes of flight time with many thousand flights per day which is too enormous to analyze.

Because of the existing real examples, statisticians nowadays always need to consider the case when the dataset is extremely huge, which can be referred to as "Big Data Problem". With the emergence of "Alpha Go" and other AI programs that show suprising achievement and accomplishment in different areas, big data has gradually become a very hot topic in data analysis. In order to solve these data analysis problems, the algorithms in machine learning, deep learning and high dimensional data analysis attracted more attention in statistics and computer science. The big data also arises in different areas in applications, such as pharmaceutical analysis, gene detecting, biology, economic optimization and so on. Although the applications are in various areas and with different background, they share some common characteristics

which people abbreviated as "3v": big volume, high velocity and large variety. Firstly, the data usually has extremely large sample size volume, which exceeds a million in most cases. Moreover, the dimension of the dataset is always very high, usually with enormous sample size at the same time. Furthermore, because of this large sample size and high dimension, the dataset is usually stored in distributed manners. There are multiple servers and each local server only stores the local information, such as local parameter, local predictors, local response and so on. In such cases, it would be helpful if we can solve the big data problem by partitioning the data into different blocks by dimensions or observations and work on each of the subproblems by distributed computing to achieve better efficiency and less computational cost.

1.2 Parallel and Distributed Computing

In the past, the information of individuals, such as age, gender, race, income and so on, are usually collected by area and time so that the data size is moderate to do analysis. However, with more and more information added to each individual in the contemporary society, a wider range of data along with time series is needed to have a big picture of those information which is with enormous contents. From another perspective, vast quantities of individual information are collected by a broad spectrum of organizations while they are commonly under the premise of privacy. Disclosures may result in harm to the data's owner and jeopardize future access to such sensitive information. This would be a severe problem when the data is extremely large while some of the information is confidential due to personal privacy, which is referred to as "privacy preserving problem". Assume that there are several pharmaceutical companies doing research on the same medicine for a type of cancer. It would be a great idea to improve the accuracy and performance of the analysis by the cooperation of those companies. However, due to privacy of the raw data, no company agree to provide the raw data. Under the scenario, by using the proposed algorithm, each company will do analysis with their raw data and upload the result to the central server. The central server will only need to use the uploaded results instead of the raw data to do statistical analysis and have a good idea of this medicine. This is also known as meta analysis. However, in meta analysis, the final result will be derived and there is no more feedback from the government. While the feedback may be needed for

another iteration in the distributed computing and privacy preserving problem. As another example, from the nature of data collection, the most common methods used is focus groups. Focus groups is not merely collecting similar data from many participants at once but a group discussion on a particular topic organized for research purposes. In this case, the data collected are blocked by those groups and it would be a tough mission to combine all of them together since each group may have its own perspective of the topic.These are introductory examples of privacy preserving problem, which is a hot topic nowadays to be solved. There are also many other types of privacy perserving problems and all of them can be solved using the proposed algorithm in this dissertation.

In order to solve these big data problems efficiently, many different algorithms have been derived to reduce the computational cost. Among all of these algorithms, the idea of parallel and distributted computing is the one widely used in statistics and engineering. If the big data can be split along its dimension or sample size to make it into small blocks, statisticians can solve the subproblems within each block, and collect and aggregate the intermediate result in each block to generate the global result. In this way, we are able to switch the big data problem into moderate data problem which is expected to obtain better efficiency and performance. In addition, if these subproblems can be solved at the same time, in other words, in parallel, the total computational time of the big data can be reduced to the time of one block of small data, which statisticians definitely prefer. Furthermore, the big data is usually stored in different locations and different servers that people cannot do analysis of the whole big data. If the data in each server can be analyzed locally and only the results are combined by the central server, the communication would be much cheaper and possible to achieve which is the reason why parallel and distributed computing needed.

In parallel and distributed computing, there are two major problems: communication and synchronization. If the server transmit uneven or wrong communication links, it will dramatically decrease the efficiency of the computation and make the cost much higher. The other case is one or some of the clients take much longer than others and will be much slower to summarize the original dataset in central server which is called the synchronization problem.

It is easy to imagine that with this problem occuring, the parallel computing will have low efficiency due to those defected machines. In this way, there are mainly three different scenarios of general parallel computing. The first scenario is the ideal one that no communication or synchronization problem occurs and the corresponding method is named embarrassingly parallel first-order methods. A simple example is the consensus ADMM in Combettes and Pesquet (2008). Moreover, if the communication problem cannot be ignored, the first-order methods with reduced or decentralized communications will be helpful. For example, Shi et al. (2015) proposed the exact first-order algorithm to handle this case. Lastly, if both communication and synchronization problem occured, we need to apply the asynchronous first-order methods with decentralized communications such as running stochastic gradient methods in parallel without locks which was proposed in Recht et al. (2011).

In recent years, several programming frameworks of parallel computing arised. MapReduce is one of those famous frameworks which is implemented in Hadoop. Hadoop is the software which provides two kernel capabilities: a reliable shared storage called Hadoop Distributed Filesystems (HDFS) and the analysis system by MapReduce. MapReduce works by breaking the processing into two phases: map and reduce. In each phase, there are key-value pairs, input, output, map function and reduce function. These functions are chosen by the programmer at the beginning of the phase. In the map phase, the input is the raw data while the information pulled out from the raw data is the output. In the reduce phase, the input is the output of map function and the output is the key value by sorting and grouping the key-value pairs. Details of MapReduce and Hadoop are discussed in White (2012). Another popular framework is the Spark. Spark is the cluster computing framework which is used to deal with the case when the working set of data across multiple parallel operations needs to be reused. The main abstraction in Spark is resilient distributed dataset (RDD) which represents a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. Spark is implemented in Scala which is a high-level programming language for Java VM. Spark is the first system to allow an efficient, general-purpose programming language to be used interactively to process large datasets on a cluster. Details of Spark and Scala can be found in Zaharia et al. (2010). The other widely used framework is openMPI which is an open

source Message Passing Interface (MPI) implementation. OpenMPI can provide a high performance, robust, parallel execution environment for a variety of computing environments. It has two functional units: Open run-time Environment (openRTE), which is used for bootstrapping operations, and openMPI communication library, which can provide efficient communication support. The details of openMPI can be found in Graham et al. (2006). In this dissertation, I applied the openMPI framework to run distributed computing. Notice that it is awkward to implement ADMM in MapReduce, while the programming language C is believed to have higher efficiency than Java. I would prefer to implement openMPI in C instead of spark in Java VM.

1.3 Model

In the applications of big data analysis, many problems can be formulated into the family of convex optimization problem. Among all of those problems, LASSO is one of the most popular problem that is widely used in statistical analysis. The least absolute shrinkage and selection operator technique, which is usually abbreviated as LASSO, was first introduced and developed by Tibshirani (1996). This technique is developed from the ordinary least squares estimates to improve the prediction accuracy which can fit the sparse parameter if such assumption holds, such as gene detecting problem, financial networks, variable selection, etc. It shrinks some coefficients of the estimated parameter β in regression and sets others to be 0, which can retain good features of subset selection and ridge regression as well. The optimization problem can be written as:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1,$$

where y is the response vector, X is the design matrix constructed by the value of predictors in each observation in the sample, λ is called the tuning parameter to modify the weight of the LASSO term. And there are many fast algorithms for solving the LASSO problem such as the least angle regression (LARS) algorithm (Efron et al. (2004)) and homotopy method (Osborne et al. (2000)). With the development of LASSO problem, different penalizations of the parameter β gradually shows up, such as the fused lasso (Rudin et al. (1992), Tibshirani

et al. (2005)), trend filtering (Steidl et al. (2006), Kim et al. (2009)), the graph fused lasso (Hoefling, 2010) and so on. And in 2011, Tibshirani and Taylor (2011) introduced generalized LASSO problem:

$$\arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D\beta\|_1,$$

and unified all the similar LASSO problems by specifying the coefficient matrix D.

In this dissertation, I will mainly focus on the generalized LASSO problem with linear equality and inequality constraints:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D\beta\|_1, \quad \text{subject to} \quad C\beta \geq d, E\beta = f, \tag{1.1}$$

where $X \in R^{n\times p}$ is the design matrix with sample size n and dimension p, $y \in R^n$ is the response, β is the parameter needed to be estimated, λ is the tunning parameter of generalized LASSO term, D gives the specific structure of β to be shrinked, and matrices C, E and vectors d, f are the coefficients in equality and inequality constraints. The linear constraints arise in LASSO problem in recent years because the existing demand in real data. For example, in the annual data on temperature anomalies, temperature appears to increase monotonically over the time which can be implemented by specifying the matrix C. As another example, the positive LASSO requires the LASSO coefficients to be nonnegative. Due to these demands, it's necessary to add the equality and inequality constraints along the generalized LASSO problem.

There are many good properties of LASSO problem and generalized LASSO problem. The algorithm applied to this model is the alternating direction method of multipliers (ADMM). The details of this algorithm will be introduced in the next chapter. Donoho et al. (1995) proved the near-minimax optimality of soft thresholding which represents the LASSO shrinkage with orthogonal predictors. It has also been proved that the $L1$ approach is able to discover the correct sparse representation of the model under certain conditions in Donoho and Elad (2003), Donoho and Huo (2001) and Donoho (2006). Meinshausen and Bühlmann (2006) also showed

that the variable selection with LASSO can be consistent if the model satisfies some good conditions. Tibshirani et al. (2013) proved the uniqueness in LASSO problem and the extension to generalized LASSO problem is also given in Ali and Tibshirani (2018). Fan and Li (2001) conjectured that the oracle properties do not hold for LASSO problem. It has been shown by Zou (2006) that the oracle property and near-minimax optimality is held in the adaptive LASSO, in which adaptive weights are used for penalizing different coefficients in the $L1$ penalty. It is interesting to consider whether the generalized LASSO could produce an oracle procedure and its corresponding suffcient and necessary condition but to the best of my knowledge, there is no paper published so far giving answer to this question.

The algorithm applied to this model is the alternating direction method of multipliers (ADMM). The details of this algorithm will be introduced in the next chapter.

1.4 Contribution

With the development of ADMM algorithm in big data problem, the application of ADMM algorithm to the LASSO problem started to show up. There are many applications of ADMM on different types of LASSO problems. For example, Li et al. (2014) and Salehani et al. (2014) applied ADMM algorithm on fused LASSO problem and Adaptive LASSO problem. In 2017, Zhu (2017) applied a revised ADMM algorithm to generalized LASSO problem with the scenario when $p > n$. However, this paper didn't consider the case associated with linear constraints, which is common in real examples. In a recent paper by Giesen and Laue (2016), they mentioned the extension of ADMM to solve the convex optimization algorithm with linear equality and inequality constraints by adding an indicator function to transfer the inequality constraints to equality constraints. However, the update of primal variables has no explicit solution and need to be solved by some other first-order methods. Similarly, in another recent paper Gaines et al. (2018), the authors also derived the algorithm for LASSO problem using ADMM. But that paper only works on the LASSO problem instead of generalized LASSO and they used the indicator function to combine the inequality and equality constraints, which is not able to be solved explicitly in some cases either.

In the dissertation, I will propose five ADMM algorithms to solve generalized LASSO problem with linear constraints. These ADMM algorithms are designed for different scenarios of the problem, including extremely large n, enormous p, n and p both large, $p > n$, and privacy preserving scenario. The details of these algorithms will be provided in Chapter 3. There are mainly five contributions of this dissertation: First, the proposed ADMM algorithm can be used for the generalized LASSO problem with constraints, which unifies all types of penalizations in LASSO problems. Next, different from Giesen and Laue (2016) and Gaines et al. (2018), the proposed algorithm can deal with the constraints without using one more layer of iteration inside of the ADMM algorithm itself. Furthermore, the proposed algorithms can cover all generalized LASSO problem in different settings of sample size and dimensions. In previous ADMM paper, the proposed algorithm usually fits well for a single scenario, either with large sample size n, large dimension p, or the scenario when $p > n$. With all the five proposed ADMM algorithms in this dissertation, all the different cases and setting can be covered very well. In addition, the proposed parallel and distributed ADMM algorithm can achieve much better performance and lower computational cost compared to existing methods, specifically, the quadratic programming. For example, when the sample size n is huge, the distributed ADMM algorithm splitting along n could reduce the time of convergence. When the dimension p is high, the distributed ADMM algorithm splitting along p can obtain the same performance with much less computational cost compared with other existing methods. Last but not least, the proposed distributed ADMM algorithm is helpful to deal with the privacy preserving problem.

This paper is organized as follows. In Chapter 2, we will introduce the basic algorithms of ADMM and give a brief review of its development. In Chapter 3, we will propose several ADMM algorithms for generalized LASSO problem with constraints. In Chapter 4, we will focus on the proof of convergence of the proposed algorithms based on the discussion in Chen et al. (2016). In Chapter 5, several issues related to the implementation of the proposed algorithms will be discussed in detail. In Chapter 6, some simulations results will be given to illustrate the performance of proposed algorithms, as well as the comparison of efficiency to quadratic programming. In Chapter 7, we will conduct the data analysis of two real examples

using the proposed algorithms. In Chapter 8, I will summarize my conclusion and dicuss the future work of my topic and its importance.

Chapter 2

ADMM

2.1 Notation

ADMM is an algorithm for solving the problems of the following form:

$$\min_{x,z} f(x) + g(z), \quad \text{subject to} \quad Ax + Bz = c, \tag{2.1}$$

where $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$ are the variables, and $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$ and $c \in \mathbb{R}^p$ are the known coefficients in the equality constraint. Here we assume function f and g are both convex functions.

In order to solve this convex optimization problem, we first find the augmented Lagrangian function given below:

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

where we call y as the dual variable. The optimal value of this problem p^* is then:

$$p^* = inf\{f(x) + g(z)|Ax + Bz = c\}.$$

The steps to find the optimal value is updated in alternating way from an initial value:

$$x^{k+1} = \arg\min_x L_\rho(x, z^k, y^k),$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, y^k),$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c).$$

It differs from the dual ascent and the method of multipliers because the updates of x and z are not updated jointly. They are updated based on alternating directions where the "AD" comes from in this algorithm. Notice that the x and z cannot be switched easily because they are not actually symmetric. The update of x is always before the update of z in each iteration which is the difference from the methods of multipliers.

In real cases, we usually work on a more convenient form which is named scaled form. If we define the residual $r = Ax + Bz - c$, then in the augmented Lagrangian function, the last two terms can be rewritten as:

$$\begin{aligned} & y^T r + \frac{\rho}{2}\|r\|_2^2 \\ = & \frac{\rho}{2}\|r + \frac{1}{\rho}y\|_2^2 - \frac{1}{2\rho}\|y\|_2^2 \\ = & \frac{\rho}{2}\|r + u\|_2^2 - \frac{\rho}{2}\|u\|_2^2, \end{aligned}$$

where $u = \frac{1}{\rho}y$ is the renewed scaled dual variable. Replacing y by u in the augmented Lagrangian function, we can have the renewed form as:

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

The corresponding ADMM algorithm can be written as:

$$\begin{aligned} x^{k+1} &= \arg\min_x (f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2), \\ z^{k+1} &= \arg\min_z (g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2), \\ u^{k+1} &= u^k + Ax^{k+1} + Bz^{k+1} - c, \end{aligned}$$

which is the scaled form of ADMM algorithm since it is expressed in terms of the scaled version of dual variable.

2.2 Convergence

According to Boyd et al. (2011) and Mota et al. (2011), we will need to make three assumptions as follows:

Assumption 1. *The functions* $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ *and* $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ *are closed, proper and convex.*

Assumption 2. *The unaugmented Lagrangian* L_0 *has a saddle point.*

Assumption 1 can guarantee that the update of x and z are solvable but may not be unique (if A and B are not with full rank). And assumption 2 can be explicitly written in the following: there exists (x^*, z^*, u^*), that

$$L_0(x^*, z^*, u) \leq L_0(x^*, z^*, u^*) \leq L_0(x, z, u^*)$$

holds for all x, z and u but not necessarily unique. And according to Mota et al. (2011), we need the additional assumption:

Assumption 3. *Matrices* A *and* B *have full column rank.*

Assumption 3 is the additional assumption made in Mota et al. (2011). The author pointed out that without the assumption 3, Boyd et al. (2011) could not come up with the proof of the claim that $\{(x^k, y^k)\}$ has a single limit point (x^*, y^*) which solves the optimization problem, i.e., the solutions of primal variables are unique.

Under these three assumptions, the ADMM algorithm would guarantee the following properties:

- Residual convergence. $r^k \to 0$ as $k \to \infty$.
- Objective convergence. $f(x^k) + g(z^k) \to p^*$ as $k \to \infty$.
- Dual variable convergence. $u^k \to u^*$ as $k \to \infty$.
- $\{(x^k, y^k)\}$ has a unique limit point (x^*, y^*) which solves (2.1).

- $\{y^k\}$ has a unique limit point y^* which solves the dual problem of (2.1).

The details of the proof can be refered to Boyd et al. (2011) and Mota et al. (2011) which is based on the proof of three inequalities including the sequence of Lyapunov function. After the inequalities established, the only property left to be proved is the uniqueness of the limit point which is discussed in Mota et al. (2011) with the additional assumption 3 made in the paper.

2.3 Stopping Criteria

From Boyd et al. (2011), if we define another residual s as:

$$s^{k+1} = \rho A^T B(z^{k+1} - z^k).$$

An efficient termination criterion of primal residual r and dual residual s can be set as:

$$\|r^k\|_2 \leq \epsilon^{pri},$$
$$\|s^k\|_2 \leq \epsilon^{dual},$$

where

$$\epsilon^{pri} = \sqrt{n}\epsilon^{abs} + \epsilon^{rel} max\{\|Ax^k\|_2, \|Bz^k\|_2, \|c\|_2\},$$
$$\epsilon^{dual} = \sqrt{p}\epsilon^{abs} + \epsilon^{rel}\|\rho A^T u^k\|_2,$$

where ϵ^{abs} is an absolute tolerance and ϵ^{rel} is a relative tolerance that both of them are usually selected as 10^{-3} or smaller depending on the application area and scale of variables.

2.4 Literature Review

ADMM is an efficient algorithm in distributed convex optimization and attracts more and more attraction in recent years because of its superior performance in big data analysis and machine learning. This algorithm is generally useful in splitting the global problem into subproblems

and solve the parallel computing of those subproblems instead of the large problem itself. In this way, this algorithm is viewed as an efficient approach to solve big data analysis with more applications with it and is widely used in statistics and machine learning now.

The ADMM was first developed by Gabay and Mercier (1975) based on the use of an Augmented Lagrangian and applied in continuum mechanics. The Augmented Lagrangian and multipliers was introduced by Hestenes (1969) at the beginning. The related numerical examples were analyzed by Miele et al. (1971) at the period. Gabay and Mercier (1975) proved the existence of unique solution (saddle point) to the augmented Lagrangian with dual approach using the class of perturbations defined in Rockafellar (1973) and Fortin (1975). He also proved the convergence of the solution of ADMM under some mild conditions on the objective function and applied the algorithm in several applications such as minimal hypersurfaces problem, Bingham fluids, Elasto-plastic torsion of a cylindrical bar problem, non-linear dirichlet problem and so on and compared this proposed method to some other existing methods to show its efficiency. In the next 10 years after that, there were several papers discussing the applications and extending proofs of convergence of ADMM. Gabay (1983) discussed many variations of ADMM and showed the proof of convergence for those variant forms such as multiplier methods for variational inequalities, the Douglas-Rachford variant of the method of multipliers which was first introduced by Mercier (1979), the peaceman-Rachford variant of the method of multipliers and so on. It was further developed later by Bertsekas (2014). He mentioned that the ADMM for convex programming decomposition is a special case of Douglas-Rachford splitting which is again a special case of proximal point algorithm that was first introduced by Rockafellar (1976) and derived a generalized ADMM by specifying the connection between Douglas-Rachford splitting and the proximal point algorithm. Fukushima (1992) presented a new decomposition algorithm for solving the separable convex programming problem based on ADMM. Eckstein and Fukushima (1994) used the ADMM to derive three convex programming decomposition methods including the problems with multiple set constraints, the epigraphic projection method for problems with block separable structure and the alternating step method for monotropic programming. Some other papers mentioned different applications and variations of ADMM. For example, Fortin and Glowinski (2000) discussed the application on

boundary-value problems, Glowinski and Le Tallec (1987) proposed the applications on variational problems, Tseng (1991) worked on the application of ADMM with variational inequality problem. More details of applications and variations of ADMM can be found in Eckstein and Fukushima (1994), Chen and Teboulle (1994), and Bioucas-Dias and Figueiredo (2010).

With the development of data analysis, big data has attracted more and more attention in these years. There are many existing papers discussing the application of ADMM in data analysis with extremely high dimension or large sample size. Such big data problems are common in statistical analysis and machine learning which can be solved by ADMM. Yin et al. (2008) and Goldstein and Osher (2009) introduced the application of ADMM in comprehensive sensing to deal with the Basic Pursuit problem or compressed sensing problem. O'Donoghue et al. (2013) proposed the algorithm on linear convex optimal control problem relying on ADMM. Xue et al. (2012) presented an algorithm to find the positive-definite $L1$-penalized covariance estimator of sparse large covariance matrices based on ADMM and established its convergence properties. Bien et al. (2013) proposed the methods based on ADMM to produce the sparse estimates of parmeter in lasso problem with strong or weak hierarchy constraint by adding a set of convex constraints to the lasso. Bogdan et al. (2013) introduced a method for sparse regression and variable selection which is inspired by multiple testing and relied on the idea of ADMM with canonical model selection procedures. Zhang and Zou (2014) proposed the constrained loss minimization for estimating high dimensional sparse precision matrices and introcued a new loss function called D-trace loss to be solved by ADMM.

Boyd et al. (2011) reviewed the existing applications on ADMM and its corresponding solutions. This paper introduced the derivation of ADMM, its convergence properties and also showed many numerical examples by applying the distributed ADMM. It introduced the application of ADMM on constrained convex optimization problem with both linear and quadratic programming, the problem with $L1$-norms like LAD, basis pursuit problem, general $L1$ regularized loss minimization problem and several lasso penalty problems, the global variable consensus optimization problem and sharing problem, the general distributed model fitting in regression, classification, logistic regression, SVM and additive models, and several nonconvex problems such as regressor selection and factor model. It also showed many implementations

to the existing algorithms and ran several numerical examples to show its good performance in convergence rate and accuracy.

In the recent paper Gaines et al. (2018), the authors showed the algorithm of using ADMM to solve LASSO optimization problem with equality and inequality constraint and mentioned the generalized LASSO problem can be converted to the constrained LASSO. However, this paper only considered the way to add an indicator function to all the constraints together. In the ADMM algorithm proposed in this dissertation, the indicator function is only used for inequality constraint and the solution will be simpler. Moreover, Gaines et al. (2018) didn't consider the possibility to split the big data along n with parallel computing which could decrease the computational cost dramatically especially for big data.

Chapter 3

ADMM Algorithms

In this chapter, I will mainly derive several ADMM algorithms for solving the generalized LASSO problem with linear constraints in (1.1). Recall that Gaines et al. (2018) formulate (1.1) in the form of (2.1) by introducing an indicator function for the feasible region determined by the constraints. However, it is better to formulate (1.1) in the form of the following:

$$\min_{x,y,z} f(x) + g(z) + h(y), \quad \text{subject to} \quad Ax + Bz + Cy = d, \tag{3.1}$$

where $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, and $y \in \mathbb{R}^s$ are variables, $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are known convex functions, and $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$, $C \in \mathbb{R}^{p\times s}$, and $d \in \mathbb{R}^p$ are known matrices or vectors.

3.1 Extended ADMM With Slack Variables

By introducing new variables $z = D\beta$ and slack variable $\omega = C\beta - d$, the problem (1.1) is equivalent to:

$$\begin{aligned} &\min_{\beta,z,\omega} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega), \\ &\text{subject to} \quad D\beta = z, \quad C\beta - \omega - d = 0, \quad E\beta = f, \end{aligned} \tag{3.2}$$

where $\Phi_{R_+^q}(\omega) = 0$ if $\omega \geq 0$ holds componentwise and $= \infty$ otherwise. That is:

$$R_+^q = \{x \in \mathbb{R}^q | x_i \geq 0, i = 1, 2, \ldots, q\},$$

where

$$\Phi_{R_+^p}(\omega) = \begin{cases} 0 & \omega \in \mathbb{R}_+^p \\ \infty & otherwise \end{cases}$$

The constraints can be written as $A_1\beta + A_2 z + A_3\omega = b$, where

$$A_1 = \begin{bmatrix} D \\ C \\ E \end{bmatrix}, \quad A_2 = \begin{bmatrix} -I_m \\ 0_q \\ 0_s \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0_m \\ -I_q \\ 0_s \end{bmatrix}, \quad b = \begin{bmatrix} 0_m \\ d \\ f \end{bmatrix}.$$

Hence (1.1), or equivalently (3.2), exactly has the form (3.1). By introducing the dual variable ρ, the augmented Lagrangian function of (3.2) is

$$\begin{aligned} \mathcal{L}_\rho(\beta, z, \omega, U) =& \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega) + \rho U^T(A_1\beta + A_2 z + A_3\omega - b) \\ &+ \frac{\rho}{2}\|A_1\beta + A_2 z + A_3\omega - b\|_2^2, \end{aligned}$$

where $U^T = (u^T, v^T, w^T)$ and $u \in \mathbb{R}^m$, $v \in \mathbb{R}^q$, $w \in \mathbb{R}^s$ are the (scaled) Lagrangian multipliers and $\rho > 0$ is a user-specified constant. Given $\beta^{(k)}$, $z^{(k)}$, $\omega^{(k)}$, and $U^{(k)}$ as the current solution to (3.2), the updating steps are as follows:

$$\begin{aligned} \beta^{(k+1)} &= \arg\min_\beta \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}\|A_1\beta + A_2 z^{(k)} + A_3\omega^{(k)} - b + U^{(k)}\|_2^2, \\ z^{(k+1)} &= \arg\min_z \lambda\|z\|_1 + \frac{\rho}{2}\|A_1\beta^{(k+1)} + A_2 z + A_3\omega^{(k)} - b + U^{(k)}\|_2^2, \\ \omega^{(k+1)} &= \arg\min_\omega \Phi_{R_+^q}(\omega) + \frac{\rho}{2}\|A_1\beta^{(k+1)} + A_2 z^{(k+1)} + A_3\omega - b + U^{(k)}\|_2^2, \\ U^{(k+1)} &= U^{(k)} + A_1\beta^{(k+1)} + A_2 z^{(k+1)} + A_3\omega^{(k+1)} - b. \end{aligned}$$

By simplifying the updating steps for each of the variable, we can update the estimates in the following way:

$$\begin{aligned}
\beta^{(k+1)} =& \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}\|D\beta - z^{(k)} + u^{(k)}\|_2^2 \\
&+ \frac{\rho}{2}\|C\beta - \omega^{(k)} - d + v^{(k)}\|_2^2 + \frac{\rho}{2}\|E\beta - f + w^{(k)}\|_2^2, \\
z^{(k+1)} =& \arg\min_{z} \lambda\|z\|_1 + \frac{\rho}{2}\|D\beta^{(k+1)} - z + u^{(k)}\|_2^2, \\
\omega^{(k+1)} =& \arg\min_{\omega} \Phi_{R_+^q}(\omega) + \frac{\rho}{2}\|C\beta^{(k+1)} - \omega - d + v^{(k)}\|_2^2, \\
u^{(k+1)} =& u^{(k)} + D\beta^{(k+1)} - z^{(k+1)}, \\
v^{(k+1)} =& v^{(k)} + C\beta^{(k+1)} - \omega^{(k+1)} - d, \\
w^{(k+1)} =& w^{(k)} + E\beta^{(k+1)} - f.
\end{aligned}$$

All the optimization problems for updating $\beta^{(k+1)}$, $z^{(k+1)}$, and $\omega^{(k+1)}$ have explicit solutions, which are discussed in details as follows.

The optimization problem for updating $\beta^{(k+1)}$ is a least squares problem with an L_2-penalty function. Some calculations yield

$$\begin{aligned}
\beta^{(k+1)} =& [X^TX + \rho(D^TD + C^TC + E^TE)]^{-1}\{X^Ty + \rho[D^T(z^{(k)} - u^{(k)}) \\
&+ C^T(\omega^{(k)} + d - v^{(k)}) + E^T(f - w^{(k)})]\}. \qquad (3.3)
\end{aligned}$$

The calculation of $[X^TX + \rho(D^TD + C^TC + E^TE)]^{-1}$ may be time-consuming when the dimension p is large. However, because its value remains constant during iterations, we may compute and cache it in the initialization step to speed up the calculation.

The optimization problem for updating $z^{(k+1)}$ can be solved componentwise. In fact,

$$\begin{aligned}
z^{(k+1)} &= \arg\min_{z} \lambda\|z\|_1 + \frac{\rho}{2}\|D\beta^{(k+1)} - z + u^{(k)}\|_2^2 \\
&= \arg\min_{z} \sum_{j=1}^{m}\{\lambda|z_j| + \frac{\rho}{2}(D_j\beta^{(k+1)} - z_j + u_j^{(k)})^2\},
\end{aligned}$$

where z_j and $u_j^{(k)}$ are the jth component of z and $u^{(k)}$, repsectively, and D_j is the jth row of matrix D. In the last expression, the objective function is separable with respect to $z_j's$, which can be calculated separately. Hence, we can find the explicit expression of updating steps of z as

$$z^{(k+1)} = (1 - \frac{\frac{\lambda}{\rho}}{|D\beta^{(k+1)} + u^{(k)}|})_+(D\beta^{(k+1)} + u^{(k)}),$$

where $(x)_+ = x$ if $x \geq 0$ and $= 0$ otherwise.

The optimization problem for updating $\omega^{(k+1)}$ has an explicit solution due to the special structure of function $\Phi_{R_+^q}(\cdot)$. Based on the simplified version, it's not hard to find

$$\omega^{(k+1)} = (C\beta^{(k+1)} - d + v^{(k)})_+,$$

Hence, starting with an initial value of $\beta^{(0)}$, $z^{(0)}$, $\omega^{(0)}$, and $U^{(0)} = (u^{(0)}, v^{(0)}, w^{(0)})$, the above derived formulas are used to update the solution iteratively until convergence. The stopping criteria is usually determined by comparing the length of residual vectors to some thresholds. In this problem, the stopping criteria are:

$$\|r_{prim}^{(k+1)}\|_2 \leq \epsilon^{prim}, \quad \|s_{dual}^{(k+1)}\|_2 \leq \epsilon^{dual},$$

where r_{prim} and s_{dual} are primary and dual residuals, respectively, given by

$$r_{prim}^{(k+1)} = A_1\beta^{(k+1)} + A_2z^{(k+1)} + A_3\omega^{(k+1)} - b = \begin{bmatrix} D\beta^{(k+1)} - z^{(k+1)} \\ C\beta^{(k+1)} - \omega^{(k+1)} - d \\ E\beta^{(k+1)} - f \end{bmatrix},$$

$$s_{dual}^{(k+1)} = \rho A_1^T A_3(\omega^{(k)} - \omega^{(k+1)}) + \rho A_1^T A_2(z^{(k)} - z^{(k+1)}) = \rho C^T(\omega^{(k)} - \omega^{(k+1)}) + \rho D^T(z^{(k)} - z^{(k+1)}).$$

Notice that ϵ^{prim} and ϵ^{dual} are some user specified positive numbers. A reasonable choice is given as follows.

$$\epsilon^{prim} = \sqrt{m+q+s}\epsilon^{abs} + \epsilon^{rel} max\{\|A_1\beta^{(k+1)}\|_2, \|A_2 z^{(k+1)}\|_2, \|A_3\omega^{(k+1)}\|_2, \|b\|_2\},$$
$$\epsilon^{dual} = \sqrt{p}\epsilon^{abs} + \epsilon^{rel}\|\rho A_1^T U^{(k+1)}\|_2$$

where ϵ^{abs} and ϵ^{rel} are absolute tolerance and relative tolerance, respectively. In practice, it is common to select both tolerances to be 10^{-3}.

The ADMM algorithm proposed in Gaines et al. (2018) solves (1.1) in a quite different way. Firstly, it formulates (1.1) in the form of (2.1). Next, both equality and inequality constraints are incorporated in the objective function as an indicator function and the resulting iteration involves solving two optimization problems, namely, a lasso problem and a projection problem onto an affine space, where both optimization problems have no explicit solutions in general. In contrast, the proposed iteration above has an explicit expression in every step and is expected to be faster than the algorithm in Gaines et al. (2018), which is confirmed in the simulation studies in Chapter 4.

The proposed algorithm solves (1.1) for a fixed value of the tuning parameter λ. In practice, the value of λ is selected adaptively to data. Usually, we first select a sequence of equally spaced values of λ_1, λ_2, ..., λ_k in an interval. Use the proposed algorithm solves (1.1) for $\lambda = \lambda_1$ and then solve the problem with the next λ using the estimated value of β solved from the previous λ as a warm start, that is, the solution at $\lambda = \lambda_i$ is used as the initial value to find the solution at $\lambda = \lambda_{i+1}$.

The proposed algorithm is below:

Algorithm 1: ADMM algorithm with slack variable

Result: β

initialization; **while**

$$\|r_{prim}^{(k+1)}\|_2 > \epsilon^{prim} \quad or \quad \|s_{dual}^{(k+1)}\|_2 > \epsilon^{dual}$$

do

$$\begin{aligned}
\beta^{(k+1)} =& [X^T X + \rho(D^T D + C^T C + E^T E)]^{-1}\{X^T y + \rho[D^T(z^{(k)} - u^{(k)}) \\
&+ C^T(\omega^{(k)} + d - v^{(k)}) + E^T(f - w^{(k)})]\} \\
z^{(k+1)} =& (1 - \frac{\frac{\lambda}{\rho}}{|D\beta^{(k+1)} + u^{(k)}|})_+(D\beta^{(k+1)} + u^{(k)}) \\
\omega^{(k+1)} =& (C\beta^{(k+1)} - d + v^{(k)})_+ \\
u^{(k+1)} =& u^{(k)} + D\beta^{(k+1)} - z^{(k+1)} \\
v^{(k+1)} =& v^{(k)} + C\beta^{(k+1)} - \omega^{(k+1)} - d \\
w^{(k+1)} =& w^{(k)} + E\beta^{(k+1)} - f
\end{aligned}$$

end

3.2 Extended ADMM With Large p

This section derives the revised ADMM algorithm when the dimension p is higher than the sample size n. In this scenario, the new constraint will be added to the extension expression in (3.2) in order to simplify the term in the update of β. The main idea is to get rid of calculating the inverse of $p \times p$ matrix $X^T X$, $C^T C$, $D^T D$ and $E^T E$ when the dimension p is very large by introducing new constraints and convert the inverse matrix into other approximate term. We have mentioned in the previous section that the result of the inverse matrix remains constant during iterations. We may compute and cache it in the initialization step to speed up the calculation. However, when p is much larger than n, the calculation of the inverse matrix in the update of β will consume much more time since the dimesnions of $X^T X$, $D^T D$, $C^T C$ and $E^T E$ are extremely large as $p \times p$. In order to convert the matrix $X^T X$ into other format,

according to Zhu (2017), we first need to get rid of the other 3 matrices C^TC, D^TD and E^TE in (3.3) and apply the proximal map technique to the update step of β to avoid the calculation of the matrix X^TX.

Recall that in the form of (3.2), there were 3 equality constraints by introducing the slack variables. According to Zhu (2017), by adding the following new constraint

$$(M - C^TC)^{\frac{1}{2}}\beta = \tilde{s},$$

where matrix $M \in \mathbb{R}^{p\times p}$ satisfies $M \succeq C^TC$, the form of (3.2) can be converted to the following

$$\begin{aligned}
&\min_{\beta,z,\omega} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),\\
&\text{subject to} \quad D\beta = z, \quad C\beta - \omega - d = 0, \quad E\beta = f, \quad (M - C^TC)^{\frac{1}{2}}\beta = \tilde{s}.
\end{aligned}$$

The constraints can be written as $A_1^*\beta + A_2^*z + A_3^*\omega = b^*$, where

$$A_1^* = \begin{bmatrix} D \\ C \\ E \\ (M - C^TC)^{\frac{1}{2}} \end{bmatrix}, \quad A_2^* = \begin{bmatrix} -I_m \\ 0_q \\ 0_s \\ 0_s \end{bmatrix}, \quad A_3^* = \begin{bmatrix} 0_m \\ -I_q \\ 0_s \\ 0_s \end{bmatrix}, \quad b^* = \begin{bmatrix} 0_m \\ d \\ f \\ \tilde{s} \end{bmatrix}.$$

Hence this new optimization problem has the form of (3.1) as well. By introducing the dual variable ρ, the augmented Lagrangian function is

$$\begin{aligned}
\mathcal{L}_\rho(\beta, z, \omega, U) =& \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega) + \rho U^{*T}(A_1^*\beta + A_2^*z + A_3^*\omega - b^*)\\
&+ \frac{\rho}{2}\|A_1^*\beta + A_2^*z + A_3^*\omega - b^*\|_2^2,
\end{aligned}$$

where $U^{*T} = (u^T, v^T, w^T, \tilde{u}^T)$ and $u \in \mathbb{R}^m$, $v \in \mathbb{R}^q$, $w \in \mathbb{R}^s$, $\tilde{u} \in \mathbb{R}^s$ are the (scaled) Lagragian multipliers and $\rho > 0$ is a user-specified constant. Given $\beta^{(k)}$, $z^{(k)}$, $\omega^{(k)}$, and $U^{*(k)}$

as the current solution to (3.2), the updating steps are as follows:

$$\beta^{(k+1)} = \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}\|A_1^*\beta + A_2^* z^{(k)} + A_3^*\omega^{(k)} - b^* + U^{*(k)}\|_2^2,$$

$$z^{(k+1)} = \arg\min_{z} \lambda\|z\|_1 + \frac{\rho}{2}\|A_1^*\beta + A_2^* z^{(k)} + A_3^*\omega^{(k)} - b^* + U^{*(k)}\|_2^2,$$

$$\omega^{(k+1)} = \arg\min_{\omega} \Phi_{R_+^q}(\omega) + \frac{\rho}{2}\|A_1^*\beta + A_2^* z^{(k)} + A_3^*\omega^{(k)} - b^* + U^{*(k)}\|_2^2,$$

$$U^{*(k+1)} = U^{*(k)} + A_1^*\beta^{(k+1)} + A_2^* z^{(k+1)} + A_3^*\omega^{(k+1)} - b^*.$$

By simplifying the updating steps for each of the variable, we can update the estimates in the following way:

$$\begin{aligned}
\beta^{(k+1)} =& \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}\|D\beta - z^{(k)} + u^{(k)}\|_2^2 + \frac{\rho}{2}\|C\beta - \omega^{(k)} - d + v^{(k)}\|_2^2 \\
&+ \frac{\rho}{2}\|E\beta - f + w^{(k)}\|_2^2 + \frac{\rho}{2}\|(M - C^TC)^{\frac{1}{2}}\beta - \tilde{s}^{(k)} + \tilde{u}^{(k)}\|_2^2, \qquad (3.4) \\
z^{(k+1)} =& \arg\min_{z} \lambda\|z\|_1 + \frac{\rho}{2}\|D\beta^{(k+1)} - z + u^{(k)}\|_2^2, \\
\omega^{(k+1)} =& \arg\min_{\omega} \Phi_{R_+^q}(\omega) + \frac{\rho}{2}\|C\beta^{(k+1)} - \omega - d + v^{(k)}\|_2^2, \\
\tilde{s}^{(k+1)} =& (M - C^TC)^{\frac{1}{2}}\beta^{(k+1)} + \tilde{u}^{(k)}, \qquad (3.5) \\
u^{(k+1)} =& u^{(k)} + D\beta^{(k+1)} - z^{(k+1)}, \\
v^{(k+1)} =& v^{(k)} + C\beta^{(k+1)} - \omega^{(k+1)} - d, \qquad (3.6) \\
w^{(k+1)} =& w^{(k)} + E\beta^{(k+1)} - f. \\
\tilde{u}^{(k+1)} =& \tilde{u}^{(k)} + (M - C^TC)^{\frac{1}{2}}\beta^{(k+1)} - \tilde{s}^{(k+1)}. \qquad (3.7)
\end{aligned}$$

Instead of finding the solution of these optimization problems, we do transformations using the above equations to derive a new group of optimization equations which has different optimization form in the update of β. Firstly, by combining equation (3.5) and equation (3.7),

$$\tilde{u}^{(k+1)} = 0, \tag{3.8}$$

$$\tilde{s}^{(k)} = (M - C^T C)^{\frac{1}{2}} \beta^{(k)}. \tag{3.9}$$

Notice that the newly added dual variable $\tilde{u}$ is 0 which can simplify the representation of the added variable $\tilde{s}$. We apply the results of (3.8) and equation (3.9) to the optimization step of β in (3.4)

$$\begin{aligned}\beta^{(k+1)} = \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}(\|C\beta - \omega^{(k)} - d + u^{(k)}\|_2^2 + \|E\beta - f + v^{(k)}\|_2^2 \\ + \|D\beta - z^{(k)} + w^{(k)}\|_2^2 + \|(M - C^T C)^{\frac{1}{2}}(\beta - \beta^{(k)})\|_2^2).\end{aligned} \tag{3.10}$$

After this transformation and simplification, the optimization problem of β no longer has any term with $\tilde{s}$. In fact, by introducing this transformation, the newly added constraints has nothing to do with the optimization problems which can be ignored and the optimization problems could be reduced to the one with only 3 constraints similar to (3.2). However, we introduce this new constraint in order to get rid of the complicated matrices in the inverse part in equation (3.3). Since this new matrix M has relationship with $C^T C$, based on the idea in Zhu (2017), we are trying to do one more transformation using (3.10) to convert the term $C^T C$ into some other term. In fact, from the optimization equation (3.6),

$$-\omega^{(k)} = v^{(k)} - v^{(k-1)} - C\beta^{(k)} + d, \tag{3.11}$$

which is a new representation of ω. In order to get rid of the term $C^T C$ in the update of β, we are trying to plug in the new equation of ω in its optimization problem and try to simplify the original form to be one without $C^T C$. In fact, by substituting the result of (3.11) into equation

(3.10),

$$\begin{aligned}\beta^{(k+1)} =& \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}(\|E\beta - f + v^{(k)}\|_2^2 + \|D\beta - z^{(k)} + w^{(k)}\|_2^2 \\ &+ (\beta - \beta^{(k)})^T M(\beta - \beta^{(k)}) + 2C^T(2v^{(k)} - v^{(k-1)})\beta),\end{aligned}$$

the term C^TC will be no longer in the optimization problem of β. In order to finally convert the matrix X^TX to some other term due to the enormous dimension p, more efforts need to be paid to deal with the D^TD and E^TE as well. By adding two more new constraints,

$$\begin{aligned}(P - D^TD)^{\frac{1}{2}}\beta &= \tilde{s_1}, \\ (Q - E^TE)^{\frac{1}{2}}\beta &= \tilde{s_2},\end{aligned}$$

where matrix $P \in \mathbb{R}^{p\times p}$ satisfies $P \succeq D^TD$, and matrix $Q \in \mathbb{R}^{p\times p}$ satisfies $Q \succeq E^TE$, the same transformations could be applied to each constraint to simplify the update of β while without adding any new variable in the optimization system. In fact, (3.3) becomes

$$\begin{aligned}\beta^{(k+1)} =& \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}(\beta - \beta^{(k)})^T(M + P + Q)(\beta - \beta^{(k)}) \\ &+ \rho D^T(2u^{(k)} - u^{(k-1)})\beta + \rho C^T(2v^{(k)} - v^{(k-1)})\beta + \rho E^T(2w^{(k)} - w^{(k-1)})\beta,\end{aligned}$$

after adding all 3 new constraints. In the finalized update of β, there is no term associated with any of C^TC, D^TD or E^TE. In fact, by solving the optimization problem for updating β which is a least square problem with an L_2-penalty function, the calculus yields

$$\begin{aligned}\beta^{k+1} =& [X^TX + \rho(M + P + Q)]^{-1}[X^Ty + \rho(M + P + Q)\beta^k \\ &- \rho D^T(2u^{(k)} - u^{(k-1)}) - \rho C^T(2v^{(k)} - v^{(k-1)}) - \rho E^T(2w^{(k)} - w^{(k-1)})].\end{aligned}$$

In the inverse part, the matrix M, P, Q are user-specified matrix satisfying the condition in the constraints. No matter how huge the dimension p is, we could always construct some "good" matrices to make the inverse easy to calculate. The only matrix left to be simplified is the X^TX

in this case. According to Zhu (2017), if the update of θ is given as follows

$$\theta^{(k+1)} = \arg\min_{\theta}(f(\theta) + (2\alpha^{(k)} - \alpha(k-1))^T A\theta + \frac{\rho}{2}(\theta - \theta^{(k)})^T D(\theta - \theta^{(k)}),$$

which satisfy $D \succeq A^T A$. By selecting $D = \delta I$ with $\delta \geq \|A\|_{op}^2$, where $\|A\|_{op}$ denotes the operator norm of A, the θ-update can be view as a proximal map of $f(\cdot)$

$$\theta^{(k+1)} = prox_{(\rho\delta)^{-1}f}\{\theta^{(k)} - (\rho\delta)^{-1}A^T(2\alpha^{(k)} - \alpha^{(k-1)})\},$$

where

$$prox_{\Omega(\cdot)}(u) = \min_{v}(\Omega(v) + \frac{1}{2}\|u - v\|_2^2),$$

for a given function $\Omega(\cdot)$. Based on this idea, the update of β can be rewritten as

$$\beta^{(k+1)} = \arg\min_{\beta}\{\|y - X\beta\|_2^2 + \rho A_1^T(2\alpha^{(k)} - \alpha^{(k-1)})\beta + \frac{\rho}{2}(\beta - \beta^{(k)})^T A_2^T(\beta - \beta^{(k)})\},$$

where

$$A_1 = \begin{bmatrix} D \\ C \\ E \end{bmatrix} \quad , \quad A_2 = \begin{bmatrix} P^T \\ M^T \\ Q^T \end{bmatrix} \quad \text{and} \quad \alpha = \begin{bmatrix} u \\ v \\ w \end{bmatrix}.$$

Denote $f(\beta) = \|y - X\beta\|_2^2$. By introducing $\delta = \|A_1\|_{op}^2$ and $A_2^T = \delta A_1^T$, the update step of β is as follows

$$\beta^{(k+1)} = prox_{(\rho\delta)^{-1}f}\{\beta^{(k)} - (\rho\delta)^{-1}A_1(2\alpha^{(k)} - \alpha^{(k-1)})\}$$

By introducing new constraints, the problem can be converted to the proximal map of a convex function $f(\cdot)$ which is much easier to calculate than the inverse matrix and the computational cost could be reduced dramatically when dimension p is extremely large.

The optimization problem of z, ω and dual variables u, v and w have the same result as derived in section 3.1 which will be skipped here. Notice that by adding new constraints, the optimization keep the same except that the update step of β is converted in a different form. In fact, the stopping criteria is also the same as the previous algorithm since the optimization problem doesn't change and we only used a different function to update β.

The finalized extended ADMM algorithm with large p is given below:

Algorithm 2: ADMM algorithm with large p ($p > n$)

Result: β

initialization; **while**

$$\|r_{prim}^{(k+1)}\|_2 > \epsilon^{prim} \quad or \quad \|s_{dual}^{(k+1)}\|_2 > \epsilon^{dual}$$

do

$$\beta^{(k+1)} = prox_{(\rho\delta)^{-1}f}\{\beta^{(k)} - (\rho\delta)^{-1}A_1(2\alpha^{(k)} - \alpha^{(k-1)})\}$$

$$z^{(k+1)} = (1 - \frac{\frac{\lambda}{\rho}}{|D\beta^{(k+1)} + u^{(k)}|})_+(D\beta^{(k+1)} + u^{(k)})$$

$$\omega^{(k+1)} = (C\beta^{(k+1)} - d + v^{(k)})_+$$

$$u^{(k+1)} = u^{(k)} + D\beta^{(k+1)} - z^{(k+1)}$$

$$v^{(k+1)} = v^{(k)} + C\beta^{(k+1)} - \omega^{(k+1)} - d$$

$$w^{(k+1)} = w^{(k)} + E\beta^{(k+1)} - f$$

end

3.3 Distributed ADMM Splitting Along n

From this section, we will mainly derive several distributed ADMM algorithm for solving (1.1). It is necessary to introduce some terms regarding distributed computing for ease of presentation. Assume that the whole system consisted of B processors. Different processor may correspond to different cores in a single computer, different computers in a cluster, or different nodes in a supercomputer. Equipped with its own CPU and memory, each processor can process or compute its own share of data independently from the other processors. There exists a protocol among the processors which allows the processors to communicate with each other. The communication indicates that some data can be exchanged among the processors. It is important to point out that the data exchanged among the processors are not the original raw data, but the intermediate results calculated during the computing process instead. Roughly speaking, in a typical algorithm of distributed computing, each processor independently processes its own data, intermediate results are then collected and aggregated together to produce a final result. The final result may be used as an initial value to repeat the above process if necessary.

Assume that the whole data are partitioned into B subsets, where the bth subset $\{y_b, X_b\}$ is stored in the bth processor. The sample size for the bth subset is n_b while the total sample size is $n = \sum_{b=1}^{B} n_b$. Based on these notations, the origianl problem (1.1) can be rewritten as

$$\min_{\beta,z} \sum_{b=1}^{B} \frac{1}{2}\|y_b - X_b\beta\|_2^2 + \lambda\|z\|_1, \quad \text{subject to } C\beta \geq d, E\beta = f. \tag{3.12}$$

Note that the parameter β and the constraints remain the same for all processors. There are several scenarios where the problem (3.12) is applicable. For example, the whole data are too large to be saved in one location or the origianl data are collected at different locations or different time period. Another popular scenario is privacy preserving. Recall the example introduced in Chapter 1 that several companies want to build a model using all data collected by them separately. However, none of the companies want to share their raw data with others due to privacy. The distributed algorithm then becomes a natural choice, which only requires

the share of intermediate calculation results, but not the original raw data to produce the global result.

In distributed computing, we will not transfer all subsets $\{y_b, X_b\}$, $b = 1, 2, \ldots, B$ to a central location, hence the algorithms derived in the previous sections are not applicable. It is necessary to derive a distributed algorithm to solve (3.12). In this algorithm, each processor only processes the local data stored in the processor. All processors needs to communicate with each other to exchange information. Roughly speaking, during each iteration, there are two major steps which includes a local step and a global step. In a local step, each processor processes the local raw data to generate the intermediate result, while in the global step, all results obtained by all processors are aggregated to produce a final update.

Firstly, we rewrite the problem (3.12) as a global consensus problem,

$$\min_{\beta,z,\omega} \sum_{b=1}^{B} \frac{1}{2}\|y_b - X_b\beta_b\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$

$$\text{subject to } D\beta_b = z, \quad C\beta_b - \omega = d, \quad E\beta_b = f, \quad \beta_b = \beta, \quad b = 1, 2, \ldots, B.$$

The augmented Lagrangian function is

$$\begin{aligned}\mathcal{L}_\rho(\beta_b, z, \omega, \beta, u_b, v_b, w_b, h_b) = &\sum_{b=1}^{B} \frac{1}{2}\|y_b - X_b\beta_b\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega) \\ &+ \sum_{b=1}^{B}\{\rho {u_b}^T(D\beta_b - z) + \frac{\rho}{2}\|D\beta_b - z\|_2^2 \\ &+ \rho {v_b}^T(C\beta_b - \omega - d) + \frac{\rho}{2}\|C\beta_b - \omega - d\|_2^2 \\ &+ \rho {w_b}^T(E\beta_b - f) + \frac{\rho}{2}\|E\beta_b - f\|_2^2 \\ &+ \rho {h_b}^T(\beta_b - \beta) + \frac{\rho}{2}\|\beta_b - \beta\|_2^2\}.\end{aligned}$$

In this problem, the local variables include $\beta_b's$, $U_b's$ where $U_b = (u_b, v_b, w_b, \beta_b)$, which are unique to each processor, while the global variables include z, ω and β, which are the same across all processors. In one typical interation, the local variables are updated within each

processor and then the global variables are updated together and simultaneously across all processors.

We update the estimators in the order of $U_b^{(k+1)}$, $\beta_b^{(k+1)}$, $z^{(k+1)}$, and $\omega^{(k+1)}$. Notice that the first two variables are local variable and the remaining three variables are global variables. The detailed calculation formulas are derived in the following.

The variable $U_b^{(k+1)}$ is updated in the bth processor using the same technique as previous sections.

$$
\begin{aligned}
u_b^{(k+1)} &= u_b^{(k)} + D\beta_b^{(k+1)} - z^{(k+1)}, \\
v_b^{(k+1)} &= v_b^{(k)} + C\beta_b^{(k+1)} - \omega^{(k+1)} - d, \\
w_b^{(k+1)} &= w_b^{(k)} + E\beta_b^{(k+1)} - f, \\
h_b^{(k+1)} &= h_b^{(k)} + \beta_b^{(k+1)} - \beta^{(k+1)}.
\end{aligned}
$$

The variable $\beta_b^{(k+1)}$ is updated in the bth processor.

$$
\begin{aligned}
\beta_b^{(k+1)} = &\arg\min_{\beta_b} \frac{1}{2}\|y_b - X_b\beta_b\|_2^2 + \frac{\rho}{2}\|D\beta_b - z^{(k)} + u_b^{(k)}\|_2^2 \\
&+ \frac{\rho}{2}\|C\beta_b - \omega^{(k)} - d + v_b^{(k)}\|_2^2 + \frac{\rho}{2}\|E\beta_b - f + w_b^{(k)}\|_2^2 \\
&+ \frac{\rho}{2}\|\beta_b - \beta^{(k)} + h_b^{(k)}\|_2^2,
\end{aligned}
$$

which is a quadratic function of β_b. The update formula is then given as

$$
\begin{aligned}
\beta_b^{(k+1)} = &[X_b^T X_b + \rho(C^T C + D^T D + E^T E + \rho I_p)]^{-1}[X_b^T y_b + \rho D^T(z^{(k)} - u_b^{(k)}) \\
&+ \rho C^T(\omega^{(k)} + d - v_b^{(k)}) + \rho E^T(f - w_b^{(k)}) + \rho(\beta^{(k)} - h_b^{(k)})].
\end{aligned}
$$

Compared with the formula to update $\beta^{(k+1)}$ in section 3.1, there is an extra penalty term $\|\beta_b - \beta^{(k)} + h_b^{(k)}\|_2^2$ in the objective function. This penalty tries to drive $\beta_b^{(k+1)}$ closer to $\beta^{(k)}$ and hence make $\beta_b^{(k+1)}$ computed by different processors get closer to each other.

The variable $\beta^{(k+1)}$ is computed globally for all processors.

$$\begin{aligned}\beta^{(k+1)} &= \arg\min_{\beta} \sum_{b=1}^{B} \frac{\rho}{2} \|\beta_b^{(k+1)} - \beta + h_b^{(k+1)}\|_2^2 \\ &= \frac{1}{B} \sum_{b=1}^{B} (\beta_b^{(k+1)} + h_b^{(k+1)}) = \bar{\beta}^{(k+1)} + \bar{h}^{(k+1)},\end{aligned}$$

where

$$\bar{\beta}^{(k+1)} = \frac{1}{B} \sum_{b=1}^{B} \beta_b^{(k+1)}, \quad \bar{h}^{(k)} = \frac{1}{B} \sum_{b=1}^{B} h_b^{(k)}.$$

It is clear that $\beta^{(k+1)}$ is essentially an average of all $\beta_b^{(k+1)}$ computed by different processors.

The variable $z^{(k+1)}$ is updated globally for all processors.

$$\begin{aligned}z^{(k+1)} &= \arg\min_{z} \lambda \|z\|_1 + \sum_{b=1}^{B} \frac{\rho}{2} \|D\beta_b^{(k+1)} - z + u_b^{(k+1)}\|_2^2 \\ &= (1 - \frac{\frac{\lambda}{\rho B}}{|D\bar{\beta}^{(k+1)} + \bar{u}^{(k+1)}|})_+ (D\bar{\beta}^{(k+1)} + \bar{u}^{(k+1)}),\end{aligned}$$

where

$$\bar{u}^{(k)} = \frac{1}{B} \sum_{b=1}^{B} u_b^{(k)}.$$

It is clear to see that the technique to find the update of z in distributed scenario is based on the same idea as section 3.1 while the intermediate results are aggregating from local processors.

The variable $\omega^{(k+1)}$ is updated globally.

$$\begin{aligned}\omega^{(k+1)} &= \arg\min_{\omega} \Phi_{R_+^q}(\omega) + \sum_{b=1}^{B} \frac{\rho}{2} \|C\beta_b^{(k+1)} - \omega - d + v_b^{(k+1)}\|_2^2 \\ &= (C\bar{\beta}^{(k+1)} - d + \bar{v}^{(k+1)})_+,\end{aligned}$$

where

$$\bar{v}^{(k)} = \frac{1}{B}\sum_{b=1}^{B} v_b^{(k)}.$$

We can see that the updating of $\beta_b^{(k+1)}$ and $U_b^{(k+1)}$ are computed with each processor locally, while the updating of $\beta^{(k+1)}$, $z^{(k+1)}$, and $\omega^{(k+1)}$ are computed globally.

The iteration will stop if the primal and dual residuals are small enough. It is important to point out that we need to consider the primal and dual residuals in different processors together in distributed algorithm. It means that all processors will stop iteration at the same time. In other words, one processor will stop the iteration if and only if all processors are converged. Based on the idea of stopping criteria introduced before, the primal and dual residuals for the bth processor are given by

$$r_{prim,b} = \begin{bmatrix} D\beta_b^{(k+1)} - z^{(k+1)} \\ C\beta_b^{(k+1)} - \omega^{(k+1)} - d \\ E\beta_b^{(k+1)} - f \\ \beta_b^{(k+1)} - \beta^{(k+1)} \end{bmatrix},$$

$$s_{dual,b} = \rho[D^T(z^{(k+1)} - z^{(k)}) + C^T(\omega^{(k+1)} - \omega^{(k)}) + (\beta^{(k+1)} - \beta^{(k)})].$$

The stopping rule is

$$\sqrt{\sum_{b=1}^{B} \|r_{prim,b}^{(k+1)}\|_2^2} \leq \epsilon^{prim}, \quad \sqrt{\sum_{b=1}^{B} \|s_{dual,b}^{(k+1)}\|_2^2} \leq \epsilon^{dual},$$

where

$$
\begin{aligned}
\epsilon^{prim} =& \sqrt{B(m+q+s+p)}\epsilon^{abs} \\
&+ \epsilon^{rel} max\{\sqrt{\sum_{b=1}^{B} \|A_1^* \beta_b^{(k+1)}\|_2^2}, \sqrt{B}\|A_2 z^{(k+1)}\|_2, \sqrt{B}\|A_3 \omega^{(k+1)}\|_2, \sqrt{B}\|\beta^{(k+1)}\|_2, \sqrt{B}\|b\|_2\} \\
=& \sqrt{B(m+q+s+p)}\epsilon^{abs} \\
&+ \epsilon^{rel} max\{\sqrt{\sum_{b=1}^{B} \|A_1^* \beta_b^{(k+1)}\|_2^2}, \sqrt{B}\|z^{(k+1)}\|_2, \sqrt{B}\|\omega^{(k+1)}\|_2, \sqrt{B}\|\beta^{(k+1)}\|_2, \sqrt{B}\|b\|_2\}, \\
\epsilon^{dual} =& \sqrt{Bp}\epsilon^{abs} + \epsilon^{rel}\sqrt{\sum_{b=1}^{B} \|\rho A_1^* U_b^{(k+1)}\|_2^2},
\end{aligned}
$$

where matrix A_2, A_3 are as defined before in section 3.1 and

$$
A_1^* = \begin{bmatrix} D \\ C \\ E \\ I_p \end{bmatrix}.
$$

The finalized algorithm with distributed ADMM splitting along n is as follows:

Algorithm 3: Distributed ADMM algorithm splitting along n

Result: β

initialization; **while**

$$\sqrt{\sum_{b=1}^{B}\|r_{prim,b}^{(k+1)}\|_2^2} > \epsilon^{prim} \quad or \quad \sqrt{\sum_{b=1}^{B}\|s_{dual,b}^{(k+1)}\|_2^2} > \epsilon^{dual}$$

do

For bth processor: $(b = 1, 2, \ldots, B)$

$$\begin{aligned}
\beta_b^{(k+1)} =& [X_b^T X_b + \rho(C^T C + D^T D + E^T E + \rho I_p)]^{-1}[X_b^T y_b + \rho D^T(z^{(k)} - u_b^{(k)}) \\
&+ \rho C^T(\omega^{(k)} + d - v_b^{(k)}) + \rho E^T(f - w_b^{(k)}) + \rho(\beta^{(k)} - h_b^{(k)})] \\
u_b^{(k+1)} =& u_b^{(k)} + D\beta_b^{(k+1)} - z^{(k+1)} \\
v_b^{(k+1)} =& v_b^{(k)} + C\beta_b^{(k+1)} - \omega^{(k+1)} - d \\
w_b^{(k+1)} =& w_b^{(k)} + E\beta_b^{(k+1)} - f \\
h_b^{(k+1)} =& h_b^{(k)} + \beta_b^{(k+1)} - \beta^{(k+1)}
\end{aligned}$$

For all processors:

$$\begin{aligned}
\beta^{(k+1)} &= \frac{1}{B}\sum_{b=1}^{B}\beta_b^{(k+1)} + h_b^{(k+1)} = \bar{\beta}^{(k+1)} + \bar{h}^{(k+1)} \\
z^{(k+1)} &= (1 - \frac{\frac{\lambda}{\rho B}}{|D\bar{\beta}^{(k+1)} + \bar{u}^{(k+1)}|})_+(D\bar{\beta}^{(k+1)} + \bar{u}^{(k+1)}) \\
\omega^{(k+1)} &= (C\bar{\beta}^{(k+1)} - d + \bar{v}^{(k+1)})_+
\end{aligned}$$

end

3.4 Distributed ADMM Splitting Along p

This section derives another distributed ADMM algorithm for solving (1.1). Assume that the design matrix are partitioned into G subsets along its column, such that $X = \begin{bmatrix} X_1 & X_2 & \ldots & X_G \end{bmatrix}$ where the gth subset $\{X_g\}$ is stored in the gth processor. The sample size for gth subset is n, while the dimension (column) for the gth subset is p_g and the total dimension is $p = \sum_{g=1}^{G} p_g$.

We need to do the same partition to the parameter β along its row, and to the matrices C, D and E along their columns respectively to obtain the subsets $\{\beta_g\}$, $\{C_g\}$, $\{D_g\}$, and $\{E_g\}$, $g = \{1, 2, \ldots, G\}$ such that

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_G \end{bmatrix}, C = \begin{bmatrix} C_1 & C_2 & \ldots & C_G \end{bmatrix}, D = \begin{bmatrix} D_1 & D_2 & \ldots & D_G \end{bmatrix}, E = \begin{bmatrix} E_1 & E_2 & \ldots & E_G \end{bmatrix},$$

where the gth subset of β, C, D and E are stored in the gth processor. Notice that the product of matrix $X\beta$, $C\beta$, $D\beta$ and $E\beta$ can be written into summation of products

$$X\beta = \sum_{g=1}^{G} X_g\beta_g, \quad C\beta = \sum_{g=1}^{G} C_g\beta_g, \quad D\beta = \sum_{g=1}^{G} D_g\beta_g, \quad E\beta = \sum_{g=1}^{G} E_g\beta_g.$$

The original problem (1.1) can be written as

$$\begin{aligned} &\min_{\beta_g} \frac{1}{2}\|y - \sum_{g=1}^{G} X_g\beta_g\|_2^2 + \lambda\|D_g\beta_g\|_1, \\ &\text{subject to} \quad \sum_{g=1}^{G} C_g\beta_g \geq d, \quad \sum_{g=1}^{G} E_g\beta_g = f, \quad g = 1, 2, \ldots, G. \end{aligned} \tag{3.13}$$

Note that only the response y remain the same for all processors. There are several scenarios where the problem (3.13) is applicable. For example, the whole data are with enormous dimension to be saved in one location or the original data are collected at different groups or locations that the explanatory variables are restricted within the group and the groups are mutually exclusive to each other. In other words, each location and group only contains an unique subset of all predictors respectively and the span of all those subsets will be the whole data. Privacy preserving along dimension is also an appropriate scenario in this framework.

In distributed computing, we will not transfer all subsets $\{X_g\}$, $g = 1, 2, \ldots, G$ to a central location, hence the algorithm derived in section 3.1 is not applicable here either. It is necessary to derive a distributed algorithm for solving (3.13). In this algorithm, the same protocol of

the processors as section 3.3 is needed that all processors needs to communicate with each other to exchange the intermediate result. Roughly speaking, during one iteration, there are also two major steps, a local step and a global step. In the local step, each processor processes information from local data, such as X_g, D_g, C_g, E_g and β_g, while in the global step, all results obtained by all processors are aggregated to produce a final update.

Firstly, we rewrite the problem (3.13) as follows.

$$\min_{\beta_g} \frac{1}{2}\|y - \sum_{g=1}^{G} X_g\beta_g\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$
$$\text{subject to} \quad \sum_{g=1}^{G} D_g\beta_g = z, \quad \sum_{g=1}^{G} C_g\beta_g - \omega = d, \quad \sum_{g=1}^{G} E_g\beta_g = f, \quad g = 1, 2, \ldots, G.$$

The augmented Lagrangian function is

$$\begin{aligned}\mathcal{L}_\rho(\beta_g, z, \omega, \beta, u, v, w) =& \frac{1}{2}\|y - \sum_{g=1}^{G} X_g\beta_g\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega) \\ &+ \rho u^T(\sum_{g=1}^{G} D_g\beta_g - z) + \frac{\rho}{2}\|\sum_{g=1}^{G} D_g\beta_g - z\|_2^2 \\ &+ \rho v^T(\sum_{g=1}^{G} C_g\beta_g - \omega - d) + \frac{\rho}{2}\|\sum_{g=1}^{G} C_g\beta_g - \omega - d\|_2^2 \\ &+ \rho w^T(\sum_{g=1}^{G} E_g\beta_g - f) + \frac{\rho}{2}\|\sum_{g=1}^{G} E_g\beta_g - f\|_2^2.\end{aligned}$$

In this distributed problem, the local variable only includes $\beta_g's$, which is unique to each processor, while the global variables include z, ω, and U where $U = (u, v, w)$, which are the same across all processors. In one typical iteration, the local variable is updated within each processor and then the global variables are updated together and simultaneously across all processors.

We update the estimator in the order of $\beta_g^{(k+1)}$, $z^{(k+1)}$, $\omega^{(k+1)}$, and $U^{(k+1)}$. Notice that the first variable is local variable and the remaining three variables are global variables. The detailed calculation formulas are derived in the following.

The variable $\beta_g^{(k+1)}$ is updated in the gth processor.

$$\begin{aligned}\beta_g^{(k+1)} = &\arg\min_{\beta_g} \frac{1}{2}\|y - \sum_{g=1}^{G} X_g\beta_g\|_2^2 + \frac{\rho}{2}\|\sum_{g=1}^{G} D_g\beta_g - z^{(k)} + u^{(k)}\|_2^2 \\ &+ \frac{\rho}{2}\|\sum_{g=1}^{G} C_g\beta_g - \omega^{(k)} - d + v^{(k)}\|_2^2 + \frac{\rho}{2}\|\sum_{g=1}^{G} E_g\beta_g - f + w^{(k)}\|_2^2,\end{aligned}$$

which is a quadratic function of β_b. The update formula is then given as

$$\begin{aligned}\beta_g^{(k+1)} =&[X_g^T X_g + \rho(C_g^T C_g + D_g^T D_g + E_g^T E_g)]^{-1}[X_g^T y - X_g^T \sum_{i\neq g} X_i\beta_i^{(k)} \\ &- \rho D_g^T(\sum_{i\neq g} D_i\beta_i^{(k)} - z^{(k)} + u^{(k)}) - \rho C_g^T(\sum_{i\neq g} C_i\beta_i^{(k)} - \omega^{(k)} - d + v^{(k)}) \\ &- \rho E_g^T(\sum_{i\neq g} E_i\beta_i^{(k)} - f + w^{(k)})].\end{aligned}$$

Compared with the formula to update $\beta^{(k+1)}$ in section 3.1, every product between matrix and parameter β is partitioned into summation of products within each processor. In the iteration, all those local products are calculated from the update variable in last iteration and updated simultaneously at the end of the iteration.

The variable $z^{(k+1)}$ is computed globally for all processors.

$$\begin{aligned}z^{(k+1)} &= \arg\min_{z} \lambda\|z\|_1 + \frac{\rho}{2}\|\sum_{g=1}^{G} D_g\beta_g^{(k+1)} - z + u^{(k)}\|_2^2 \\ &= (1 - \frac{\frac{\lambda}{\rho}}{|\sum_{g=1}^{G} D_g\beta_g^{(k+1)} + u^{(k)}|})_+(\sum_{g=1}^{G} D_g\beta_g^{(k+1)} + u^{(k)}).\end{aligned}$$

The update step of $z^{(k+1)}$ is computed based on the idea in Boyd et al. (2011) as well.

The variable $\omega^{(k+1)}$ is updated globally for all processors.

$$\begin{aligned}\omega^{(k+1)} &= \arg\min_{\omega} \Phi_{R_+^q}(\omega) + \frac{\rho}{2}\|\sum_{g=1}^{G} C_g\beta_g^{(k+1)} - \omega - d + v^{(k)}\|_2^2 \\ &= (\sum_{g=1}^{G} C_g\beta_g^{(k+1)} - d + v^{(k)})_+.\end{aligned}$$

The variable $U^{(k+1)}$ is updated globally for all processors.

$$u^{(k+1)} = u^{(k)} + \sum_{g=1}^{G} D_g \beta_g^{(k+1)} - z^{(k+1)},$$
$$v^{(k+1)} = v^{(k)} + \sum_{g=1}^{G} C_g \beta_g^{(k+1)} - \omega^{(k+1)} - d,$$
$$w^{(k+1)} = w^{(k)} + \sum_{g=1}^{G} E_g \beta_g^{(k+1)} - f.$$

We can see that the updating of $\beta_g^{(k+1)}$ is computed within each processor locally, while the updating of $z^{(k+1)}$, $\omega^{(k+1)}$ and $U^{(k+1)}$ are computed globally based on the intermediate results of $\beta_g^{(k+1)}$ from all processors simultaneously.

The iteration will stop if the primal and dual residuals are very close to 0. It is important to point out that we need to consider the primal and dual residuals in different processors together. In this way, all processors will stop iterations at the same time. However, since the local processor can only generate one subset of the final update of β, while the residuals have to be evaluated with the complete parameter β, the stopping rule should be the same as the one in section 3.1, that is,

$$\|r_{prim}^{(k+1)}\|_2 \leq \epsilon^{prim}, \quad \|s_{dual}^{(k+1)}\|_2 \leq \epsilon^{dual},$$

where r_{prim} and s_{dual} are primary and dual residuals, respectively, given by

$$r_{prim}^{(k+1)} = A_1\beta^{(k+1)} + A_2 z^{(k+1)} + A_3\omega^{(k+1)} - b = \begin{bmatrix} D\beta^{(k+1)} - z^{(k+1)} \\ C\beta^{(k+1)} - \omega^{(k+1)} - d \\ E\beta^{(k+1)} - f \end{bmatrix},$$

$$s_{dual}^{(k+1)} = \rho A_1^T A_3(\omega^{(k)} - \omega^{(k+1)}) + \rho A_1^T A_2(z^{(k)} - z^{(k+1)}) = \rho C^T(\omega^{(k)} - \omega^{(k+1)}) + \rho D^T(z^{(k)} - z^{(k+1)}),$$

where $\beta^{(k+1)} = \begin{bmatrix} \beta_1^{(k+1)} \\ \beta_2^{(k+1)} \\ \vdots \\ \beta_g^{(k+1)} \end{bmatrix}$, and

$$\epsilon^{prim} = \sqrt{m+q+s}\epsilon^{abs} + \epsilon^{rel} max\{\|A_1\beta^{(k+1)}\|_2, \|A_2 z^{(k+1)}\|_2, \|A_3\omega^{(k+1)}\|_2, \|b\|_2\},$$
$$\epsilon^{dual} = \sqrt{p}\epsilon^{abs} + \epsilon^{rel}\|\rho A_1^T U^{(k+1)}\|_2,$$

where ϵ^{abs} and ϵ^{rel} are absolute tolerance and relative tolerance.

Notice that if there is a multicollinearity problem, the penalization in the ridge regression could be added to the global problem to mitigate the its effect. Although in each local processor, only a subset of β is estimated, by adding the constraint $\sum_{i=1}^{p} \beta_i = c$ in the original problem, we can control the effect of multicollinearity in local processor as well as across multiple processors.

As another remark, there will not be a case when the same variables are measured at different locations. As I showed in the proof of convergence in the next chapter, the convergence could be achieved only if the variables in each local processor are independent to other local processors. If there are shared variables in several local processors, the sufficient condition discussed in Chen et al. (2016) will not be held and the convergence may not be achieved.

The finalized algorithm with distributed ADMM splitting along p is as follows:

Algorithm 4: Distributed ADMM algorithm splitting along p

Result: β

initialization; **while**

$$\|r_{prim}^{(k+1)}\|_2 > \epsilon^{prim} \quad or \quad \|s_{dual}^{(k+1)}\|_2 > \epsilon^{dual}$$

do

For gth processor: $(g = 1, 2, \ldots, G)$

$$\begin{aligned}\beta_g^{(k+1)} =&[X_g^T X_g + \rho(C_g^T C_g + D_g^T D_g + E_g^T E_g)]^{-1}[X_g^T y - X_g^T \sum_{i\neq g} X_i\beta_i^{(k)} \\ &- \rho D_g^T(\sum_{i\neq g} D_i\beta_i^{(k)} - z^{(k)} + u^{(k)}) - \rho C_g^T(\sum_{i\neq g} C_i\beta_i^{(k)} - \omega^{(k)} - d + v^{(k)}) \\ &- \rho E_g^T(\sum_{i\neq g} E_i\beta_i^{(k)} - f + w^{(k)})]\end{aligned}$$

For all processors:

$$\begin{aligned}z^{(k+1)} &= (1 - \frac{\frac{\lambda}{\rho}}{|\sum_{g=1}^G D_g\beta_g^{(k+1)} + u^{(k)}|})_+(\sum_{g=1}^G D_g\beta_g^{(k+1)} + u^{(k)}) \\ \omega^{(k+1)} &= (\sum_{g=1}^G C_g\beta_g^{(k+1)} - d + v^{(k)})_+ \\ u^{(k+1)} &= u^{(k)} + \sum_{g=1}^G D_g\beta_g^{(k+1)} - z^{(k+1)} \\ v^{(k+1)} &= v^{(k)} + \sum_{g=1}^G C_g\beta_g^{(k+1)} - \omega^{(k+1)} - d \\ w^{(k+1)} &= w^{(k)} + \sum_{g=1}^G E_g\beta_g^{(k+1)} - f\end{aligned}$$

end

3.5 Distributed ADMM Splitting Along n and p

This section derives another distributed algorithm for solving (1.1). It is important to point out that in this algorithm, there are two layers of iterations. In the first layer, there are several clusters of processors. Each processor will independently compute the intermediate result using the local data and the intermediate numbers are collected and aggregated together to produce

another intermediate result in each cluster. Then the intermediate result generated in each cluster will be collected and aggregated together one more time to produce the final global update. For example, if there are totally $B \times G$ number of processors, and the processors are partitioned into B clusters where each cluster includes G processors. Then in the iteration, at the beginning, each of the G processors in the 1st block will run simultaneously using the local data and the intermediate results will be collected and aggregated to produce the final result in the 1st cluster. Similarly, cluster 2, cluster 3, ..., cluster B will calculate their corresponding local results in each processor which will be aggregated as a final result within each cluster. After this step, each cluster will have only one intermediate result which is the final result aggregated by the G processors in each cluster. These new intermediate results will be collected and aggregated again by communication among all clusters to produce the global result.

Assume that the whole data are partitioned into B subsets along rows, and G subsets along columns, where the gth subset within the bth subset is stored in the gth processor of the bth cluster of processors. The sample size of the local data in the bth cluster is n_b and the total sample size is $n = \sum_{b=1}^{B} n_b$. The dimension for the the gth subset in each cluster is p_g and the total dimension is $p = \sum_{g=1}^{G} p_g$. We need to do the same partition to the parameter β along its row, and to the matrices C, D and E along their columns repsectively to obatin the subsets $\{\beta_g\}$, $\{C_g\}$, $\{D_g\}$, and $\{E_g\}$, $g = \{1, 2, \ldots, G\}$ such that

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_g \end{bmatrix}, C = \begin{bmatrix} C_1 & C_2 & \ldots & C_g \end{bmatrix}, D = \begin{bmatrix} D_1 & D_2 & \ldots & D_g \end{bmatrix}, E = \begin{bmatrix} E_1 & E_2 & \ldots & E_g \end{bmatrix},$$

where the gth subset of β, C, D and E are stored in the gth processor in each cluster of processors. Notice that the product of matrix $X\beta$, $C\beta$, $D\beta$ and $E\beta$ can be written into summation of products

$$X\beta = \sum_{g=1}^{G} X_g\beta_g, C\beta = \sum_{g=1}^{G} C_g\beta_g, D\beta = \sum_{g=1}^{G} D_g\beta_g, E\beta = \sum_{g=1}^{G} E_g\beta_g.$$

The original problem (1.1) can be written as

$$\min_{\beta_g} \sum_{b=1}^{B} \frac{1}{2}\|y_b - \sum_{g=1}^{G} X_{bg}\beta_g\|_2^2 + \lambda\|D_g\beta_g\|_1,$$
$$\text{subject to} \quad \sum_{g=1}^{G} C_g\beta_g \geq d, \quad \sum_{g=1}^{G} E_g\beta_g = f, \quad g = 1, 2, \ldots, G, \quad b = 1, 2, \ldots, B. \tag{3.14}$$

Notice that parameter β_g, penalty function, equality constraint and inequality constraint will all change for different processors in different clusters. This is the most general situation of distributed ADMM and there are several scenarios where the problem (3.14) is applicable. For example, the whole data have extremely large sample size as well as enormous dimesion to be saved in one location or the original data are collected at different locations and time period and stored according to different group of predictors, this approach will be needed. This is common in big data problem. Another scenario is also privacy preserving. For example, several pharmaceutical companies want to build a model for the same medicine. However, on one hand, no company want to share the raw data to others due to privacy. On the other hand, within each company, the raw data are stored in several different servers corresponding to different stage in the clinical trial. In this case, the data could not be collected along either n or p completely. A distributed algorithm splitting along n and p together will become a natural choice, which only require the share of intermediate calculation results, which is neither the original raw data within each pharmaceutical company, nor the original raw data within each stage of clinical trial in the company.

In this algorithm, the technique is to combine the steps we used in section 3.3 and 3.4 together. Each processor only processes the local data stored in the processor within the cluster. All processors needs to communicate with each other to exchange some information, not only within each cluster, but also among the clusters. Roughly speaking, during each iteration, there are $(B + 1)$ major steps, which include B local steps and one global step. In each local step, each processor within the same cluster processes information based on local data to obtain B intermediate result, while in the global step, the intermediate results generated from B clusters will be aggregated to produce a final update.

Firstly, we rewrite the problem (3.14) as a global consensus problem

$$\min_{\beta_{bg},z,\omega,\beta_g^c} \sum_{b=1}^{B} \frac{1}{2}\|y - \sum_{g=1}^{G} X_{bg}\beta_{bg}\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$

$$\text{subject to} \quad \sum_{g=1}^{G} D_g\beta_{bg} = z, \quad \sum_{g=1}^{G} C_g\beta_{bg} - \omega = d, \quad \sum_{g=1}^{G} E_g\beta_{bg} = f, \quad \beta_{bg} = \beta_g^c,$$

$$g = 1, 2, \ldots, G, \quad b = 1, 2, \ldots, B.$$

The augmented Lagrangian function is

$$\begin{aligned}
\mathcal{L}_\rho(\beta_{bg}, z, \omega, \beta_g^c, u_b, v_b, w_b, h_b) = &\sum_{b=1}^{B} \frac{1}{2}\|y - \sum_{g=1}^{G} X_{bg}\beta_{bg}\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega) \\
&+ \sum_{b=1}^{B}\{\rho {u_b}^T(\sum_{g=1}^{G} D_g\beta_{bg} - z) + \frac{\rho}{2}\|\sum_{g=1}^{G} D_g\beta_{bg} - z\|_2^2\} \\
&+ \sum_{b=1}^{B}\{\rho {v_b}^T(\sum_{g=1}^{G} C_g\beta_{bg} - \omega - d) + \frac{\rho}{2}\|\sum_{g=1}^{G} C_g\beta_{bg} - \omega - d\|_2^2\} \\
&+ \sum_{b=1}^{B}\{\rho {w_b}^T(\sum_{g=1}^{G} E_g\beta_{bg} - f) + \frac{\rho}{2}\|\sum_{g=1}^{G} E_g\beta_{bg} - f\|_2^2\} \\
&+ \sum_{b=1}^{B}\{\rho {h_{bg}}^T(\beta_{bg} - \beta_g^c) + \frac{\rho}{2}\|\beta_{bg} - \beta_g^c\|_2^2\}.
\end{aligned}$$

In this problem, the local variables include β_{bg}, h_{bg}, and β_g^c which are unique to each processor within each cluster. The other local dual variable $U_b = (u_b, v_b, w_b)$ remains the same across all processors within each cluster, which is unique to each cluster of processors. The global variables include z, ω which are the same across all processors. In one typical iteration, the local variables within each cluster are updated within each processor which are aggregated together to the cluster to obtain the intermediate result with respect to each cluster, and then the local variable corresponding to cluster of processors will be updated based on the intermediate result. Finally, the global variables are updated together and simultaneously across all processors.

We update the estimators in the order of $\beta_{bg}^{(k+1)}$, $\beta_g^{c(k+1)}$, $h_{bg}^{(k+1)}$, $U_b^{(k+1)}$, $z^{(k+1)}$, and $\omega^{(k+1)}$. Notice that the first three variables are local variables within each cluster, the fourth variable is

cluster-level local variable, and the remaining two variables are global variables. The detailed calculation formulas are derived in the following.

The variable $\beta_{bg}^{(k+1)}$ is updated in the gth processor within the bth cluster.

$$\begin{aligned}\beta_{bg}^{(k+1)} =& \arg\min_{\beta_{bg}} \sum_{b=1}^{B} \frac{1}{2}\|y - \sum_{g=1}^{G} X_{bg}\beta_{bg}\|_2^2 + \sum_{b=1}^{B} \frac{\rho}{2}\|\sum_{g=1}^{G} D_g\beta_{bg} - z^{(k)} + u_b^{(k)}\|_2^2 \\ &+ \sum_{b=1}^{B} \frac{\rho}{2}\|\sum_{g=1}^{G} C_g\beta_{bg} - \omega^{(k)} + v_b^{(k)}\|_2^2 + \sum_{b=1}^{B} \frac{\rho}{2}\|\sum_{g=1}^{G} E_g\beta_{bg} - f + w_b^{(k)}\|_2^2 \\ &+ \sum_{b=1}^{B} \frac{\rho}{2}\|\beta_{bg} - \beta_g^{c(k)} + h_{bg}^{(k)}\|_2^2,\end{aligned}$$

which is a quadratic function of β_{bg}. The update formula is then given as

$$\begin{aligned}\beta_{bg}^{(k+1)} =& [X_{bg}^T X_{bg} + \rho(C_g^T C_g + D_g^T D_g + E_g^T E_g + I_{p_g})]^{-1}\{X_{bg}^T y_b - X_{bg}^T \sum_{i\neq g} X_{bi}\beta_{bi}^{(k)} \\ &- \rho D_g^T(\sum_{i\neq g} D_i\beta_{bi}^{(k)} - z^{(k)} + u_b^{(k)}) - \rho C_g^T(\sum_{i\neq g} C_i\beta_{bi}^{(k)} - \omega^{(k)} - d + v_b^{(k)}) \\ &- \rho E_g^T(\sum_{i\neq g} E_i\beta_{bi}^{(k)} - f + w_b^{(k)}) - \rho(-\beta_g^{c(k)} + h_{bg}^{(k)})\}.\end{aligned}$$

Compared with the formula to update $\beta^{(k+1)}$ in section 3.1, there is an extra penalty term in the objective function and every product between matrix and parameter β is partitioned into summation of products within each processor. In the iteration, all those local products are calculated from the update variable in last iteration and updated simultaneously at the end of the iteration.

The variable $\beta_g^{c(k+1)}$ is computed in the gth processor with each cluster.

$$\begin{aligned}\beta_g^{c(k+1)} =& \arg\min_{\beta_g^c} \sum_{b=1}^{B} \frac{\rho}{2}\|\beta_{bg}^{(k+1)} - \beta_g^c + h_{bg}^{(k)}\|_2^2 \\ =& \frac{\sum_{b=1}^{B}(\beta_{bg}^{(k+1)} + h_{bg}^{(k)})}{B} = \beta_g^{(\bar{k}+1)} + h_g^{(\bar{k})},\end{aligned}$$

where

$$\bar{\beta_g^{(k+1)}} = \frac{\sum_{b=1}^{B} \beta_{bg}^{(k+1)}}{B},$$
$$\bar{h_g^{(k)}} = \frac{\sum_{b=1}^{B} h_{bg}^{(k)}}{B}.$$

It is clear that $\beta_g^{c(k+1)}$ is essentially an average of all $\beta_{bg}^{(k+1)}$ computed by different processors in different clusters.

The variable $h_{bg}^{(k+1)}$ is updated in the gth processor within the bth cluster of processors.

$$h_{bg}^{(k+1)} = h_{bg}^{(k)} + \beta_{bg}^{(k+1)} - \beta_g^{c(k+1)}.$$

The variable $U_b^{(k+1)}$ is computed for all processors in the bth cluster of processors.

$$u_b^{(k+1)} = u_b^{(k)} + \sum_{g=1}^{G} D_g \beta_{bg}^{(k+1)} - z^{(k)},$$
$$v_b^{(k+1)} = v_b^{(k)} + \sum_{g=1}^{G} C_g \beta_{bg}^{(k+1)} - \omega^{(k)} - d,$$
$$w_b^{(k+1)} = w_b^{(k)} + \sum_{g=1}^{G} E_g \beta_{bg}^{(k+1)} - f,$$

The variable $z^{(k+1)}$ is updated globally for all processors in all clusters.

$$\begin{aligned} z^{(k+1)} &= \arg\min_z \lambda\|z\|_1 + \sum_{b=1}^{B} \frac{\rho}{2} \|\sum_{g=1}^{G} D_g \beta_{bg}^{(k+1)} - z + u_b^{(k+1)}\|_2^2 \\ &= (1 - \frac{\frac{\lambda}{\rho B}}{|\overline{\sum_{g=1}^{G} D_g \beta_g^{(k+1)}} + \overline{u^{(k+1)}}|})_+ (\overline{\sum_{g=1}^{G} D_g \beta_g^{(k+1)}} + \overline{u^{(k+1)}}), \end{aligned}$$

where

$$\overline{\sum_{g=1}^{G} D_g \beta_g^{(k+1)}} = \frac{\sum_{b=1}^{B} (\sum_{g=1}^{G} D_g \beta_{bg}^{(k+1)})}{B},$$
$$\overline{u^{(k+1)}} = \frac{\sum_{b=1}^{B} u_b^{(k+1)}}{B}.$$

The update of $z^{(k+1)}$ is based on the combination of the techniques used in section 3.3 and section 3.4.

The update of $\omega^{(k+1)}$ is computed globally for all processors in all clusters.

$$\begin{aligned}\omega^{(k+1)} &= \arg\min_{\omega} \Phi_{R_+^q}(\omega) + \sum_{b=1}^{B} \frac{\rho}{2} \| \sum_{g=1}^{G} C_g \beta_{bg}^{(k+1)} - \omega + v_b^{(k+1)} \|_2^2 \\ &= (\overline{\sum_{g=1}^{G} C_g \beta_g^{(k+1)}} - d + \overline{v^{(k+1)}})_+,\end{aligned}$$

where

$$\overline{\sum_{g=1}^{G} C_g \beta_g^{(k+1)}} = \frac{\sum_{b=1}^{B} (\sum_{g=1}^{G} C_g \beta_{bg}^{(k+1)})}{B},$$

$$\overline{v^{(k+1)}} = \frac{\sum_{b=1}^{B} v_b^{(k+1)}}{B}.$$

We can see that the updating of $\beta_{bg}^{(k+1)}$, $\beta_g^{c(k+1)}$, and $h_{bg}^{(k+1)}$ are within each processor within each cluster locally, while the updating of $U_b^{(k+1)}$ is updated for all processors within each cluster locally, and the updating of $z^{(k+1)}$ and $\omega^{(k+1)}$ are computed globally for all processors in all clusters.

The iteration will stop if the primal and dual residuals are small enough. We have showed the stopping rule for distributed ADMM with splitting along n and p in section 3.3 and 3.4 respectively. It is easy to see that splitting along p will not affect the result from the stopping rule of section 3.1 and we only need to consider the primal and dual residual in different processors together among the clusters, that is to say, from splitting along n. In this way, all processors will stop iterations at the same time. The stopping rule should be the same as the one we derived in

section 3.3, and the primal and dual residuals for th bth cluster are given by

$$r_{prim,b} = \begin{bmatrix} D\beta_b^{(k+1)} - z^{(k+1)} \\ C\beta_b^{(k+1)} - \omega^{(k+1)} - d \\ E\beta_b^{(k+1)} - f \\ \beta_b^{(k+1)} - \beta_c^{(k+1)} \end{bmatrix},$$

$$s_{dual,b} = \rho[D^T(z^{(k+1)} - z^{(k)}) + C^T(\omega^{(k+1)} - \omega^{(k)}) + (\beta_c^{(k+1)} - \beta_c^{(k)})],$$

where

$$\beta_b^{(k+1)} = \begin{bmatrix} \beta_{b1}^{(k+1)} \\ \beta_{b2}^{(k+1)} \\ \vdots \\ \beta_{bG}^{(k+1)} \end{bmatrix}, \quad \text{and} \quad \beta_c^{(k+1)} = \begin{bmatrix} \beta_1^{c(k+1)} \\ \beta_2^{c(k+1)} \\ \vdots \\ \beta_G^{c}{}^{(k+1)} \end{bmatrix}.$$

The stopping rule is

$$\sqrt{\sum_{b=1}^{B} \|r_{prim,b}^{(k+1)}\|_2^2} \leq \epsilon^{prim}, \quad \sqrt{\sum_{b=1}^{B} \|s_{dual,b}^{(k+1)}\|_2^2} \leq \epsilon^{dual},$$

where

$$\begin{aligned}
\epsilon^{prim} =& \sqrt{B(m+q+s+p)}\epsilon^{abs} \\
&+ \epsilon^{rel} max\{\sqrt{\sum_{b=1}^{B} \|A_1^* \beta_b^{(k+1)}\|_2^2}, \sqrt{B}\|A_2 z^{(k+1)}\|_2, \sqrt{B}\|A_3 \omega^{(k+1)}\|_2, \sqrt{B}\|\beta_c^{(k+1)}\|_2, \sqrt{B}\|b\|_2\} \\
=& \sqrt{B(m+q+s+p)}\epsilon^{abs} \\
&+ \epsilon^{rel} max\{\sqrt{\sum_{b=1}^{B} \|A_1^* \beta_b^{(k+1)}\|_2^2}, \sqrt{B}\|z^{(k+1)}\|_2, \sqrt{B}\|\omega^{(k+1)}\|_2, \sqrt{B}\|\beta_c^{(k+1)}\|_2, \sqrt{B}\|b\|_2\}, \\
\epsilon^{dual} =& \sqrt{Bp}\epsilon^{abs} + \epsilon^{rel}\sqrt{\sum_{b=1}^{B} \|\rho A_1^* U_b^{*(k+1)}\|_2^2},
\end{aligned}$$

where matrices A_2, A_3 are as defined before in section 3.1, A_1^* is defined in section 3.3, and

$$U_b^{*(k+1)} = \begin{bmatrix} u_b^{(k+1)} \\ v_b^{(k+1)} \\ w_b^{(k+1)} \\ h_b^{(k+1)} \end{bmatrix},$$

where

$$h_b^{(k+1)} = \begin{bmatrix} h_{b1}^{(k+1)} \\ h_{b2}^{(k+1)} \\ \vdots \\ h_{bG}^{(k+1)} \end{bmatrix}.$$

The finalized algorithm with distributed ADMM splitting along n and p is as follows:

Algorithm 5: Distributed ADMM algorithm splitting along n and p

Result: β

initialization; **while**

$$\sqrt{\sum_{b=1}^{B}\|r_{prim,b}^{(k+1)}\|_2^2} > \epsilon^{prim} \quad or \quad \sqrt{\sum_{b=1}^{B}\|s_{dual,b}^{(k+1)}\|_2^2} > \epsilon^{dual}$$

do

For bth cluster, $b = 1, 2, \ldots, B$

For gth processor, $g = 1, 2, \ldots, G$

$$\beta_{bg}^{(k+1)} = [X_{bg}^T X_{bg} + \rho(C_g^T C_g + D_g^T D_g + E_g^T E_g + I_{p_g})]^{-1}\{X_{bg}^T y_b - X_{bg}^T \sum_{i\neq g} X_{bi}\beta_{bi}^{(k)}$$

$$- \rho D_g^T(\sum_{i\neq g} D_i\beta_{bi}^{(k)} - z^{(k)} + u_b^{(k)}) - \rho C_g^T(\sum_{i\neq g} C_i\beta_{bi}^{(k)} - \omega^{(k)} - d + v_b^{(k)})$$

$$- \rho E_g^T(\sum_{i\neq g} E_i\beta_{bi}^{(k)} - f + w_b^{(k)}) - \rho(-\beta_g^{c(k)} + h_{bg}^{(k)})\}$$

$$\beta_g^{c(k+1)} = \bar{\beta_g^{(k+1)}} + \bar{h_g^{(k)}}$$

$$h_{bg}^{(k+1)} = h_{bg}^{(k)} + \beta_{bg}^{(k+1)} - \beta_g^{c(k+1)}$$

$$u_b^{(k+1)} = u_b^{(k)} + \sum_{g=1}^{G} D_g\beta_{bg}^{(k+1)} - z^{(k)}$$

$$v_b^{(k+1)} = v_b^{(k)} + \sum_{g=1}^{G} C_g\beta_{bg}^{(k+1)} - \omega^{(k)} - d$$

$$w_b^{(k+1)} = w_b^{(k)} + \sum_{g=1}^{G} E_g\beta_{bg}^{(k+1)} - f$$

For all processors:

$$z^{(k+1)} = (1 - \frac{\frac{\lambda}{\rho B}}{|\overline{\sum_{g=1}^{G} D_g\beta_g^{(k+1)}} + \overline{u^{(k+1)}}|})_+(\overline{\sum_{g=1}^{G} D_g\beta_g^{(k+1)}} + \overline{u^{(k+1)}})$$

$$\omega^{(k+1)} = (\overline{\sum_{g=1}^{G} C_g\beta_g^{(k+1)}} - d + \overline{v^{(k+1)}})_+$$

end

Chapter 4

Proof of Convergence

In this section, we will show the proof of convergence mainly based on the discussion of sufficient condition of convergence in Chen et al. (2016). In this paper, the author proposed a sufficient condition ensuring the convergence of the solution in (3.1), which is

$$A^TB = 0, \quad or \quad B^TC = 0 \quad , or \quad A^TC = 0 \tag{4.1}$$

which means that there exists one orthogonal pair of coefficient matrix in the constraint. Specifically, if two coefficient matrices in consecutive order are orthogonal, i.e. $A^TB = 0$ or $B^TC = 0$. The problem (3.1) can be reduced to a special case of the original format of ADMM problem as (2.1). Suppose we have the condition that $A^TB = 0$, the augmented Lagrangian function of (3.1) is

$$\mathcal{L}_\rho(x, z, y, U) = f(x) + g(z) + h(y) + \rho U^T(Ax + Bz + Cy - d) + \frac{\rho}{2}\|Ax + Bz + Cy - d\|_2^2,$$

where $U^T = (u^T, v^T, w^T)$ are the (scaled) Lagrangian multiplier with respect to x, z and y and $\rho > 0$ is the user-specified constant. The update scheme of this problem is

$$
\begin{aligned}
x^{(k+1)} &= \arg\min_x \{\mathcal{L}_\rho(x, z^{(k)}, y^{(k)}, U^{(k)}) | x \in \mathcal{X}\}, \\
z^{(k+1)} &= \arg\min_z \{\mathcal{L}_\rho(x^{(k+1)}, z, y^{(k)}, U^{(k)}) | z \in \mathcal{Z}\}, \\
y^{(k+1)} &= \arg\min_y \{\mathcal{L}_\rho(x^{(k+1)}, z^{(k+1)}, y, U^{(k)}) | y \in \mathcal{Y}\}, \\
U^{(k+1)} &= U^{(k)} - \rho(Ax^{(k+1)} + Bz^{(k+1)} + Cy^{(k+1)} - d).
\end{aligned}
$$

According to the first-order optimality conditions of the minimization problem above,

$$
\begin{aligned}
&f(x) - f(x^{(k+1)}) + (x - x^{(k+1)})^T \{-A^T [U^{(k)} - \rho(Ax^{(k+1)} + Bz^{(k)} + Cy^{(k)} - d)]\} \geq 0, \forall x \in \mathcal{X}, \\
&g(z) - g(z^{(k+1)}) + (z - z^{(k+1)})^T \{-B^T [U^{(k)} - \rho(Ax^{(k+1)} + Bz^{(k+1)} + Cy^{(k)} - d)]\} \geq 0, \forall z \in \mathcal{Z}, \\
&h(y) - h(y^{(k+1)}) + (y - y^{(k+1)})^T \{-C^T [U^{(k)} - \rho(Ax^{(k+1)} + Bz^{(k+1)} + Cy^{(k+1)} - d)]\} \geq 0, \forall y \in \mathcal{Y}.
\end{aligned}
$$

Under the assumption $A^T B = 0$, the optimality conditions can be simplified as

$$
\begin{aligned}
&f(x) - f(x^{(k+1)}) + (x - x^{(k+1)})^T \{-A^T [U^{(k)} - \rho(Ax^{(k+1)} + Cy^{(k)} - d)]\} \geq 0, \forall x \in \mathcal{X}, \\
&g(z) - g(z^{(k+1)}) + (z - z^{(k+1)})^T \{-B^T [U^{(k)} - \rho(Bz^{(k+1)} + Cy^{(k)} - d)]\} \geq 0, \forall z \in \mathcal{Z}, \\
&h(y) - h(y^{(k+1)}) + (y - y^{(k+1)})^T \{-C^T [U^{(k)} - \rho(Ax^{(k+1)} + Bz^{(k+1)} + Cy^{(k+1)} - d)]\} \geq 0, \forall y \in \mathcal{Y},
\end{aligned}
$$

which is the first-order optimality conditions of the scheme

$$(x^{(k+1)}, z^{(k+1)}) = \arg\min_{x,z}\{F(x,z) - (U^{(k)})^T D \begin{bmatrix} x \\ z \end{bmatrix} + \frac{\rho}{2}\|D \begin{bmatrix} x \\ z \end{bmatrix} + Cy^{(k)} - d\|_2^2\},$$

$$y^{(k+1)} = \arg\min_{y}\{\mathcal{L}_\rho((x^{(k+1)}, z^{(k+1)}), y, U^{(k)})\},$$

$$U^{(k+1)} = U^{(k)} - \rho(D \begin{bmatrix} x \\ z \end{bmatrix} + Cy^{(k+1)} - d),$$

where

$$D = \begin{bmatrix} A & B \end{bmatrix},$$

$$F(x,z) = f(x) + g(z).$$

If we define a new variable $x^* = \begin{bmatrix} x \\ z \end{bmatrix}$, then this problem can be rewritten as

$$\min_{x^*,z} F(x^*) + h(y), \quad \text{subject to} \quad Dx^* + Cy = d,$$

which is the original format of ADMM algorithm in (2.1). Then the convergence of solution in this algorithm could be proved based on the discussion in Boyd et al. (2011) and Mota et al. (2011). Similar transformation of the first-order optimality conditions can be done with the assumption $B^TC = 0$ as well. In general, if there are two consecutive matrices in the constraint that are orthogonal, the variables associated with the matrices could be combined to a new variable so that the number of variables could be reduced by 1. Based on this technique, when there are more than 3 variables in the objective function and constraint, we need to find as much as consecutively orthogonal matrices to reduce the number of variables in the optimization problem. If there are only 3 variables left at the end and there is at least one pair of matrices orthogonal, in other words, if the condition (4.1) is satisfied, the convergence of the extended ADMM is guaranteed.

4.1 Extended ADMM With Slack Variables

Following the discussion above, in this section, we will prove the convergence of the algorithm introduced in section 3.1. The optimization problem is

$$\min_{\beta,z,\omega} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$
$$\text{subject to} \quad D\beta = z, \quad C\beta - \omega - d = 0, \quad E\beta = f.$$

The constraints can be written as $A_1\beta + A_2 z + A_3\omega = b$, where

$$A_1 = \begin{bmatrix} D \\ C \\ E \end{bmatrix}, \quad A_2 = \begin{bmatrix} -I_m \\ 0_q \\ 0_s \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0_m \\ -I_q \\ 0_s \end{bmatrix}, \quad b = \begin{bmatrix} 0_m \\ d \\ f \end{bmatrix}.$$

Define $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$, $g(z) = \lambda\|z\|_1$, and $h(\omega) = \Phi_{R_+^q}(\omega)$, the problem is equivalent to

$$\min_{\beta,z,\omega} f(\beta) + g(z) + h(\omega),$$
$$\text{subject to} \quad A_1\beta + A_2 z + A_3\omega = b.$$

According to Chen et al. (2016), since there are three variables in the system, if there exists one pair of orthogonal matrix, the convergence can be guaranteed. In fact, It is apprantly that A_2 and A_3 are orthogonal matrices. In conclusion, the solution of this algorithm will converge to a limit point.

4.2 Extended ADMM With Large p

In this section, we will focus on the convergence of the algorithm discussed in section 3.2. Recall that after adding all three new constraints with respect to M, P, Q, which are used to

simplify the matrix C^TC, D^TD and E^TE, the optimization problem is given as

$$\min_{\beta,z,\omega} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$

$$\text{subject to} \quad D\beta = z, \quad C\beta - \omega - d = 0, \quad E\beta = f,$$

$$(M - C^TC)^{\frac{1}{2}}\beta = \tilde{s}, \quad (P - D^TD)^{\frac{1}{2}}\beta = \tilde{u}, \quad (Q - E^TE)^{\frac{1}{2}}\beta = \tilde{v}.$$

The constraints can be written as $A_1\beta + A_2z + A_3\omega = b$, where

$$A_1 = \begin{bmatrix} D \\ C \\ E \\ (M - C^TC)^{\frac{1}{2}} \\ (P - D^TD)^{\frac{1}{2}} \\ (Q - E^TE)^{\frac{1}{2}} \end{bmatrix}, \quad A_2 = \begin{bmatrix} -I_m \\ 0_q \\ 0_s \\ 0_p \\ 0_p \\ 0_p \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0_m \\ -I_q \\ 0_s \\ 0_p \\ 0_p \\ 0_p \end{bmatrix}, \quad b = \begin{bmatrix} 0_m \\ d \\ f \\ \tilde{s} \\ \tilde{u} \\ \tilde{v} \end{bmatrix}.$$

Define $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$, $g(z) = \lambda\|z\|_1$, and $h(\omega) = \Phi_{R_+^q}(\omega)$, the problem is equivalent to

$$\min_{\beta,z,\omega} f(\beta) + g(z) + h(\omega),$$

$$\text{subject to} \quad A_1\beta + A_2z + A_3\omega = b.$$

Since matrices A_2 and A_3 are orthogonal and there are only three variables in the system, the convergence will be guaranteed.

4.3 Distributed ADMM Splitting Along n

In this section, we will focus on the convergence of the distributed algorithm discussed in section 3.3. Recall the global consensus problem

$$\min_{\beta_b,z,\omega} \sum_{b=1}^{B} \frac{1}{2}\|y_b - X_b\beta_b\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$

$$\text{subject to } D\beta_b = z, \quad C\beta_b - \omega = d, \quad E\beta_b = f, \quad \beta_b = \beta, \quad b = 1, 2, \ldots, B.$$

The constraints can be written as

$$\sum_{b=1}^{B} A_b\beta_b + Pz + Q\omega + R\beta = b_0,$$

where

$$A_b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ D \\ C \\ E \\ I_p \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, P = \begin{bmatrix} -I_m \\ 0 \\ 0 \\ 0 \\ -I_m \\ 0 \\ 0 \\ 0 \\ \vdots \\ -I_m \\ 0 \\ 0 \\ 0 \end{bmatrix}, Q = \begin{bmatrix} 0 \\ -I_q \\ 0 \\ 0 \\ 0 \\ -I_q \\ 0 \\ 0 \\ \vdots \\ 0 \\ -I_q \\ 0 \\ 0 \end{bmatrix}, R = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -I_p \\ 0 \\ 0 \\ 0 \\ -I_p \\ \vdots \\ 0 \\ 0 \\ 0 \\ -I_p \end{bmatrix}, \quad \text{and} \quad b_0 = \begin{bmatrix} 0 \\ d \\ f \\ 0 \\ 0 \\ d \\ f \\ 0 \\ \vdots \\ 0 \\ d \\ f \\ 0 \end{bmatrix},$$

where matrix D is on the $[4 \times (b-1)+1]$th block of A_b, $-I_m$ is on every $[4 \times (b-1)+1]$th block of P, $-I_q$ is on every $[4 \times (b-1)+2]$th block of Q, $-I_p$ is on every $[4 \times b]$th block of R, and d is on every $[4 \times (b-1)+2]$th block of b_0, for $b = 1, 2, \ldots, B$. Notice that there are more than three variables in the optimization problem. We need to apply the technique discussed at the beginning of this chapter to reduce the number of variables.

Firstly, it is not hard to find that the matrices $\{A_b\}, b = 1, 2, \ldots, B$ are mutually orthogonal. Applying the technique to them could result in the new matrix and associated new variable

$$A = \begin{bmatrix} D \\ C \\ E \\ I_p \\ D \\ C \\ E \\ I_p \\ \vdots \\ D \\ C \\ E \\ I_p \end{bmatrix}, \quad \beta^* = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta^B \end{bmatrix}^T .$$

Furthermore, notice that matrices P, Q and R are mutually orthogonal, the new matrix with corresponding variable

$$W = \begin{bmatrix} -I_m \\ -I_q \\ 0 \\ -I_p \\ -I_m \\ -I_q \\ 0 \\ -I_p \\ \vdots \\ -I_m \\ -I_q \\ 0 \\ -I_p \end{bmatrix}, \quad \alpha = \begin{bmatrix} z \\ \omega \\ \beta \end{bmatrix}^T,$$

could be derived from the discussed technique. Define

$$F(\beta^*) = \sum_{b=1}^{B} \frac{1}{2} \|y_b - X_b \beta_b\|_2^2,$$
$$G(\alpha) = \lambda \|z\|_1 + \Phi_{R_+^q}(\omega).$$

After the dimension reduction of the optimization problem, the renewed problem can be written as

$$\min_{\beta^*, \alpha} F(\beta^*) + G(\alpha),$$
$$\text{subject to} \quad A\beta^* + W\alpha = b_0,$$

which is in the form of (2.1) that includes only two variables. Then the convergence can be proved referring to Boyd et al. (2011) and Mota et al. (2011).

4.4 Distributed ADMM Splitting Along p

In this section, we will focus on the convergence of the distributed algorithm discussed in section 3.4. Recall the global consensus problem

$$\min_{\beta_g,z,\omega} \frac{1}{2}\|y - \sum_{g=1}^{G} X_g\beta_g\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$
$$\text{subject to} \quad \sum_{g=1}^{F} D_g\beta_g = z, \quad \sum_{g=1}^{G} C_g\beta_g - \omega = d, \quad \sum_{g=1}^{G} E_g\beta_g = f, \quad g = 1, 2, \ldots, G.$$

The constraints can be written as

$$\sum_{g=1}^{G} A_g\beta_g + Pz + Q\omega = b_0,$$

where

$$A_g = \begin{bmatrix} D_g \\ C_g \\ E_g \end{bmatrix}, \quad P = \begin{bmatrix} -I_m \\ 0 \\ 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 \\ -I_q \\ 0 \end{bmatrix}, \quad \text{and} \quad b_0 = \begin{bmatrix} 0 \\ d \\ f \end{bmatrix}.$$

Notice that matrices P and Q are consecutively orthogonal. The new matrix and associated variable

$$W = \begin{bmatrix} -I_m \\ -I_q \\ 0 \end{bmatrix}, \quad \alpha = \begin{bmatrix} z \\ \omega \end{bmatrix}^T,$$

can be obtained by applying the technique discussed before. However, we need to do something to the variables β_g to reduce the number to at least two or the sufficient condition in Chen et al.

(2016) could not be applied and the convergence will not be guaranteed. In order to apply the dimension reduction, we need to have the following assumption

Assumption 4. *matrices D, C and E have full column rank, while the number of rows in D, C and E, i.e. m, q and s, should be not less than the shared number of columns p.*

This assumption is similar to the assumption in section 2.2 which is used to prove the uniqueness of solution by Mota et al. (2011). It is important to point out that this assumption is reasonable. Since the raw data is stored in each processor, there should be unique linear constraint associated with each of them, which means that the number of constraints should be at least the same as the length of β. And in order to reduce the communication among all the processors and protect the privacy, the constraint in each processor should be only related to the corresponding β_g itself. In fact, we use D_g as an example, which can be extended to C_g and E_g as well, that there should be only one row with nonzero entries, while the other rows are all 0. The nonzero entries are associated with the unique constraint of β_g. In such case, the matrix

$$
\begin{aligned}
D &= \begin{bmatrix} D_1 & D_2 & \dots & D_G \end{bmatrix} \\
&= \begin{bmatrix} \begin{pmatrix} a_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ a_2 \\ \vdots \\ 0 \end{pmatrix} & \cdots & \begin{pmatrix} 0 \\ 0 \\ \vdots \\ a_G \end{pmatrix} \end{bmatrix},
\end{aligned}
$$

where $a_g, g = 1, 2, \dots, G$ are the unique vector of constraint corresponding to β_g which is on the gth block of D_g. The same expansion can be applied to C and E. Then D, C, E are all with full rank, or in other words, the matrices $\{D_g\}$, $\{C_g\}$, and $\{E_g\}$ are mutually orthogonal within each set, which gives the mutually orthogonal of A_g.

Under the assumption 4, matrices A_g are mutually orthogonal as discussed above. By applying the reduction technique, the new matrix and corresponding variable

$$A = \begin{bmatrix} D \\ C \\ E \end{bmatrix}, \quad \text{and} \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_G \end{bmatrix},$$

are obtained. Define

$$F(\beta) = \frac{1}{2}\|y - \sum_{g=1}^{G} X_g\beta_g\|_2^2,$$

$$H(\alpha) = \lambda\|z\|_1 + \Phi_{R_+^q}(\omega).$$

The optimization problem can be written as

$$\min_{\beta,\alpha} F(\beta) + H(\alpha),$$

$$\text{subject to} \quad A\beta + W\alpha = b_0,$$

which only contains two variables to update. According to Boyd et al. (2011) and Mota et al. (2011), the convergence is proved.

4.5 Distributed ADMM Splitting Along n and p

In this section, we will focus on the convergence of the distributed algorithm discussed in section 3.5. Recall the global consensus problem

$$\min_{\beta_{bg},z,\omega,\beta_g^c} \sum_{b=1}^{B} \frac{1}{2}\|y - \sum_{g=1}^{G} X_{bg}\beta_{bg}\|_2^2 + \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$

$$\text{subject to} \quad \sum_{g=1}^{G} D_g\beta_{bg} = z, \quad \sum_{g=1}^{G} C_g\beta_{bg} - \omega = d, \quad \sum_{g=1}^{G} E_g\beta_{bg} = f, \quad \beta_{bg} = \beta_g^c,$$

$$g = 1, 2, \ldots, G, \quad b = 1, 2, \ldots, B.$$

The constraints can be written as

$$\sum_{b=1}^{B}\{\sum_{g=1}^{G} A_{bg}\beta_{bg}\} + Pz + Q\omega + \sum_{g=1}^{G} R_g\beta_g^c = b_0,$$

where

$$A_{bg} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ D_g \\ C_g \\ E_g \\ I_{p_g} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, P = \begin{bmatrix} -I_m \\ 0 \\ 0 \\ 0 \\ -I_m \\ 0 \\ 0 \\ 0 \\ \vdots \\ -I_m \\ 0 \\ 0 \\ 0 \end{bmatrix}, Q = \begin{bmatrix} 0 \\ -I_q \\ 0 \\ 0 \\ 0 \\ -I_q \\ 0 \\ 0 \\ \vdots \\ 0 \\ -I_q \\ 0 \\ 0 \end{bmatrix}, R = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -I_{p_g} \\ 0 \\ 0 \\ 0 \\ -I_{p_g} \\ \vdots \\ 0 \\ 0 \\ 0 \\ -I_{p_g} \end{bmatrix}, \quad \text{and} \quad b_0 = \begin{bmatrix} 0 \\ d \\ f \\ 0 \\ 0 \\ d \\ f \\ 0 \\ \vdots \\ 0 \\ d \\ f \\ 0 \end{bmatrix},$$

where matrix D_g is on the $[4 \times (b-1)+1]$th block of A_{bg}, $-I_m$ is on every $[4 \times (b-1)+1]$th block of P, $-I_q$ is on every $[4 \times (b-1)+2]$th block of Q, $-I_{p_g}$ is on every $[4 \times b]$th block of R_g, and d is on every $[4 \times (b-1)+2]$th block of b_0, for $b = 1, 2, \ldots, B, g = 1, 2, \ldots, G$.

The matrices P, Q are consecutively orthogonal, which can be combined into

$$W = \begin{bmatrix} -I_m \\ -I_q \\ 0 \\ 0 \\ -I_m \\ -I_q \\ 0 \\ 0 \\ \vdots \\ -I_m \\ -I_q \\ 0 \\ 0 \end{bmatrix}, \quad \text{associated with} \quad \alpha = \begin{bmatrix} z \\ \omega \end{bmatrix}^T .$$

Notice that I_{p_g} is the subset of several columns of identity matrix, they are mutually orthogonal to each other. Then it is easy to find that the matrices R_g are mutually orthogonal, which derive

$$R = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -I_p \\ 0 \\ 0 \\ 0 \\ -I_p \\ \vdots \\ 0 \\ 0 \\ 0 \\ -I_p \end{bmatrix}, \quad \text{associated with} \quad \beta_c = \begin{bmatrix} \beta_1^c \\ \beta_2^c \\ \vdots \\ \beta_G^c \end{bmatrix}.$$

So far, we have reduced the variables z, ω and $\beta_g^c, g = 1, 2, \ldots, G$ into two variables α and β_c. We need to reduce the variables $\beta_{bg}, b = 1, 2, \ldots, B, g = 1, 2, \ldots, G$ into one variable so that the optimization problem will contain three variables. Notice that the new matrices W and R are consecutively orthogonal, according to the sufficient condition (4.1), the convergence will be guaranteed.

For fixed b, under assumtption 4, the matrices $A_{bg}, g = 1, 2, \ldots, G$ are mutually orthogonal as we discussed in the previous section. In this way, A_{bg} can be grouped into B clusters

and simplified as B matrices

$$A_1 = \begin{bmatrix} D \\ C \\ E \\ I_p \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ D \\ C \\ E \\ I_p \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \ldots, \quad A_B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ D \\ C \\ E \\ I_p \end{bmatrix},$$

which are associated with the variables

$$\beta_1 = \begin{bmatrix} \beta_{11} \\ \beta_{12} \\ \vdots \\ \beta_{1G} \end{bmatrix}, \quad \beta_2 = \begin{bmatrix} \beta_{21} \\ \beta_{22} \\ \vdots \\ \beta_{2G} \end{bmatrix}, \quad \ldots, \quad \beta_B = \begin{bmatrix} \beta_{B1} \\ \beta_{B2} \\ \vdots \\ \beta_{BG} \end{bmatrix}$$

respectively. Note that generated new matrices $A_b, b = 1, 2, \ldots, B$ are also mutually orthogonal, which can be combined as

$$A = \begin{bmatrix} D \\ C \\ E \\ I_p \\ D \\ C \\ E \\ I_p \\ \vdots \\ D \\ C \\ E \\ I_p \end{bmatrix}, \quad \text{associated with the new variable} \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_B \end{bmatrix}^T.$$

Finally, the original global consensus problem is reduced to three variables β, α and β_c. Denote

$$F(\beta) = \sum_{b=1}^{B} \frac{1}{2} \|y - \sum_{g=1}^{G} X_{bg}\beta_{bg}\|_2^2,$$
$$H(\alpha) = \lambda\|z\|_1 + \Phi_{R_+^q}(\omega),$$

and the corresponding new optimization problem can be written as

$$\min_{\beta,\alpha,\beta_c} F(\beta) + H(\alpha),$$
$$\text{subject to} \quad A\beta + W\alpha + R\beta_c = b_0,$$

which is in the form of (3.1). Since matrice W and R are orthogonal, i.e. $W^T R = 0$, which is the sufficient condition in Chen et al. (2016). The convergence of the solution to this distributed ADMM algorithm is proved.

Chapter 5

Implementation

This chapter will mainly discuss several issues related to the implementation of the proposed algorithms in Chapter 3.

5.1 Selection of Tuning Parameters

The implementation of the proposed algorithms required the selection of a proper value of ρ. Roughly speaking, a large value of ρ places a larger penalty on the violation of primal feasibility and so tends to produce small primal residuals. Conversely, a small value of ρ tends to produce small dual residuals. Ideally, we expect both primal and dual residuals reduce at a comparable rate, and hence leads to a faster convergence.

The idea is to make these two residuals hold similar magnitude of the length. Increase ρ if primal residual are too large and decrease ρ if dual residual are too large. According to He et al. (2000) and Wang and Liao (2001), a simple scheme is to choose ρ as follows.

$$\rho^{(k+1)} = \begin{cases} \eta_{inc}\rho^{(k)} & if \|r_{prim}^{(k)}\|_2 > \mu \|s_{dual}^{(k)}\|_2, \\ \frac{\rho^{(k)}}{\eta_{dec}} & if \|s_{dual}^{(k)}\|_2 > \mu \|r_{prim}^{(k)}\|_2, \\ \rho^{(k)}, & otherwise \end{cases}$$

where $\mu > 1$, $\eta_{inc} > 1$, and $\eta_{dec} > 1$ are user-specified constants. A typical choice is $\mu = 10$ and $\eta_{inc} = \eta_{dec} = 2$.

5.2 Sparse Matrices

A sparse matrix is a matrix with only a few nonzero entries. Because of the sparsity structure, a sparse matrix can be stored in terms of its nonzero entries and the location of these entries. In contrast, a dense matrix has to be stored with all entries. Therefore, a sparse matrix may need much smaller size of memory. The computational cost could be saved dramatically especially when the size of the matrix is extremely large.The matrix calculation involving sparse matrices may be achieved in a much faster algorithm. For example, in the generalized LASSO problem, the computational complexity to compute $D\beta$ is $O(np^2)$ if D is dense matrix, while the complexity reduces to $O(np)$ when D is diagonal matrix (the traditional LASSO problem). In real examples, the linear constraints and penalty term should be with easier form, i.e. we could always assume that the matrix C, D and E are sparse matrices in applications. Instead of storing the information of all entries, we could allocate the memory only for the nonzero entries and its associated positions. In such case, the computational cost could be saved greatly, especially in the big data problem.

5.3 Distributed Computing

The distributed ADMM algorithms derived in section 3.3, 3.4 and 3.5 are implemented using openMPI(`https://www.open-mpi.org/`), which is the open source implementation of message passing interface. The openMPI provides a function called Allreduce. When this function is called, each processor will send the data (ex. $\beta_b^{(k+1)}$) to other processors and some operator (ex. summation) can be applied to aggregate the data and the intermediate results (ex. $\sum_{b=1}^{B} \beta_b^{(k+1)}$) will be sent back to all processors. The global variables are computed in the same framework.

Chapter 6

Simulation

In this chapter, we will report some simulation studies, which are used to illustrate the performance of the proposed algorithms, as well as some comparisons with existing algorithms.

6.1 Influence of ρ

This simulation example is used to demonstrate the influence of ρ on the rate of convergence of the proposed algorithm.

Randomly generate a design matrix X with $n = 550$ and $p = 500$ where each entry is independently sampled from $N(0, 1)$. The response vector $y = X\beta + \epsilon$, where $\beta = (1.0, 0.5, -1.0, 0.0, \ldots, 0.0, 1.0, 0.5, -1.0, 0.0, \ldots, 0.0)^T$ contains nonzero elements in the $1st$, $2nd$, $3rd$, $11th$, $12th$ and $13th$ positions and ϵ is sampled from $N(0, 1)$ as the random error. Using the LASSO model

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1,$$

subject to

$$\beta_1 + \beta_2 + \beta_3 \geq 0, \qquad \beta_1 + \beta_3 + \beta_{11} + \beta_{13} = 0,$$
$$\beta_2 + \beta_5 + \beta_{11} \geq 1, \qquad \beta_2 + \beta_8 + \beta_{12} = 1.$$

Figure 6.1 shows the result when $\lambda = 5$ and the possible value of ρ is selected to be 1, 5, and 1000. When $\rho = 1$, primal residuals are larger than dual residuals. When $\rho = 1000$,

dual residuals are larger than primal residuals. When $\rho = 5$, both primal and dual residuals decreases at a similar magnitude. It is clear to see that $\rho = 5$ converges much faster than the cases when $\rho = 1$ and $\rho = 1000$.

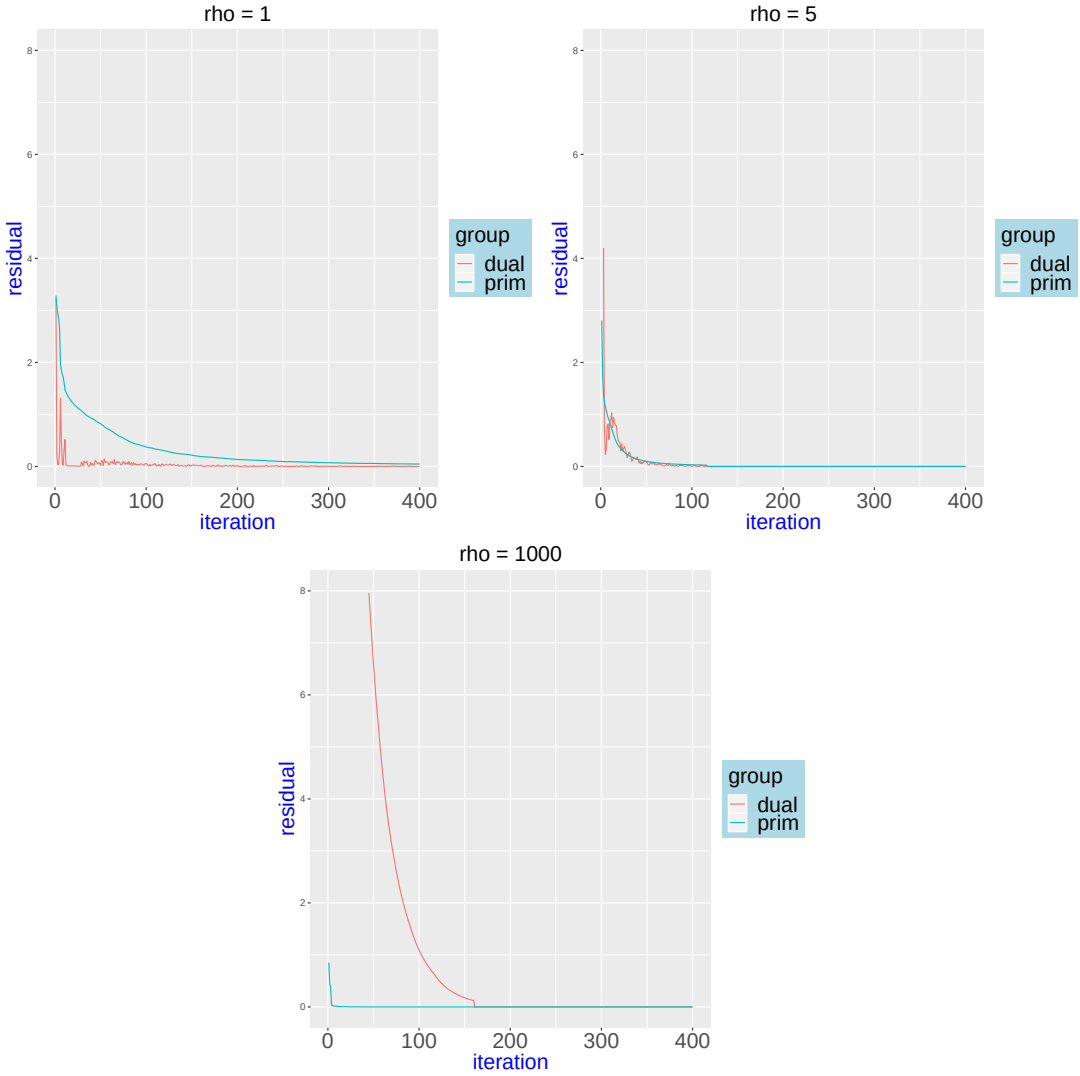

Figure 6.1: Influence of ρ

6.2 Influence of λ

This example is used to demonstrate the influence of λ on the degree of freedom, which will affect the evaluation based on information criterion (IC). The same setting in the previous section will be used while ρ is fixed at 5. The set of possible values of tuning parameter

$\lambda = (0.01, 0.1, 1, 5, 10, 20)$. According to Zeng et al. (2017), the BIC can be defined as

$$BIC(\hat{y}) = \frac{\|y - \hat{y}\|_2^2}{n\sigma^2} + \frac{log(n)}{n} df(\hat{y}), \tag{6.1}$$

where $\hat{y}$ is the estimated response $X\hat{\beta}$ and n is the sample size. The degree of freedom is as follows

$$df(\hat{y}) = \mathbb{E}[dim(col(XP_{null(G_{-\mathcal{A},\mathcal{B}})}))],$$

where $P_{null(G_{-\mathcal{A},\mathcal{B}})}$ is the projection matrix associated with $null(G_{-\mathcal{A},\mathcal{B}})$, and

$$\mathcal{A} = \{i : D_i\hat{\beta} \neq 0\}, \quad \mathcal{B} = \{k : C_k\hat{\beta} = d_k\}$$

that $\mathcal{A}$ is the set of indexes corresponding to non-zero components of $D\hat{\beta}$ and $\mathcal{B}$ is the set of indexes corresponding to active inequality constraints. Additionaly, define

$$G_{-\mathcal{A},\mathcal{B}} = (D_{-\mathcal{A}}^T, -C_{\mathcal{B}}^T, -E^T)^T.$$

In practice, this degree of freedom can be obtained in the following steps. We obtain $\mathcal{A}$ and $\mathcal{B}$ according to their definition and estimate degree of freedom by the rank of XP_{null}. Since P_{null} is the projection matrix, i. e., $XP_{null} = X(I_p - \mathcal{Q}\mathcal{Q}^T)$, where $\mathcal{Q}$ is the $\mathcal{Q}$-matrix of the QR-decomposition of $G_{-\mathcal{A},\mathcal{B}}^T$. Then the degree of freedom is the rank of XP_{null} which can be determined by applying the QR-decomposition with pivoting to XP_{null}.

Figure 6.2 shows the result of BIC with different tuning parameters λ. When $\lambda < 1$, the BIC will decrease as the tuning parameter increase. When $\lambda > 1$, the BIC will be an increasing function with respect to λ. The best λ in this scenario should be 1. It is important to point out that the most accurate value of the best λ should be between 0.1 and 5. If we have more training values of λ, we could see the trend of λ between 0.1 to 1 and 1 to 5 more clearly and may be able to find a more accurate tuning parameter.

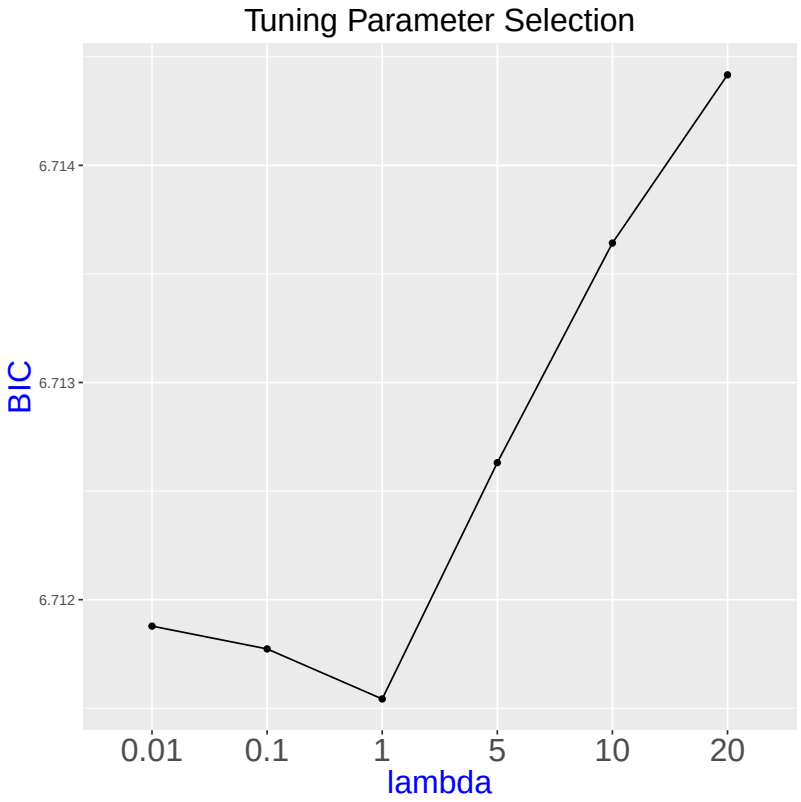

Figure 6.2: Influence of λ

6.3 Influence of number of subgroups

This example is used to demonstrate the influence of the number of subgroups when splitting along n in ADMM.

Randomly generate a design matrix X with $n = 600$ and $p = 40$, where each entry is independently sampled from $N(0, 1)$. The response vector $y = X\beta + \epsilon$, where $\beta = (1.0, -1.0, 0.5, 2, 1, \ldots, 1)^T$ that all the elements after the 4th position are all 1 and ϵ is sampled from $N(0, 1)$. C matrix is a diagonal matrix with the diagonal entries repeating from 1 to 10 by 4 times except the 2nd diagonal element with -1 and vector d is a 0 vector. Matrix E is set to be I_p and $f = E\beta$. D is also an diagonal matrix with all elements equal to 1. $\lambda = 1$ and $\rho = 0.8$ are fixed tuning parameters. We used 4 different ways to partition the design matrix X: equally divided in 2 subgroups, 4 subgroups, 6 subgroups and 8 subgroups along its rows. We run the algorithm in 3.1 and 3.3 to the same dataset as generated above and use the relative

norm to measure the difference of number of subgroups which is defined as

$$R = \frac{\|\hat{\beta}_n - \beta\|_2}{\|\hat{\beta}_s - \beta\|_2}$$

where $\hat{\beta}_n$ is the estimate from the algorithm splitting along n and $\hat{\beta}_s$ is estimated by applying the Extended ADMM with slack variables. The result is given below in figure 6.3.

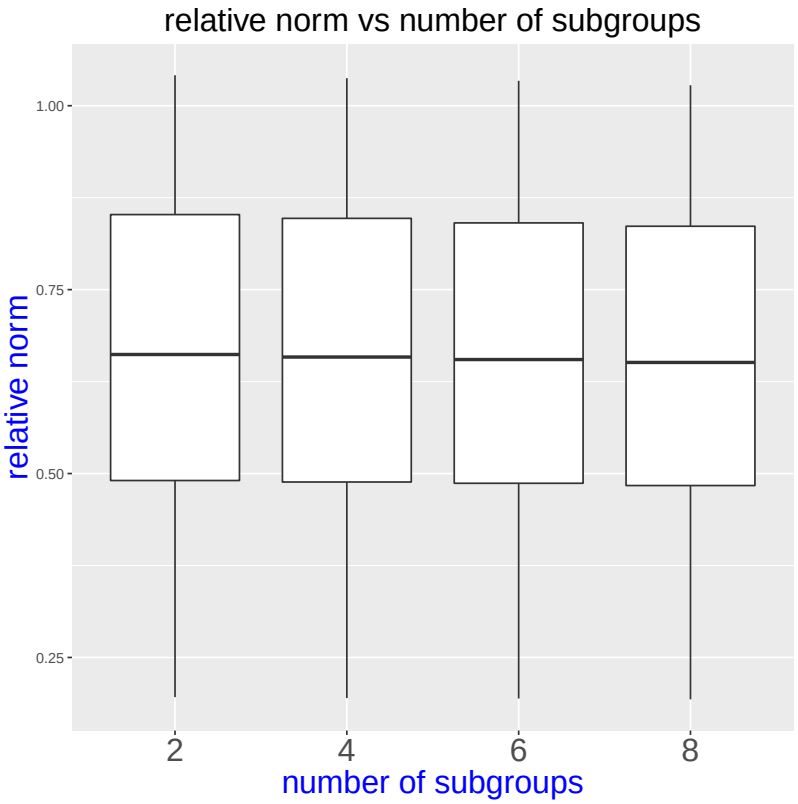

Figure 6.3: Influence of λ

It's clear to see that when the number of subgroups increase from 2 to 8, the relative norm is not changing much, which means the estimate generated by ADMM splitting along n is robust to the number of subgroups partitioned in design matrix X. The similar result could be found when we use the ADMM splitting along p.

Notice that when the number of subgroups increase, the numerator of the relative norm is decreasing which means the accuracy of the estimated β in distributed ADMM along n is increasing. This is reasonable since both estimation of β in this two algorithms highly converge to the true parameter β, and the 2-norm differences are both very close to 0. When the number

of subgroups change, the 2-norm difference in the numerator will change with difference less than 0.00001. However, the denominator is very close to 0 so that this little change will affect the relative norm with a comparably large differece. In fact, the accuracy can be achieved in the proposed algorithms are in the level of 0.001. If I set the 2-norm in the numerator to be $max(\|\hat{\beta}_n - \beta\|_2, 0.001)$, then this relative norm will keep the same for all different scenarios.

6.4 Influence of n and p

In this section, we will mainly focus on the demonstration of the performance of the extended ADMM algorithms introduced in Chapter 3 as sample size n and dimension p increase.

The design matrix X is generated independently from $N(0, 1)$ and the response $y = X\beta + \epsilon$, where β is the same as the previous section, as well as the matrices C, D, E and vectors d and f in the linear constraints. $\lambda = 1$ and $\rho = 0.8$ are still the selected fixed values for the tuning parameters. The number of subgroups to partition along n and p are both set to be 4. In this simulation, we will generate the results for 3 differen scenarios.

- Dimension $p = 400$ fixed, while sample size n with increasing sequence 4000, 8000, and 16000.
- Sample size $n = 4000$ fixed, while the dimension p with increasing sequence 400, 800, 1600.
- Sample size n and dimension p are both increasing in the same rate with the sequences n = (4000, 8000, 16000) and p = (400, 800, 1600).

The number of iteration is set to be 100 for each case. The following metrics will be evaluted in the simulation.

- Angle between true parameter β and estimated $\hat{\beta}$.

$$\theta = \frac{< \beta, \hat{\beta} >}{\|\beta\|_2 \|\hat{\beta}\|_2}$$

- Relative error between true parameter β and estimated $\hat{\beta}$.

$$r = \frac{\|\beta - \hat{\beta}\|_2}{\|\beta\|_2}$$

- Distance between true parameter β and estimated $\hat{\beta}$.

$$d = \|\beta - \hat{\beta}\|_2$$

- Objective function based on the estimated $\hat{\beta}$.

$$\hat{f} = \frac{1}{2}\|y - x\hat{\beta}\|_2^2 + \|D\hat{\beta}\|_1$$

- Inequality constraint based on the estimated $\hat{\beta}$.

$$I_1 = mean(C\hat{\beta} - d)$$

- Equality constraint based on the estimated $\hat{\beta}$.

$$I_2 = mean(E\hat{\beta} - f)$$

The Figure 6.4, Figure 6.5 and Figure 6.6 are the generated boxplots of the simulation in the scenarios above where the extended ADMM with slack variable is $ADMM$, extended ADMM with large p is $ADMMz$, distributed ADMM splitting along n is $ADMMn$, and distributed ADMM splitting along p is $ADMMp$ and distributed ADMM splitting along n and p is $ADMMnp$.

The performance of Distributed ADMM along n and along n and p are better than the other ADMM algorithms. However, these distributed ADMM cannot achieve smaller objective function compared to the others since they both strictly hold the equality and inequality constraints. From the graph, we can see that the distributed ADMM along n and np are more robust to sample size n and dimension p than other approaches as n and p increase. At the same

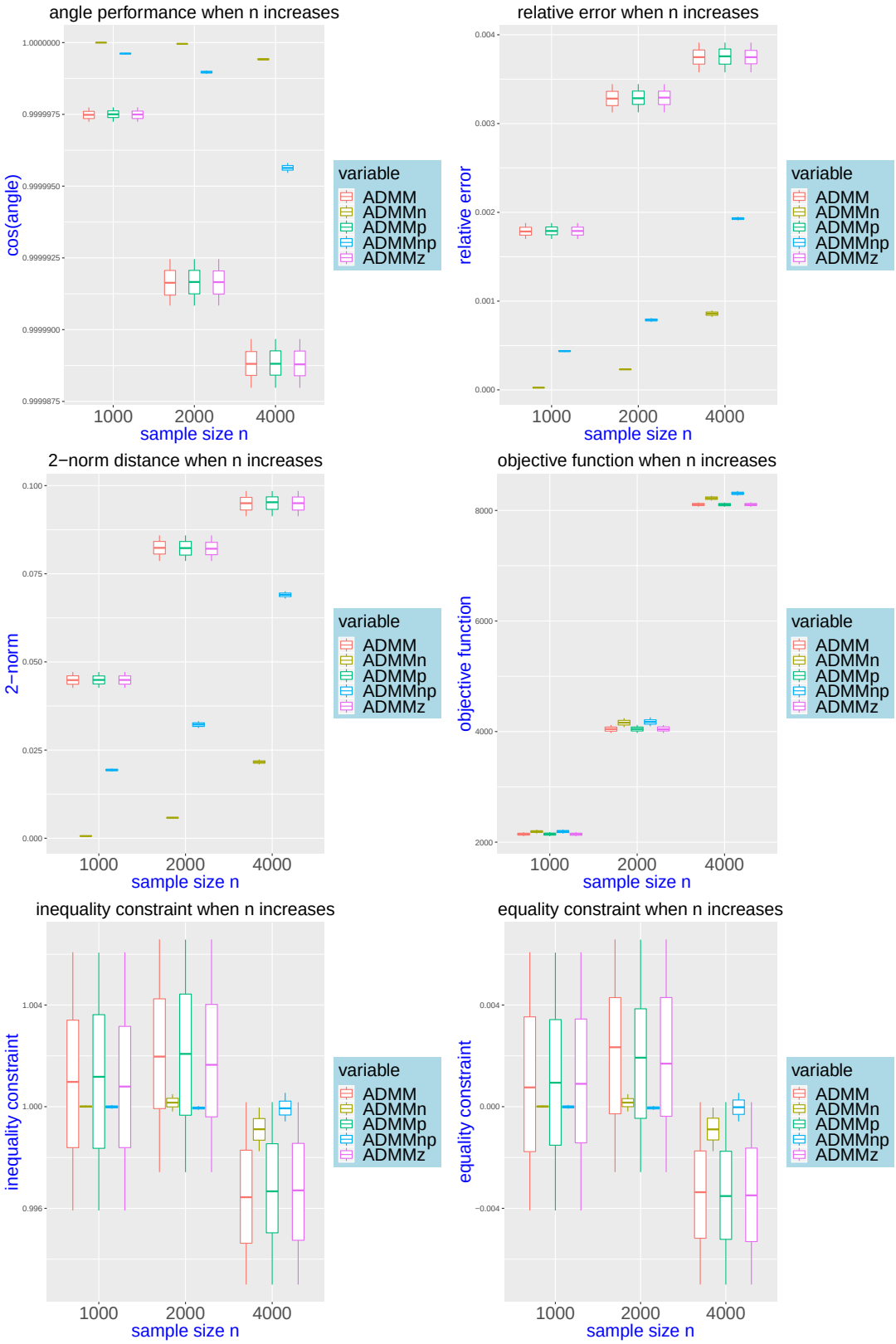

Figure 6.4: Performance when n increases

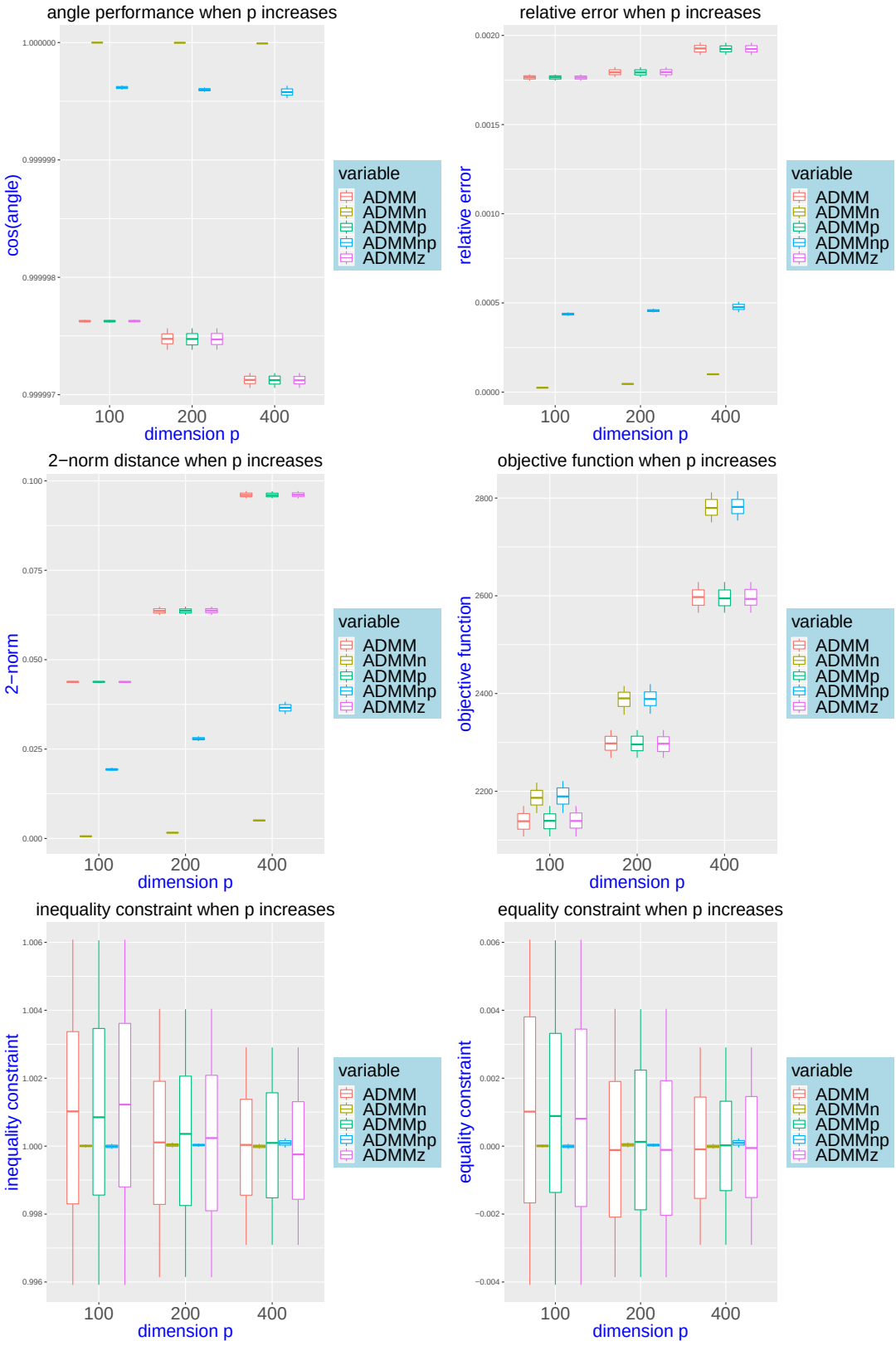

Figure 6.5: Performance when n increases

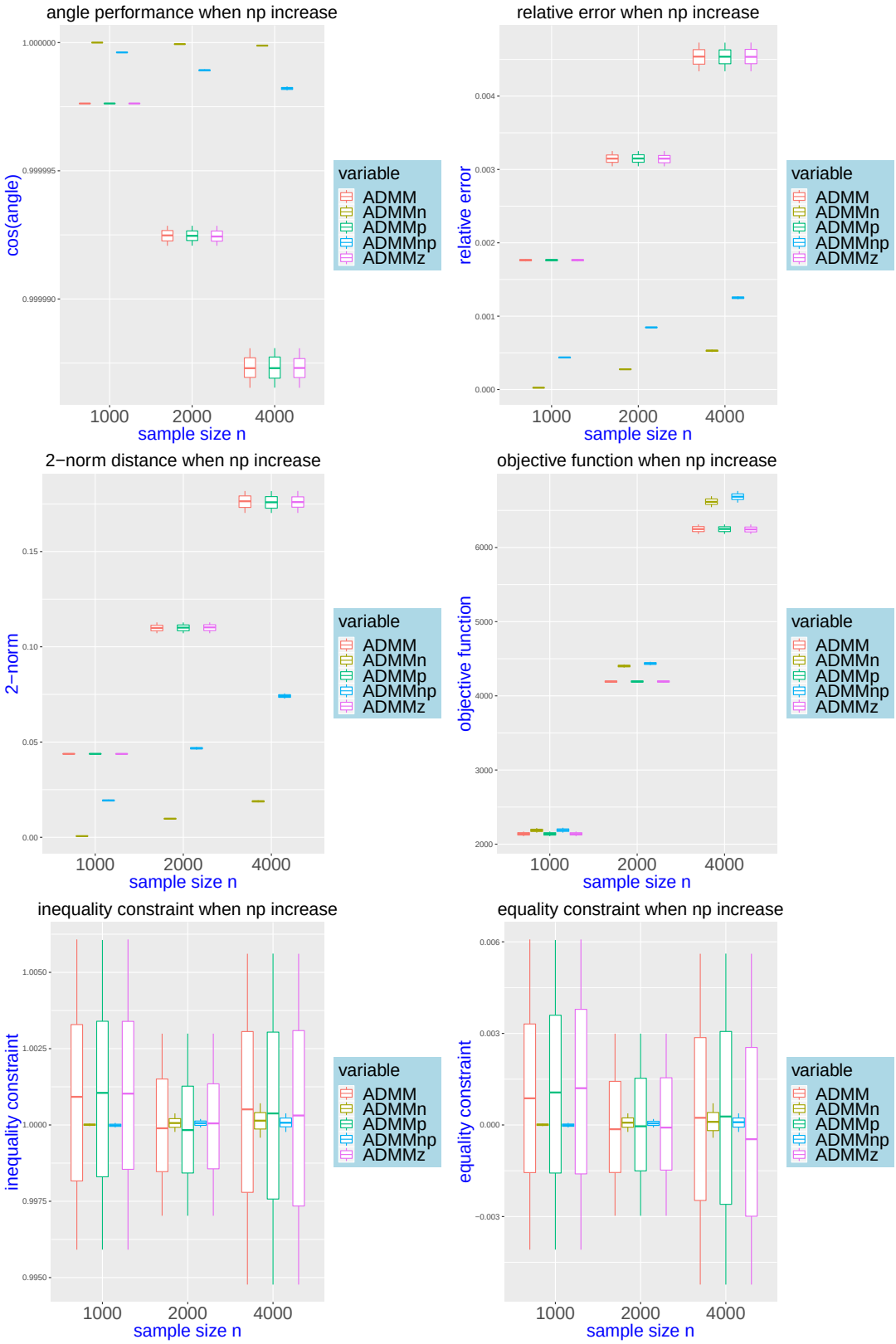

Figure 6.6: Performance when n increases

time, we expect to have higher efficiency in distributed ADMM. When splitting along n is not available to be used in the real dataset, we prefer to run the distributd ADMM along p instead of the extended ADMM with slack variables since they preserve similar performance. When the situation is $p > n$, the proposed algorithm in section 3.2 needs to be considered.

Remark that in the simulation, it is without the case when $p > n$ since the uniqueness will not be guaranteed in this scenario. Distributed ADMM along p (section 3.4) and Distributed ADMM along n and p (section 3.5) can split the dataset along column to make the p in each local processor less than n in order to hold the uniqueness, while the other approaches are not able to obtain that. Although I introduced the proof of convergence for the algorithm proposed in section 3.2, the uniqueness needs to be considered and only if some good conditions are held, the application of that algorithm is reasonable to achieve higher efficiency.

The mean and standard deviation of the $\cos(Angle)$ and estimated objective function for all algorithms when n increase can be found in Table 6.1 while the mean and standard deviation of the inequality and equality constraints can be found in Table 6.2.

Table 6.1: Mean and Standard Deviation of Performance Metrics

Item		$\cos(Angle)$		Item		Objective Function	
Algorithm	Sample Size	Mean	Standard Deviation	Algorithm	Sample Size	Mean	Standard Deviation
$ADMM$	$n = 1000$	0.99	1.44×10^{-7}	$ADMM$	$n = 1000$	2142.37	21.59
	$n = 2000$	0.99	4.65×10^{-7}		$n = 2000$	4047.85	42.62
	$n = 4000$	0.99	4.85×10^{-7}		$n = 4000$	8106.04	23.48
$ADMMz$	$n = 1000$	0.99	1.43×10^{-7}	$ADMMz$	$n = 1000$	2144.74	21.82
	$n = 2000$	0.99	4.76×10^{-7}		$n = 2000$	4041.93	43.35
	$n = 4000$	0.99	4.99×10^{-7}		$n = 4000$	8105.06	22.53
$ADMMn$	$n = 1000$	1	3.58×10^{-11}	$ADMMn$	$n = 1000$	2190.56	20.49
	$n = 2000$	1	1.27×10^{-9}		$n = 2000$	4159.10	47.59
	$n = 4000$	0.99	2.05×10^{-8}		$n = 4000$	8222.48	24.73
$ADMMp$	$n = 1000$	0.99	1.47×10^{-7}	$ADMMp$	$n = 1000$	2144.67	21.43
	$n = 2000$	0.99	4.74×10^{-7}		$n = 2000$	4044.15	41.22
	$n = 4000$	0.99	4.88×10^{-7}		$n = 4000$	8104.44	23.01
$ADMMnp$	$n = 1000$	0.99	9.72×10^{-9}	$ADMMnp$	$n = 1000$	2191.42	20.91
	$n = 2000$	0.99	3.46×10^{-8}		$n = 2000$	4179.27	45.62
	$n = 4000$	0.99	9.77×10^{-8}		$n = 4000$	8310.33	23.60

6.5 Comparison of ADMM and QP

This section is intended to demonstrate the performance and speed of the proposed ADMM algorithms. As a benchmark, the problem (1.1) is also solved as a quadratic programming problem, which is computed using Gurobi (`https://www.gurobi.com/`); refer to Zeng

Table 6.2: Mean and Standard Deviation of Linear Constraints

Item		Inequality Constraint		Item		Equality Constraint	
Algorithm	Sample Size	Mean	Standard Deviation	Algorithm	Sample Size	Mean	Standard Deviation
$ADMM$	$n = 1000$	1.001	0.002	$ADMM$	$n = 1000$	7.835×10^{-4}	2.947×10^{-3}
	$n = 2000$	1.001	0.003		$n = 2000$	1.916×10^{-3}	2.653×10^{-3}
	$n = 4000$	0.997	0.002		$n = 4000$	-3.443×10^{-3}	2.093×10^{-3}
$ADMMz$	$n = 1000$	1.001	0.003	$ADMMz$	$n = 1000$	1.098×10^{-3}	2.973×10^{-3}
	$n = 2000$	1.002	0.003		$n = 2000$	1.933×10^{-3}	2.631×10^{-3}
	$n = 4000$	0.997	0.002		$n = 4000$	-3.323×10^{-3}	2.008×10^{-3}
$ADMMn$	$n = 1000$	1.000	2.528×10^{-5}	$ADMMn$	$n = 1000$	6.026×10^{-6}	2.589×10^{-5}
	$n = 2000$	1.000	1.967×10^{-4}		$n = 2000$	1.454×10^{-4}	1.998×10^{-4}
	$n = 4000$	1.000	4.845×10^{-4}		$n = 4000$	-9.197×10^{-4}	4.894×10^{-4}
$ADMMp$	$n = 1000$	1.001	0.003	$ADMMp$	$n = 1000$	8.950×10^{-4}	2.976×10^{-3}
	$n = 2000$	1.002	0.003		$n = 2000$	1.905×10^{-3}	2.688×10^{-3}
	$n = 4000$	0.997	0.002		$n = 4000$	-3.315×10^{-3}	2.086×10^{-3}
$ADMMnp$	$n = 1000$	1.000	4.264×10^{-5}	$ADMMnp$	$n = 1000$	-5.483×10^{-6}	4.270×10^{-5}
	$n = 2000$	$1,000$	4.245×10^{-5}		$n = 2000$	1.293×10^{-3}	7.749×10^{-4}
	$n = 4000$	1.000	3.192×10^{-4}		$n = 4000$	-1.196×10^{-5}	3.252×10^{-4}

et al. (2017) for details. Gurobi is one of the fastest solvers for linear and quadratic programming problems.

Randomly generate a design matrix X with $n = 4000$ rows and $p = 400$ columns, where each entry is independently sampled from $N(0, 1)$. The response vector $y = X\beta + \epsilon$, where β, matrices C, D, E, vectors d and f are the same as the setting in the previous section. $\lambda = 1$ and $\rho = 0.8$ are fixed values and the number of subgroups along n and p are fixed as 4. In this simulation, we apply the 5 proposed extended ADMM algorithms introduced in Chapter 3 and we track the estimated objective function

$$\frac{1}{2}\|y - X\hat{\beta}\|_2^2 + \rho\|D\hat{\beta}\|_1$$

as the iteration increases from 1 to the max value which is determined as 10000. The faster the objective function decrease, the more efficiency the algorithm could achieve. After the simulation, the result can be found in figure 6.7.

It is clear to find that the extended ADMM with slack variable ($ADMM$), extended ADMM with large p ($ADMMz$), distributed ADMM splitting along n ($ADMMn$), and distributed ADMM splitting along p ($ADMMp$) converge to the minimum value at the first several iterations while the Distributed ADMM splitting along n and p ($ADMMnp$) finally converge to the minimum objective function at around 250's iteration. At the same time, the quadratic programming could have the minimum objective function in the 1st iteration which is not shown in

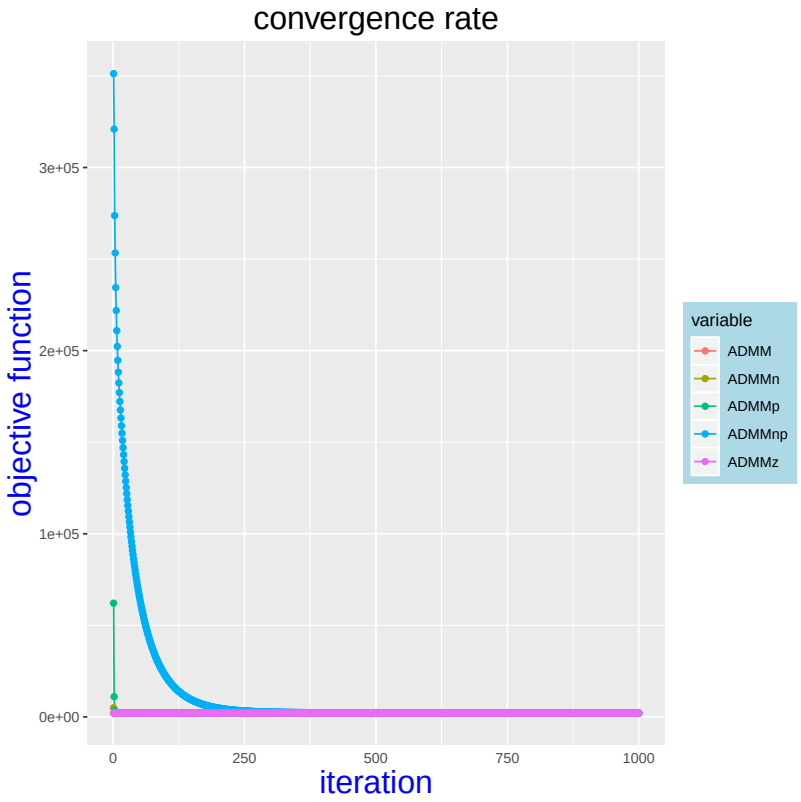

Figure 6.7: Convergence rate of extended ADMM algorithms

this figure. However, since the distributed ADMM can solve those subgroups at the same time, the time of each iteration should be much less than the time in each iterationin the extended ADMM or quadratic programming which needs to consider the whole data. In our setting, the number of subgroups along n and p are both 4, which means that the iteration time in each iteration of distributed ADMM along n and p should be $\frac{1}{4\times 4} = \frac{1}{16}$ of the time in each iteration to the non-distributed case. Although it converges at around 250's iteration, but the total time length of these iterations are about $250 \times \frac{1}{16} \approx 15$ times of the time length in each iteration of non-distributed case. Notice that the computational complexity in the algorithm in $O(np^2)$, which means the efficiency in each subgroup is larger than 16 times of the efficiency in the whole data. As sample size n and dimension p goes larger, this difference will be more obvious and the distributed ADMM will be much more efficient which is more appropriate to be used to the big data.

Chapter 7

Real Example

In this chapter, I will mainly discuss the performance of the extended ADMM algorithm on real datasets. The first dataset is the one related to the cooperative spectrum sensing for cognitive radio networks (CRN) and the other one is from the National Health and Nutrition Examination Survey (NHANES). In the dataset of CRN, I applied the ADMM with slack variables and the ADMM splitting along n and ADMM splitting along p are both implemented in the dataset of NHANES.

7.1 Cooperative spectrum sensing for cognitive radio networks

With the proliferation in radio communication systems, scarce bandwidth resources and high cost licenses are getting more common which could limit the access to emergent wireless applications. There is evidence to state that the perceived under-utilization of the specturm is generated by the access policy where applications are assigned with fixed frequency bands. This fact motivates the development of cognitive radio networks' capability to sense the psectrum and access it opportunistically.

According to Mateos et al. (2010), after devising the cooperative approach to the sensing task of CR networks, the basic expansion model for the spectrum is given below:

$$\Phi_r(f) = \sum_{s=1}^{N_s} g_{sr}\Phi_s(f) = \sum_{s=1}^{N_s} g_{sr} \sum_{b=1}^{N_b} \beta_{bs}\phi_b(f),$$

where $\Phi_r(f)$ denotes the power spectrum density (PSD) at frequency f at the location of the rth CR, which is generated by the superposition of PSDs $\Phi_s(f)$ from N_s sources. The coefficient

g_{sr} represents the channel gain which models the average propagation loss between source s and the CR r. $\phi_b(f)$ represents the rectangular pulses of unit height and the parameter β_{bs} denotes how much power is emitted by source s in the frequency band spanned by the basis $\phi_b(f)$.

Furthermore, in the cooperative scenario, N_r sensing CRs collect smoothed periodogram samples $\{y_{rk}\}_{r=1}^{N_r}$ of the received signal at frequencies $\{f_k\}_{k=1}^{N_f}$. The model is :

$$y_{rk} = \Phi_r(f_k) + \eta_{rk},$$

where the noise η_{rk} is modeled as a Gaussian random variable.

By combining these two models, the model used for ADMM and convex optimization is given below:

$$y_{rk} = \sum_{s=1}^{N_s} g_{sr} \sum_{b=1}^{N_b} \beta_{bs}\phi_b(f_k) + \eta_{rk}$$

After rewriting this into matrix form, the following equation can be obtained:

$$Y = A\beta + \eta$$

where

$$Y = \begin{bmatrix} y_{1,1} \\ \vdots \\ y_{1,N_f} \\ y_{2,1} \\ \vdots \\ y_{2,N_f} \\ \vdots y_{N_r,1} \\ \vdots \\ y_{N_r,N_f} \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_{1,1} \\ \vdots \\ \beta_{N_b,1} \\ \beta_{1,2} \\ \vdots \\ \beta_{N_b,2} \\ \vdots \beta_{1,N_s} \\ \vdots \\ \beta_{N_b,N_s} \end{bmatrix} \qquad \eta = \begin{bmatrix} \eta_{1,1} \\ \vdots \\ \eta_{1,N_f} \\ \eta_{2,1} \\ \vdots \\ \eta_{2,N_f} \\ \vdots \eta_{N_r,1} \\ \vdots \\ \eta_{N_r,N_f} \end{bmatrix}$$

And

$$A = \begin{bmatrix} g_{11}\phi_1(f_1) & \dots & g_{11}\phi_b(f_1) & g_{21}\phi_1(f_1) & \dots & g_{21}\phi_b(f_1) & \dots & g_{N_s1}\phi_b(f_1) \\ g_{11}\phi_1(f_2) & \dots & g_{11}\phi_b(f_2) & g_{21}\phi_1(f_2) & \dots & g_{N_s1}\phi_1(f_2) & \dots & g_{N_s1}\phi_b(f_2) \\ \dots & & & & & & & \\ g_{11}\phi_1(f_{N_f}) & \dots & g_{11}\phi_b(f_{N_f}) & g_{21}\phi_1(f_{N_f}) & \dots & g_{N_s1}\phi_1(f_{N_f}) & \dots & g_{N_s1}\phi_b(f_{N_f}) \\ g_{12}\phi_1(f_1) & \dots & g_{12}\phi_b(f_1) & g_{22}\phi_1(f_1) & \dots & g_{N_s2}\phi_1(f_1) & \dots & g_{N_s2}\phi_b(f_1) \\ \dots & & & & & & & \\ g_{12}\phi_1(f_{N_f}) & \dots & g_{12}\phi_b(f_{N_f}) & g_{22}\phi_1(f_{N_f}) & \dots & g_{N_s2}\phi_1(f_{N_f}) & \dots & g_{N_s2}\phi_b(f_{N_f}) \\ \dots & & & & & & & \\ g_{1N_r}\phi_1(f_1) & \dots & g_{1N_r}\phi_b(f_1) & g_{2N_r}\phi_1(f_1) & \dots & g_{N_sN_r}\phi_1(f_1) & \dots & g_{N_sN_r}\phi_b(f_1) \\ \dots & & & & & & & \\ g_{1N_r}\phi_1(f_{N_f}) & \dots & g_{1N_r}\phi_b(f_{N_f}) & g_{2N_r}\phi_1(f_{N_f}) & \dots & g_{N_sN_r}\phi_1(f_{N_f}) & \dots & g_{N_sN_r}\phi_b(f_{N_f}) \end{bmatrix}$$

Following Mateos et al. (2010), the numerical example is generated by 5 sources and each source's PSD $\Phi_s(f)$ corresponds to one of $N_b = 9$ non-overlapping rectangular pulses $\phi_i(f_j) = (i \times j)$ MHz, $i = 1, 2, \dots, b, j = 1, 2, \dots, N_f$. Samples of the PSD field at $N_f = 8$ frequencies are acquired by $N_r = 50$ CRs. The CRs collaborate to locate the sources on a rectangular grid of $N_s = 121$ candidate positions. The gain is selected as $g_{sr} = min\{1, (\frac{200}{\|x_s - x_r\|_2})^3\}$. Based on the definition of these quantities, it would be reasonable to have the following assumption:

- only a few β are nonzero. The band nature of source-PSDs is narrow, and in this way, could only span a few frequency bands that most of them are zeros. Under this condition, in the LASSO problem, the penalty matrix D could be set to identity matrix.

- all the elements in β should be nonnegative. According to the definition of β, each component is the power emitted by source s in the frequency band. In common sense, the emitting power could only be nonnegative value since it is not reasonable to emit negative power. Based on this assumption, in the LASSO problem, the matrix C could be set to be identity matrix and vector d could be a zero vector.

- x_s and x_r are the position in the coordinate plane. Based on the introduction of this dataset, function g_{sr} represents the channel gain modeling the average propagation loss between the source s and the CR r which is known function of their distance. In this case, the data could be generated when the positions of those sources are fixed in the plane.

After generating the data of x_s and x_r randomly and the corresponding periodogram samples y_{rk}, the objective function could be rewritten in the following form:

$$\begin{aligned} &\min_{\beta} \frac{1}{2}\|Y - A\beta\|_2^2 + \lambda\|D\beta\|_1, \\ &\text{subject to} \quad C\beta \geq d, \end{aligned}$$

where

$$D = I_{N_b N_s}, \quad C = I_{N_b N_s} \quad \text{and} \quad d = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

In this way, the basic expansion model for the spectrum could be transfered into the generalized LASSO problem with inequality constraint. There is no equality constraint in this example. The matrix E is then assumed to be zero matrix which can be ignored in this problem.

Using the proposed ADMM algorithm with slack variables, the tuning parameter selection plot associated with BIC can be obtained in Figure 7.1. In the tuning parameter selection, the BIC is defined in (6.1). From Figure 7.1, $\lambda = 1.5$ is selected as the best tuning parameter to estimate the parameter β.

To evaluate the performance of the estimation with the selected λ, we can calculate the average prediction error (APE) of the fitted model. The full datasets $T = \{(A_i, Y_i), i = 1, 2, \ldots, n\}$, where $n = N_r * N_f = 400$ in our case, can be divided randomly into cross-validation training set $T - T^k$ and test sets T^k, $k = 1, 2, 3, 4$. The parameters are estimated

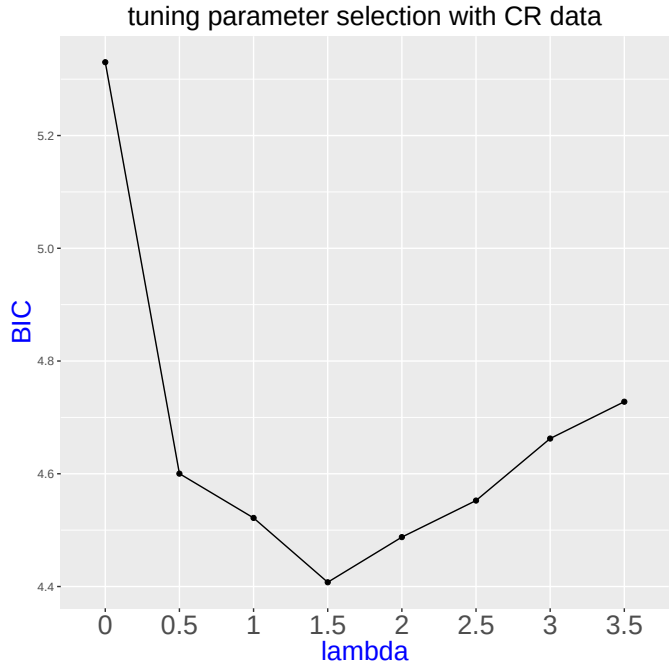

Figure 7.1: Tuning parameter selection with CR data

using the training sets and the prediction errors are calculated using the test sets. The APE is defined as

$$APE = \frac{1}{4}\sum_{k=1}^{4}\{\frac{1}{n_k}\sum_{i\in T^k}\|Y_i - A_i\hat{\beta}_{T-T^k}\|_2\},$$

where n_k is the sample size of test set T^k. We set $n_k = 100$ for all $k = 1, 2, 3, 4$ here. In conclusion, the APE of the model with linear constraints is 12.02. Remark that by applying the Extended ADMM with new constraint as introduced in section 3.2, the computational complexity could be reduced while the estimation error willl not vary much.

7.2 National health and nutrition examination survey

The second real data analysis is based on the National Health and Nutrition Examination Survey: `https://www.cdc.gov/nchs/nhanes/ContinuousNhanes/Default.aspx?BeginYear=2015`. Body fatness has been an important psychosocial issue among humans for a long time, which is difficult to quantify. That is, each individual has his/her own perception of how fat he/she should be. The Body Mass Index (BMI) is the metric currently

in use to represent the index of an individual's fatness which is defined as $\frac{weight}{(height)^2}$. It also is widely used as a risk factor for the development of or the prevalence of several health issues. In addition, it is widely used in determining public health policies. The BMI has been useful in population-based studies by virtue of its wide acceptance in defining specific categories of body mass as a health issue. We would like to apply the extended ADMM algorithm to show the significance of several variables to the metric BMI, in other words, the fatness of an individual.

In this analysis, I used the data from 2011 to 2016 and selected several predictors to analyze the linear regression with the response BMI. The predictors include: alkaline phosphatase (x_1), total calcium (x_2), globulin (x_3), glucose (x_4), iron (x_5), potassium (x_6), sodium (x_7), total protein (x_8) and uric acid (x_9), albumin in urine (x_{10}), creatinine in urine (x_{11}), blood pressure (x_{12}), age in years of the participant at the time of screening (x_{13}), length of time the participant has been in the US (x_{14}), total cholesterol (x_{15}), triglyceride (x_{16}). The objective of this analysis is to use the following two methods: extended ADMM with slack variables and distributed ADMM splitting along n to solve this linear regression problem. In the distributed computing case, I used the year of examination survey to divide the dataset into 3 pieces: the individuals that take the examination survey between 2011 - 2012, between 2013 - 2014 and between 2015 - 2016. Using these slices, the parallel computing using ADMM with splitting along n could be applied to the dataset. There are many existing paper discussing the association between the predictors and BMI. For example, in Khan et al. (2015), a significant linear relationship with p-value less than 0.0001 has been found between alkaline phosphatase and BMI, while in Dos Santos et al. (2005), a significant negative correlation between the calcium intake and BMI has been analyzed. More details can be found in Madhuvanthi and Lathadevi (2016), Innocent et al. (2013), Eftekhari et al. (2009), Elfassy et al. (2018), Pimpin et al. (2015), Honggang et al. (2014), Gerchman et al. (2009), Linderman et al. (2018), Mitchell et al. (2013), Faheem et al. (2010) and Frazee et al. (2015) for the associations between BMI and other predictors. Based on the information, the assumed value of matrix C, D, and vectors d are as follows:

$D = I_{18}$, $d = 0_{16}$, and C is diagonal matrix with entries:

$$\{1, -1, -1, 1, -1, -1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1\}$$

on the diagonal.

Transfer this linear regression problem to the following optimization problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D\beta\|_1,$$
$$\text{subject to} \quad C\beta \geq d,$$

where y is the vector of responses in BMI and X is the matrix with the observations of the predictors. This is a generalized LASSO problem with inequality constraints which can be solved by using the ADMM algorithms proposed in Chapter 3. After applying the proposed algorithms in section 3.1 and section 3.3 to this dataset, Figure 7.2 can be obtained.

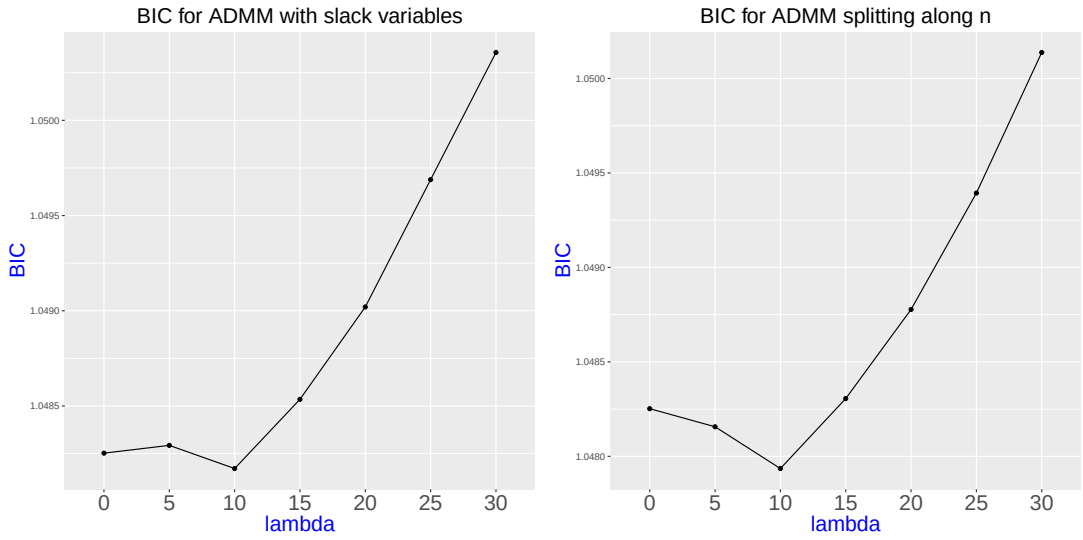

Figure 7.2: Tuning parameter selection with NHNES data

In this analysis, the BIC is based on (6.1) and ρ is fixed at 1. From Figure 7.2, we can find that the best tuning parameter λ for the both approaches should be 10. By selecting the tuning parameter $\lambda = 10$ in both scenarios, the estimated linear regression for BMI and the other predictors are given as follows:

$$y = 0.010x_1 - 0.160x_2 + 0.006x_3 + 0.125x_4 - 0.146x_5 + 0.033x_6$$
$$- 0.002x_7 + 0.002x_8 + 0.161x_9 - 0.069x_{10} + 0.140x_{11}$$
$$+ 0.080x_{12} + 0.089x_{13} + 0.033x_{14} + 0.028x_{15} + 0.127x_{16}$$

and

$$y = 0.009x_1 - 0.162x_2 + 0.013x_3 + 0.123x_4 - 0.145x_5 + 0.011x_6$$
$$- 0.002x_7 + 0.005x_8 + 0.159x_9 - 0.069x_{10} + 0.140x_{11}$$
$$+ 0.080x_{12} + 0.088x_{13} + 0.033x_{14} + 0.027x_{15} + 0.127x_{16}$$

Based on the estimated linear equations above, the proposed ADMM with slack variables and ADMM with splitting along n can be demonstrated that most of the parameters are with the similar coefficients to predict the response except for those with little significance compared with other predictors. Based on the analysis, we can see that the BMI has positive association with alkaline phosphatase, globulin, glucose, potassium, total protein, uric acid, creatinine in urine, blood pressure, age in years of the participant at the time of screening, length of time the participant has been in the US, total cholesterol and triglyceride. On the other hand, BMI has negative association with total calcium, iron, sodium and albumin in urine. It is easy to find that the significance of total calcium, glucose, iron, urine acid,creatinine in urine and triglyceride are much higher than the other predictors while alkaline phosphatase, globulin, sodium and total protein has little significance to BMI.

Chapter 8

Future Work

There are several other scenarios and problems in this research that we expect to accomplish in the future. First, we expect to apply the extended ADMM algorithms proposed in Chapter 3 to some other penalized regression models and do simulations with increase n and p to see whether the proposed algorithms can work on other models as well or is there anything we could revise to make it work. Specifically, the Huber loss function and the Logistic regression, which is a popular model in survival analysis and other pharmaceutical area, will be considered and tested here. Next, we also need to consider the asymptotic behavior of these algorithm. Notice that in the proof of convergence in Chapter 4, we only showed the case when n and p are fixed with assumption 4 assumed to be true. we are going to work on the asymptotic case when n and p both increase to infinity in the same rate, for example, $\frac{p}{n} = \frac{1}{2}$. We know that several researchers have work in this area and have showed that some existing algorithm will no longer work in this scenario. We are wondering if the convergence could still be held when this situation occurs. Moreover, since we have 5 extended ADMM algorithms proposed in this dissertation, a package in R needs to be developed which should compile all these algorithms. In fact, in order to apply the openMPI to run parallel computing, C language is also used and the C code with distributed computing worked as the source file in R to read. Furthermore, we will come up with more real examples to show the availability of our proposed algorithms with different scenarios. Since we didn't use the extended ADMM with new constraint, distributed ADMM splitting along p or distributed ADMM splitting along n and p in the real data, we may need to find some dataset that fits well for each of them to do analysis. Moreover, the BMI example didn't show a signficant variable selection which is common in generalied LASSO problem.

Another dataset is needed to illustrate of variable selection of the proposed algorithms. Last but not least, in the proof of convergence in Chapter 4, we applied the sufficient condition in Chen et al. (2016). We would like to do one more step to find the sufficient and necessary condition of the convergence of extended ADMM so that we may not need to have the strong assumption 4 in the proof of distributed ADMM splitting along p and distributed ADMM splitting along n and p. If the sufficient and necessary condition can be found, those two algorithms may achieve convergence naturally without any further assumption which can also help in the proof of convergence when we apply the ADMM algorithms to other penalized models, such as logistic regression.

References

Ali, A. and R. J. Tibshirani (2018). The generalized lasso problem and uniqueness. *arXiv preprint arXiv:1805.07682*.

Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

Bien, J., J. Taylor, and R. Tibshirani (2013). A lasso for hierarchical interactions. *Annals of statistics 41*(3), 1111.

Bioucas-Dias, J. M. and M. A. Figueiredo (2010). Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2010 2nd Workshop on*, pp. 1–4. IEEE.

Bogdan, M., E. v. d. Berg, W. Su, and E. Candes (2013). Statistical estimation and testing via the sorted l1 norm. *arXiv preprint arXiv:1310.1969*.

Boyd, S., N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning 3*(1), 1–122.

Chen, C., B. He, Y. Ye, and X. Yuan (2016). The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming 155*(1-2), 57–79.

Chen, G. and M. Teboulle (1994). A proximal-based decomposition method for convex minimization problems. *Mathematical Programming 64*(1-3), 81–101.

Combettes, P. L. and J.-C. Pesquet (2008). A proximal decomposition method for solving convex variational inverse problems. *Inverse problems 24*(6), 065014.

Donoho, D. L. (2006). For most large underdetermined systems of equations, the minimal 1-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences 59*(7), 907–934.

Donoho, D. L. and M. Elad (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization. *Proceedings of the National Academy of Sciences 100*(5), 2197–2202.

Donoho, D. L. and X. Huo (2001). Uncertainty principles and ideal atomic decomposition. *IEEE transactions on information theory 47*(7), 2845–2862.

Donoho, D. L., I. M. Johnstone, G. Kerkyacharian, and D. Picard (1995). Wavelet shrinkage: asymptopia? *Journal of the Royal Statistical Society: Series B (Methodological) 57*(2), 301–337.

Dos Santos, L. C., L. Araújo Martini, I. Pádua Cintra, and M. Fisberg (2005). Relationship between calcium intake and body mass index in adolescents. *Archivos latinoamericanos de nutricion 55*(4), 345–349.

Eckstein, J. and M. Fukushima (1994). Some reformulations and applications of the alternating direction method of multipliers. In *Large scale optimization*, pp. 115–134. Springer.

Efron, B., T. Hastie, I. Johnstone, R. Tibshirani, et al. (2004). Least angle regression. *The Annals of statistics 32*(2), 407–499.

Eftekhari, M., H. Mozaffari-Khosravi, and F. Shidfar (2009). The relationship between bmi and iron status in iron-deficient adolescent iranian girls. *Public health nutrition 12*(12), 2377–2381.

Elfassy, T., Y. Mossavar-Rahmani, L. Van Horn, M. Gellman, D. Sotres-Alvarez, N. Schneiderman, M. Daviglus, J. M. Beasley, M. M. Llabre, P. A. Shaw, et al. (2018). Associations of

sodium and potassium with obesity measures among diverse us hispanic/latino adults: results from the hispanic community health study/study of latinos. *Obesity 26*(2), 442–450.

Faheem, M., S. Qureshi, J. Ali, H. Hameed, Z. Zahoor, F. Abbas, A. M. Gul, and M. Hafizullah (2010). Does bmi affect cholesterol, sugar, and blood pressure in general population? *Journal of Ayub Medical College Abbottabad 22*(4), 74–77.

Fan, J. and R. Li (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association 96*(456), 1348–1360.

Fortin, M. (1975). Minimization of some non-differentiable functionals by the augmented lagrangian method of hestenes and powell. *Applied Mathematics and Optimization 2*(3), 236–250.

Fortin, M. and R. Glowinski (2000). *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*, Volume 15. Elsevier.

Frazee, E. N., E. M. Nystrom, M. M. McMahon, E. E. Williamson, and J. M. Miles (2015). Relationship between triglyceride tolerance, body mass index, and fat depots in hospitalized patients receiving parenteral nutrition. *Journal of Parenteral and Enteral Nutrition 39*(8), 922–928.

Fukushima, M. (1992). Application of the alternating direction method of multipliers to separable convex programming problems. *Computational Optimization and Applications 1*(1), 93–111.

Gabay, D. (1983). Applications of the method of multipliers to variational inequalities. *Studies in Mathematics and Its Applications*, 299–331.

Gabay, D. and B. Mercier (1975). *A dual algorithm for the solution of non linear variational problems via finite element approximation*. Institut de recherche d'informatique et d'automatique.

Gaines, B. R., J. Kim, and H. Zhou (2018). Algorithms for fitting the constrained lasso. *Journal of Computational and Graphical Statistics 27*(4), 861–871.

Gerchman, F., J. Tong, K. M. Utzschneider, S. Zraika, J. Udayasankar, M. J. McNeely, D. B. Carr, D. L. Leonetti, B. A. Young, I. H. de Boer, et al. (2009). Body mass index is associated with increased creatinine clearance by a mechanism independent of body fat distribution. *The Journal of Clinical Endocrinology & Metabolism 94*(10), 3781–3788.

Giesen, J. and S. Laue (2016). Distributed convex optimization with many convex constraints. *arXiv preprint arXiv:1610.02967*.

Glowinski, R. and P. Le Tallec (1987). Augmented lagrangian methods for the solution of variational problems. Technical report, WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER.

Goldstein, T. and S. Osher (2009). The split bregman method for l1-regularized problems. *SIAM journal on imaging sciences 2*(2), 323–343.

Graham, R. L., G. M. Shipman, B. W. Barrett, R. H. Castain, G. Bosilca, and A. Lumsdaine (2006). Open mpi: A high-performance, heterogeneous mpi. In *2006 IEEE International Conference on Cluster Computing*, pp. 1–9. IEEE.

He, B., H. Yang, and S. Wang (2000). Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications 106*(2), 337–356.

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of optimization theory and applications 4*(5), 303–320.

Hoefling, H. (2010). A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics 19*(4), 984–1006.

Honggang, W., W. Lizhen, X. Rui, D. Weijie, G. Chengcheng, S. Peng, F. Z. Xiaodan HUANG, Y. Xiaozhong, and J. Guozhong (2014). Association of serum uric acid with body mass index: a cross-sectional study from jiangsu province, china. *Iranian journal of public health 43*(11), 1503.

Innocent, O., O. O. ThankGod, E. O. Sandra, and I. E. Josiah (2013). Correlation between body mass index and blood glucose levels among some nigerian undergraduates. *HOAJ Biology 2*(1), 4.

Khan, A. R., F. R. Awan, S. Najam, M. Islam, T. Siddique, and M. Zain (2015). Elevated serum level of human alkaline phosphatase in obesity. *Age (years) 48*(8.8), 42–5.

Kim, S.-J., K. Koh, S. Boyd, and D. Gorinevsky (2009). \ell_1 trend filtering. *SIAM review 51*(2), 339–360.

Li, X., L. Mo, X. Yuan, and J. Zhang (2014). Linearized alternating direction method of multipliers for sparse group and fused lasso models. *Computational Statistics & Data Analysis 79*, 203–221.

Linderman, G. C., J. Lu, Y. Lu, X. Sun, W. Xu, K. Nasir, W. Schulz, L. Jiang, and H. M. Krumholz (2018). Association of body mass index with blood pressure among 1.7 million chinese adults. *JAMA network open 1*(4), e181271–e181271.

Madhuvanthi, M. and G. Lathadevi (2016). Serum proteins alteration in association with body mass index in human volunteers. *Journal of clinical and diagnostic research: JCDR 10*(6), CC05.

Mateos, G., J. A. Bazerque, and G. B. Giannakis (2010). Distributed sparse linear regression. *IEEE Transactions on Signal Processing 58*(10), 5262–5276.

Meinshausen, N. and P. Bühlmann (2006). Variable selection and high-dimensional graphs with the lasso. *Annals of Statistics 34*, 1436–1462.

Mercier, B. (1979). *Topics in finite element solution of elliptic problems*. Springer.

Miele, A., E. Cragg, R. Iyer, and A. Levy (1971). Use of the augmented penalty function in mathematical programming problems, part 1. *Journal of Optimization Theory and Applications 8*(2), 115–130.

Mitchell, J. A., D. Rodriguez, K. H. Schmitz, and J. Audrain-McGovern (2013). Greater screen time is associated with adolescent obesity: a longitudinal study of the bmi distribution from ages 14 to 18. *Obesity 21*(3), 572–575.

Mota, J. F., J. M. Xavier, P. M. Aguiar, and M. Püschel (2011). A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. *arXiv preprint arXiv:1112.2295*.

O'Donoghue, B., G. Stathopoulos, and S. Boyd (2013). A splitting method for optimal control. *IEEE Transactions on Control Systems Technology 21*(6), 2432–2442.

Osborne, M. R., B. Presnell, and B. A. Turlach (2000). On the lasso and its dual. *Journal of Computational and Graphical statistics 9*(2), 319–337.

Pimpin, L., S. Jebb, L. Johnson, J. Wardle, and G. L. Ambrosini (2015). Dietary protein intake is associated with body mass index and weight up to 5 y of age in a prospective cohort of twins, 2. *The American journal of clinical nutrition 103*(2), 389–397.

Recht, B., C. Re, S. Wright, and F. Niu (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pp. 693–701.

Rockafellar, R. T. (1973). A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical Programming 5*(1), 354–373.

Rockafellar, R. T. (1976). Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of operations research 1*(2), 97–116.

Rudin, L. I., S. Osher, and E. Fatemi (1992). Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena 60*(1-4), 259–268.

Salehani, Y. E., S. Gazor, S. Yousefi, and I.-M. Kim (2014). Adaptive lasso hyperspectral unmixing using admm. In *Communications (QBSC), 2014 27th Biennial Symposium on*, pp. 159–163. IEEE.

Shi, W., Q. Ling, G. Wu, and W. Yin (2015). Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization 25*(2), 944–966.

Steidl, G., S. Didas, and J. Neumann (2006). Splines in higher order tv regularization. *International journal of computer vision 70*(3), 241–255.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Tibshirani, R., M. Saunders, S. Rosset, J. Zhu, and K. Knight (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*(1), 91–108.

Tibshirani, R. J. et al. (2013). The lasso problem and uniqueness. *Electronic Journal of Statistics 7*, 1456–1490.

Tibshirani, R. J. and J. Taylor (2011). *The solution path of the generalized lasso*, Volume 39. Stanford University.

Tseng, P. (1991). Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization 29*(1), 119–138.

Wang, S. and L. Liao (2001). Decomposition method with a variable parameter for a class of monotone variational inequality problems. *Journal of optimization theory and applications 109*(2), 415–429.

White, T. (2012). *Hadoop: The definitive guide*. " O'Reilly Media, Inc.".

Xue, L., S. Ma, and H. Zou (2012). Positive-definite 1-penalized estimation of large covariance matrices. *Journal of the American Statistical Association 107*(500), 1480–1491.

Yin, W., S. Osher, D. Goldfarb, and J. Darbon (2008). Bregman iterative algorithms for l1-minimization with applications to compressed sensing: Siam journal on imaging sciences, 1 (1), 143–168. *CrossRef Google Scholar*.

Zaharia, M., M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica (2010). Spark: Cluster computing with working sets. *HotCloud 10*(10-10), 95.

Zeng, P., Q. Hu, and X. Li (2017). Geometry and degrees of freedom of linearly constrained generalized lasso. *Scandinavian Journal of Statistics 44*(4), 989–1008.

Zhang, T. and H. Zou (2014). Sparse precision matrix estimation via lasso penalized d-trace loss. *Biometrika 101*(1), 103–120.

Zhu, Y. (2017). An augmented admm algorithm with application to the generalized lasso problem. *Journal of Computational and Graphical Statistics 26*(1), 195–204.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association 101*(476), 1418–1429.