The Pennsylvania State University

The Graduate School

MULTI-BLOCK ADMM ALGORITHMS FOR HIGH-DIMENSIONAL SPARSE ESTIMATION

A Dissertation in

Statistics

by

Jiawei Wen

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

December 2020

The dissertation of Jiawei Wen was reviewed and approved by the following:

Runze Li
Eberly Family Chair Professor of Statistics and Professor of Public Health Sciences
Dissertation Co-Advisor
Co-Chair of Committee

Ethan Xingyuan Fang
Assistant Professor of Statistics
Dissertation Co-Advisor
Co-Chair of Committee

Lingzhou Xue
Associate Professor of Statistics

Helen Kamens
Associate Professor of Biobehavioral Health

Ephraim Hanks
Associate Professor of Statistics
Chair of Graduate Studies

Abstract

The alternating direction method of multipliers (ADMM) has drawn considerable attention due to its applicability to massive optimization problems. The optimization problems associated with a large number of statistical models can be formulated as convex programs. In many cases, objective functions are non-smooth, which presents computational challenges for high-dimensional data analysis. We design efficient and parallelizable algorithms for high-dimensional sparse estimation. The proposed algorithm addresses the computational challenges through feature split and multi-block ADMM algorithm.

In the first part of the dissertation, we present the multi-block ADMM algorithms for sparse quantile regression and sparse support vector machines. When the regularization term is non-convex, we apply the one-step LLA procedure and verify the strong oracle property of the resulting estimator. We establish the rate of convergence of the proposed ADMM method and compare it with existing solvers in various high dimensional settings. The proposed method can be implemented distributedly across multiple processors, which alleviates the storage and scalability limitations of a single machine in the large-scale data processing. In the second part, we apply the proposed algorithmic framework to find the Dantzig selector and linear programming discriminant (LPD) rule. We propose their non-convex generalizations and study the strong oracle property of the estimator constructed by the one-step LLA algorithm. Computationally, we approach two problems through feature partition and multi-block ADMM. We compare our feature-split algorithm with a standard linear programming solver that uses the full data. The numerical results suggest that linear programming solver fails for dimensions of tens of thousands, and our proposed algorithms work well for such high dimensions.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my special appreciation to my research advisors, Dr. Runze Li and Dr. Ethan Fang, for their guidance and endless support throughout my research and career development. Their expert knowledge of statistics and great spirit helped me overcome numerous challenges throughout my graduate study. Without their supervision, it would be impossible for me to complete this dissertation. I would also like to extend my gratitude to my committee members: Dr. Lingzhou Xue and Dr. Helen Kamens, for their insightful recommendations and suggestions on my research.

I gratefully acknowledge the funding sources that have made this dissertation work possible. This work was financially supported by the Biomedical Big Data to Knowledge (B2D2K) Pre-doctoral Training Program. I would like to express my thanks to my cohort and Dr. Cooduvalli Shashikant for many pleasant meetings and helpful discussions.

Last but not least, I would like to sincerely thank my parents and family members for their unconditional love and support. Thank you for cherishing with me every precious moment and encouraging me all the time. I would also like to thank my friends. Thank you for being a part of my journey and filling my time in Pennsylvania and California with so many colorful memories.

Chapter 1

Introduction

Nowadays, enormous large-scale datasets have aroused frequently in many fields of modern scientific research, from genomics and biomedical science to finance and machine learning. This presents challenges to classic statistical methods that become ineffective or infeasible when the dimension p is very large. More discussions on challenges imposed by big data in statistical analysis can be found in [1] and [2]. In response to the advent of big data, studies on distributed learning have drawn considerable attention in recent years. The critical procedure is to split a dataset into disjoint subsets and distribute them to multiple processors; then each worker processor will only use a subset of data and update a local variable using properly designed learning algorithms. The updated local variables are periodically collected from worker processors and aggregated at the master processor where global variables reside. Recent advances in distributed algorithms include [3, 4, 5, 6, 7, 8, 9, 10], where they target on the large n problem and the data is partitioned across samples to a manageable size. By appropriately aggregating the local estimators, the resulting estimators can attain nice theoretical and computational properties.

In many scientific applications, collected datasets are common to have a huge number of features rather than samples, in which case it is more reasonable to split across the feature space. For example, [11] devises a split-and-merge approach (SAM) for variable selection in high dimensional data analysis. SAM first partitions the ultrahigh dimensional dataset into a number of lower dimensional subsets, performs variable selection on each subset, and then performs a second Bayesian variable selection for variables survive the first stage selection. One drawback of SAM is that when strong correlations among predictors exist, the first step tends to select a large number of correlated variables that are irrelevant to the response variable. Dealing with strong correlations among features is an inherent challenge of feature space partition, where naive partition may result in biased estimation. To address this issue for penalized regression models, [12] proposes a parallel computing framework for high dimensional data which performs one-step decorrelation before partition.

Recently, the alternating direction method of multipliers (ADMM) has drawn considerable attention due to its applicability to massive optimization problems. For distributed optimization problems, the distributed ADMM and its many variations have been rigorously studied in the literature. In distributed ADMM, the original problem is partitioned into subproblems, each containing a subset of training samples or learning parameters. At each iteration, the worker processors solve the subproblems and send the up-to-date variables to the master, who then summarizes and broadcasts the results to the workers. Hence, a given large-scale learning problem can be solved in parallel. In the first part of the dissertation, we adopt a specially designed 3-block ADMM algorithm as the building block to solve a class of optimization problems represented by sparse quantile regression

and support vector machine. Generally speaking, the problems of interest fall into the following framework,

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, L(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \alpha_j |\beta_j|, \\ &\coloneqq \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, L(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \|\boldsymbol{\alpha} \circ \boldsymbol{\beta}\|_1,\end{aligned} \tag{1.1}$$

where $L(\boldsymbol{\beta})$ is a convex but non-smooth loss function, $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_p)$ is unknown feature coefficients, and λ is the penalty parameter. $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_p)^T$ is the weight parameter associated with the coefficients. When $\boldsymbol{\alpha} = \mathbf{1}_p$, the procedure reduces to the Lasso-penalized estimation. Weighted ℓ_1 penalty is closely related to nonconvex penalized estimation problems. In particular, folded concave penalties such as SCAD [13] has been shown to perform better than Lasso in many high dimensional settings. ℓ_1 penalty tends to over-penalize large coefficients and thus does not possess the desirable oracle property. SCAD penalty is defined as

$$P_\lambda^{'}(|\beta|) = \lambda \left\{ I(|\beta| \leq \lambda) + \frac{(a\lambda - |\beta|)_+}{(a-1)\lambda} I(|\beta| > \lambda) \right\}, \quad \text{for } a > 2, \tag{1.2}$$

with $P_\lambda'(0) \coloneqq P_\lambda'(0+) \geq \lambda$. In general, folded concave penalties satisfy the following properties:

1. $P_\lambda(t)$ is non-decreasing and concave for $t \in [0, \infty)$ with $P_\lambda(0) = 0$,
2. $P_\lambda(t)$ is differentiable in $(0, \infty)$ and $P_\lambda'(0) := P_\lambda'(0+) \geq a_1\lambda$,
3. $P_\lambda'(t) \geq a_1\lambda$ for $t \in (0, a_2\lambda]$ and $P_\lambda'(t) = 0$ for $t \in [a\lambda, \infty)$ with $a > a_2$.

For nonconvex penalized estimation, the objective function is nonconvex and typically multiple local minimizers exist. The technical difficulty here is to show

that the obtained solution from a certain algorithm is the local solution with desired properties. [14] proposes to inexactly solve nonconvex penalized estimation problem by combining ℓ_1 estimation with the local linear approximation (LLA) algorithm. LLA approximates the penalty by $P_\lambda(|\beta_j|) \approx P_\lambda(|\beta_j^{(0)}|) + P'_\lambda(|\beta_j^{(0)}|)(|\beta_j| - |\beta_j^{(0)}|)$, for $\beta_j \approx \beta_j^{(0)}$, where $\beta_j^{(0)}$ is a properly chosen initial value. LLA is the best convex majorization of the concave penalty function. Given a loss function $L(\boldsymbol{\beta})$, LLA iteratively updates $\boldsymbol{\beta}^{(k+1)}$ until convergence,

$$\boldsymbol{\beta}^{(k+1)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ L(\boldsymbol{\beta}) + \sum_{j=1}^{p} P'_\lambda(|\beta_j^{(k)}|)|\beta_j| \right\}, \tag{1.3}$$

which is essentially a sequence of weighted ℓ_1-penalized estimations. Furthermore, [14] proposes to use the properly initialized one-step LLA estimator as the final estimate instead of the computationally more intensive fully converged solution. Therefore one-step LLA has the advantage on both computational and statistical efficiency.

In the second part of the dissertation, we apply the proposed computational framework to a broader range of statistical problems with the following form,

$$\begin{aligned} \min_{\boldsymbol{\beta} \in \mathcal{R}^p} \quad & f(\boldsymbol{\beta}) \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{\beta} - \boldsymbol{b}\|_\infty \leq c, \end{aligned} \tag{1.4}$$

where $\boldsymbol{\beta} \in \mathcal{R}^p$ is the learning parameter and $\boldsymbol{b} = (b_1, \cdots b_p)^T \in \mathcal{R}^p, c > 0$ are given. $\boldsymbol{A} \in \mathcal{R}^{p \times p}$ is a positive semidefinite matrix. $f(\cdot) : \mathcal{R}^p \to (-\infty, \infty]$ is a convex, possibly non-smooth and piecewise separable function, i.e., $f(\boldsymbol{\beta}) = \sum_{i=1}^{K} f_i(\boldsymbol{\beta}_i)$ for some $1 < K < p$. The optimization problem in (1.4) is applicable to modeling a variety of statistical and machine learning problems. Typical examples include

Dantzig selector (DS) and the linear programming discriminant (LPD) rule, where $f(\cdot)$ is the ℓ_1-norm.

The rest of the dissertation is organized as follows. In Chapter 3, we provide algorithmic solutions through feature-split for solving sparse quantile regression and support vector machine for ultra-high dimensional data. In Section 3.1, we introduce a 3-block semi-proximal ADMM algorithm to solve sparse quantile regression regularized by weighted ℓ_1 and folded concave penalties. We provide linear rate of convergence analysis for the proposed algorithm. In Section 3.2, we apply the proposed algorithm to solve sparse linear support vector machine. In Section 3.3, we demonstrate their competitive performances through simulation studies and real data examples. We present the derivations of algorithms and proofs of theorems in Appendix 3.4.

In Chapter 4, we study the strong oracle property of the nonconvex Dantzig selector and Linear Programming Discriminant (LPD) rule obtained via one-step LLA algorithm and then apply the algorithmic framework proposed in Chapter 3 to efficiently compute the solutions. In Section 4.1, we verify the strong oracle property of nonconvex Dantzig selector obtained from one-step LLA. In Section 4.2, we introduce the computational framework that nests a 3-block ADMM into one-step LLA to find the nonconvex Dantzig selector. In Section 4.3, we apply the proposed algorithm to the LPD rule for high dimensional linear discriminant analysis. In Section 4.4, we compare the proposed approach with package *lpSolve* which implements linear programming using the full set of features. The numerical experiment results suggest that the proposed feature-split approach performs comparably well to *lpSolve* on moderate dimensional datasets. While *lpSolve* may fail for datasets that have tens of thousands of features due to intensive memory

usage, the proposed method maintains good performance for such dimensions.

In Chapter 5, we conclude the dissertation work.

Chapter 2

Literature Review

2.1 ADMM and Distributed Learning

The explosion of modern data-intensive applications promotes the study on distributed learning of convex problems. The alternating direction method of multiplier (ADMM) was developed in the 1970s and received enormous attention recently due to its application to massive optimization problems, including distributed learning. In this chapter, we first review the general procedure of ADMM algorithm, along with some of its variants, and then show their applications to distributed statistical learning.

2.1.1 The ADMM Algorithm

The ADMM was studied as early as in [15] and [16]. [17] gives a systematic review of ADMM. Recently, it has drawn remarkable attention thanks to its simplicity in implementation and scalability to large-scale data. ADMM has a wide range of applications in many statistical and machine learning problems. Some recent developments and applications in high dimensional statistics include

[18, 19, 20, 21, 22, 23, 24].

The classic ADMM solves the problem of the following form,

$$\begin{aligned} &\min_{\boldsymbol{v}_1,\boldsymbol{v}_2} f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) \\ &\text{s.t. } \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 = \boldsymbol{c}, \end{aligned} \tag{2.1}$$

with variables $\boldsymbol{v}_1 \in \mathcal{R}^{p_1}$ and $\boldsymbol{v}_2 \in \mathcal{R}^{p_2}$, where $\boldsymbol{A}_1 \in \mathcal{R}^{n\times p_1}$, $\boldsymbol{A}_2 \in \mathcal{R}^{n\times p_2}$ and $\boldsymbol{c} \in \mathcal{R}^n$ are given. A common assumption is that both f_1 and f_2 are proper closed convex functions. A function $f : R^n \to R$ is proper if $\mathrm{dom}(f)$ is not an empty set, and $f(\boldsymbol{v}) < +\infty$ for at least one $\boldsymbol{v} \in R^n$ and $f(\boldsymbol{v}) > -\infty$ for every $\boldsymbol{v} \in R^n$; a function $f : R^n \to R$ is said to be closed if $\forall t \in R$, the set $\{\boldsymbol{v} \in \mathrm{dom} f | f(\boldsymbol{v}) \leq t\}$ is a closed set.

ADMM works by iteratively minimizing an augmented Lagrangian function defined as

$$\mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{\gamma}) = f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) + \langle \boldsymbol{\gamma}, \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\rangle + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\|_2^2, \tag{2.2}$$

where $\boldsymbol{\gamma} \in \mathcal{R}^n$ is the dual variable and $\phi > 0$ is the parameter for the quadratic penalty of the constraints. Given an initial point $(\boldsymbol{v}_1^0, \boldsymbol{v}_2^0, \boldsymbol{\gamma}^0)$, the classic iterative update for (2.1) is given by

$$\begin{aligned} \boldsymbol{v}_1^{k+1} &= \operatorname*{argmin}_{\boldsymbol{v}_1} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2^k; \boldsymbol{\gamma}^k) \\ \boldsymbol{v}_2^{k+1} &= \operatorname*{argmin}_{\boldsymbol{v}_2} \mathcal{L}_\phi(\boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2; \boldsymbol{\gamma}^k) \\ \boldsymbol{\gamma}^{k+1} &= \boldsymbol{\gamma}^k + \phi(\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}). \end{aligned} \tag{2.3}$$

In this scheme, instead of the a joint update of variables $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, the primal

variables $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are updated alternatively via a single Gauss-Seidel pass, which accounts for the name *alternating direction*. The dual variable $\boldsymbol{\gamma}$ is updated using gradient ascent with ϕ being the step size. To see this, recall that given Lagrangian function (2.2), the dual function is

$$g_\phi(\boldsymbol{\gamma}) = \inf_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}). \tag{2.4}$$

Then the dual problem is defined as

$$\max_{\boldsymbol{\gamma}} g_\phi(\boldsymbol{\gamma}) = \max_{\boldsymbol{\gamma}} \inf_{\boldsymbol{v}_1, \boldsymbol{v}_2} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{\gamma}). \tag{2.5}$$

Under the assumption of strong duality, the optimal values of the primal and dual problems will be identical. If there is only one minimizer of $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}^*)$, one can obtain the optimal $\boldsymbol{\beta}^*$ from an optimal $\boldsymbol{\gamma}^*$,

$$\boldsymbol{\beta}^* = \operatorname*{argmin}_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}^*).$$

Assuming that dual function is differentiable, given $\boldsymbol{v}_1^{k+1}$ and $\boldsymbol{v}_2^{k+1}$, the gradient of dual function $\nabla g_\phi(\boldsymbol{\gamma}^k) = \boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}$. Therefore we can view procedure (2.3) as a dual ascent algorithm, which first computes $\boldsymbol{v}_1^k$ and $\boldsymbol{v}_2^k$ using augmented Lagrangian, and then updates $\boldsymbol{\gamma}^k$ through gradient descent.

Compared with the original unaugmented Lagrangian function $\mathcal{L}_0(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{\gamma}) = f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) + \langle \boldsymbol{\gamma}, \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c} \rangle$, the augmented Lagrangian adds an extra penalty term $\frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\|_2^2$ and is equivalent to the unaugmented

Lagrangian for the following problem,

$$\begin{aligned} &\min_{\boldsymbol{v}_1,\boldsymbol{v}_2} f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\|_2^2 \\ &\text{s.t. } \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 = \boldsymbol{c}. \end{aligned} \tag{2.6}$$

The advantage of including the extra penalty term is that dual function $g_\phi(\boldsymbol{\gamma})$ is differentiable under rather mild conditions and the algorithm converges under more general conditions including situations when f_1 or f_2 take $+\infty$ or are not strictly convex. The improved convergence performance comes at the cost of less decomposibility. For example, when f_1 is a separable function among K blocks of $\boldsymbol{v}_1 = (\boldsymbol{v}_{1,1}^T, \cdots, \boldsymbol{v}_{1,K}^T)^T$, the augmented Lagrangian $\mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{\gamma})$ is no longer separable unless $\boldsymbol{A}_1$ is block diagonal conformably with the partition. That being the case, the optimization step for $\boldsymbol{v}_1$ cannot be executed in parallel with respect to $\boldsymbol{v}_{1,j}, j = 1, \cdots, K$.

In respect to updating primal variables $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, Jacobi-type method is another type of iteration scheme. The difference is that Gauss–Seidel utilizes the newest updated values within one iteration while Jacobi updates variables jointly together and uses the previous iterates. Examples of Jacobi-type methods include the method of multiplier, which updates problem (2.1) as

$$\begin{aligned} (\boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2^{k+1}) &= \operatorname*{argmin}_{\boldsymbol{v}_1,\boldsymbol{v}_2} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{\gamma}^k) \\ \boldsymbol{\gamma}^{k+1} &= \boldsymbol{\gamma}^k + \phi(\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}). \end{aligned} \tag{2.7}$$

Gauss-Seidel and Jacobi methods have different theoretical properties. For Gauss-Seidel ADMM, switching the order of the updates of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ results in a different algorithm as the roles of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are not exactly symmetric. In fact, separating

the minimization over $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ allows for further decomposition when f_1 or g_1 are separable. On the other hand, (2.7) uses less updated information within one iteration and could be divergent even in the two-block case.

ADMM is well-suited to solving a wide range of statistical and machine learning models. For example, a collection of regularization models can be formulated as the following optimization problem,

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) + g(\boldsymbol{\beta}), \tag{2.8}$$

where $\boldsymbol{\beta} \in \mathcal{R}^p$ is the learning parameter, $L(\cdot)$ is a convex loss function, not necessarily smooth and $g(\cdot)$ is a regularization function. We can reform (2.8) into a two-block optimization problem that ADMM can handle,

$$\begin{aligned} &\min L(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) + g(\boldsymbol{z}) \\ &\text{s.t. } \boldsymbol{\beta} - \boldsymbol{z} = 0. \end{aligned} \tag{2.9}$$

Then ADMM can be directly applied to solve (2.9). For example, in Lasso linear regression, $L(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) = \frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ and $g(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$, and by applying (2.3) we have the following update scheme,

$$\begin{aligned} \boldsymbol{\beta}^{k+1} &= (\boldsymbol{X}^T\boldsymbol{X} + \phi\boldsymbol{I})^{-1}(\boldsymbol{X}^T\boldsymbol{y} + \phi(\boldsymbol{z}^k - \tilde{\boldsymbol{\gamma}}^k)) \\ \boldsymbol{z}^{k+1} &= S_{\lambda/\phi}(\boldsymbol{\beta}^{k+1} + \tilde{\boldsymbol{\gamma}}^k) \\ \tilde{\boldsymbol{\gamma}}^{k+1} &= \tilde{\boldsymbol{\gamma}}^k + \boldsymbol{\beta}^{k+1} - \boldsymbol{z}^{k+1}, \end{aligned} \tag{2.10}$$

where $\tilde{\boldsymbol{\gamma}} = \boldsymbol{\gamma}/\phi$ is the scaled dual variable. $S_{\lambda/\phi}(\boldsymbol{\beta}^{k+1} + \tilde{\boldsymbol{\gamma}}^k) = (\boldsymbol{\beta}^{k+1} + \tilde{\boldsymbol{\gamma}}^k - \lambda/\phi)_+ - (-\boldsymbol{\beta}^{k+1} - \tilde{\boldsymbol{\gamma}}^k - \lambda/\phi)_+$ is the soft thresholding rule.

2.1.2 ADMM for Distributed Model Fitting

Distributed data collection, storage and computation have become increasingly common in the big data era, fostering active studies on distributed optimization algorithms. In the distributed algorithms, the original problem is partitioned into subproblems, each containing a subset of data. At each iteration, the worker processors solve the subproblems and send the up-to-date variables to the master machine, who then summarizes and broadcasts the results to the workers. Hence, a given large-scale learning problem can be solved in a parallel way. [17] has argued that ADMM, along with its multiple variations, is well suited to distributed convex optimization. In this section, we describe how to apply ADMM to solve two canonical problems, consensus and sharing, in a distributed manner.

The consensus problem is a type of minimization problem with a single global variable $\boldsymbol{v} \in \mathcal{R}^p$ and K different convex functions $f_i : \mathcal{R}^p \to (-\infty, +\infty], i = 1, \cdots, K$,

$$\min_{\boldsymbol{v} \in \mathcal{R}^p} f(\boldsymbol{v}) = \min_{\boldsymbol{v} \in \mathcal{R}^p} \sum_{i=1}^{K} f_i(\boldsymbol{v}) \tag{2.11}$$

with $f_i, i = 1, \cdots, K$ possibly sit on different nodes in a network structure. With a huge-scale training set, it is often desirable to scale up the optimization algorithms through parallel computing. Specifically, in a network of worker machines, we are interested in solving (2.11) in such a way that at each iteration, each worker handles a smaller-scale subproblem in parallel and communicates information with others on a global machine. Since the objective function is separable, we can make a copy of data $\boldsymbol{v}$ for each node and connect them using a global variable $\boldsymbol{z}$, which

gives rise to the following constrained optimization problem,

$$\begin{aligned} &\min_{\boldsymbol{v}_i \in \mathcal{R}^p} \sum_{i=1}^{K} f_i(\boldsymbol{v}_i) \\ &\text{s.t. } \boldsymbol{v}_i - \boldsymbol{z} = 0,\ i = 1, \cdots, K. \end{aligned} \tag{2.12}$$

Common variations include adding a regularization term to the objective function,

$$\begin{aligned} &\min_{\boldsymbol{v}_i, \boldsymbol{z} \in \mathcal{R}^p} \sum_{i=1}^{K} f_i(\boldsymbol{v}_i) + g(\boldsymbol{z}) \\ &\text{s.t. } \boldsymbol{v}_i - \boldsymbol{z} = 0,\ i = 1, \cdots, K. \end{aligned} \tag{2.13}$$

The sharing problem, on the other hand, is a dual problem for the regularized consensus problem given by,

$$\min_{\boldsymbol{v}_i \in \mathcal{R}^p} \sum_{i=1}^{K} f_i(\boldsymbol{v}_i) + g(\sum_{i=1}^{K} \boldsymbol{v}_i), \tag{2.14}$$

with $\boldsymbol{v}_i \in \mathcal{R}^p,\ i = 1, \cdots, K$. f_i is a local cost function on different machines (nodes) and g is shared by all nodes in the network. (2.14) can also be put into an ADMM actionable form,

$$\begin{aligned} &\min_{\boldsymbol{v}_i, \boldsymbol{z}_i \in \mathcal{R}^p} \sum_{i=1}^{K} f_i(\boldsymbol{v}_i) + g(\sum_{i=1}^{K} \boldsymbol{z}_i) \\ &\text{s.t. } \boldsymbol{v}_i - \boldsymbol{z}_i = 0, i = 1, \cdots, K. \end{aligned} \tag{2.15}$$

Consensus and sharing problems create generic frameworks for distributed optimization and ADMM-based methods can be applied naturally to solve them using distributed optimization. We refer to [17] for a more detailed discussion on consensus and sharing problems.

In contrast to problem (2.1) that only has two-block variables $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, (2.12) and (2.15) have multi-block variables in the objective function. Therefore, ADMM algorithms for solving (2.12) and (2.15) are referred to as multi-block ADMM. A direct extension from classic two-block ADMM to multi-block setting contains two categories: the Gauss-Seidel multi-block ADMM and the Jacobi multi-block ADMM. Consider a q-block ($q \geq 3$) optimization problem given by

$$\min_{\boldsymbol{v}_1,\cdots,\boldsymbol{v}_q} \left\{ \sum_{i=1}^{q} f_i(\boldsymbol{v}_i) \mid \sum_{i=1}^{q} \boldsymbol{A}_i \boldsymbol{v}_i = \boldsymbol{c} \right\}, \tag{2.16}$$

where $q \geq 3$ and $f_i : \mathcal{R}^{p_i} \mapsto (-\infty, +\infty], i = 1, \cdots, q$ are closed proper convex functions; $\boldsymbol{A}_i \in \mathcal{R}^{n \times p_i}$ and $\boldsymbol{c} \in \mathcal{R}^n$. Let $\phi > 0$ be given penalty parameter, then the augmented Lagrange function for (2.16) is defined as

$$\mathcal{L}_\phi(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_q; \boldsymbol{\gamma}) = \sum_{i=1}^{q} f_i(\boldsymbol{v}_i) + \langle \boldsymbol{\gamma}, \sum_{i=1}^{q} \boldsymbol{A}_i \boldsymbol{v}_i - \boldsymbol{c} \rangle + \frac{\phi}{2} \| \sum_{i=1}^{q} \boldsymbol{A}_i \boldsymbol{v}_i - \boldsymbol{c} \|_2^2,$$

with $\boldsymbol{v}_i \in \mathcal{R}^{p_i}, i = 1, \cdots, q$, and $\boldsymbol{\gamma} \in \mathcal{R}^n$. The Gauss-Seidel multi-block ADMM is a natural extension of the classic 2-block Gauss-Seidel ADMM. At the iteration k, it consists of

$$\boldsymbol{v}_i^{k+1} = \underset{\boldsymbol{v}_i \in \mathcal{R}^{p_i}}{\operatorname{argmin}} \mathcal{L}_\phi(\boldsymbol{v}_1^{k+1}, \cdots, \boldsymbol{v}_{i-1}^{k+1}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1}^{k}, \cdots, \boldsymbol{v}_q^{k}; \boldsymbol{\gamma}^k), \quad \text{for } i = 1, \cdots q, \tag{2.17}$$

where the primal variables $\boldsymbol{v}_i$ are updated in a sequential fashion.

Many statistical modeling procedure can be formulated as convex optimization problems and solved efficiently by ADMM-type algorithms. Consider the following

statistical model fitting problem,

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) + g(\boldsymbol{\beta}), \tag{2.18}$$

where $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)^T \in \mathcal{R}^{n \times p}$ is the feature matrix, $\boldsymbol{\beta}$ is the learning parameter, and $L(\cdot)$ is any convex loss function coupled with a regularization term $g(\cdot)$. We assume that $L(\cdot)$ and $g(\cdot)$ are separable with respect to different blocks of data or variables. Two basic approaches to enable parallel computation are splitting across samples and splitting across features.

2.1.2.1 Splitting across Samples

This part describes the multi-block ADMM for solving problem (2.18) in circumstances where there are a modest number of features but massive training samples. To develop an algorithm that is effectively structured for model parallelization, we partition the feature matrix $\boldsymbol{X}$ and response vector $\boldsymbol{y}$ by rows (samples),

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_K \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_K \end{bmatrix}, \tag{2.19}$$

with $\boldsymbol{x}_i \in R^{n_i \times p}$ and $\boldsymbol{y}_i \in R^{n_i}$, where $\sum_{i=1}^{K} n_i = n$. Each $\boldsymbol{v}_i$ and $\boldsymbol{y}_i$ will be assigned to the i-th processor. Under the assumption that L is additive, i.e., $L(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{K} l_i(\boldsymbol{x}_i\boldsymbol{\beta} - \boldsymbol{y}_i)$, where l_i is the loss function for the i-th block

of data, problem (2.18) can be formulated into a consensus problem,

$$\begin{aligned} &\min_{\boldsymbol{\beta}_{(i)},\boldsymbol{z}} \sum_{i=1}^{K} l_i(\boldsymbol{x}_i\boldsymbol{\beta}_{(i)} - \boldsymbol{y}_i) + g(\boldsymbol{z}) \\ &\text{s.t.} \;\; \boldsymbol{\beta}_{(i)} - \boldsymbol{z} = 0, \quad \text{for } i = 1, \cdots, K, \end{aligned} \tag{2.20}$$

where $\boldsymbol{\beta}_{(i)} \in \mathcal{R}^p$ is the estimator obtained from the i-th processor. The directly extended multi-block ADMM algorithm for (2.20) is given by

$$\begin{aligned} \boldsymbol{\beta}_{(i)}^{k+1} &:= \arg\min_{\boldsymbol{\beta}_{(i)}} \left(l_i(\boldsymbol{x}_i\boldsymbol{\beta}_{(i)} - \boldsymbol{y}_i) + \frac{\phi}{2}\|\boldsymbol{\beta}_{(i)} - \boldsymbol{z}^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2 \right), \; i = 1, \cdots, K \\ \boldsymbol{z}^{k+1} &:= \arg\min_{\boldsymbol{z}} \left(g(\boldsymbol{z}) + \frac{K\phi}{2}\|\boldsymbol{z} - \bar{\boldsymbol{\beta}}^{k+1} - \frac{\bar{\boldsymbol{\gamma}}^k}{\phi}\|_2^2 \right) \\ \boldsymbol{\gamma}_i^{k+1} &:= \boldsymbol{\gamma}_i^k + \phi(\boldsymbol{\beta}_{(i)}^{k+1} - \boldsymbol{z}^{k+1}), \end{aligned} \tag{2.21}$$

where $\bar{\boldsymbol{\gamma}} = \frac{1}{K}\sum_{i=1}^{K}\boldsymbol{\gamma}_i$, $\bar{\boldsymbol{\beta}} = \frac{1}{K}\sum_{i=1}^{K}\boldsymbol{\beta}_{(i)}$. The $\boldsymbol{\beta}_{(i)}, i = 1, \cdots, K$, update is an ℓ_2-regularized model fitting problem and can be carried out independently from different processors. The $\boldsymbol{z}$-update is the shrinkage process, which requires knowledge of $\bar{\boldsymbol{\beta}}$. Therefore, the master processor will gather variables information from the K worker processors in this step. When g is assumed to be fully separable, the minimization in the $\boldsymbol{z}$ update often admits a closed-form solution. This is a very intuitive algorithm in the sense that the dual variables $\boldsymbol{\gamma}_i$ are updated separately to drive the variables into consensus, and the quadratic regularization terms in the updates of $\boldsymbol{\beta}$ and $\boldsymbol{z}$ help pull the variables towards their average value.

2.1.2.2 Splitting across Features

In many applications, collected datasets have a modest size of samples and a huge number of features. If we partition the parameter vector $\boldsymbol{\beta}$ as $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_K$

with $\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}$ and $\sum_{i=1}^{K} p_i = p$, and conformably partition the data matrix $\boldsymbol{X}$ as $\boldsymbol{X} = [\boldsymbol{X}_1 \cdots \boldsymbol{X}_K]$ with $\boldsymbol{X}_i \in \mathcal{R}^{n \times p_i}$, $g(\boldsymbol{\beta}) = \sum_{i=1}^{K} g_i(\boldsymbol{\beta}_i)$, then it follows that $\boldsymbol{X}\boldsymbol{\beta} = \sum_{i=1}^{K} \boldsymbol{X}_i\boldsymbol{\beta}_i$, and the model fitting problem (2.18) becomes a sharing problem,

$$\min L(\sum_{i=1}^{K} \boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{y}) + \sum_{i=1}^{K} g_i(\boldsymbol{\beta}_i), \tag{2.22}$$

or equivalently written as

$$\begin{aligned} &\min L(\sum_{i=1}^{K} \boldsymbol{z}_i - \boldsymbol{y}) + \sum_{i=1}^{K} g_i(\boldsymbol{\beta}_i). \\ &\text{s.t.} \quad \boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{z}_i = 0, \quad i = 1, \cdots, K, \end{aligned} \tag{2.23}$$

with new slack variables $\boldsymbol{z}_i \in R^n$ introduced. The iterative scheme of the directly extended multi-block ADMM is

$$\begin{aligned} \boldsymbol{\beta}_i^{k+1} &:= \underset{\boldsymbol{\beta}_i}{\operatorname{argmin}} \left(g_i(\boldsymbol{\beta}_i) + \frac{\phi}{2} \|\boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{z}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2 \right), \ i = 1, \cdots, K \\ \boldsymbol{z}^{k+1} &:= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(L(\sum_{i=1}^{K} \boldsymbol{z}_i - \boldsymbol{y}) + \sum_{i=1}^{K} \frac{\phi}{2} \|\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{z}_i + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2 \right) \\ \boldsymbol{\gamma}_i^{k+1} &:= \boldsymbol{\gamma}_i^k + \phi(\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{z}_i^{k+1}). \end{aligned}$$

As discussed in [17], to simplify the algorithm, consider updating the average of $\boldsymbol{z}_i$ as $\overline{\boldsymbol{z}}^{k+1} := \operatorname{argmin}_{\overline{\boldsymbol{z}}} \left(L(K\overline{\boldsymbol{z}} - \boldsymbol{y}) + (K\phi/2) \|\overline{\boldsymbol{X}\boldsymbol{\beta}}^{k+1} - \overline{\boldsymbol{z}}^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2 \right)$ with $\overline{\boldsymbol{X}\boldsymbol{\beta}}^{k+1} = 1/K \sum_{i=1}^{K} \boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1}$. Then we have ${\boldsymbol{z}_i}^{k+1} := \overline{\boldsymbol{z}}^{k+1} + \boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} + \boldsymbol{\gamma}_i^{k+1} - \overline{\boldsymbol{X}\boldsymbol{\beta}}^{k+1} - \bar{\boldsymbol{\gamma}}^k$. Plugging $\boldsymbol{z}_i$ into the $\boldsymbol{\gamma}_i$ update, it turns out that all $\boldsymbol{\gamma}_i$ have identical update and can be merged into one single $\boldsymbol{\gamma}$. The final multi-block ADMM update for problem

2.23 is given by

$$\begin{aligned}
\boldsymbol{\beta}_i^{k+1} &:= \underset{\boldsymbol{\beta}_i}{\operatorname{argmin}} \left(g_i(\boldsymbol{\beta}_i) + (\phi/2) \| \boldsymbol{X}_i \boldsymbol{\beta}_i - \boldsymbol{X}_i \boldsymbol{\beta}_i^k - \overline{\boldsymbol{z}}^k + \overline{\boldsymbol{X}\boldsymbol{\beta}}^k + \frac{\boldsymbol{\gamma}_i^k}{\phi} \|_2^2 \right) \\
\overline{\boldsymbol{z}}^{k+1} &:= \underset{\overline{\boldsymbol{z}}}{\operatorname{argmin}} \left(L(K\overline{\boldsymbol{z}} - \boldsymbol{y}) + (K\phi/2) \| \overline{\boldsymbol{X}\boldsymbol{\beta}}^{k+1} - \overline{\boldsymbol{z}}^k + \frac{\boldsymbol{\gamma}_i^k}{\phi} \|_2^2 \right) \\
\boldsymbol{\gamma}^{k+1} &:= \boldsymbol{\gamma}^k + \phi(\overline{\boldsymbol{X}\boldsymbol{\beta}}^{k+1} - \overline{\boldsymbol{z}}^{k+1}).
\end{aligned} \tag{2.24}$$

2.2 Convergence of the Multi-block ADMM

The convergence of the classic 2-block ADMM is well studied in the literature. A recent work in [25] provides a linear rate of convergence analysis under a variety of scenarios where at least one of the objective functions are strongly convex and has Lipschitz continuous gradient. Their results are applicable to various generalizations of ADMM. Detailed discussions can be found in [16, 15, 26, 27, 28, 25].

The directly extended multi-block ADMM functions well in many practical cases. However, the convergence of multi-block ADMM has remained unclear for long. For example, it was shown by [29] that the direct extension of Gauss-Seidel multi-block ADMM is not necessarily convergent. To resolve this issue, a number of studies have been conducted mainly in two strands. One strand of literature focuses on establishing convergence theory under specific conditions. Examples include [30], which proves the convergence of the directly extended multi-block ADMM under restrictive assumptions such as a sufficiently small step length θ. Since practical computations prefer a larger step size for faster convergence, this result is mainly of theoretical importance. [31] proves the sublinear convergence rate of the Gauss-Seidel multi-block ADMM under the assumption that $q-1$ functions $f_2, \cdots, f_q$ are strongly convex and ϕ is restricted to a certain region. [32] further proves the global linear convergence of the Gauss-Seidel multi-block

ADMM under certain assumptions on the strong convexity of functions f_i, and the rank of $\boldsymbol{A}_i$. [29] shows that extending ADMM straightforwardly from two-block with a Gauss-seidel pass to q blocks with a Jacobian update will preserve convergence if matrices $\boldsymbol{A}_i, i = 1, \cdots, q$ are mutually near-orthogonal and have full column-rank. The other strand explores various modifications on the Gauss-Seidel and Jacobian multi-block ADMM and their convergence behaviors. We refer to [33, 34, 35, 36] for more details on this topic. Among the studies on variants of ADMM, [37] proposes a symmetric Gauss-Seidel based semi-proximal ADMM (sGS-sPADMM) for convex programming problems, which enjoys both theoretical convergence guarantee and superior numerical efficiency over the directly extended multi-block ADMM. This convergent semi-proximal ADMM has three separable blocks in the objective function, with the third part being linear and takes a particular block coordinate descent cycle with the order $1 \to 3 \to 2 \to 3$ for updating the variable blocks, so one only needs to update the third variable block twice to obtain convergence.

In this part, we briefly summarize some existing works on sufficient conditions that ensure the convergence of multi-block ADMM. Consider the following problem

$$\min_{\boldsymbol{v}_i \in \mathcal{R}^{p_i}} \left\{ \sum_{i=1}^{q} f_i(\boldsymbol{v}_i) \mid \sum_{i=1}^{q} \boldsymbol{A}_i \boldsymbol{v}_i = \boldsymbol{c} \right\}, \tag{2.25}$$

where $f_i : \mathcal{R}^{p_i} \mapsto (-\infty, +\infty]$, $\boldsymbol{A}_i \in \mathcal{R}^{n \times p_i}$ and $\boldsymbol{c} \in \mathcal{R}^n$. [38] proves the global linear convergence of Gauss-Seidel ADMM (2.17) when f_i are all strongly convex and ϕ is restricted to a specific region. [31] proves the sublinear convergence rate of the Gauss-Seidel multi-block ADMM (2.17) when $K-1$ functions $f_i : \mathcal{R}^{p_i} \mapsto (-\infty, +\infty]$, $i = 2, \cdots, q$ are strongly convex and f_1 is convex, not necessarily strongly convex. No assumption is imposed on $\boldsymbol{A}_1, \cdots \boldsymbol{A}_q$. Moreover, [31] shows

that the Gauss-Seidel ADMM (2.17) converges with rate $\mathcal{O}(1/t)$ in the ergodic sense and $o(1/t)$ in the non-ergodic sense. For ease of presentation, this study only provides results when $q = 3$, yet the generalization to $q \geq 3$ should be straightforward. [32] further establishes the global linear convergence of the Gauss-Seidel multi-block ADMM (2.17) under different circumstances:

1. $f_2, \cdots, f_q$ are strongly convex; ∇f_q is Lipschitz continuous, and $\boldsymbol{A}_q$ has full row rank.

2. $f_1, \cdots, f_q$ are strongly convex; $\nabla f_1, \cdots, \nabla f_q$ are Lipschitz continuous.

3. $f_2, \cdots, f_q$ are strongly convex; $\nabla f_1, \cdots, \nabla f_q$ are Lipschitz continuous, and $\boldsymbol{A}_1$ has full column rank.

Jacobi-type ADMM is amendable to parallelization at the cost of being more divergent. To ensure its convergence, one needs to impose additional and sometimes stringent assumptions. For example, [36] shows that Jacobian ADMM will converge globally if matrices $\boldsymbol{A}_i, i = 1, \cdots, q$ are mutually near-orthogonal and have full column-rank. For more general cases, they propose the proximal Jacobi ADMM summarized in Algorithm (2.1) as a convergent variant of Jacobian ADMM.We will briefly review proximal ADMM in Section 2.2.1. Proximal Jacobi ADMM algorithm adds a proximal term $\frac{1}{2}\|\boldsymbol{v}_i - \boldsymbol{v}_i^k\|_{P_i}^2$ for each $\boldsymbol{v}_i$-subproblem, and use an extra parameter $\theta > 0$ when update $\boldsymbol{\gamma}$. Under a proper choice of $\mathcal{P}_i$ and θ, it is proven that Algorithm 2.1 has $o(1/k)$ rate of convergence. The proximal term helps make the $\boldsymbol{v}_i$ subproblem strongly convex, and thus more stable and easy to solve.

Algorithm 2.1 Proximal Jacobi ADMM

Initialization: $\boldsymbol{v}_1^0, \cdots \boldsymbol{v}_q^0$ and $\boldsymbol{\gamma}^0$; $\phi > 0$ and $\theta > 0$ are given.

while the stopping criterion is not satisfied, **do**

Update $\boldsymbol{v}_1, \cdots \boldsymbol{v}_q$ in parallel by

$\boldsymbol{v}_i^{k+1} = \operatorname{argmin} f_i(\boldsymbol{v}_i) - \langle \boldsymbol{\gamma}^k, \boldsymbol{A}_i \boldsymbol{v}_i \rangle + \| \sum_{j \neq i} \boldsymbol{A}_j \boldsymbol{v}_j^k - \boldsymbol{c} - \frac{\boldsymbol{\gamma}^k}{\phi} \|_2^2 + \frac{1}{2} \|\boldsymbol{v}_i - \boldsymbol{v}_i^k\|_{\mathcal{P}_i}^2$

Update $\boldsymbol{\gamma}^{k+1} = \boldsymbol{\gamma}^k - \theta\phi(\sum_{i=1}^K \boldsymbol{A}_i \boldsymbol{v}_i^{k+1} - \boldsymbol{c})$

end while

2.2.1 Variants of ADMM

In this part, we review relevant analysis on the variants of ADMM. A notable work in the convergence study of multi-block ADMM is [30], where the authors investigate the convergence of a variant of ADMM with an additional parameter θ. Specifically, [30] proposes to modify the dual update in (2.17)

$$\boldsymbol{\gamma}^{k+1} = \boldsymbol{\gamma}^k + \phi(\sum_{i=1}^q \boldsymbol{A}_i \boldsymbol{v}_i^{k+1} - \boldsymbol{c})$$

by including a new relaxation parameter $\theta > 0$,

$$\boldsymbol{\gamma}^{k+1} = \boldsymbol{\gamma}^k + \theta\phi(\sum_{i=1}^q \boldsymbol{A}_i \boldsymbol{v}_i^{k+1} - \boldsymbol{c}).$$

[30] proves the linear convergence of the directly extended multi-block ADMM when θ is small enough such than an error-bound condition holds. Since practical computations prefer a larger step size for faster convergence, this result is mainly of theoretical importance. Moreover, since th choice of θ is bounded by some parameters introduced in the theoretical error bound condition, it can be challenging to choose θ in practice.

Another important variant is the proximal ADMM, which has been an active

research topic in recent years. Consider a two-block convex optimization problem,

$$\begin{aligned} &\min_{\boldsymbol{v}_1,\boldsymbol{v}_2} f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) \\ &\text{s.t. } \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 = \boldsymbol{c}, \end{aligned} \tag{2.26}$$

where $f_1 : R^{p_1} \to (-\infty, +\infty]$, $f_2 : R^{p_2} \to (-\infty, +\infty]$, $\boldsymbol{A}_1 : R^{p_1} \to R^n$, and $\boldsymbol{A}_2 : R^{p_2} \to R^n$. In each iteration, ADMM iteratively solves the subproblems through Gauss-Seidel pass. The success of ADMM algorithm heavily depends on the solvability of the subproblem, that is, either the subproblem endows closed-form solutions or can be solved numerically with high precision. In the latter case, efficient algorithm needs to be handy for finding an approximate solution and the convergence of ADMM with approximate solutions of its subproblems still hold as long as subproblems can be solved up to a high accuracy. In many situations, the subproblems do not have simple closed-form solutions, nor were there any efficient numerical algorithms, which impedes an efficient execution of ADMM. One solution is to add proximal terms to make subproblems easier to solve, and the resulting algorithm is called semi-proximal ADMM algorithm. (Algorithm 2.2). In algorithm

Algorithm 2.2 sPADMM

Initialization: $(\boldsymbol{v}_1^0, \boldsymbol{v}_2^0, \boldsymbol{z}^0) \in \mathrm{dom}(f_1) \times \mathrm{dom}(f_2) \times \mathcal{C}$,
$\mathcal{S}$ and $\mathcal{T}$ are two properly chosen self-adjoint positive semidefinite operators.
$\phi > 0$ is the penalty parameter, and $\theta \in (0, \frac{1+\sqrt{5}}{2})$ controls the step size.

while the stopping criterion is not satisfied, **do**

$\boldsymbol{v}_1^{k+1} = \mathrm{argmin}_{\boldsymbol{v}_1} f_1(\boldsymbol{v}_1) + \langle \boldsymbol{z}^k, \boldsymbol{A}_1\boldsymbol{v}_1 \rangle + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2^k - \boldsymbol{c}\|_2^2 + \frac{1}{2}\|\boldsymbol{v}_1 - \boldsymbol{v}_1^k\|_{\mathcal{S}}^2$

$\boldsymbol{v}_2^{k+1} = \mathrm{argmin}_{\boldsymbol{v}_2} f_2(\boldsymbol{v}_2) + \langle \boldsymbol{z}^k, \boldsymbol{A}_2\boldsymbol{v}_2 \rangle + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\|_2^2 + \frac{1}{2}\|\boldsymbol{v}_2 - \boldsymbol{v}_2^k\|_{\mathcal{T}}^2$

$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \theta\phi(\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c})$

end while

2.2, $\mathrm{dom}(f_1), \mathrm{dom}(f_2)$ denote the domain of functions f_1 and f_2, respectively. $\mathcal{S}$

and $\mathcal{T}$ are positive semi-definite operators and assure that generated sequences of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are bounded sequences, which are critical in many convergence analysis. It can be seen that the classic ADMM is a special case of proximal ADMM with $\mathcal{S} = \mathcal{T} = 0$. The choices of $\mathcal{S}$ and $\mathcal{T}$ are usually problem dependent. A general principal is that $\mathcal{S}$ and $\mathcal{T}$ should be as small as possible while $\boldsymbol{v}_1^{k+1}$ and $\boldsymbol{v}_2^{k+1}$are still relatively easy to compute. For example, the $\boldsymbol{v}_i$-update involves a quadratic term $\frac{\phi}{2}\boldsymbol{v}_i\boldsymbol{A}_i^T\boldsymbol{A}_i\boldsymbol{v}_i, i = 1, 2$. Assume that $\boldsymbol{A}_1^T\boldsymbol{A}_1$ is ill-conditioned, then we can let $\mathcal{S}$ takes $\phi(\boldsymbol{D}_1 - \boldsymbol{A}_1^T\boldsymbol{A}_1)$ and thus $\frac{\phi}{2}\boldsymbol{v}_1^T\boldsymbol{A}_1^T\boldsymbol{A}_1\boldsymbol{v}_1$ will be cancelled out and replaced by $\boldsymbol{v}_1^T\boldsymbol{D}_1\boldsymbol{v}_1$. $\boldsymbol{D}_1$ is often chosen to be a simple, invertible operator. The linearized ADMM is actually a special case of proximal ADMM with $\boldsymbol{D}_1 = \eta\phi\boldsymbol{I} - \phi\boldsymbol{A}_1^T\boldsymbol{A}_1$ and $\eta > \lambda_{max}(\boldsymbol{A}_1^T\boldsymbol{A}_1)$, where λ_{max} denotes the largest eigenvalue of $\boldsymbol{A}_1^T\boldsymbol{A}_1$. [39] studies the optimal step size in the linearized ADMM. They find that the requirement on η being greater than the $\lambda_{\max}(\boldsymbol{A}_1^T\boldsymbol{A}_1)$ and can be relaxed by a factor of 0.75. This inspiring result was derived when $\theta = 1$. [40] establishes a unified convergence result for the proximal ADMM by requiring $\mathcal{S}$ and $\mathcal{T}$ to be positive semidefinite only.

Next we review some definitions and concepts that are relevant to our analysis.

Definition 2.1. *(Relative Interior) Let $C \subset \mathcal{R}^n$, a point $x \in C$ is a relative interior point for C, if C contains the intersection of a small enough ball centered at x with* $\text{aff}(C)$*:*

$$\exists r > 0, B_r(x) \cap \text{aff}(C) \equiv \{y \mid y \in \text{aff}(C), |y - x| \leq r\} \subset C,$$

where $\text{aff}(C) = \left\{ \sum_{i=1}^k \alpha_i x_i \,\middle|\, k > 0,\, x_i \in C,\, \alpha_i \in \mathbb{R},\, \sum_{i=1}^k \alpha_i = 1 \right\}$*. The set consists of all relative interior points of C is called the relative interior of C, denoted as ri(C).*

Constraint Qualification condition:

$$\text{There exists } (\boldsymbol{v}_1^0, \boldsymbol{v}_2^0) \in \text{ri}(\text{dom}(f_1) \times \text{dom}(f_2)) \text{ and } \boldsymbol{A}_1\boldsymbol{v}_1^0 + \boldsymbol{A}_2\boldsymbol{v}_2^0 = \boldsymbol{c}, \qquad \text{(CQ)}$$

where $\text{ri}(C)$ denotes that relative interior of set $\boldsymbol{c}$.

Under CQ, the KKT optimality condition holds, and we have $(\bar{\boldsymbol{v}}_1, \bar{\boldsymbol{v}}_2)$ is an optimal solution to (2.26) if and only if there exists a $\bar{\boldsymbol{z}}$ such that

$$\boldsymbol{A}_1^T\bar{\boldsymbol{z}} \in \partial f_1(\bar{\boldsymbol{v}}_1), \quad \boldsymbol{A}_2^T\bar{\boldsymbol{z}} \in \partial f_2(\bar{\boldsymbol{v}}_2), \quad \boldsymbol{A}_1\bar{\boldsymbol{v}}_1 + \boldsymbol{A}_2\bar{\boldsymbol{v}}_2 = \boldsymbol{c}, \tag{2.27}$$

where ∂f_1 and ∂f_2 are the subdifferential mappings of f_1 and f_2, respectively. Since f_1 and f_2 are two closed proper convex functions, ∂f_1 and ∂f_2 are maximal monotone, which means that there exist two self-adjoint and positive semidefinite operators Σ_{f_1} and Σ_{f_2} such that for all $x, x' \in \mathcal{R}^{p_1}, u \in \partial f_1(x), u' \in \partial f_1(x')$,

$$\langle u - u', x - x' \rangle \geq \|x - x'\|^2_{\Sigma_{f_1}}, \tag{2.28}$$

and for all $y, y' \in \mathcal{R}^{p_2}, v \in \partial f_2(y), v' \in \partial f_2(y')$,

$$\langle v - v', y - y' \rangle \geq \|y - y'\|^2_{\Sigma_{f_2}}. \tag{2.29}$$

Definition 2.2. *(Halfspace) A half-space is the set of all points x such that $a^T x \leq b$ for some $a \in \mathcal{R}^n$ and $b \in \mathcal{R}$.*

Definition 2.3. *(Polyhedron) A polyhedron in $\mathcal{R}^n$ is defined as the intersection of finitely many half-spaces; or equivalently, it can be defined as the set $\{x | \boldsymbol{A}x \leq b\}$ for a matrix $\boldsymbol{A} \in \mathcal{R}^{m \times n}$ and a vector $b \in R^m$.*

Definition 2.4. *(Piecewise linear-quadratic) A continuous function $f : D \to \mathcal{R}$ defined on a subset of $\mathcal{R}^n$ is piecewise linear-quadratic (PLQ) if there exists M quadratic functions $\{g_i\}, i = 1, \cdots M$ and M polyhedra $\{P_i\}, i = 1, \cdots M$ with $\bigcup_{i=1}^{M} P_i = D$ such that for all $x \in P_i$, $f(x) = g_i(x)$.*

Definition 2.5. *(Piecewise Polyhedral) A set $S \in \mathcal{R}^n$ is piecewise polyhedral if it is the union of finitely many polyhedra each of which is called a (polyhedral) piece of S. Thus the domain of a PLQ function is piecewise polyhedral.*

The set of piecewise linear-quadratic functions contains a wide range of functions, for instance, quadratic forms, indicators of polyhedral sets, polyhedral norms such as ℓ_1, and several loss functions such as the Huber loss and hinge loss.

An important class of piecewise polyhedral mapping is the subdifferential of convex piecewise linear-quadratic functions as discussed in [41]. We restate their result as the following proposition.

Proposition 2.1. *Let $F : \mathcal{X} \to (-\infty, +\infty]$ be a closed proper convex function. Then F is piecewise linear quadratic if and only if the graph of ∂F is piecewise polyhedral.*

2.2.2 Convergence of proximal ADMM

Denote $\Theta(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{z}) = (\theta\phi)^{-1}\|\boldsymbol{z}-\bar{\boldsymbol{z}}\|^2+\|\boldsymbol{v}_1-\bar{\boldsymbol{v}}_1\|_{\mathcal{S}}^2+\|\boldsymbol{v}_2-\bar{\boldsymbol{v}}_2\|_{\mathcal{T}}^2+\phi\|\boldsymbol{A}_2(\boldsymbol{v}_2-\bar{\boldsymbol{v}}_2)\|^2$, [40] establishes the convergence theory for proximal ADMM in Theorem 2.1.

Theorem 2.1. *Assume that the solution set of (2.26) is nonempty, and CQ condition holds. Assume also both $\Sigma_{f_1} + \mathcal{S} + \phi\boldsymbol{A}_1^T\boldsymbol{A}_1$ and $\Sigma_{f_2} + \mathcal{T} + \phi\boldsymbol{A}_2^T\boldsymbol{A}_2$ are positive definite. Let $\{(\boldsymbol{v}_1^k, \boldsymbol{v}_2^k, \boldsymbol{z}^k)\}$ be the sequence generated by proximal ADMM,*

and denote

$$\begin{aligned}
\delta_{k+1} &= \min(\theta, 1+\theta-\theta^2)\phi\|\boldsymbol{A}_2(\boldsymbol{v}_2^{k+1}-\boldsymbol{v}_2^k)\|^2 + \|\boldsymbol{v}_2^{k+1}-\boldsymbol{v}_2^k\|_{\mathcal{T}}^2,\\
t_{k+1} &= \delta_{k+1} + \|\boldsymbol{v}_1^{k+1}-\boldsymbol{v}_1^k\|_{\mathcal{S}}^2 + 2\|\boldsymbol{v}_1^{k+1}-\bar{\boldsymbol{v}}_1\|_{\Sigma_{f_1}}^2 + 2\|\boldsymbol{v}_2^{k+1}-\bar{\boldsymbol{v}}_2\|_{\Sigma_{f_2}}^2,\\
\xi_{k+1} &= \Theta(\boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2^{k+1}, \boldsymbol{z}^{k+1}) + \|\boldsymbol{v}_2^{k+1}-\boldsymbol{v}_2^k\|_{\mathcal{T}}^2,
\end{aligned} \tag{2.30}$$

then we have the following results,

1. *when $\theta \in (0,1]$, for $k \geq 1$,*

$$\begin{aligned}
&\xi_{k+1} + (1-\theta)\phi\|\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}\|^2\\
&- [\xi_k + (1-\theta)\phi\|\boldsymbol{A}_1\boldsymbol{v}_1^k + \boldsymbol{A}_2\boldsymbol{v}_2^k - \boldsymbol{c}\|^2]\\
&\leq -[t_{k+1} + \phi\|\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}\|^2];
\end{aligned} \tag{2.31}$$

2. *when $\theta > 1$, for $k \geq 1$,*

$$\begin{aligned}
&\xi_{k+1} + (1-\theta^{-1})\phi\|\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}\|^2\\
&- [\xi_k + (1-\theta^{-1})\phi\|\boldsymbol{A}_1\boldsymbol{v}_1^k + \boldsymbol{A}_2\boldsymbol{v}_2^k - \boldsymbol{c}\|]\\
&\leq -[t_{k+1} + \theta^{-1}(1+\theta-\theta^2)\phi\|\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}\|^2];
\end{aligned} \tag{2.32}$$

3. *when $\theta \in (0, \frac{1+\sqrt{5}}{2})$, the sequence $\{(\boldsymbol{v}_1^k, \boldsymbol{v}_2^k)\}$ converges to an optimal solution of (2.26), and $\{\boldsymbol{z}^k\}$ converges to the optimal solution of the dual problem.*

Proximal ADMM addresses the potential issue arising from the non-solvability of subproblems in the classic ADMM. Theorem 2.1 justifies the application of proximal ADMM to a broader range of optimization problems and covers many existing convergence results.

[42] extends the semi-proximal ADMM to solve the 3-block separable convex

minimization problems

$$\min_{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{v}_3} \left\{ f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) + f_3(\boldsymbol{v}_3) \;\middle|\; \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 + \boldsymbol{A}_3\boldsymbol{v}_3 = \boldsymbol{c} \right\}, \tag{2.33}$$

where $\boldsymbol{v}_i \in \mathcal{V}_i, i = 1,2,3$ and $\mathcal{V}_i$ are real finite-dimensional Euclidean spaces. $f_i : \mathcal{V}_i \to (-\infty, +\infty]$ are closed proper convex functions and $\boldsymbol{A}_i : \mathcal{V}_i \to \mathcal{C}$ are linear operators. $\mathcal{C}$ is a real finite-dimensional Euclidean space. Then the augmented Lagrangian function is

$$\begin{aligned}\mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{z}) = f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) + f_3(\boldsymbol{v}_3) + \langle \boldsymbol{z}, \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 + \boldsymbol{A}_3\boldsymbol{v}_3 - \boldsymbol{c}\rangle \\ + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 + \boldsymbol{A}_3\boldsymbol{v}_3 - \boldsymbol{c}\|^2.\end{aligned} \tag{2.34}$$

[42] propose a specially designed 3-block sPADMM for solving (2.33) as presented in Algorithm 2.3.

Algorithm 2.3 3-block sPADMM

Initialization: $(\boldsymbol{v}_1^0, \boldsymbol{v}_2^0, \boldsymbol{v}_3^0, \boldsymbol{z}^0) \in \mathcal{V}_1 \times \mathcal{V}_2 \times \mathcal{V}_3 \times \mathcal{C}$; $\phi \in (0, +\infty), \theta \in (0, +\infty)$ are given tuning parameters. $\mathcal{T}_i, i = 1,2,3$ are given self-adjoint and positive semidefinite linear operators defined on $\mathcal{V}_i$, respectively.

while the stopping criterion is not satisfied, **do**

$$\boldsymbol{v}_1^{k+1} = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2^k, \boldsymbol{v}_3^k, \boldsymbol{z}^k) + \tfrac{1}{2}\|\boldsymbol{v}_1 - \boldsymbol{v}_1^k\|_{\mathcal{T}_1},$$

$$\boldsymbol{v}_2^{k+1} = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2, \boldsymbol{v}_3^k, \boldsymbol{z}^k) + \tfrac{1}{2}\|\boldsymbol{v}_2 - \boldsymbol{v}_2^k\|_{\mathcal{T}_2},$$

$$\boldsymbol{v}_3^{k+1} = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2^{k+1}, \boldsymbol{v}_3, \boldsymbol{z}^k) + \tfrac{1}{2}\|\boldsymbol{v}_3 - \boldsymbol{v}_3^k\|_{\mathcal{T}_3},$$

$$\boldsymbol{z}^{k+1} = z^k + \theta\phi(\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} + \boldsymbol{A}_3\boldsymbol{v}_3^{k+1} - \boldsymbol{c}).$$

end while

Under the assumption that f_2 is strongly convex, [42] establishes the global convergence of sPADMM when $\theta \in (0, (1+\sqrt{5})/2)$. Furthermore, if θ is smaller than a threshold and the $\boldsymbol{A}_1$ and $\boldsymbol{A}_3$ are injective, then all the $\mathcal{T}_i, i = 1,2,3$ can be dropped, in which case the 3-block sPADMM reduces to the directly extended

3-block ADMM.

[43] and [44] propose a symmetric Gauss-Seidel iteration based semi-proximal ADMM (sGS-sPADMM). The sGS-sPADMM is a convergent multi-block semi-proximal ADMM and is more efficient than the directly extended semi-proximal ADMM in solving large-scale convex quadratic conic programming problems. We will review this algorithm in Chapter 3 and 4.

Chapter 3

Distributed Alternating Direction Method of Multipliers for Large-scale Problems

In this chapter, we introduce an efficient and parallelizable algorithm based on multi-block ADMM for solving a class of high dimensional machine learning problems represented by sparse quantile regression and sparse support vector machine.

Quantile regression, proposed in the seminar article [45], allows a more comprehensive portrayal of the relationship between the response variable and predictors than the least squares mean regression. Given data $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^T$ and $\boldsymbol{y}$, the loss function $L(\boldsymbol{\beta})$ is defined as

$$L(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}), \tag{3.1}$$

where $\rho_\tau(z) = z[\tau - \mathcal{I}(z < 0)] = \tau \mathbf{1}^T(z)_+ + (1-\tau)\mathbf{1}^T(z)_-$ with $\mathcal{I}(\cdot)$ being the indicator function. For high-dimensional data where $p \gg n$, $\hat{\boldsymbol{\beta}}$ estimated from

minimizing (3.1) is generally inconsistent. This motivates investigations on quantile regression in the high dimensional sparse models framework (1.1). As a leading work, [46] discusses several theoretical properties of the ℓ_1-penalized quantile regression estimator ($\ell_1 - QR$). $\ell_1 - QR$ can be implemented by *rqPen* package [47] and the algorithm is similar to the Lasso code introduced in [48]. [49] proposes a computationally efficient algorithm called QICD for quantile regression with nonconvex penalties. They apply a convex majorization function on the concave penalty term, and iteratively solve the majorized objective function by coordinate descent. *rqPen* package will run QICD for penalized quantile regression when p is larger than 50. Computational aspects of QR-related models can be found in numerous work, see [50] and [51] as two examples.

Support vector machine (SVM) [52] is an important tool used in binary classification. Consider random samples of size n, $(\boldsymbol{x}_i, y_i), i = 1, \cdots, n$ with $\boldsymbol{x}_i \in \mathcal{R}^{p+1}$ and class label $y \in \{-1, 1\}$. The learning parameter is a $(p+1)$-vector $\boldsymbol{\beta}^T = (\beta_0, (\boldsymbol{\beta}^+)^T)$ and β_0 is the intercept term. The primary goal of linear SVM is to estimate a separating hyperplane $\boldsymbol{X}\boldsymbol{\beta} = 0$ for two classes. Denote the feature matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_n)^T = (\boldsymbol{X}_1, \cdots \boldsymbol{X}_p)$ and labels $\boldsymbol{y} = (y_1, y_2, \cdots y_n)$. The standard ℓ_2-norm support vector machine estimates $\boldsymbol{\beta}$ via solving the following optimization problem,

$$\min_{\boldsymbol{\beta}^+} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i \boldsymbol{x}_i^T \boldsymbol{\beta}^+ - y_i \beta_0)_+ + \lambda_n \|\boldsymbol{\beta}^+\|_2^2, \tag{3.2}$$

where the first addend is the non-smooth empirical "hinge" loss. In high dimensional problems, the performance of the standard ℓ_2-norm SVM can be adversely affected by irrelevant variables in the model. This motivates research on sparse support vector machine such as the ℓ_1-norm support vector machine discussed in

[53],

$$\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i \boldsymbol{x}_i^T \boldsymbol{\beta}^+ - y_i\beta_0)_+ + \lambda_n \|\boldsymbol{\beta}^+\|_1. \tag{3.3}$$

The ℓ_1-norm SVM can be formulated as a linear program and solved by standard linear programming solver such as *lpSolve* when data dimension is moderate.

In this chapter, we propose a 3-block ADMM algorithm to solve the sparse QR and SVM for ultra-high dimensional datasets. While the aforementioned solvers function well in moderate dimensions, they may break down when dimension is higher than tens of thousands. One way to get around this issue is through feature splitting. The basic idea of our proposed method is to split feature space into smaller subsets, carry out computation independently on subproblems, and coordinate local solutions to find the global solution. In Section 3.1, we present the distributed computational framework based on the 3-block ADMM for sparse quantile regression and derive the linear rate of convergence of the algorithm. In Section 3.2, we apply the framework to sparse support vector machine and re-visit the strong oracle property of the nonconvex penalized SVM estimator obtained by LLA. In Section 3.3, we demonstrate the numerical and statistical efficiency of the proposed framework especially in ultra-high dimensional setting through a number of simulation studies and real data analysis.

Throughout the chapter, we use the following notations. For a matrix $\boldsymbol{M} = (m_{ij})_{s\times t}$, $\|\boldsymbol{M}\|_{\max} = \max_{(i,j)} |m_{ij}|$ is the entry-wise maximum absolute value, and $\|\boldsymbol{M}\|_{\min} = \min_{(i,j)} |m_{ij}|$ is the entry-wise minimum absolute value. $\lambda_{\min}(\boldsymbol{M})$ and $\lambda_{\max}(\boldsymbol{M})$ denote the smallest and largest eigenvalues of $\boldsymbol{M}$, respectively. $\boldsymbol{X}_{\boldsymbol{A}}$ denotes the sub-matrix of $\boldsymbol{X}$ containing the columns indexed by the set $\boldsymbol{A}$. For a positive semidefinite operator $\boldsymbol{P}$, $\|\boldsymbol{X}\|_{\boldsymbol{P}}^2 = \boldsymbol{X}^T\boldsymbol{P}\boldsymbol{X}$.

3.1 Sparse Quantile Regression

We start with the weighted ℓ_1-penalized quantile regression problem in (3.4) and extend the algorithm for folded concave penalized estimation later via LLA algorithm. The estimator from the weighted ℓ_1-penalized QR is defined as

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} L(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \|\boldsymbol{\alpha} \circ \boldsymbol{\beta}\|_1, \tag{3.4}$$

where $L(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$ is the check loss and $\boldsymbol{\alpha}$ is the weights vector.

3.1.1 Review of ADMM

ADMM was proposed originally in [15, 16] and later [17] gives a systematic review on this topic. The non-smoothness of the objective function in (3.4) hinders an efficient application of gradient-based methods. ADMM is one of the decomposition-coordination procedures that naturally decouple the non-smooth parts in the computation. In this respect, we decentralize problem (3.4) into the following constrained optimization problem,

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \boldsymbol{z}} L(\boldsymbol{z}) + \lambda \|\boldsymbol{\alpha} \circ \boldsymbol{\beta}\|_1 \\ &\text{s.t.} \quad \boldsymbol{z} + \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}. \end{aligned} \tag{3.5}$$

The decentralization of variables transforms the problem into a natural candidate of two-block ADMM algorithm. One key ingredient in ADMM is the augmented Lagrangian function,

$$\mathcal{L}_\phi(\boldsymbol{z}, \boldsymbol{\beta}; \boldsymbol{\gamma}) = L(\boldsymbol{z}) + \lambda \|\boldsymbol{\alpha} \circ \boldsymbol{\beta}\|_1 + \langle \boldsymbol{\gamma}, \boldsymbol{z} + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y} \rangle + \frac{\phi}{2} \|\boldsymbol{z} + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2,$$

where $\boldsymbol{\gamma} \in \mathcal{R}^n$ is the Lagrangian multiplier and $\phi > 0$ is the parameter associated with the quadratic term. The classic iterative scheme at the iteration k for two-block ADMM is given by

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \mathcal{L}_\phi(\boldsymbol{\beta}, \boldsymbol{z}^k; \boldsymbol{\gamma}^k) \\
\boldsymbol{z}^{k+1} &= \underset{\boldsymbol{z}}{\operatorname{argmin}}\, \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}; \boldsymbol{\gamma}^k) \\
\boldsymbol{\gamma}^{k+1} &= \boldsymbol{\gamma}^k + \theta\phi(\boldsymbol{\beta}^{k+1} + \boldsymbol{z}^{k+1} - \boldsymbol{y}),
\end{aligned}$$

In this scheme, the primal variables $\boldsymbol{\beta}$ and $\boldsymbol{z}$ are updated alternatively via a single Gauss-Seidel pass. The dual variable $\boldsymbol{\gamma}$ is updated using the gradient ascent method, where θ is a tuning parameter controlling the step size. In convergence analysis, θ is often restricted to $(0, \frac{1+\sqrt{5}}{2})$. We favor a larger θ for faster convergence in practice, but not too large to retain theoretical convergence; thus a natural choice is to set $\theta = \frac{1+\sqrt{5}}{2}$. The convergence of the classic 2-block ADMM has been widely studied in literature. We refer to [16, 15, 26, 54, 27, 28, 25] for some detailed discussions. Recently, [55] has developed a package *qrADMM* that utilizes a two-block proximal ADMM to compute ℓ_1-penalized quantile regression. *qrADMM* has superior performances compared to *rqPen* package. One potential limitation is that the two-block formulation does not naturally parallelize variable updates and may suffer from memory-intensive operations such as large matrices multiplication.

3.1.2 A 3-block semi-proximal ADMM

Major computational cost of two-block ADMM for solving (3.5) comes from the $\boldsymbol{\beta}$ update, which takes up to $O(np)$ operations and could impede an efficient

execution of the algorithm in case p is very large. To overcome this issue, we propose a new 3-block semi-proximal ADMM framework that capacitates a parallel update of $\boldsymbol{\beta}$.

Suppose we can split the feature matrix and coefficients into $K > 1$ parts,

$$\boldsymbol{X} = (\boldsymbol{X}_1, \cdots, \boldsymbol{X}_K), \quad \boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_K \end{pmatrix}, \quad \boldsymbol{X}\boldsymbol{\beta} = \sum_{i=1}^{K} \boldsymbol{X}_i \boldsymbol{\beta}_i.$$

Then problem (3.5) can be rewritten as a 3-block constrained optimization problem,

$$\begin{aligned} \min_{\boldsymbol{\beta}, \boldsymbol{z}, \boldsymbol{\omega}} \quad & L(\boldsymbol{z}) + \sum_{i=1}^{K} \lambda \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1, \\ \text{s.t.} \quad & \boldsymbol{X}_1 \boldsymbol{\beta}_1 + \boldsymbol{z} + \boldsymbol{\omega}_2 + \cdots + \boldsymbol{\omega}_K = \boldsymbol{y}, \\ & \boldsymbol{\omega}_i = \boldsymbol{X}_i \boldsymbol{\beta}_i, \quad i = 2, \cdots, K. \end{aligned} \tag{3.6}$$

Intuitively, slack variables $\boldsymbol{\omega}_i, i = 2, \cdots, K$ store information of each local update $\boldsymbol{\beta}_i$. Note that although $\boldsymbol{\beta}_i$ will be updated separately, they account for a single variable block in the problem. Likewise, all $\boldsymbol{\omega}_i$ together make up the third variable block.

Remark. *There may exist multiple ways to transform a problem into a form that ADMM can handle, and different formulations of the slack variables and constraints may give rise to different algorithms. For formulation* (3.6)*, the role of* $\boldsymbol{X}_1\boldsymbol{\beta}_1$ *is not special and* $\boldsymbol{X}_i\boldsymbol{\beta}_i, i = 1, \cdots, K$ *are exchangable.*

The augmented Lagrangian function for (3.6) is given by

$$\begin{aligned}\mathcal{L}\phi(\boldsymbol{\beta}, \boldsymbol{z}, \boldsymbol{\omega}; \boldsymbol{\gamma}) = \frac{1}{n}[\tau \mathbf{1}^T(\boldsymbol{z})_+ + (1-\tau)\mathbf{1}^T(\boldsymbol{z})_-] + \lambda \sum_{i=1}^{K} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 \\ + \boldsymbol{\gamma}_1^T(\boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{z} + \boldsymbol{\omega}_2 + \cdots + \boldsymbol{\omega}_K - \boldsymbol{y}) + \sum_{i=2}^{K} \boldsymbol{\gamma}_i^T(\boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i) \\ + \frac{\phi}{2}\|\boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{z} + \boldsymbol{\omega}_2 + \cdots + \boldsymbol{\omega}_K - \boldsymbol{y}\|_2^2 + \frac{\phi}{2}\sum_{i=2}^{K} \|\boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i\|_2^2.\end{aligned} \tag{3.7}$$

As we can see from (3.7), each $\boldsymbol{\beta}_i$ is decoupled in the quadratic term, which allows a natural parallelization for $\boldsymbol{\beta}$ updates.

Two-block ADMM can be directly extended for solving (3.6). At iteration k, it alternatingly minimizes the augmented Lagrangian between $\boldsymbol{\beta}$ and $\boldsymbol{z}$ and is referred to as Gauss-Seidel multi-block ADMM,

$$\begin{cases} \boldsymbol{\beta}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}, \boldsymbol{z}^k, \boldsymbol{\omega}^k; \boldsymbol{\gamma}^k) \\ \boldsymbol{z}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}, \boldsymbol{\omega}^k; \boldsymbol{\gamma}^k) \\ \boldsymbol{\omega}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\omega}; \boldsymbol{\gamma}^k) \\ \boldsymbol{\gamma}_1^{k+1} & = \boldsymbol{\gamma}_1^k + \theta\phi(\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z}^{k+1} + \textstyle\sum_{i=2}^{K} \boldsymbol{\omega}_i^{k+1} - \boldsymbol{y}) \\ \boldsymbol{\gamma}_i^{k+1} & = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 2, \cdots, K. \end{cases} \tag{3.8}$$

Procedure (3.8) functions well in many practical cases; however, its theoretical convergence has remained unclear for long. In fact, it was shown by [29] that Gauss-Seidel multi-block ADMM is not necessarily convergent. Recently, [37] proposes a convergent symmetric Gauss-Seidel based semi-proximal ADMM for convex programming problems with three separable blocks and the third part being linear. It takes a special block coordinate descent cycle for updating each block and

update the third block variable twice. We adopt their procedure and obtain the following update scheme for (3.6),

$$\begin{cases} \boldsymbol{\beta}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}, \boldsymbol{z}^k, \boldsymbol{\omega}^k; \boldsymbol{\gamma}^k) + \frac{\phi}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^k\|^2_{\mathcal{T}_f} \\ \boldsymbol{\omega}^{k+\frac{1}{2}} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\omega}; \boldsymbol{\gamma}^k) \\ \boldsymbol{z}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}, \boldsymbol{\omega}^{k+\frac{1}{2}}; \boldsymbol{\gamma}^k) + \frac{\phi}{2}\|\boldsymbol{z} - \boldsymbol{z}^k\|^2_{\mathcal{T}_g} \\ \boldsymbol{\omega}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\omega}; \boldsymbol{\gamma}^k) \\ \boldsymbol{\gamma}_1^{k+1} & = \boldsymbol{\gamma}_1^k + \theta\phi(\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z}^{k+1} + \sum_{i=2}^K \boldsymbol{\omega}_i^{k+1} - \boldsymbol{y}) \\ \boldsymbol{\gamma}_i^{k+1} & = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 2, \cdots, K, \end{cases} \tag{3.9}$$

A major difference of (3.9) from the classic semi-proximal ADMM is that (3.9) performs an extra intermediate step to compute $\boldsymbol{\omega}^{k+\frac{1}{2}}$ before computing $\boldsymbol{z}^{k+1}$. Therefore, the practical success and efficiency of the algorithm relies heavily on the computational cost to update $\boldsymbol{\omega}$ and we will show that this extra cost is negligible in our case. $\mathcal{T}_f$ and $\mathcal{T}_g$ are two optional self-adjoint positive semidefinite operators. One desirable reason for including $\mathcal{T}_f$ and $\mathcal{T}_g$ is to ensure that $\{\boldsymbol{\beta}^{k+1}\}$ and $\{\boldsymbol{z}^{k+1}\}$ are well defined. A general principle is that $\mathcal{T}_f$ and $\mathcal{T}_g$ should be as small as possible, while the optimization problems are still easy to compute. The effect of tuning parameter θ on the algorithm convergence has been discussed in a number of works including [40] and [26], where algorithms convergences are established when θ is constrained to $(0, \frac{1+\sqrt{5}}{2})$. In our numerical experiments, we set $\theta = \frac{1+\sqrt{5}}{2}$. Next we show how to update each variable step by step.

$\boldsymbol{\beta}$ update: $\boldsymbol{\beta}_i, i = 1, \cdots, K$ are obtained via solving

$$\boldsymbol{\beta}_1^{k+1} = \underset{\boldsymbol{\beta}_1 \in \mathcal{R}^{p_1}}{\operatorname{argmin}} \lambda\|\boldsymbol{\alpha}_1 \circ \boldsymbol{\beta}_1\|_1 + \frac{\phi}{2}\|\boldsymbol{X}_1\boldsymbol{\beta}_1 + \sum_{i=1}^K \boldsymbol{\omega}_i^k + \boldsymbol{z}^k - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\|_2^2, \tag{3.10}$$

$$\boldsymbol{\beta}_i^{k+1} = \underset{\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}}{\operatorname{argmin}} \lambda \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{\phi}{2} \|\boldsymbol{X}_i \boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2, \quad i = 2, \cdots, K. \tag{3.11}$$

Thus $\boldsymbol{\beta}$ subproblems are basically a series of ℓ_1-penalized "least-square" problems. Since in high dimensional settings, each $\boldsymbol{X}_i \in \mathcal{R}^{n \times p_i}$ does not necessarily have full column rank, the generated sequences may not be well-defined. This concern can be addressed with an additional assumption called general position condition [56]. If the columns of $\boldsymbol{X}_i$ are in general positions, then for any $\lambda > 0$, the Lasso solution is well-defined. Then coordinate descent algorithm can be applied to solve (3.10) and (3.11) efficiently. We call the resulting algorithm *ADMM-CD* and summarize it in Algorithm 3.4. As will be shown in our experiments, ADMM-CD has favorable practical performances.

On the other hand, to make solutions of subproblems (3.10) and (3.11) automatically well-defined, one can add extra proximal terms to the $\boldsymbol{\beta}$-subproblems,

$$\boldsymbol{\beta}_1^{k+1} = \underset{\boldsymbol{\beta}_1 \in \mathcal{R}^{p_1}}{\operatorname{argmin}} \lambda \|\boldsymbol{\alpha}_1 \circ \boldsymbol{\beta}_1\|_1 + \frac{\phi}{2} \|\boldsymbol{X}_1 \boldsymbol{\beta}_1 + \sum_{i=1}^{K} \boldsymbol{\omega}_i^k + \boldsymbol{z}^k - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\|_2^2 + \frac{1}{2} \|\boldsymbol{\beta}_1 - \boldsymbol{\beta}_1^k\|_{\boldsymbol{\mathcal{T}_1}}^2, \tag{3.12}$$

$$\boldsymbol{\beta}_i^{k+1} = \underset{\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}}{\operatorname{argmin}} \lambda \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{\phi}{2} \|\boldsymbol{X}_i \boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2 + \frac{1}{2} \|\boldsymbol{\beta}_i - \boldsymbol{\beta}_i^k\|_{\boldsymbol{\mathcal{T}_i}}^2, \quad i = 2, \cdots, K, \tag{3.13}$$

where the proximal operators $\boldsymbol{\mathcal{T}_i} \succ \boldsymbol{0}, i = 1, \cdots, p$. The positive definiteness of $\boldsymbol{\mathcal{T}_i}$ will make $\{\boldsymbol{\beta}^k\}$ automatically well-defined. One widely adopted choice of $\boldsymbol{\mathcal{T}_i}$ is $\eta_i \boldsymbol{I}_{p_i} - \phi \boldsymbol{X}_i^T \boldsymbol{X}_i$ with $\eta_i > \phi \lambda_{\max}(\boldsymbol{X}_i^T \boldsymbol{X}_i)$. This essentially is a linearization step of the $\boldsymbol{\beta}$ update, as it uses $\eta_i \boldsymbol{I}_{p_i}$ to approximate the hessian matrix. Hence, the modified minimization problem admits a closed-form solution, which can be

carried out component-wisely,

$$\begin{aligned}\boldsymbol{\beta}_1^{k+1} &= \underset{\boldsymbol{\beta}_1\in\mathcal{R}^{p_1}}{\text{argmin}} \quad \lambda\|\boldsymbol{\alpha}_1\circ\boldsymbol{\beta}_1\|_1 \\ &\qquad + \frac{\eta_1}{2}\|\boldsymbol{\beta}_1 - \frac{\eta_1\boldsymbol{\beta}_1^k - \phi\boldsymbol{X}_1^T(\boldsymbol{X}_1\boldsymbol{\beta}_1^k + \sum_{i=2}^K \boldsymbol{\omega}_i^k + \boldsymbol{z}^k - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi})}{\eta_1}\|_2^2 \\ &= \text{Shrink} \quad \Big(\boldsymbol{\beta}_1^k - \frac{\phi}{\eta_1}\boldsymbol{X}_1^T(\boldsymbol{X}_1\boldsymbol{\beta}_1^k + \sum_{i=2}^K \boldsymbol{\omega}_i^k + \boldsymbol{z}^k - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}), \frac{\boldsymbol{\alpha}_1\lambda}{\eta_1}\Big),\end{aligned} \tag{3.14}$$

$$\begin{aligned}\boldsymbol{\beta}_i^{k+1} &= \underset{\boldsymbol{\beta}_i\in\mathcal{R}^{p_i}}{\text{argmin}} \quad \lambda\|\boldsymbol{\alpha}_i\circ\boldsymbol{\beta}_i\|_1 + \frac{\eta_i}{2}\|\boldsymbol{\beta}_i - \frac{\eta_i\boldsymbol{\beta}_i^k - \phi\boldsymbol{X}_i^T(\boldsymbol{X}_i\boldsymbol{\beta}_i^k - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi})}{\eta_i})\|_2^2 \\ &= \text{Shrink} \quad \Big(\boldsymbol{\beta}_i^k - \frac{\phi}{\eta_i}\boldsymbol{X}_i^T(\boldsymbol{X}_i\boldsymbol{\beta}_i^k - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}), \frac{\boldsymbol{\alpha}_i\lambda}{\eta_i}\Big),\end{aligned} \tag{3.15}$$

where $\text{Shrink}(x,t) = \text{sign}(x)(|x|-t)I(|x|>t)$ is the soft thresholding function. Updates in (3.14) and (3.15) manifest one advantage of splitting feature space into lower dimension. With variable split, η_i are relatively small compared to the "un-splitted" η, which should be larger than $\phi\lambda_{\max}(\boldsymbol{X}^T\boldsymbol{X})$. For high dimensional datasets, η increases significantly with p, and the step size for the update (i.e., $\frac{1}{\eta}$) could become rather small and slow down the convergence of the algorithm. In this respect, variable split improves the convergence speed. We use *ADMM-prox* to denote the algorithm with $\boldsymbol{\mathcal{T}_i} \succ 0$ and summarize it in Algorithm 3.5. Note that $\boldsymbol{\mathcal{T}_i} \succ 0$ is also required in the proof of $\{\boldsymbol{\beta}^k\}$ convergence.

$\boldsymbol{\omega}$ and $\boldsymbol{z}$ update: $\boldsymbol{\omega}$ and $\boldsymbol{z}$ are updated using the following cycle. The derivations of updates are given in the Appendix 3.4.1.

$$\boldsymbol{\omega}_i^{k+\frac{1}{2}} = \frac{1}{K}(\boldsymbol{y} - \boldsymbol{z}^k + K\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \sum_{j=1}^K \boldsymbol{X}_j\boldsymbol{\beta}_j^{k+1}), \quad i = 2,\cdots,K \tag{3.16}$$

$$\boldsymbol{z}^{k+1} = \Big(\boldsymbol{y} - \boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} - \sum_{i=2}^K \boldsymbol{\omega}_i^{k+\frac{1}{2}} - \frac{\boldsymbol{\gamma}_1^k}{\phi} - \frac{\tau}{n\phi}\Big)_+ \tag{3.17}$$

$$- \Big(- \boldsymbol{y} + \boldsymbol{X}_1 \boldsymbol{\beta}_1^{k+1} + \sum_{i=2}^{K} \boldsymbol{\omega}_i^{k+\frac{1}{2}} + \frac{\boldsymbol{\gamma}_1^k}{\phi} + \frac{\tau - 1}{n\phi} \Big)_+ \tag{3.18}$$

$$\boldsymbol{\omega}_i^{k+1} = \frac{1}{K}(\boldsymbol{y} - \boldsymbol{z}^{k+1} + K\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \sum_{j=1}^{K} \boldsymbol{X}_j \boldsymbol{\beta}_j^{k+1}), \quad i = 2, \cdots, K. \tag{3.19}$$

$\boldsymbol{\gamma}$ **update**: we update $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_i$ via gradient ascent,

$$\boldsymbol{\gamma}_1^{k+1} = \boldsymbol{\gamma}_1^k + \theta\phi(\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z}^{k+1} + \sum_{i=2}^{K} \boldsymbol{\omega}_i^{k+1} - \boldsymbol{y}) \tag{3.20}$$

$$\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 2, \cdots, K. \tag{3.21}$$

In our algorithm implementation, $\boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_K$ and updated using multiple processors/cores, while other variables are located in one processor and updated using the aggregated information from worker processors.

Algorithm 3.4 ADMM-CD for weighted ℓ_1-penalized QR

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.
  while the stopping criterion is not satisfied, **do**
    Compute $\boldsymbol{\beta}^{k+1}$ by (3.10) and (3.11) using coordinate descent algorithm.
    Compute $\boldsymbol{\omega}^{k+\frac{1}{2}}$ by (3.16), then $\boldsymbol{z}^{k+1}$ by (3.17), and $\boldsymbol{\omega}^{k+1}$ by (3.19).
    Update $\boldsymbol{\gamma}^{k+1}$ by (3.20) and (3.21).
  end while

Algorithm 3.5 ADMM-prox for weighted ℓ_1-penalized QR

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given. $\boldsymbol{\mathcal{T}_i} = \eta_i \boldsymbol{I}_{p_i} - \phi \boldsymbol{X}_i^T \boldsymbol{X}_i$ with $\eta_i > \phi\lambda_{\max}(\boldsymbol{X}_i^T\boldsymbol{X}_i)$.
  while the stopping criterion is not satisfied, **do**
    Compute $\boldsymbol{\beta}^{k+1}$ by (3.14) and (3.15).
    Compute $\boldsymbol{\omega}^{k+\frac{1}{2}}$ by (3.16), then $\boldsymbol{z}^{k+1}$ by (3.17), and $\boldsymbol{\omega}^{k+1}$ by (3.19).
    Update $\boldsymbol{\gamma}^{k+1}$ by (3.20) and (3.21).
  end while

Now we can present the linear rate of convergence for Algorithm 3.5. Although Algorithm 3.4 performs well in practice, the proximal term is necessary in estab-

lishing the theory. Proofs of theorem are given in Appendix 3.4.3.

Theorem 3.1. *For $\theta \in (0, (1+\sqrt{5})/2)$, the sequence $(\boldsymbol{\beta}^k, \boldsymbol{z}^k, \boldsymbol{\omega}^k, \boldsymbol{\gamma}^k)$ generated by Algorithm 3.5 converges to a limit point $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\gamma}})$, where $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}})$ is the primal optimal and $\bar{\boldsymbol{\gamma}}$ is the dual optimal. Furthermore, there exists a constant $\mu \in (0,1)$ such that*

$$Dist^{k+1} \leq \mu Dist^k,$$

where $Dist^k$ at the k-th iteration is defined as,

$$\begin{aligned} Dist^k = &\sum_{i=1}^{K} \left(\left\| \boldsymbol{X}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i) - \frac{1}{K}(\boldsymbol{X}\boldsymbol{\beta}^k - \boldsymbol{X}\bar{\boldsymbol{\beta}}) \right\|_2^2 + \|\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i\|_{\boldsymbol{\mathcal{T}}_i}^2 \right) \\ &+ \|\boldsymbol{z}^k - \bar{\boldsymbol{z}}\|_2^2 + \frac{K-1}{K}\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|_2^2 \\ &+ \frac{m_1}{K} \left\| \sum_{i=1}^{K} \boldsymbol{X}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i) + (\boldsymbol{z}^k - \bar{\boldsymbol{z}}) \right\|_2^2, \end{aligned}$$

where $m_1 = 1 + d_1 - d_1\theta - (1-d_1)\min\{\theta, \frac{1}{\theta}\}$ and $d_1 \in (0, \frac{1}{2})$.

Remark. *The linear rate of convergence $O(\mu^k)$ only needs $O(\log(1/\epsilon))$ iterations to achieve ϵ accuracy. The effect of K on the convergence is two-fold. On the one hand, increasing K will reduce the dimension of subproblems and the value of η, thus accelerate the computation of each sub-problem; on the other hand, increasing K will lead to an increased number of sub-problems and may raise the value of μ, which slow down the convergence both practically and theoretically. In practice, without additional hardware constraints, we find that choosing K from 5 to 10 works well for dimensions ranging from thousands to tens of thousands.*

3.1.3 Extension to Nonconvex Penalties

In this part, we combine one-step LLA with the proposed algorithms to solve sparse quantile regression with a class of folded concave penalties. [50] has proven that LLA initialized by Lasso estimator $\hat{\boldsymbol{\beta}}^{Lasso}$ can find the oracle estimator after one step with overwhelming probability for sparse quantile regression. The effectiveness of one-step LLA in finding oracle estimator alleviates the concern about the multiple local minimizers associated with nonconvex penalties. With the aid of one-step LLA, the nonconvex optimization problem reduces to a convex optimization problem. To initialize LLA, we compute the ℓ_1-penalized quantile regression estimator $\hat{\boldsymbol{\beta}}^{Lasso}$ by Algorithm 3.4 or 3.5 with $\boldsymbol{\alpha} = \mathbf{1}_p$,

$$\hat{\boldsymbol{\beta}}^{Lasso} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\big[\tau \mathbf{1}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})_+ + (1-\tau)\mathbf{1}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})_-\big] + \lambda_{Lasso}\|\boldsymbol{\beta}\|_1. \quad (3.22)$$

Next we apply Algorithm 3.4 or 3.5 with weights $\alpha_j = \frac{1}{\lambda}P_\lambda'(|\hat{\beta}_j^{(0)}|)$ for $j = 1, \cdots, p$ to get the final estimator $\hat{\boldsymbol{\beta}}^{(1)}$.

$$\hat{\boldsymbol{\beta}}^{(1)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\big[\tau \mathbf{1}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})_+ + (1-\tau)\mathbf{1}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})_-\big] + \sum_{j=1}^{p} P_\lambda'(|\hat{\beta}_j^{Lasso}|)|\beta_j|. \quad (3.23)$$

As we can see, the algorithm is a double-loop algorithm. The outer loop performs one-step LLA and the inner loop implements the proposed ADMM algorithm for weighted ℓ_1-penalized QR. For the inner loop, we have shown that the generated sequence will converge to the optimal solution. For the outer loop, Corollary 8 in [50] theoretically justifies that the algorithm will find the oracle estimator among multiple local minimums with overwhelming probability. Note that it is critical to tune λ_{Lasso} properly such that the resulting $\hat{\boldsymbol{\beta}}^{Lasso}$ is accurate enough for the

second step to reach the oracle estimator. We follow the study in [57] and set $\lambda_{Lasso} = \upsilon\lambda$, where λ is the regularization parameter in (3.23) and $\upsilon > 0$ is tuning parameter to increase the estimation accuracy. We summarize ADMM algorithms for nonconvex penalized quantile regression in Algorithm 3.8 and 3.9, which can be found in Appendix 3.4.5.

3.2 Sparse Linear Support Vector Machines

In this part, we apply the proposed algorithm to sparse linear support vector machine. The separating hyperplane $\boldsymbol{X}\boldsymbol{\beta}^*$ is parameterized by $\boldsymbol{\beta}^* = (\beta_0^*, (\boldsymbol{\beta}^{+*})^T)^T$, which is the minimizer of the population hinge loss, i.e.,

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(1 - Y\boldsymbol{X}\boldsymbol{\beta})_+. \tag{3.24}$$

The weighted ℓ_1-penalized SVM estimation problem is given by,

$$\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i\boldsymbol{x}_i^T\boldsymbol{\beta}^+ - y_i\beta_0)_+ + \lambda\|\boldsymbol{\alpha} \circ \boldsymbol{\beta}^+\|_1. \tag{3.25}$$

3.2.1 Weighted ℓ_1-penalized SVM

We first outline the algorithm for the weighted ℓ_1-penalized SVM. Divide the learning parameter $\boldsymbol{\beta}^+$ and the matrix $\boldsymbol{y}\boldsymbol{X}^T$ into K blocks,

$$\boldsymbol{\beta}^T = (\beta_0, (\boldsymbol{\beta}^+)^T) = (\beta_0, \boldsymbol{\beta}_1^T, \cdots, \boldsymbol{\beta}_K^T), \quad \boldsymbol{A} = \begin{pmatrix} y_1\boldsymbol{x}_1^T \\ y_2\boldsymbol{x}_2^T \\ \vdots \\ y_n\boldsymbol{x}_n^T \end{pmatrix} = \Big(\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_K\Big),$$

where each $\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}$, $\boldsymbol{A}_0 = \boldsymbol{y}$ and $\boldsymbol{A}_i \in \mathcal{R}^{n\times p_i}, \sum_{i=1}^K p_i = p$. Then problem (3.25) can be reformulated as a 3-block constraint minimization problem,

$$\begin{aligned} \min_{\boldsymbol{\beta},\boldsymbol{z},\boldsymbol{\omega}} \quad & \sum_{i=1}^K \lambda\|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{1}{n}\mathbf{1}_n^T(\boldsymbol{z})_+ \\ \text{s.t.} \quad & \boldsymbol{z} + \sum_{i=1}^K \boldsymbol{\omega}_i + \boldsymbol{A}_0\beta_0 = \mathbf{1}_n, \\ & \boldsymbol{A}_i\boldsymbol{\beta}_i = \boldsymbol{\omega}_i, \quad i = 1, \cdots K. \end{aligned} \tag{3.26}$$

Through a similar discussion of the quantile regression in Section 3.1, $\boldsymbol{\beta}$ updates consist of the following problems,

$$\beta_0^{k+1} = \frac{1}{\|\boldsymbol{y}\|^2}\boldsymbol{y}^T(\mathbf{1}_n - \boldsymbol{z}^k - \sum_{i=1}^K \boldsymbol{\omega}_i^k - \frac{\boldsymbol{\gamma}_0^k}{\phi}), \tag{3.27}$$

$$\boldsymbol{\beta}_i^{k+1} = \underset{\boldsymbol{\beta}_i\in\mathcal{R}^{p_i}}{\operatorname{argmin}} \lambda\|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{\phi}{2}\|\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2, \quad i = 1, \cdots, K. \tag{3.28}$$

By introducing proximal terms $\boldsymbol{\mathcal{T}_i} = \eta_i \boldsymbol{I}_{p_i} - \phi \boldsymbol{A}_i^T\boldsymbol{A}_i$ with $\eta_i > \phi\lambda_{\max}(\boldsymbol{A}_i^T\boldsymbol{A}_i)$ to subproblem (3.28), we obtain the proximal update given by

$$\begin{aligned} \boldsymbol{\beta}_i^{k+1} &= \underset{\boldsymbol{\beta}_i\in\mathcal{R}^{p_i}}{\operatorname{argmin}} \lambda\|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{\eta_i}{2}\|\boldsymbol{\beta}_i - \frac{\eta_i\boldsymbol{\beta}_i^k - \phi\boldsymbol{A}_i^T(\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi})}{\eta_i})\|_2^2 \\ &= \text{Shrink}\Big(\boldsymbol{\beta}_i^k - \frac{\phi}{\eta_i}\boldsymbol{A}_i^T(\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}), \frac{\alpha_i\lambda}{\eta_i}\Big)_{j=1,\cdots,p_i}. \end{aligned} \tag{3.29}$$

The update for $\boldsymbol{z}$ endorses the following closed form solution,

$$\begin{aligned}\boldsymbol{z}^{k+1} &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{1}{n}\mathbf{1}_n^T(\boldsymbol{z})_+ + \boldsymbol{\gamma}_0^T\boldsymbol{z} + \frac{\phi}{2}\|\boldsymbol{y}\beta_0^{k+1} + \boldsymbol{z} + \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \mathbf{1}_n\|^2 \\ &= \Big(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \frac{\boldsymbol{\gamma}_0^k}{\phi} - \frac{1}{n\phi}\Big)_+ \\ &- \Big(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \frac{\boldsymbol{\gamma}_0^k}{\phi}\Big)_-\end{aligned} \tag{3.30}$$

We summarize the proximal algorithm in Algorithm 3.6. When $\boldsymbol{\beta}$-subproblem is solved by coordinate descent, we summarize it in Algorithm 3.7.

Algorithm 3.6 ADMM-prox for weighted ℓ_1-norm SVM

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.
  while the stopping criterion is not satisfied, **do**
    Compute $\boldsymbol{\beta}^{k+1}$ by (3.27) and (3.29).
    Compute $\boldsymbol{\omega}^{k+\frac{1}{2}}$ by
      $\boldsymbol{\omega}_i^{k+\frac{1}{2}} = \frac{1}{K+1}(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \boldsymbol{z}^k + (K+1)\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \sum_{i=1}^{K}\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1})$.
    Compute $\boldsymbol{z}^{k+1}$ by (3.30).
    Compute $\boldsymbol{\omega}^{k+1}$ by
      $\boldsymbol{\omega}_i^{k+1} = \frac{1}{K+1}(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \boldsymbol{z}^{k+1} + (K+1)\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \sum_{i=1}^{K}\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1})$.
    Update $\boldsymbol{\gamma}^{k+1}$ by
      $\boldsymbol{\gamma}_0^{k+1} = \boldsymbol{\gamma}_0^k + \theta\phi(\boldsymbol{y}\beta_0^{k+1} + \boldsymbol{z}^{k+1} + \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+1} - \mathbf{1}_n)$,
      $\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 1, \cdots, K.$
  end while

We present the result of convergence rate in Theorem 3.2 for Algorithm 3.6. The proof of Theorem 3.2 can be derived similarly as Theorem 3.1 and we omit it for brevity.

Theorem 3.2. *For $\theta \in (0, (1+\sqrt{5})/2)$, the sequence $(\boldsymbol{\beta}^k, \boldsymbol{z}^k, \boldsymbol{\omega}^k, \boldsymbol{\gamma}^k)$ generated by Algorithm 3.6 converges to a limit point $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\gamma}})$ that solves the optimization problem* (3.26) *and its dual problem. Furthermore, there exists a constant $\mu \in (0, 1)$*

Algorithm 3.7 ADMM-CD for weighted ℓ_1-norm SVM

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.

while the stopping criterion is not satisfied, **do**

Compute $\boldsymbol{\beta}^{k+1}$ by (3.27) and (3.28).

Compute $\boldsymbol{\omega}^{k+\frac{1}{2}}$ by

$\boldsymbol{\omega}_i^{k+\frac{1}{2}} = \frac{1}{K+1}(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \boldsymbol{z}^k + (K+1)\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \sum_{i=1}^K \boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1})$.

Compute $\boldsymbol{z}^{k+1}$ by (3.30).

Compute $\boldsymbol{\omega}^{k+1}$ by

$\boldsymbol{\omega}_i^{k+1} = \frac{1}{K+1}(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \boldsymbol{z}^{k+1} + (K+1)\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \sum_{i=1}^K \boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1})$.

Update $\boldsymbol{\gamma}^{k+1}$ by

$\boldsymbol{\gamma}_0^{k+1} = \boldsymbol{\gamma}_0^k + \theta\phi(\boldsymbol{y}\beta_0^{k+1} + \boldsymbol{z}^{k+1} + \sum_{i=1}^K \boldsymbol{\omega}_i^{k+1} - \mathbf{1}_n)$,

$\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 1, \cdots, K.$

end while

such that

$$Dist^{k+1} \leq \mu Dist^k,$$

where $Dist^k$ *is defined as*

$$\begin{aligned} Dist^k = &\sum_{i=0}^K \left(\left\| \boldsymbol{A}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i) - \frac{1}{K+1}\boldsymbol{A}(\boldsymbol{\beta}^k - \bar{\boldsymbol{\beta}}) \right\|_2^2 + \|\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i\|_{\boldsymbol{\mathcal{T}}_i}^2 \right) \\ &+ \sum_{i=0}^K \|\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i\|_{\boldsymbol{\mathcal{T}}_i}^2 + \|\boldsymbol{z}^k - \bar{\boldsymbol{z}}\|_2^2 + \frac{K}{K+1}\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|_2^2 \\ &+ \frac{m_1}{K+1}\|\sum_{i=0}^K \boldsymbol{A}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i) + (\boldsymbol{z}^k - \bar{\boldsymbol{z}})\|_2^2, \end{aligned}$$

with $m_1 = 1 + d_1 - d_1\theta - (1 - d_1)\min\{\theta, \frac{1}{\theta}\}$ *and* $d_1 \in (0, \frac{1}{2})$.

3.2.2 Sparse SVM with Nonconvex Penalty

In this part, we first revisit the strong oracle property of the folded concave penalized SVM estimator obtained by one-step LLA and then generalize ADMM-CD and ADMM-prox to the nonconvex penalized SVM with the help of LLA accordingly. [58] establishes the oracle property of the nonconvex penalized SVM

and shows that with probability tending to one, the LLA algorithm is able to find the oracle estimator in one step with an appropriate initial estimator. Denote the support set as $\mathcal{A} = \{j : \boldsymbol{\beta}_j^* \neq 0\}$ and the cardinality $|\mathcal{A}| = q$. Under the sparsity assumption, we have $q \ll p$. The oracle estimator is defined as $\hat{\boldsymbol{\beta}}^{oracle} = (\hat{\boldsymbol{\beta}}_{\mathcal{A}}^T, \mathbf{0}^T)^T$, where

$$\hat{\boldsymbol{\beta}}_{\mathcal{A}} = \underset{\boldsymbol{\beta}_{\mathcal{A}} \in R^q, \boldsymbol{\beta}_{\mathcal{A}^c} = \mathbf{0}}{\operatorname{argmin}} L(\boldsymbol{\beta}) = \underset{\boldsymbol{\beta}_{\mathcal{A}} \in R^q}{\operatorname{argmin}} \sum_{i=1}^{n} (1 - Y_i \boldsymbol{X}_{\mathcal{A}} \boldsymbol{\beta}_{\mathcal{A}})_+. \tag{3.31}$$

Since the objective function in (3.31) is piecewise linear, the solution may not be unique given finite n, in which case, $\hat{\boldsymbol{\beta}}^{oracle}$ can be chosen to be any minimizer. Let $u_{\mathcal{A}^c} = \max_{(i,j): j \in \mathcal{A}^c} |x_{ij}|$ and $U_{\mathcal{A}} = \max_{i \in [n]} q^{-1} \|\boldsymbol{X}_{\mathcal{A},i}\|_2^2$, where $\boldsymbol{X}_{\mathcal{A},i}$ denotes the i-th row of $\boldsymbol{X}_{\mathcal{A}}$. Let $\hat{\boldsymbol{\beta}}^{Lasso}$ be the solution of ℓ_1-norm SVM. Theorem 4 in [59] proves that $\hat{\boldsymbol{\beta}}^{Lasso}$ is consistent at the near-oracle rate $O(\sqrt{q \log p / n})$. As a complementary work to [58], we provide non-asymptotic probability bounds of the one-step LLA reaching the oracle estimator when initialized by Lasso estimator. Consider the following events E_1 and E_2 and denote P_i as the probability of E_i,

$$E_1 = \left\{ \|\nabla_{\mathcal{A}^c} L(\hat{\boldsymbol{\beta}}^{oracle})\|_{\max} > a_1 \lambda \right\}, \; E_2 = \left\{ \|\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle}\|_{\min} \leq a\lambda \right\}. \tag{3.32}$$

We assume that conditions of Theorem 4 in [59] hold in our analysis and restate them as regularity conditions (A1)-(A5) in Appendix 3.4.4. To ease the notation, we denote $r = \|\boldsymbol{\beta}_{\mathcal{A}}^*\|_{\min} - a\lambda$ and $\lambda_{\min} = \lambda_{\min}(\frac{1}{n} \boldsymbol{X}_{\mathcal{A}}^T \boldsymbol{X}_{\mathcal{A}})$. Denote the conditional distribution density of $\boldsymbol{X}\boldsymbol{\beta}^*$ given $Y = +1$ and $Y = -1$ as $f(\cdot)$ and $g(\cdot)$ respectively.

Theorem 3.3. *Suppose conditions (A1)-(A5) hold and assume that* $\|\boldsymbol{\beta}_{\mathcal{A}}^*\|_{\min} \geq (a+1)\lambda$ *and* $\|\hat{\boldsymbol{\beta}}^{Lasso} - \boldsymbol{\beta}^*\|_{\max} \leq a_0 \lambda$. *Then if* λ *satisfies* $\log(p) = o(n\lambda^2)$, $(qU_{\mathcal{A}})^{1/2} \|\boldsymbol{\beta}_{\mathcal{A}}^* - a\lambda\|_2 + 2 \leq u_1$, $u_{\mathcal{A}^c} U_{\mathcal{A}} q \sqrt{\frac{\log n}{n}} = o(\lambda)$, *the LLA algorithm initialized*

by Lasso estimator $\hat{\boldsymbol{\beta}}^{Lasso}$ *converges to* $\hat{\boldsymbol{\beta}}^{oracle}$ *after one iteration with probability* $1 - P_1 - P_2$*, where*

$$
\begin{aligned}
P_1 &\leq C_1 \exp\Big[\log(p-q) + 4q\log(n) - C_2 n a_1 \lambda\Big] \\
&\quad + 4\exp(-\frac{1}{2}\lambda_{\min}^2 \kappa^2 \log n) + (p-q)\exp(-C_3 n a_1^2 \lambda^2), \\
P_2 &\leq 4\exp\left(-\frac{n\lambda_{\min}^2 \kappa^2 r^2}{18 q U_{\mathcal{A}}}\right),
\end{aligned}
$$

with κ *defined in Condition (A5).*

The proof of theorem 3.3 is given in the Appendix 3.4.4. The algorithms for SCAD-penalized support vector machine are presented in the Appendix 3.4.6.

3.3 Numerical Experiments

In this section, we evaluate the performance of the proposed algorithms on synthetic datasets and real data. For all ADMM-based methods, we implement the warm-start technique introduced in [60, 61], which uses the solution from the previous λ to initialize computation at the current λ. The stopping criterion of ADMM-based algorithms is introduced in Appendix 3.4.2.

3.3.0.1 Synthetic Study for Quantile Regression

We compare the proposed ADMM-based algorithm with R package *rqPen* [47] and *qradmm*[55]. Since *qradmm* package is written in FORTRAN, we re-implement its core algorithm, i.e., a 2-block proximal ADMM, in R for a relatively fair comparison.

The simulation setting is similar to that of [49]. We generate $(\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_p)^T$

from the multivariate normal distribution $\boldsymbol{N}_p(\mathbf{0}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} = (\sigma_{ij})_{p\times p}$ with $\sigma_{ij} = 0.5^{|i-j|}$. Then set $\boldsymbol{X}_1 = \Phi(\boldsymbol{Z}_1)$ and $\boldsymbol{X}_j = \boldsymbol{Z}_j$ for $j = 2, \cdots, p$, where $\Phi(\cdot)$ is the cumulative distribution function of standard normal random variables. The response variable $\boldsymbol{Y}$ is generated from the following heteroscedastic regression model,

$$\boldsymbol{Y} = \boldsymbol{X}_6 + \boldsymbol{X}_{100} + \boldsymbol{X}_{500} + \boldsymbol{X}_{1000} + 0.7\boldsymbol{X}_1\boldsymbol{\varepsilon}, \tag{3.33}$$

where $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{I})$ is the random error. We consider three different quantiles $\tau = 0.3, 0.5$ and 0.7. Note that $\boldsymbol{X}_1$ does not affect the center of the conditional distribution $Y|\boldsymbol{X}$, but will affect the conditional distribution when $\tau = 0.3, 0.7$. λ is chosen by HBIC criterion proposed in [62],

$$\text{HBIC}(\lambda) = \log\big(\sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})\big) + |\mathcal{A}|\frac{\log(\log n)}{n} \times \log(p), \tag{3.34}$$

where $|\mathcal{A}|$ is the cardinality of active set. We select the λ that minimizes HBIC. We compare the performances of different methods in terms of the following criteria.

1. $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\hat{\beta}_j - \beta_j|$;
2. Size: the average number of nonzero estimated coefficients $\hat{\beta}_j \neq 0$, for $j = 1, \cdots, p$;
3. P1: the proportion of simulation runs that select all active features except for $\boldsymbol{X}_1$;
4. P2: the proportion of simulation runs that select $\boldsymbol{X}_1$.

We expect P2 to be 0 when $\tau = 0.5$, and be 1 when $\tau = 0.3$ and 0.7. The simulation results over 500 replications are summarized in Table 3.1 and 3.2. Compared to

the ℓ_1-penalized quantile regression, in general, SCAD-penalized quantile regression produce models with significantly smaller absolute error and better selection accuracy. ADMM-CD and ADMM-prox have the best performances with respect to estimation and variable selection accuracy. When $p = 1000$, three ADMM-based methods perform comparably well and outperform *rqPen* by a significant margin. *rqPen* results in relatively larger estimation errors and is more likely to miss $\boldsymbol{X}_1$ when $\tau = 0.3, 0.7$. The current version of *rqPen* throws errors when solving SCAD penalized quantile regression, as noted in the table. Nonetheless, when $p = 50000$, both *Qradmm* and *rqPen* fail due to their demanding memory usage. In fact, we notice that the efficiency of *Qradmm* would deteriorate sharply when p increases. Figure 3.1 plots the curves of $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_1$ with respect to the iteration steps averaged over 100 replications when $n = 400, p = 1000, \tau = (0.3, 0.5, 0.7)$. We can see that Algorithm 3.5 and 3.4 converge to true $\boldsymbol{\beta}$ within approximate 20 iterations.

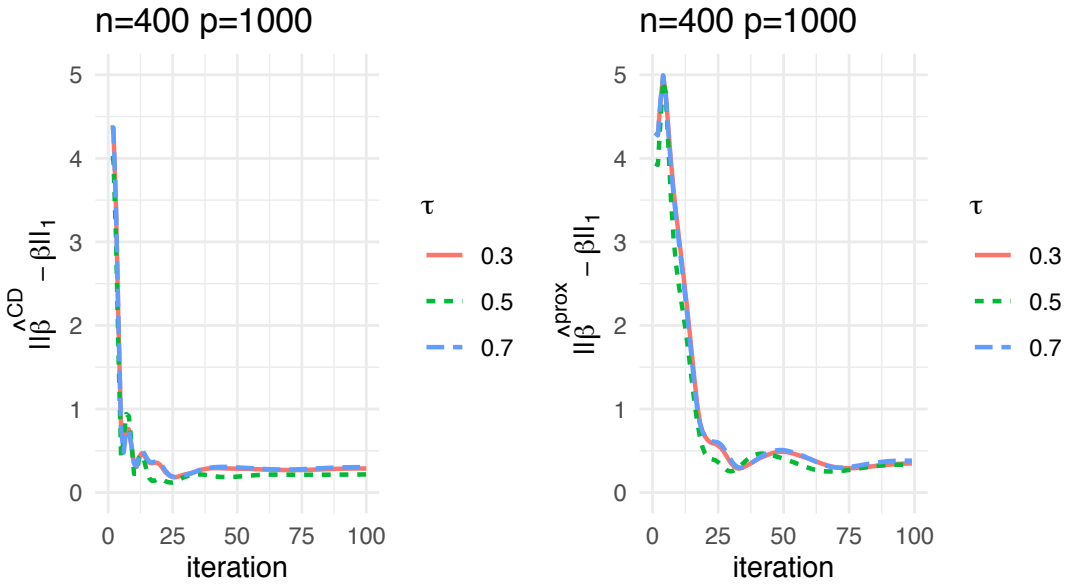

Figure 3.1. Convergence curves of $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_1$ of ADMM-CD (left panel) and ADMM-prox (right panel) over 100 replications.

Furthermore, we measure the quality of each algorithm's output by the average

objective values of ℓ_1-QR given in (3.4) over 500 replications. We fix λ at 0.05, which is the center of the candidate λ sequence. We use the same stopping condition for three ADMM-based algorithms, and use the internal default stopping criterion for *rqPen*. ϕ is set to be $1/p$ for three ADMM-based algorithms. As shown in table 3.3, three ADMM-based algorithms lead to nearly identical objective values, lower than that of *rqPen*. ADMM-CD has the highest proportion of resulting in the lowest values of the objective function over all replications, followed by ADMM-prox.

3.3.0.2 Synthetic Study for SVM

We benchmark the performances of ADMM-CD and ADMM-prox against the standard LP solver *lpSolve* on a synthetic dataset. We generate $\boldsymbol{X} \sim \boldsymbol{MN}_p(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = (\sigma_{ij})$ with $\sigma_{ij} = 0.4^{|i-j|}$ for $1 \leq i \neq j \leq p$, $P(Y = 1) = \Phi(\boldsymbol{X}\boldsymbol{\beta})$ where $\Phi(\cdot)$ is the CDF of the standard normal distribution. Let $q = 4$ and true active set $\mathcal{A} = \{50, 1000, 1500, 2000\}$. Let $\boldsymbol{\beta}_j^+ = 1.1$ for $j \in \mathcal{A}$. We set $(n, p) = (300, 3000)$ and $(300, 50000)$. Regularization parameter λ is chosen by the SVMIC$_H$ criterion proposed in [63], which is defined as

$$\text{SVMIC}_H(\lambda_n) = \sum_{i=1}^{n}(1 - y_i\hat{\beta}_0(\lambda_n) - y_i\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}_+(\lambda_n))_+ + L_n|\hat{\mathcal{A}}(\lambda_n)|\log(n), \quad (3.35)$$

where $\hat{\mathcal{A}}(\lambda_n) = \{\hat{\boldsymbol{\beta}}_j(\lambda_n) \neq 0, j = 1, \cdots p\}$. As suggested by [63], we choose $L_n = \log(\log(n))$ and select the λ that minimizes SVMIC$_H$. We also generate an independent dataset of the same sample size to evaluate the test performance. We evaluate the performances of different algorithms averaged over 500 replications by the following criteria.

Table 3.1. Numerical comparisons for sparse quantile regression when $p = 1000$ over 500 replications. ✗ indicates the algorithm runs out of memory.

$n = 400, p = 1000$	τ	$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_1$	P1	P2	Size
ℓ_1-ADMM-CD	0.3	0.295 (0.003)	100%	100%	5.56 (0.03)
	0.5	0.210 (0.003)	100%	5.4%	4.36 (0.03)
	0.7	0.281 (0.003)	100%	100%	5.56 (0.03)
ℓ_1-ADMM-prox	0.3	0.295 (0.003)	100%	100%	5.62 (0.03)
	0.5	0.198 (0.003)	100%	4.6%	4.34 (0.02)
	0.7	0.301 (0.003)	100%	100%	5.56 (0.03)
ℓ_1-qradmm	0.3	0.310 (0.003)	100%	100%	5.68 (0.03)
	0.5	0.230 (0.003)	100%	9%	5.32 (0.06)
	0.7	0.327 (0.005)	100%	100 %	6.73 (0.08)
ℓ_1-rqPen	0.3	0.598 (0.004)	100%	61.2%	5.10 (0.04)
	0.5	0.267 (0.003)	100%	0%	4.23 (0.02)
	0.7	0.601 (0.004)	100%	56.6%	5.04 (0.04)
SCAD-ADMM-CD	0.3	0.119 (0.002)	100%	100%	5.00 (0.00)
	0.5	0.035 (0.001)	100%	0.2%	4.00 (0.00)
	0.7	0.125 (0.002)	100%	100%	5.00 (0.00)
SCAD-ADMM-prox	0.3	0.115 (0.002)	100%	100%	5.00 (0.00)
	0.5	0.040 (0.001)	100%	0.2%	4.00 (0.00)
	0.7	0.123 (0.001)	100%	100%	5.00 (0.00)
SCAD-qradmm	0.3	0.122 (0.002)	100%	100%	5.00 (0.00)
	0.5	0.038 (0.001)	100%	0.4%	4.00 (0.00)
	0.7	0.129 (0.002)	100%	100%	5.00 (0.00)
SCAD-rqPen	0.3		✗		
	0.5		✗		
	0.7		✗		

- Test error: testing misclassification error rate.
- Signal: the number of selected relevant features, i.e., $\hat{\beta}_j \neq 0$ with $j \in \{50, 1000, 1500, 2000\}$.
- Noise: the number of selected irrelevant features, i.e., $\hat{\beta}_j \neq 0$ with $j \notin \{50, 1000, 1500, 2000\}$.
- AAC: the absolute value of the sample correlation between $\boldsymbol{X}\boldsymbol{\beta}^*$ and $\boldsymbol{X}\hat{\boldsymbol{\beta}}$.

Table 3.2. Numerical comparisons for sparse quantile regression when $p = 50000$ over 500 replications. ✗ indicates the algorithm runs out of memory.

$n = 400, p = 50000$	τ	$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_1$	P1	P2	Size
ℓ_1-ADMM-CD	0.3	0.320 (0.003)	100%	98.2%	5.34 (0.02)
	0.5	0.250 (0.003)	100%	2%	4.25 (0.03)
	0.7	0.349 (0.003)	100%	100%	5.15 (0.03)
ℓ_1-ADMM-prox	0.3	0.326 (0.003)	100%	92.4%	4.93 (0.01)
	0.5	0.121 (0.001)	100%	0%	4.01 (0.00)
	0.7	0.394 (0.002)	100%	95.6%	5.01 (0.11)
ℓ_1-qradmm	0.3		✗		
	0.5		✗		
	0.7		✗		
ℓ_1-rqPen	0.3		✗		
	0.5		✗		
	0.7		✗		
SCAD-ADMM-CD	0.3	0.180 (0.003)	100%	98.8%	4.99 (0.00)
	0.5	0.047 (0.001)	100%	0%	4.00 (0.00)
	0.7	0.172 (0.003)	100%	99.6%	5.00 (0.03)
SCAD-ADMM-prox	0.3	0.158 (0.002)	100%	100%	5.00 (0.00)
	0.5	0.069 (0.005)	100%	2.2%	7.31 (0.47)
	0.7	0.244 (0.007)	100%	99.2%	6.64 (0.14)
SCAD-qradmm	0.3		✗		
	0.5		✗		
	0.7		✗		
SCAD-rqPen	0.3		✗		
	0.5		✗		
	0.7		✗		

AAC is an accuracy measure used in [64]. An AAC close to 1 implies that the estimated direction matches that of Bayes rule.We denote the solution from *lpSolve* by $\hat{\boldsymbol{\beta}}^{LP}$ and we expect that the sequence generated by the ADMM-based algorithm will be close to the solution provided by linear programming. In figure 3.2, we plot the ℓ_2 difference $\|\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}^{LP}\|_2^2$ against iterations averaged over 100 replications for ℓ_1-ADMM-CD and ℓ_1-ADMM-prox, respectively, and the results align with our expectation. We summarize the evaluation performances of three methods in

Table 3.3. Upper table: objective values of ℓ_1-QR when $\lambda = 0.05$ over 500 replications. Lower table: proportions of each method producing the lowest objective values for ℓ_1-QR when $\lambda = 0.05$ over 500 replications.

	ADMM-CD	ADMM-prox	qradmm	rqPen
0.3	0.3221 (0.0003)	0.3224 (0.0002)	0.3297(0.0007)	0.3397 (0.0006)
0.5	0.3354 (0.0003)	0.3356 (0.0002)	0.3446 (0.0006)	0.3396 (0.0004)
0.7	0.3222 (0.0003)	0.3224 (0.0002)	0.3326 (0.0010)	0.3395 (0.0007)

	ADMM-CD	ADMM-prox	qradmm	rqPen
0.3	0.928	0.072	0	0
0.5	0.884	0.064	0.052	0
0.7	0.922	0.068	0.010	0

table 3.4. SCAD-penalized SVM improves ℓ_1-norm SVM on test accuracy, signal selection and AAC. When $p = 3000$, ADMM-CD, ADMM-prox and *lpSolve* have very similar performances. When p is increased to 50000, ADMM-CD and ADMM-prox result in better AAC than *lpSolve*. The computational efficiency of *lpSolve* is severely affected by the high dimensions and SCAD-*lpSolve* fails due to excessive memory usage.

3.3.0.3 Chinese Supermarket Data

We evaluate the performance of three methods on a real dataset in [65]. The dataset contains 464 day samples of the number of customers and the sale volumes of 6391 products. The main purpose of this study is to find an appropriate model for the response variable, the number of customers. The response vector and the predictors have been standardized. We randomly split observations into training and testing datasets of size 300 and 164, respectively. We fit sparse quantile regression model with ℓ_1 and SCAD penalty on the training data with $\tau = 0.3, 0.5$ and 0.7. We report the averaged predictive performance on the test data over

Table 3.4. Numerical comparisons for sparse support vector machine.

$n = 300, p = 3000$

	Test error	Signal	Noise	AAC
ℓ_1-ADMM-CD	0.16 (0.00)	3.99 (0.00)	1.38 (0.02)	0.97 (0.00)
ℓ_1-ADMM-prox	0.16 (0.00)	3.98 (0.00)	1.26 (0.02)	0.97 (0.00)
ℓ_1-*lpSolve*	0.16 (0.00)	3.99 (0.00)	0.94 (0.04)	0.97 (0.00)
SCAD-ADMM-CD	0.15 (0.00)	4.00 (0.00)	1.23 (0.02)	0.99 (0.00)
SCAD-ADMM-prox	0.16 (0.00)	3.99 (0.00)	1.53 (0.03)	0.97 (0.00)
SCAD-*lpSolve*	0.16 (0.00)	4.00 (0.00)	0.95 (0.04)	0.98 (0.00)

$n = 300, p = 50000$

	Test error	Signal	Noise	AAC
ℓ_1-ADMM-CD	0.17 (0.00)	3.95 (0.01)	1.41 (0.02)	0.96 (0.00)
ℓ_1-ADMM-prox	0.17 (0.00)	3.96 (0.01)	1.28 (0.03)	0.96 (0.00)
ℓ_1-*lpSolve*	0.17 (0.00)	3.95 (0.01)	0.90 (0.04)	0.95 (0.00)
SCAD-ADMM-CD	0.17 (0.00)	3.98 (0.00)	1.45 (0.03)	0.96 (0.00)
SCAD-ADMM-prox	0.16 (0.00)	3.95 (0.01)	1.70 (0.03)	0.96 (0.00)
SCAD-*lpSolve*		✗		

100 replications which are summarized in Table 3.5. The predictive error is measured by the check loss function, namely, $\frac{1}{164}\sum_{i=1}^{164}\rho_\tau(y_i - \hat{y}_i)$. We compare the performances for different algorithms when λ is fixed at 0.05. We observe that ADMM-based algorithms have similar performances on this dataset and outperform *rqPen*. ADMM-CD and ADMM-prox produce smaller models than *qradmm*. We can also notice that SCAD-penalized quantile regression models give rise to better check loss compared to the ℓ_1 penalty.

3.4 Appendix

In this part, we present the derivation of each variable update and the convergence theory of the proposed algorithm under a more general three-block framework. We only discuss the penalized quantile regression here, as there is no sub-

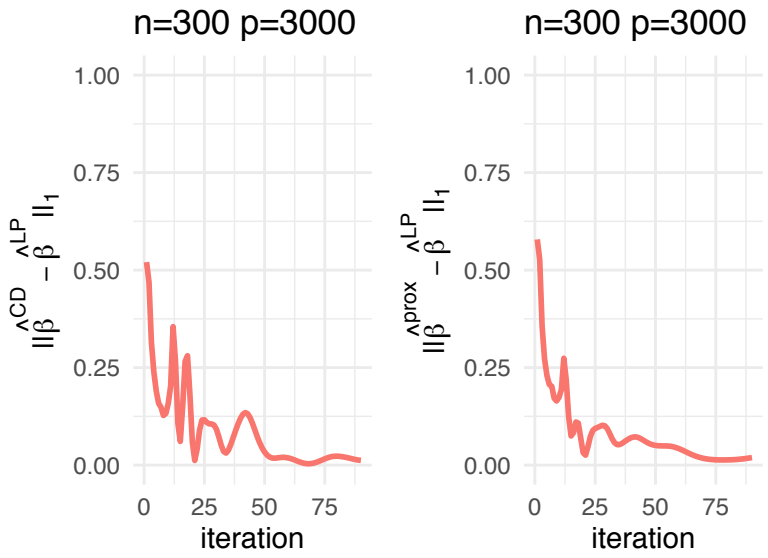

Figure 3.2. Convergence curves of $\|\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}^{LP}\|_2^2$ of ADMM-CD(left panel) and ADMM-prox (right panel) over 100 replications.

stantial difference when working with the other applications. We also present the algorithms for SCAD penalized support vector machine.

3.4.1 Sub-problems in the Algorithm

In this part, we derive the updates for $\boldsymbol{\beta}, \boldsymbol{z}$ and $\boldsymbol{\omega}$ for quantile regression. For ease of notation, define a set of functions f, g, h and linear operators F, G, H as follows,

$$f(\boldsymbol{\beta}) = n\lambda \sum_{i=1}^{K} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1, \quad h(\boldsymbol{\omega}) = \mathbf{0}, \quad g(\boldsymbol{z}) = \tau \mathbf{1}^T (\boldsymbol{z})_+ + (1-\tau)\mathbf{1}^T (\boldsymbol{z})_-, \quad (3.36)$$

Table 3.5. Performances of ADMM and *lpSolve* of sparse quantile regression on the Chinese Supermarket Data.

	τ	$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i-\hat{y}_i)$	Size
ℓ_1-ADMM-CD	0.3	0.117 (0.001)	95.95 (0.59)
	0.5	0.115 (0.001)	102.56 (0.57)
	0.7	0.129 (0.001)	101.40 (0.65)
ℓ_1-ADMM-prox	0.3	0.117 (0.001)	101.23 (1.01)
	0.5	0.110 (0.001)	120.84 (0.96)
	0.7	0.127 (0.001)	93.06 (1.17)
ℓ_1-qradmm	0.3	0.115 (0.000)	119.40 (0.53)
	0.5	0.115 (0.001)	122.40 (0.34)
	0.7	0.131 (0.000)	129.40 (0.60)
ℓ_1-rqPen	0.3	0.117 (0.001)	116.57 (0.67)
	0.5	0.114 (0.001)	119.17 (0.58)
	0.7	0.127 (0.001)	122.31 (0.68)
SCAD-ADMM-CD	0.3	0.111 (0.000)	65.04 (0.43)
	0.5	0.110 (0.001)	71.32 (0.49)
	0.7	0.117 (0.001)	70.74 (0.55)
SCAD-ADMM-prox	0.3	0.116 (0.000)	99.99 (1.01)
	0.5	0.109 (0.000)	101.90 (0.97)
	0.7	0.113 (0.000)	94.96 (0.87)
SCAD-qradmm	0.3	0.112 (0.001)	482.63 (2.34)
	0.5	0.113 (0.001)	488.86 (2.11)
	0.7	0.130 (0.001)	507.13 (2.06)
SCAD-rqPen	0.3	✗	
	0.5	✗	
	0.7	✗	

$$
\begin{aligned}
F &= Diag(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_K),\\
G &= (\boldsymbol{I}_n, 0, \cdots, 0)^T,\\
H &= \begin{pmatrix}
\boldsymbol{I}_n & \boldsymbol{I}_n & \cdots & \boldsymbol{I}_n \\
-\boldsymbol{I}_n & 0 & \cdots & 0 \\
\vdots & \ddots & & \vdots \\
0 & \cdots & -\boldsymbol{I}_n & 0 \\
0 & 0 & \cdots & -\boldsymbol{I}_n
\end{pmatrix}.
\end{aligned}
\tag{3.37}
$$

Clearly, f, g, h are closed proper convex functions. Then problem (3.6) can be expressed as a general three-block constrained optimization problem,

$$\min_{\boldsymbol{\beta},\boldsymbol{z},\boldsymbol{\omega}} \{f(\boldsymbol{\beta}) + g(\boldsymbol{z}) + h(\boldsymbol{\omega}) \mid F\boldsymbol{\beta} + G\boldsymbol{z} + H\boldsymbol{\omega} = \boldsymbol{c}\}, \tag{3.38}$$

where by the definition of F, G and H,

$$F\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{X}_1\boldsymbol{\beta}_1 \\ \boldsymbol{X}_2\boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{X}_K\boldsymbol{\beta}_K \end{pmatrix}, \quad G\boldsymbol{z} = \begin{pmatrix} \boldsymbol{z} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad H\boldsymbol{\omega} = \begin{pmatrix} \boldsymbol{\omega}_2 + \cdots + \boldsymbol{\omega}_K \\ -\boldsymbol{\omega}_2 \\ \vdots \\ -\boldsymbol{\omega}_K \end{pmatrix}. \tag{3.39}$$

Recall that algorithm (3.9) updates the 3-block variables using a special cycle,

$$\begin{cases} \boldsymbol{\beta}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}, \boldsymbol{z}^k, \boldsymbol{\omega}^k; \boldsymbol{\gamma}^k) + \frac{\phi}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^k\|^2_{\mathcal{T}_f}, \\ \boldsymbol{\omega}^{k+\frac{1}{2}} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\omega}; \boldsymbol{\gamma}^k) = (H^TH)^{-1}H(\boldsymbol{c} - F\boldsymbol{\beta}^{k+1} - G\boldsymbol{z}^k), \\ \boldsymbol{z}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}, \boldsymbol{\omega}^{k+\frac{1}{2}}; \boldsymbol{\gamma}^k) + \frac{\phi}{2}\|\boldsymbol{z} - \boldsymbol{z}^k\|^2_{\mathcal{T}_g}, \\ \boldsymbol{\omega}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\omega}; \boldsymbol{\gamma}^k) = (H^TH)^{-1}H(\boldsymbol{c} - F\boldsymbol{\beta}^{k+1} - G\boldsymbol{z}^{k+1}), \\ \boldsymbol{\gamma}^{k+1} & = \boldsymbol{\gamma}^k + \theta\phi(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} + H\boldsymbol{\omega}^{k+1} - \boldsymbol{c}). \end{cases} \tag{3.40}$$

To update $\boldsymbol{\omega}$, we need to compute $(H^TH)^{-1}$. Since

$$H^TH = \begin{pmatrix} I_p & & \\ & \ddots & \\ & & I_p \end{pmatrix} + \begin{pmatrix} I_p \\ \vdots \\ I_p \end{pmatrix} \begin{pmatrix} I_p \\ \vdots \\ I_p \end{pmatrix}^T,$$

we apply the Sherman-Morrison-Woodebury formula to compute $(H^TH)^{-1}$ and it

follows that

$$\begin{aligned}\boldsymbol{\omega}_i^{k+\frac{1}{2}} &= (H^TH)^{-1}H(\boldsymbol{c} - F\boldsymbol{\beta}^{k+1} - G\boldsymbol{z}^k) \\ &= \frac{1}{K}(\boldsymbol{y} - \boldsymbol{z}^k + K\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \sum_{j=1}^{K}\boldsymbol{X}_j\boldsymbol{\beta}_j^{k+1}), \quad i = 2, \cdots, K,\end{aligned} \tag{3.41}$$

In the $\boldsymbol{z}$-subproblem, we set $\mathcal{T}_g = 0$ and we have

$$\begin{aligned}\boldsymbol{z}^{k+1} &= \underset{\boldsymbol{z}}{\operatorname{argmin}}\, \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}, \boldsymbol{\omega}^{k+\frac{1}{2}}; \boldsymbol{\gamma}^k) \\ &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{i=1}^{n}\rho_\tau(z_i) + \boldsymbol{\gamma}_1^T(\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z} + \sum_{i=2}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \boldsymbol{y}) \right. \\ &\left. + \frac{\phi}{2}\|\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z} + \sum_{i=2}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \boldsymbol{y}\|_2^2 \right\} \\ &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\rho_\tau(z_i) + \frac{\phi}{2}\|\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z} + \sum_{i=2}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\|_2^2.\end{aligned} \tag{3.42}$$

The closed-form solution of the $\boldsymbol{z}$-subproblem can be easily derived as

$$\begin{aligned}\boldsymbol{z}^{k+1} &= \max\Big(\boldsymbol{y} - \boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} - \sum_{i=2}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \frac{\boldsymbol{\gamma}_1^k}{\phi} - \frac{\tau}{n\phi}, 0\Big) \\ &\quad - \max\Big(- \big(\boldsymbol{y} - \boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} - \sum_{i=2}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \frac{\boldsymbol{\gamma}_1^k}{\phi} - \frac{\tau}{n\phi}\big) + \frac{1}{n\phi}\big), 0\Big).\end{aligned} \tag{3.43}$$

3.4.2 Convergence Study and Stopping Criterion

We discuss the linear rate convergence of the sequence $(\boldsymbol{\beta}^k, \boldsymbol{z}^k, \boldsymbol{\omega}^k, \boldsymbol{\gamma}^k)$ generated by Algorithm 3.4. As elaborated in [37], we need the following two assumptions for obtaining theoretical guarantee on feasibility and convergence.

Assumption 3.1. *H^TH is positive definite.*

Assumption 3.2. *There exists $(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{\omega}}) \in \mathcal{R}^{p_1} \times \mathcal{R}^{p_2} \times \mathcal{R}^{p_3}$, such that $F\hat{\boldsymbol{\beta}} + G\hat{\boldsymbol{z}} +$*

$H\hat{\boldsymbol{\omega}} = \boldsymbol{c}$.

For algorithm (3.40), the projection matrix $\mathcal{P} = H(H^TH)^{-1}H^T$ plays a very important role in the convergence analysis. Let $\mathcal{Q} = \mathcal{I} - \mathcal{P}$. Since $\boldsymbol{\omega}$ can be expressed as $\boldsymbol{\omega}(\boldsymbol{\beta}, \boldsymbol{z}) = (H^TH)^{-1}H^T(\boldsymbol{c} - F\boldsymbol{\beta} - G\boldsymbol{z})$, it follows that $H\boldsymbol{\omega} = \mathcal{P}(\boldsymbol{c} - F\boldsymbol{\beta} - G\boldsymbol{z})$, and then we re-write (3.38) as two-block optimization problem,

$$\min_{\boldsymbol{\beta},\boldsymbol{z}}\{f(\boldsymbol{\beta}) + g(\boldsymbol{z}) \mid \mathcal{Q}(F\boldsymbol{\beta} + G\boldsymbol{z} - \boldsymbol{c}) = 0\}. \tag{3.44}$$

3.4.2.1 Stopping Criterion

The primal and dual residuals are often used in characterizing the convergence stage. We use the same criterion as the one introduced in [17]. Define $\boldsymbol{r}^{k+1} = \mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c}) = 1/K\mathbf{1}_K \otimes (\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{z} - \boldsymbol{c})$ as the primal residual and $\boldsymbol{s}^{k+1} = \phi F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) = \phi/K(\boldsymbol{X}_1^T, \cdots, \boldsymbol{X}_K^T)^T\text{diag}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)$ as the dual residual at iteration $k+1$. The termination criterion is

$$\|\boldsymbol{r}^k\|_2 \leq \epsilon^{\text{pri}} \quad \text{and} \quad \|\boldsymbol{s}^k\|_2 \leq \epsilon^{\text{dual}}, \tag{3.45}$$

where $\epsilon^{\text{pri}} > 0$ and $\epsilon^{\text{dual}} > 0$ are feasibility tolerances chosen as

$$\epsilon^{\text{pri}} = \sqrt{n}\epsilon^{\text{abs}} + \frac{\epsilon^{\text{rel}}}{\sqrt{K}}\max(\|\boldsymbol{X}\boldsymbol{\beta}^k\|_2, \|\boldsymbol{z}^k\|_2, \|\boldsymbol{c}\|_2),$$
$$\epsilon^{\text{dual}} = \sqrt{p}\epsilon^{\text{abs}} + \frac{\epsilon^{\text{rel}}}{K}\|F^T\mathcal{Q}\boldsymbol{\gamma}^k\|_2.$$

One common choice is $\epsilon^{\text{abs}} = 0.001$ and $\epsilon^{\text{rel}} = 0.001$.

3.4.2.2 Rate of Convergence

The augmented Lagrangian function for (3.44) is given by

$$\mathcal{L}_\phi(\boldsymbol{z},\boldsymbol{\beta};\boldsymbol{\gamma}) = f(\boldsymbol{\beta}) + g(\boldsymbol{z}) + \langle \boldsymbol{\gamma}, \mathcal{Q}(F\boldsymbol{\beta} + G\boldsymbol{z} - \boldsymbol{c})\rangle + \frac{\phi}{2}\|\mathcal{Q}(F\boldsymbol{\beta} + G\boldsymbol{z} - \boldsymbol{c})\|_2^2.$$

It has been proved in [37] that applying Algorithm 3.40 to problem (3.38) is equivalent to applying the following 2-block semi-proximal ADMM to (3.44),

$$\begin{cases} \boldsymbol{\beta}^{k+1} &= \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}, \boldsymbol{z}^k; \boldsymbol{\gamma}^k) + \frac{\phi}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F + \mathcal{T}_f}, \\ \boldsymbol{z}^{k+1} &= \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}; \boldsymbol{\gamma}^k) + \frac{\phi}{2}\|\boldsymbol{z} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G + \mathcal{T}_g}, \\ \boldsymbol{\gamma}^{k+1} &= \boldsymbol{\gamma}^k + \theta\phi\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c}). \end{cases} \tag{3.46}$$

The KKT optimality condition of (3.44) is that

$$0 \in (\mathcal{Q}F)^T\boldsymbol{\gamma} + \partial f(\boldsymbol{\beta}), \quad 0 \in (\mathcal{Q}G)^T\boldsymbol{\gamma} + \partial g(\boldsymbol{z}), \qquad \mathcal{Q}(\boldsymbol{c} - F\boldsymbol{\beta} - G\boldsymbol{z}) = 0. \tag{3.47}$$

Denote the solution set to (3.47) as $\bar{\Omega}$, then we can replace Assumption 3.2 by assuming that $\bar{\Omega}$ is non-empty. Let $\bar{\boldsymbol{u}} = (\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\gamma}})$ be an optimal solution to (3.44). We have the following lemma on the convergence of the proposed algorithm by utilizing its equivalence to the Algorithm 3.46.

Lemma 3.4. *Suppose Assumption 3.1 and 3.2 hold. $\mathcal{T}_f$ and $\mathcal{T}_g$ are chosen such that $\mathcal{T}_f + F^TF$ and $\mathcal{T}_g + G^TG$ are positive definite. Then under the condition $\theta \in (0, (1+\sqrt{5})/2)$, the sequence $(\boldsymbol{\beta}^k, \boldsymbol{z}^k, \boldsymbol{\omega}^k, \boldsymbol{\gamma}^k)$ generated by algorithm (3.40) converges to a limit point $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\gamma}})$ with $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}})$ solving (3.38) and $\bar{\boldsymbol{\gamma}}$ is the dual optimal.*

Lemma 3.4 can be easily derived from Theorem 3.2 in [66]. To further study the

rate of convergence of the proposed algorithms, we need the following assumption to bound the error.

Assumption 3.3. *Suppose $\boldsymbol{u}^k$ converges to $\bar{\boldsymbol{u}} \in \bar{\Omega}$. There exists a positive constant q such that*

$$\begin{aligned}\|\boldsymbol{u}^k - \bar{\boldsymbol{u}}\|_2^2 \leq q^2\Big(&\|\boldsymbol{\beta}^k - prox_f(\boldsymbol{\beta}^k - (\mathcal{Q}F)^T\boldsymbol{\gamma}^k)\|_2^2 + \|\boldsymbol{z}^k - prox_g(\boldsymbol{z}^k - (\mathcal{Q}G)^T\boldsymbol{\gamma}^k)\|_2^2 \\ &+ \|\mathcal{Q}(\boldsymbol{c} - F\boldsymbol{\beta}^k - G\boldsymbol{z}^k)\|_2^2\Big),\end{aligned} \tag{3.48}$$

for sufficiently large k.

For any convex function P, $\text{prox}_P(\cdot)$ denotes the proximal mapping associated with P, i.e.,

$$\text{prox}_P(x) = \underset{y}{\text{argmin}} \left\{ \frac{1}{2}\|x - y\|_2^2 + P(y) \right\}. \tag{3.49}$$

Denote

$$\mathcal{H} = C \times \text{Diag}(F^T\mathcal{P}F + \mathcal{T}_f, G^T\mathcal{P}G + \mathcal{T}_g, (\theta\phi)^{-2}I),$$

where

$$\begin{aligned}3C = \max\big\{&3\phi^2\|F^T\mathcal{P}F + \mathcal{T}_f\|_2, 3\phi\lambda_{\max}(FF^T), \\ &2\phi^2\|G^T\mathcal{P}G + \mathcal{T}_g\|_2, \\ &3(1-\theta)^2\phi\lambda_{\max}(\mathcal{Q}FF^T\mathcal{Q})) + 2(1-\theta)^2\phi\lambda_{\max}(\mathcal{Q}GG^T\mathcal{Q}) + \frac{1}{\phi}\big\},\end{aligned}$$

then we have the following relationship.

Lemma 3.5. *Suppose the sequence $\boldsymbol{u}^k = (\boldsymbol{\beta}^k, \boldsymbol{z}^k, \boldsymbol{\gamma}^k)$ is generated by algorithm (3.40) and Assumption 3.3 holds, then $\forall k \geq 0$,*

$$\|\boldsymbol{u}^{k+1} - \bar{\boldsymbol{u}}\|_2^2 \leq q^2\|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_{\mathcal{H}}^2. \tag{3.50}$$

Proof. Consider the optimality conditions of subproblems in (3.46), we have

$$\begin{aligned}
0 \in \quad & \partial f(\boldsymbol{\beta}^{k+1}) + (\mathcal{Q}F)^T\boldsymbol{\gamma}^k + \phi(\mathcal{Q}F)^T\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^k - \boldsymbol{c}) \\
& + \phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k), \\
0 \in \quad & \partial g(\boldsymbol{z}^{k+1}) + (\mathcal{Q}G)^T\boldsymbol{\gamma}^k + \phi(\mathcal{Q}G)^T\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c}) \\
& + (G^T\mathcal{P}G + \mathcal{T}_g)(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k), \\
0 = \quad & (\theta\phi)^{-1}(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) - \mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c}),
\end{aligned} \tag{3.51}$$

and then it follows that

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} \quad & = \mathrm{prox}_f\Big(\boldsymbol{\beta}^{k+1} - (\mathcal{Q}F)^T\big(\boldsymbol{\gamma}^k + \theta^{-1}(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) - \phi\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\big) \\
& \quad + \phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)\Big), \\
\boldsymbol{z}^{k+1} \quad & = \mathrm{prox}_g\Big(\boldsymbol{z}^{k+1} - (\mathcal{Q}G)^T(\boldsymbol{\gamma}^k + \theta^{-1}(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) \\
& \quad - (G^T\mathcal{P}G + \mathcal{T}_g)(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\Big), \\
\boldsymbol{\gamma}^{k+1} \quad & = \boldsymbol{\gamma}^k + \theta\phi\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c}).
\end{aligned} \tag{3.52}$$

Under Assumption 3.3, it suffices to show that

$$\begin{aligned}
& \|\boldsymbol{\beta}^{k+1} - \mathrm{prox}_f(\boldsymbol{\beta}^{k+1} - (\mathcal{Q}F)^T\boldsymbol{\gamma}^{k+1})\|_2^2 + \|\boldsymbol{z}^{k+1} - \mathrm{prox}_g(\boldsymbol{z}^{k+1} - (\mathcal{Q}G)^T\boldsymbol{\gamma}^{k+1})\|_2^2 \\
& + \|\mathcal{Q}(\boldsymbol{c} - F\boldsymbol{\beta}^{k+1} - G\boldsymbol{z}^{k+1})\|_2^2 \\
& \leq \|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_{\mathcal{H}}^2.
\end{aligned} \tag{3.53}$$

We first bound the term $\|\boldsymbol{\beta}^{k+1} - \mathrm{prox}_f(\boldsymbol{\beta}^{k+1} - (\mathcal{Q}F)^T\boldsymbol{\gamma}^{k+1})\|_2^2$. By the fact that the proximal mapping is Lipschitz continuous with constant 1, i.e., $\|\mathrm{prox}_h(x) -$

$\text{prox}_h(y)\| \leq \|x-y\|_2$ for any mapping h,

$$
\begin{aligned}
&\|\boldsymbol{\beta}^{k+1} - \text{prox}_f(\boldsymbol{\beta}^{k+1} - (\mathcal{Q}F)^T\boldsymbol{\gamma}^{k+1})\|_2^2 \\
&\leq \|\boldsymbol{\beta}^{k+1} - (\mathcal{Q}F)^T\big(\boldsymbol{\gamma}^k + \theta^{-1}(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) - \phi\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\big) \\
&\quad + \phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k) - \boldsymbol{\beta}^{k+1} + (QF)^T\boldsymbol{\gamma}^{k+1}\|_2^2 \\
&= \|\phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k) + \phi F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \\
&\quad + (1 - \frac{1}{\theta})(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\|_2^2 \\
&= \|\phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)\|_2^2 + \|\phi F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2 \\
&\quad + (1 - \frac{1}{\theta})^2\|(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\|_2^2 \\
&\quad + 2\phi^2(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)(F^T\mathcal{P}F + \mathcal{T}_f)^T F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \\
&\quad + 2(1 - \frac{1}{\theta})\phi(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)(F^T\mathcal{P}F + \mathcal{T}_f)^T(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) \\
&\quad + 2(1 - \frac{1}{\theta})\phi(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)^T G^T\mathcal{Q}F(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)
\end{aligned}
\tag{3.54}
$$

By taking into account the fact that

$$
\begin{aligned}
&2\phi^2(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)(F^T\mathcal{P}F + \mathcal{T}_f)^T F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \\
&\leq \|\phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)\|_2^2 + \|\phi F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2,
\end{aligned}
$$

$$
\begin{aligned}
&2(1 - \frac{1}{\theta})\phi(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)(F^T\mathcal{P}F + \mathcal{T}_f)^T(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) \\
&\leq \|\phi(F^T\mathcal{P}F + \mathcal{T}_f)(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k)\|_2^2 + (1 - \frac{1}{\theta})^2\|(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\|_2^2,
\end{aligned}
$$

$$
\begin{aligned}
&2(1 - \frac{1}{\theta})\phi(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)^T G^T\mathcal{Q}F(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k) \\
&\leq (1 - \frac{1}{\theta})^2\|(QF)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\|_2^2 + \|\phi F^T\mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2,
\end{aligned}
$$

and the inequality that

$$\begin{aligned}\|F^T\mathcal{Q}G(\boldsymbol{z}^{k+1}-\boldsymbol{z}^k)\|_2^2 &= (\mathcal{Q}G(\boldsymbol{z}^{k+1}-\boldsymbol{z}^k))^T FF^T(\mathcal{Q}G(\boldsymbol{z}^{k+1}-\boldsymbol{z}^k))^T \\ &\leq \lambda_{\max}(FF^T)\|\mathcal{Q}G(\boldsymbol{z}^{k+1}-\boldsymbol{z}^k)\|_2^2,\end{aligned}$$

where $\lambda_{\max}(FF^T)$ is the largest eigenvalue of FF^T, (3.54) can be reduced to

$$\begin{aligned}&\|\boldsymbol{\beta}^{k+1}-\text{prox}_f(\boldsymbol{\beta}^{k+1}-(\mathcal{Q}F)^T\boldsymbol{\gamma}^{k+1})\|_2^2 \\ &\leq 3\phi^2\|F^T\mathcal{P}F+\mathcal{T}_f\|_2\|\boldsymbol{\beta}^{k+1}-\boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}+3\phi^2\lambda_{\max}(FF^T)\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{Q}G} \\ &+3(1-\frac{1}{\theta})^2\|(QF)^T(\boldsymbol{\gamma}^{k+1}-\boldsymbol{\gamma}^k)\|_2^2.\end{aligned} \tag{3.55}$$

Similarly we can bound the term $\|\boldsymbol{z}^{k+1}-\text{prox}_g(\boldsymbol{z}^{k+1}-(\mathcal{Q}G)^T\boldsymbol{\gamma}^{k+1})\|_2^2$,

$$\begin{aligned}&\|\boldsymbol{z}^{k+1}-\text{prox}_g(\boldsymbol{z}^{k+1}-(\mathcal{Q}G)^T\boldsymbol{\gamma}^{k+1})\|_2^2 \\ &\leq 2\phi^2\|G^T\mathcal{P}G+\mathcal{T}_g\|_2\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}+2(1-\frac{1}{\theta})^2\|(\mathcal{Q}G)^T(\boldsymbol{\gamma}^{k+1}-\boldsymbol{\gamma}^k)\|_2^2.\end{aligned} \tag{3.56}$$

From the update of $\boldsymbol{\gamma}$, we have

$$\|\mathcal{Q}(\boldsymbol{c}-F\boldsymbol{\beta}^{k+1}-G\boldsymbol{z}^{k+1})\|_2^2=(\theta\phi)^{-2}\|\boldsymbol{\gamma}^{k+1}-\boldsymbol{\gamma}^k\|_2^2. \tag{3.57}$$

Combining (3.55), (3.56) and (3.57), we can obtain that

$$\begin{aligned}
&\|\boldsymbol{\beta}^{k+1} - \mathrm{prox}_f(\boldsymbol{\beta}^{k+1} - (\mathcal{Q}F)^T\boldsymbol{\gamma}^{k+1})\|_2^2 + \|\boldsymbol{z}^{k+1} - \mathrm{prox}_g(\boldsymbol{z}^{k+1} - G\boldsymbol{\gamma}^{k+1})\|_2^2 \\
&+ \|\mathcal{Q}(\boldsymbol{c} - F\boldsymbol{\beta}^{k+1} - G\boldsymbol{z}^{k+1})\|_2^2 \\
&\leq 3\phi^2\|F^T\mathcal{P}F + \mathcal{T}_f\|_2\|\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+\mathcal{T}_f} + 3\phi^2\lambda_{\max}(FF^T)\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{Q}G} \\
&+ (\theta\phi)^{-2}\|\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k\|_2^2 \\
&+ 3(1 - \frac{1}{\theta})^2\|(\mathcal{Q}F)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\|_2^2 + 2\phi^2\|G^T\mathcal{P}G + \mathcal{T}_g\|_2\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g} \\
&+ 2(1 - \frac{1}{\theta})^2\|(\mathcal{Q}G)^T(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\|_2^2 \\
&\leq C \times \big(\|\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+\mathcal{T}_f} + \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^TG+\mathcal{T}_g} + \theta^{-2}\phi^{-1}\|\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k\|_2^2\big).
\end{aligned} \tag{3.58}$$

□

Lemma 3.6. *Suppose that Assumptions 3.2 and 3.3 hold, and assume that both* $F^TF + \mathcal{T}_f$ *and* $G^TG + \mathcal{T}_g$ *are positive definite. Then for all* k *sufficiently large and* $\theta \in (0, \frac{1+\sqrt{5}}{2})$*, there exists* $\mu \in (0,1)$ *such that*

$$\|\boldsymbol{u}^{k+1} - \bar{\boldsymbol{u}}\|_{\mathcal{H}_1} + \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g} \leq \mu\Big(\|\boldsymbol{u}^k - \bar{\boldsymbol{u}}\|_{\mathcal{H}_1} + \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}\Big), \tag{3.59}$$

where

$$\mathcal{H}_1 = \begin{pmatrix} F^T(m_1\mathcal{Q} + \mathcal{P})F + \mathcal{T}_f & m_1F^T\mathcal{Q}G & \cdots & \mathbf{0} \\ m_1G^T\mathcal{Q}F & G^T(\mathcal{P} + (m_1+1)\mathcal{Q})G & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \theta^{-1}\phi^{-2}\boldsymbol{I} \end{pmatrix} \quad \textit{with} \quad m_1 \in (0,1). \tag{3.60}$$

Proof. From Theorem 3.1 in [66], we can derive the following results.

$$
\begin{aligned}
&\Big\{\|\boldsymbol{\beta}^k-\bar{\boldsymbol{\beta}}\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}+\|\boldsymbol{z}^k-\bar{\boldsymbol{z}}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}+\|\boldsymbol{z}^k-\boldsymbol{z}^{k-1}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}\\
&+(1-\min\{\theta,\frac{1}{\theta}\})\|\mathcal{Q}(F\boldsymbol{\beta}^k+G\boldsymbol{z}^k-\boldsymbol{c})\|_2^2+\theta^{-1}\phi^{-2}\|\boldsymbol{\gamma}^k-\bar{\boldsymbol{\gamma}}\|_2^2\Big\}\\
&-\Big\{\|\boldsymbol{\beta}^{k+1}-\bar{\boldsymbol{\beta}}\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}+\|\boldsymbol{z}^{k+1}-\bar{\boldsymbol{z}}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}+\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}\\
&+(1-\min\{\theta,\frac{1}{\theta}\})\|\mathcal{Q}(F\boldsymbol{\beta}^{k+1}+G\boldsymbol{z}^{k+1}-\boldsymbol{c})\|_2^2+\theta^{-1}\phi^{-2}\|\boldsymbol{\gamma}^{k+1}-\bar{\boldsymbol{\gamma}}\|_2^2\Big\}\\
&\geq\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g+(\theta-\theta^2+\min(\theta^2,1))G\mathcal{Q}^TG}+\|\boldsymbol{\beta}^{k+1}-\boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+T_f}\\
&+(1-\theta+\min\{\theta,\theta^{-1}\})\|\mathcal{Q}(F\boldsymbol{\beta}^{k+1}+G\boldsymbol{z}^{k+1}-\boldsymbol{c})\|_2^2.
\end{aligned}
\tag{3.61}
$$

When $\theta \in (0, \frac{1+\sqrt{5}}{2})$, $(1-\theta+\phi\min\{\theta,\theta^{-1}\}) > 0$. Let $d_1 \in (0, \frac{1}{2})$, then we have

$$
\begin{aligned}
& \Big\{ \|\boldsymbol{\beta}^k - \bar{\boldsymbol{\beta}}\|^2_{F^T\mathcal{P}F+\mathcal{T}_f} + \|\boldsymbol{z}^k - \bar{\boldsymbol{z}}\|^2_{G^TG+\mathcal{T}_g} + \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g} \\
& + (1 + d_1 - d_1\theta - (1-d_1)\min\{\theta, \frac{1}{\theta}\})\|\mathcal{Q}(F\boldsymbol{\beta}^k + G\boldsymbol{z}^k - \boldsymbol{c})\|_2^2 \\
& + \theta^{-1}\phi^{-2}\|\boldsymbol{\gamma}^k - \bar{\boldsymbol{\gamma}}\|_2^2 \Big\} \\
& - \Big\{ \|\boldsymbol{\beta}^{k+1} - \bar{\boldsymbol{\beta}}\|^2_{F^T\mathcal{P}F+\mathcal{T}_f} + \|\boldsymbol{z}^{k+1} - \bar{\boldsymbol{z}}\|^2_{G^TG+\mathcal{T}_g} + \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g} \\
& + (1 + d_1 - d_1\theta - (1-d_1)\min\{\theta, \frac{1}{\theta}\})\|\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c})\|_2^2 \\
& + \theta^{-1}\phi^{-2}\|\boldsymbol{\gamma}^{k+1} - \bar{\boldsymbol{\gamma}}\|_2^2 \Big\} \\
\geq\ & \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g+(\theta-\theta^2+\min(\theta^2,1))G^T\mathcal{Q}G} + \|\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+T_f} \\
& + (1-d_1)(1-\theta+\min\{\theta,\theta^{-1}\})\|\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c})\|_2^2 \\
& + d_1(1-\theta+\min\{\theta,\theta^{-1}\})\|\mathcal{Q}(F\boldsymbol{\beta}^k + G\boldsymbol{z}^k - \boldsymbol{c})\|_2^2 \\
=\ & \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g+(\theta-\theta^2+\min(\theta^2,1))G^T\mathcal{Q}G} + \|\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+T_f} \\
& + (1-2d_1)(1-\theta+\min\{\theta,\theta^{-1}\})\theta^{-2}\phi^{-2}\|\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k\|_2^2 \\
& + d_1(1-\theta+\min\{\theta,\theta^{-1}\})(\|\mathcal{Q}(F\boldsymbol{\beta}^k + G\boldsymbol{z}^k - \boldsymbol{c})\|_2^2 \\
& + \|\mathcal{Q}(F\boldsymbol{\beta}^{k+1} + G\boldsymbol{z}^{k+1} - \boldsymbol{c})\|_2^2) \\
\geq\ & \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g+(\theta-\theta^2+\min(\theta^2,1))G^T\mathcal{Q}G} + \|\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+T_f} \\
& + (1-2d_1)(1-\theta+\min\{\theta,\theta^{-1}\})\theta^{-2}\phi^{-2}\|\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k\|_2^2 \\
& + \frac{1}{2}d_1(1-\theta+\min\{\theta,\theta^{-1}\})\|\mathcal{Q}F(\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k) + \mathcal{Q}G(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2
\end{aligned}
\tag{3.62}
$$

Note that $\mathcal{Q}(F\boldsymbol{\beta}^{k+1}+G\boldsymbol{z}^{k+1}-\boldsymbol{c})=\mathcal{Q}F(\boldsymbol{\beta}^{k+1}-\bar{\boldsymbol{\beta}})+\mathcal{Q}G(\boldsymbol{z}^{k+1}-\bar{\boldsymbol{z}})$, and we have

$$
\begin{aligned}
&\Big\{\|\boldsymbol{\beta}^k-\bar{\boldsymbol{\beta}}\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}+\|\boldsymbol{z}^k-\bar{\boldsymbol{z}}\|^2_{G^TG+\mathcal{T}_g}+\|\boldsymbol{z}^k-\boldsymbol{z}^{k-1}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}\\
&+(1+d_1-d_1\theta-(1-d_1)\min\{\theta,\frac{1}{\theta}\})\|\mathcal{Q}F(\boldsymbol{\beta}^k-\bar{\boldsymbol{\beta}})+\mathcal{Q}G(\boldsymbol{z}^k-\bar{\boldsymbol{z}})\|_2^2\\
&+\theta^{-1}\phi^{-2}\|\boldsymbol{\gamma}^k-\bar{\boldsymbol{\gamma}}\|_2^2\Big\}\\
&-\Big\{\|\boldsymbol{\beta}^{k+1}-\bar{\boldsymbol{\beta}}\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}+\|\boldsymbol{z}^{k+1}-\bar{\boldsymbol{z}}\|^2_{G^TG+\mathcal{T}_g}+\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}\\
&+(1+d_1-d_1\theta-(1-d_1)\min\{\theta,\frac{1}{\theta}\})\|\mathcal{Q}F(\boldsymbol{\beta}^{k+1}-\bar{\boldsymbol{\beta}})+\mathcal{Q}G(\boldsymbol{z}^{k+1}-\bar{\boldsymbol{z}}))\|_2^2\\
&+\theta^{-1}\phi^{-2}\|\boldsymbol{\gamma}^{k+1}-\bar{\boldsymbol{\gamma}}\|_2^2\Big\}\\
\geq\;&\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g+(\theta-\theta^2+\min(\theta^2,1))G^T\mathcal{Q}G}+\|\boldsymbol{\beta}^{k+1}-\boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+T_f}\\
&+(1-2d_1)(1-\theta+\min\{\theta,\theta^{-1}\})\theta^{-2}\phi^{-2}\|\boldsymbol{\gamma}^{k+1}-\boldsymbol{\gamma}^k\|_2^2\\
&+\frac{1}{2}d_1(1-\theta+\min\{\theta,\theta^{-1}\})\|\mathcal{Q}F(\boldsymbol{\beta}^{k+1}-\boldsymbol{\beta}^k)+\mathcal{Q}G(\boldsymbol{z}^{k+1}-\boldsymbol{z}^k)\|_2^2\\
\geq\;&(1-\theta+\min\{\theta,\theta^{-1}\})\min\{\frac{1}{2}d_1,1-2d_1\}\Big(\|\boldsymbol{\beta}^{k+1}-\boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}\\
&+\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^TG+\mathcal{T}_g}+\theta^{-2}\phi^{-2}\|\boldsymbol{\gamma}^{k+1}-\boldsymbol{\gamma}^k\|_2^2\Big)
\end{aligned}
\tag{3.63}
$$

Let $m_1=1+d_1-d_1\theta-(1-d_1)\min\{\theta,\frac{1}{\theta}\}$ in $\mathcal{H}_1$ defined in (3.60), and $m_2=(1-\theta+\min\{\theta,\theta^{-1}\})\min\{\frac{1}{2}d_1,1-2d_1\}$. Note that when $\theta\in(0,\frac{1+\sqrt{5}}{2})$, the following relationship holds.

$$F^TF+T_f\succ 0 \text{ and } G^TG+T_g\succ 0\Longleftrightarrow H_1\succ 0.$$

Combining with Lemma 3.5, we have

$$\begin{aligned}
& \|\boldsymbol{u}^k-\bar{\boldsymbol{u}}\|_{\mathcal{H}_1}+\|\boldsymbol{z}^k-\boldsymbol{z}^{k-1}\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}-(\|\boldsymbol{u}^{k+1}-\bar{\boldsymbol{u}}\|_{\mathcal{H}_1}+\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}) \\
\geq\ & \frac{m_2}{C}\Big(C\times\big(\|\boldsymbol{\beta}^{k+1}-\boldsymbol{\beta}^k\|^2_{F^T\mathcal{P}F+\mathcal{T}_f}+\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g} \\
& +\theta^{-2}\phi^{-2}\|\boldsymbol{\gamma}^{k+1}-\boldsymbol{\gamma}^k\|_2^2\Big) \\
=\ & \frac{m_2}{C}\|\boldsymbol{u}^{k+1}-\boldsymbol{u}^k\|^2_{\mathcal{H}}\geq\frac{m_2d_2}{Cq^2}\|\boldsymbol{u}^{k+1}-\bar{\boldsymbol{u}}\|_2^2+\frac{m_2(1-d_2)}{Cq^2}\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g} \\
\geq\ & \frac{m_2d_2}{Cq^2\lambda_{\max}(\mathcal{H}_1)}\|\boldsymbol{u}^{k+1}-\bar{\boldsymbol{u}}\|^2_{\mathcal{H}_1}+\frac{m_2(1-d_2)}{Cq^2}\|\boldsymbol{z}^{k+1}-\boldsymbol{z}^k\|^2_{G^T\mathcal{P}G+\mathcal{T}_g}.
\end{aligned} \tag{3.64}$$

Take $d_2=\frac{\lambda_{\max}(\mathcal{H}_1)}{1+\lambda_{\max}(\mathcal{H}_1)}$, then we can obtain (3.59) with $\mu=\left[1+\frac{m_2}{Cq^2(1+\lambda_{\max}(\mathcal{H}_1))}\right]^{-1}$.

□

3.4.3 Proof of Theorem 3.1

In this part, we present the proof for Theorem 3.1 as a direct application of Lemma 3.4, 3.5 and 3.6.

Proof. By the definition of H in (3.37), Assumption 3.1 holds. Since $f=\|\cdot\|_1$ and $g=\frac{1}{n}\mathbf{1}_n^T(\cdot)_+$ are piecewise linear-quadratic functions, this implies that both $\mathrm{prox}_f(\cdot)$ $\mathrm{prox}_g(\cdot)$ are piecewise polyhedral and thus Assumption 3.3 also holds. Since we take $\boldsymbol{\mathcal{T}_i}=\eta_i\boldsymbol{I}_{p_i}-\boldsymbol{X}_i^T\boldsymbol{X}_i, i=1,\cdots,K$, which indicates $\mathcal{T}_f+F^TF=\mathrm{Diag}(\eta_1\boldsymbol{I}_{p_1},\cdots,\eta_K\boldsymbol{I}_{p_K})$ are positive definite, and this together with the fact that $G^TG=\boldsymbol{I}_n\succ 0$ imply that the sequence $(\boldsymbol{\beta}^k,\boldsymbol{z}^k,\boldsymbol{\omega}^k,\boldsymbol{\gamma}^k)$ is automatically well defined. By Lemma 3.4, under the condition $\theta\in(0,(1+\sqrt{5})/2)$, the sequence $(\boldsymbol{\beta}^k,\boldsymbol{z}^k,\boldsymbol{\omega}^k,\boldsymbol{\gamma}^k)$ generated by algorithm (3.40) converges to a limit point $(\bar{\boldsymbol{\beta}},\bar{\boldsymbol{z}},\bar{\boldsymbol{\omega}},\bar{\boldsymbol{\gamma}})$ with $(\bar{\boldsymbol{\beta}},\bar{\boldsymbol{z}},\bar{\boldsymbol{\omega}})$ solving (3.6) and $\bar{\boldsymbol{\gamma}}$ is the dual optimal.

To derive the rate of convergence, we first compute $\mathcal{H}_1$. By definition,

$$\mathcal{P} = H(H^T H)^{-1} H^T = \frac{1}{K} \begin{pmatrix} (K-1)I & -I & \cdots & -I \\ -I & (K-1)I & \cdots & -I \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ -I & -I & \cdots & (K-1)I \end{pmatrix},$$

then it follows that

$$\begin{aligned} &\|\boldsymbol{\beta}^{k+1} - \bar{\boldsymbol{\beta}}\|^2_{F^T \mathcal{P} F} = \sum_{i=1}^{K} \|\boldsymbol{X}_i(\boldsymbol{\beta}_i^{k+1} - \bar{\boldsymbol{\beta}}_i)\|_2^2 - \frac{1}{K}\|\sum_{i=1}^{K} \boldsymbol{X}_i^T(\boldsymbol{\beta}_i^{k+1} - \bar{\boldsymbol{\beta}}_i)\|_2^2, \\ &m_1\|\mathcal{Q}F(\boldsymbol{\beta}^k - \bar{\boldsymbol{\beta}}) + \mathcal{Q}G(\boldsymbol{z}^k - \bar{\boldsymbol{z}})\|_2^2 = \frac{m_1}{K}\|\sum_{i=1}^{K} \boldsymbol{X}_i(\boldsymbol{\beta}_i^{k+1} - \bar{\boldsymbol{\beta}}_i) + (\boldsymbol{z}^{k+1} - \bar{\boldsymbol{z}})\|_2^2, \\ &\|\boldsymbol{z}^{k+1} - \bar{\boldsymbol{z}}\|^2_{G^T G + \mathcal{T}_g} = \|\boldsymbol{z}^{k+1} - \bar{\boldsymbol{z}}\|_2^2, \\ &\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2_{G^T \mathcal{P} G + \mathcal{T}_g} = \frac{K-1}{K}\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|_2^2. \end{aligned} \tag{3.65}$$

Plugging equations in (3.65) back into (3.64), we can derive the results in theorem 3.1 easily. □

3.4.4 Proof of Theorem 3.3

To facilitate the technical analysis, the following regularity conditions are required.

(A1) $q = O(n^{c_1})$ for some $0 \leq c_1 \leq 1/2$.

(A2) The densities of $\boldsymbol{X}^+$ given $Y = +1$ and $Y = -1$ are continuous with common support and have finite second moments. In addition, there exists $M > 0$ such that $\|\boldsymbol{X}_j\|_{\max} \leq M$.

(A3) There exists a constant $M_1 > 0$ such that $\lambda_{\max}(n^{-1}\boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}}) \leq M_1$ almost surely. Furthermore, X_{ij} are sub-gaussian random variables for $j \in \mathcal{A}^c$.

(A4) There exits a constant $M_2 > 0$ such that $\lambda_{\min}(H(\boldsymbol{\beta}^*)) \geq M_2$, where $\lambda_{\min}$ denotes the smallest eigenvalue and H denotes the Hessian matrix.

(A5) There exist constants $u_1, f_{\min}, g_{\min} \in (0, \infty)$ such that

$$0 < f_{\min} \leq \min_{|u-1|\leq u_1} f(u) \leq \max_{|u-1|\leq u_1} f(u) \leq f_{\max} < \infty$$

$$0 < g_{\min} \leq \min_{|u+1|\leq u_1} g(u) \leq \max_{|u+1|\leq u_1} g(u) \leq g_{\max} < \infty.$$

Let $\kappa = \min(f_{\min}, g_{\min})$.

Remark. *The conditions (A1), (A2) and (A4) are required to ensure that the oracle estimator is consistent with diverging p. Condition (A4) holds if $H(\boldsymbol{\beta}^*)$ is positive definite. Condition (A3) is also commonly used and has been discussed in [67]. Condition (A5) basically states that there should be sufficient information around the non-differentiable point of the hinge loss.*

Proof. [59] has shown that if λ_{Lasso} satisfies condition (8) in [59] and $\|\hat{\boldsymbol{\beta}}^{Lasso} - \boldsymbol{\beta}^*\|_{\max} \leq a_0\lambda$, then with probability at least $1 - P_1 - P_2$, the LLA algorithm initiated by $\hat{\boldsymbol{\beta}}^{Lasso}$ finds the oracle estimator $\hat{\boldsymbol{\beta}}^{oracle}$ after two iterations. Therefore, it remains to give a clear bound for P_1 and P_2. We write $\hat{\varepsilon}_i = 1 - Y_i\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}^{oracle}$ and $\varepsilon^* = 1 - Y_i\boldsymbol{x}_i^T\boldsymbol{\beta}^*$. Denote $\mathcal{E} = \{i : \hat{\varepsilon}_i = 0\}$. An important property about $\mathcal{E}$ is that if we assume $(\boldsymbol{X}_{\mathcal{A}}, \boldsymbol{y})$ is in general positions, then $|\mathcal{E}| = q + 1$ with probability 1.

First we bound $P_2 = P(\|\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle}\|_{\min} \leq a\lambda)$. We follow a similar argument in [50] for quantile regression. Let $B(r) = \{\Delta \in \mathcal{R}^P : \|\Delta_{\mathcal{A}}\|_2 \leq r, \Delta_{\mathcal{A}^c} = \mathbf{0}\}$, with $r = \|\boldsymbol{\beta}_{\mathcal{A}}^*\|_{\min} - a\lambda$. Define $F(\Delta) = L(\boldsymbol{\beta}^* + \Delta) - L(\boldsymbol{\beta}^*)$. Denote $\hat{\Delta}^{oracle} =$

$\hat{\boldsymbol{\beta}}^{oracle} - \boldsymbol{\beta}^*$, then we have $\hat{\Delta}^{oracle} = \text{argmin}_{\Delta_{\mathcal{A}^c}=\mathbf{0}} F(\Delta)$. Since $P(\|\hat{\boldsymbol{\beta}}^{oracle}_{\mathcal{A}}\|_{\min} \leq a\lambda) \leq 1 - P(\|\hat{\Delta}^{oracle}_{\mathcal{A}}\|_2 \leq r) \leq 1 - P(\inf_{\Delta\in\partial B(r)} F(\Delta) \geq 0)$, we now bound $P(\inf_{\Delta\in\partial B(r)} F(\Delta) \geq 0)$.

$$
\begin{aligned}
F(\Delta) &= \frac{1}{n}\sum_{i=1}^{n}\Big\{[1 - Y_i\boldsymbol{x}_i^T(\boldsymbol{\beta}^* + \Delta)]_+ - [1 - Y_i\boldsymbol{x}_i^T\boldsymbol{\beta}^*]_+\Big\} \\
&= \frac{1}{n}\sum_{i=1}^{n}\Big\{(\varepsilon_i^* - Y_i\boldsymbol{x}_i^T\Delta)_+ - (\varepsilon_i^*)_+\Big\} \\
&= \frac{1}{n}\sum_{i=1}^{n}\Big\{(\varepsilon_i^* - Y_i\boldsymbol{x}_i^T\Delta)[I_{\{\varepsilon_i^* > Y_i\boldsymbol{x}_i^T\Delta\}} - I_{\{\varepsilon_i^*>0\}}] \\
&\quad + (\varepsilon_i^* - Y_i\boldsymbol{x}_i^T\Delta)I_{\{\varepsilon_i^*>0\}} - \varepsilon_i^* I_{\{\varepsilon_i^*>0\}}\Big\} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\Big\{Y_i\boldsymbol{x}_i^T\Delta I_{\{\varepsilon_i^*>0\}}\Big\} \\
&\quad + \frac{1}{n}\sum_{i=1}^{n}\Big\{(\varepsilon_i^* - Y_i\boldsymbol{x}_i^T\Delta)[I_{\{\varepsilon_i^* > Y_i\boldsymbol{x}_i^T\Delta\}} - I_{\{\varepsilon_i^*>0\}}]\Big\} \\
&= I_1 + I_2.
\end{aligned}
\tag{3.66}
$$

We now bound I_1 and I_2 respectively. To bound I_1, note that when $\Delta \in \partial B(r)$, $|Y_i\boldsymbol{x}_i^T\Delta I_{\{\varepsilon_i^*\geq 0\}}| \leq \|\boldsymbol{X}_{\mathcal{A},i}\|_2\|\Delta_{\mathcal{A}}\|_2 \leq (qU_{\mathcal{A}})^{1/2}r$. Denote the gradient vector of the population hinge loss function as $S(\boldsymbol{\beta}) = -E[I(1-Y\boldsymbol{X}^T\boldsymbol{\beta} \geq 0)Y\boldsymbol{X}]$, then we have $S(\boldsymbol{\beta}^*) = 0$. Therefore, $E[-Y_i\boldsymbol{x}_i^T\Delta I_{\{\varepsilon_i^*\geq 0\}}] = \sum_{j=1}^{p}\Delta_j \cdot E[-Y_i x_{ij} I(\varepsilon_i^* \geq 0)] = 0$. Thus by the Hoeffding's inequality, we have

$$
P\left(|I_1| \geq \frac{1}{6}\lambda_{\min}\kappa r^2\right) \leq 2\exp\left(-\frac{n\lambda_{\min}^2\kappa^2 r^2}{18qU_{\mathcal{A}}}\right). \tag{3.67}
$$

Next we bound I_2. We first denote $R_i(\Delta) = (\varepsilon_i^* - Y_i\boldsymbol{x}_i^T\Delta)[I_{\{\varepsilon_i^* \geq Y_i\boldsymbol{x}_i^T\Delta\}} - I_{\{\varepsilon_i^*\geq 0\}}]$. Then we have $I_2 = \frac{1}{n}\sum_{i=1}^{n} R_i(\Delta) = \bar{R}(\Delta)$. Using the inequality (27) in [68], we

have

$$|R_i(\Delta)| \leq \left|\boldsymbol{x}_i^T\Delta\right| I_{\{|\varepsilon_i^*|\leq|\boldsymbol{x}_i^T\Delta|\}} \leq U_{\mathcal{A}}^{1/2} q^{1/2} r. \tag{3.68}$$

Then we can apply the Hoeffding's inequality to bound $|I_2 - E[I_2]|$,

$$\begin{aligned} P\left(|I_2 - E[I_2]| \geq \frac{1}{6}\lambda_{\min}\cdot\kappa\cdot r^2\right) &= P\left(|\bar{R}(\Delta) - E[\bar{R}(\Delta)]| \geq \frac{1}{6}\lambda_{\min}\cdot\kappa\cdot r^2\right) \\ &\leq 2\exp\left(-\frac{\lambda_{\min}^2\kappa^2 r^2 n}{18q\cdot U_{\mathcal{A}}}\right). \end{aligned} \tag{3.69}$$

By the Knights's identity in [69],

$$I_2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i\boldsymbol{x}_i^T\Delta - \varepsilon_i^*)[I_{\{\varepsilon_i^*\leq Y_i\boldsymbol{x}_i^T\Delta\}} - I_{\{\varepsilon_i^*\leq 0\}}] = \frac{1}{n}\sum_{i=1}^{n}\int_0^{Y_i\boldsymbol{x}_i^T\Delta} I_{\{\varepsilon_i^*\leq s\}} - I_{\{\varepsilon_i^*\leq 0\}} ds. \tag{3.70}$$

Thus, by applying Fibini's theorem and mean-value theory, we can derive that

$$\begin{aligned} E[I_2] = &\ \frac{1}{n}\sum_{i}^{n}\int_0^{Y_i\boldsymbol{x}_i^T\Delta}[F_i(1) - F_i(1-s)]P(Y_i=1)ds \\ &+ \frac{1}{n}\sum_{i}^{n}\int_0^{Y_i\boldsymbol{x}_i^T\Delta}[G_i(s-1) - G_i(-1)]P(Y_i=-1)ds \\ = &\ \frac{1}{n}\sum_{i}^{n}\int_0^{Y_i\boldsymbol{x}_i^T\Delta} f_i(m_1(s))\cdot sP(Y_i=1)ds \\ &+ \frac{1}{n}\sum_{i}^{n}\int_0^{Y_i\boldsymbol{x}_i^T\Delta} g_i(m_2(s))\cdot sP(Y_i=-1)ds, \end{aligned} \tag{3.71}$$

where both $m_1(s)$ and $m_2(s)$ are on the line segment $[1-s, 1]$ and $[-1, s-1]$

respectively. Note that for $l = 1, 2$,

$$\begin{aligned} |m_l(s) \pm 1| &\leq \max\{|2 - s|, |s|, 2\} \\ &\leq \max\{|2 - Y_i \boldsymbol{x}_i^T \Delta|, |Y_i \boldsymbol{x}_i^T \Delta|, 2\} \\ &\leq (qU_{\mathcal{A}})^{1/2} r + 2 \\ &\leq u_1, \end{aligned} \tag{3.72}$$

then we can bound $E[I_2]$ as

$$\begin{aligned} E[I_2] &\geq \frac{1}{n} \sum_{i=1}^{n} \int_0^{Y_i \boldsymbol{x}_i^T \Delta} f_{\min} \cdot s P(Y_i = 1) ds + \frac{1}{n} \sum_{i=1}^{n} \int_0^{Y_i \boldsymbol{x}_i^T \Delta} g_{\min} \cdot s P(Y_i = -1) ds \\ &\geq \frac{1}{n} \sum_{i=1}^{n} \int_0^{Y_i \boldsymbol{x}_i^T \Delta} \kappa \cdot s ds \\ &= \frac{1}{2n} \kappa \sum_{i=1}^{n} (Y_i \boldsymbol{x}_{\mathcal{A},i}^T \Delta_{\mathcal{A}})^2 \\ &\geq \frac{1}{2} \lambda_{\min} \kappa r^2. \end{aligned} \tag{3.73}$$

Therefore for any $\Delta \in \partial B(r)$, as long as $|I_1| \leq \frac{1}{6} \lambda_{\min} \kappa r^2$ and $|I_2 - E[I_2]| \leq \frac{1}{6} \lambda_{\min} \kappa r^2$, we have

$$F(\Delta) = I_1 + I_2 \geq -|I_1| + E[I_2] + (I_2 - E[I_2]) \geq \frac{1}{6} \lambda_{\min} \kappa r^2 > 0. \tag{3.74}$$

Hence,

$$\begin{aligned} P_2 &\leq 1 - P\Big(\inf_{\Delta \in \partial B(r)} F(\Delta) > 0 \Big) \\ &\leq P\Big(|I_1| > \frac{1}{6} \lambda_{\min} \kappa r^2 \Big) + P\Big(|I_2 - E[I_2]| > \frac{1}{6} \lambda_{\min} \kappa r^2 \Big) \\ &\leq 4 \exp\left(-\frac{n \lambda_{\min}^2 \kappa^2 r^2}{18 q U_{\mathcal{A}}} \right). \end{aligned} \tag{3.75}$$

In the next step, we bound $P_1 = P(\|\nabla_{\mathcal{A}^c} L(\hat{\boldsymbol{\beta}}^{oracle})\|_{\max} \geq a_1 \lambda)$. Rewrite the

subgradient of the empirical hinge loss at the oracle estimator for $j = 1, \cdots, p$ as

$$\begin{aligned} \nabla_j L(\hat{\boldsymbol{\beta}}^{oracle}) &= -\frac{1}{n}\sum_{i=1}^{n} Y_i X_{ij} I(\hat{\varepsilon}_i \geq 0) - \frac{1}{n}\sum_{i\in\mathcal{E}} Y_i X_{ij}(v_j - 1), \\ &= D_{1,j} + D_{2,j}, \end{aligned} \tag{3.76}$$

where $v_j \in [-1, 0]$ if $j \in \mathcal{E} = \{i : \hat{\varepsilon}_i = 0\}$ and 0 elsewhere. Hence we can bound $|D_{2,j}|$ as

$$\max_{j\in\mathcal{A}^c} |D_{2,j}| \leq \max_{j\in\mathcal{A}^c} \frac{1}{n}\sum_{i\in\mathcal{E}} |Y_i X_{ij}||v_j - 1| \leq \frac{2(q+1)}{n} u_{\mathcal{A}^c} \leq \frac{a_1}{4}\lambda.$$

Thus P_1 can be bounded as

$$P_1 = P(\max_{j\in\mathcal{A}^c} |D_{1,j} + D_{2,j}| \geq a_1\lambda) \leq P\Big(\max_{j\in\mathcal{A}^c} |D_{1,j}| \geq \frac{3a_1}{4}\lambda\Big). \tag{3.77}$$

Then it remains to bound $\max_{j\in\mathcal{A}^c} |D_{1,j}|$.

$$\begin{aligned} D_{1,j} = &-\frac{1}{n}\sum_{i=1}^{n} Y_i X_{ij}\Big[I(\hat{\varepsilon}_i \geq 0) - I(\varepsilon_i^* \geq 0) - P(\hat{\varepsilon}_i \geq 0) + P(\varepsilon_i^* \geq 0)\Big] \\ &-\frac{1}{n}\sum_{i=1}^{n} Y_i X_{ij}\Big[P(\hat{\varepsilon}_i \geq 0) - P(\varepsilon_i^* \geq 0)\Big] - \frac{1}{n}\sum_{i=1}^{n} Y_i X_{ij} I(\varepsilon_i^* \geq 0). \end{aligned} \tag{3.78}$$

Therefore, by union bound we have

$$
\begin{aligned}
&P\Big(\max_{j\in\mathcal{A}^c}|D_{1,j}|) > \frac{3a_1}{4}\lambda\Big) \\
&\le P\Big(\max_{j\in\mathcal{A}^c}\frac{1}{n}\Big|\sum_{i=1}^{n} Y_i X_{ij}\Big[I(\hat{\varepsilon}_i \ge 0) - I(\varepsilon_i^* \ge 0) - P(\hat{\varepsilon}_i \ge 0) + P(\varepsilon_i^* \ge 0)\Big| > \frac{a_1}{4}\lambda\Big) \\
&+ P\Big(\max_{j\in\mathcal{A}^c}\frac{1}{n}\Big|\sum_{i=1}^{n} Y_i X_{ij}\Big[P(\hat{\varepsilon}_i \ge 0) - P(\varepsilon_i^* \ge 0)\Big]\Big| > \frac{a_1}{4}\lambda\Big) \\
&+ P\Big(\max_{j\in\mathcal{A}^c}\frac{1}{n}\Big|\sum_{i=1}^{n} Y_i X_{ij} I(\varepsilon_i^* \ge 0)\Big| > \frac{a_1}{4}\lambda\Big) \\
&= P\Big(\max_{j\in\mathcal{A}^c}|D_{1,j}^1| > \frac{a_1}{4}\lambda\Big) + P\Big(\max_{j\in\mathcal{A}^c}|D_{1,j}^2| > \frac{a_1}{4}\lambda\Big) + P\Big(\max_{j\in\mathcal{A}^c}|D_{1,j}^3| > \frac{a_1}{4}\lambda\Big)
\end{aligned}
\tag{3.79}
$$

Since $E(Y_i X_{ij} I(1 - Y_i \boldsymbol{X}_{\mathcal{A},i}^T \boldsymbol{\beta}_{\mathcal{A}}^* \ge 0)) = 0$, by using the Bernstein's inequality, we have

$$
\begin{aligned}
P\Big(\max_{j\in\mathcal{A}^c}|D_{1,j}^3| > \frac{a_1}{4}\lambda\Big) &= P\Big(\max_{j\in\mathcal{A}^c}\frac{1}{n}|\sum_{i=1}^{n} Y_i X_{ij} I(1 - Y_i \boldsymbol{X}_{\mathcal{A},i}^T \boldsymbol{\beta}_{\mathcal{A}}^* \ge 0)| > \frac{a_1}{4}\lambda\Big) \\
&\le (p-q)\exp(-C_3 n a_1^2 \lambda^2)
\end{aligned}
\tag{3.80}
$$

for some constant $C_3 > 0$.

Let $\varepsilon_i(\boldsymbol{\beta}_{\mathcal{A}}) = 1 - y_i \boldsymbol{X}_{\mathcal{A},i}^T \boldsymbol{\beta}_{\mathcal{A}}$. To bound the first term, define a ball $B^* = \{\boldsymbol{\beta} \in \mathcal{R}^q : \|\boldsymbol{\beta}_{\mathcal{A}} - \boldsymbol{\beta}_{\mathcal{A}}^*\|_{\ell_2} \le r_1\}$, where we take $r_1 = 3\sqrt{\frac{U_{\mathcal{A}} q \log(n)}{n}}$. Under the event that $\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle} \in B^*$,

$$
|D_{1,j}^1| \le \sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}\left|\frac{1}{n}\sum_{i=1}^{n} Y_i X_{ij}\Big[I_{\{\varepsilon_i(\boldsymbol{\beta}_{\mathcal{A}})\ge 0\}} - I_{\{\varepsilon_i^*\ge 0\}} - P(\varepsilon_i(\boldsymbol{\beta}_{\mathcal{A}}) \ge 0) + P(\varepsilon_i^* \ge 0)\Big]\right|.
\tag{3.81}
$$

By Lemma 7.2 in [58], we can derive that

$$
\begin{aligned}
&P\Big(\max_{j\in\mathcal{A}^c}|D^1_{1,j}|>\frac{a_1}{4}\lambda\Big)\\
&\le P\Bigg(\max_{j\in\mathcal{A}^c}\sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}\bigg|\frac{1}{n}\sum_{i=1}^n Y_iX_{ij}\Big[I_{\{\varepsilon_i(\boldsymbol{\beta}_{\mathcal{A}}\ge 0)\}}-I_{\{\varepsilon_i^*\ge 0\}}-P(\varepsilon_i(\boldsymbol{\beta}_{\mathcal{A}})\ge 0)\\
&\qquad +P(\varepsilon_i^*\ge 0)\Big]\bigg|>\frac{a_1}{4}\lambda\Bigg)\\
&\le (p-q)\cdot d_1n^{4q}\exp\Big(-\frac{n^2a_1^2\lambda^2/4}{d_2qU_{\mathcal{A}}\sqrt{n\log(n)}+d_3na_1\lambda}\Big)\\
&\le C_1\exp\Big[\log(p-q)+4q\log(n)-C_2na_1\lambda\Big].
\end{aligned}
\tag{3.82}
$$

for some constants $d_i\ge 0, i=1,2,3$, $C_1>0$ and $C_2>0$.

Next we bound $|D^2_{1,j}|$. By the mean-value theory,

$$
\begin{aligned}
|D^2_{1,j}|&=\bigg|\frac{1}{n}\sum_{i=1}^n Y_iX_{ij}\Big[P(\varepsilon_i(\boldsymbol{\beta}_{\mathcal{A}})\ge 0)-P(\varepsilon_i^*\ge 0)\Big]\bigg|\\
&\le \sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}\frac{u_{\mathcal{A}^c}}{n}\sum_{i=1}^n\Big[|F_i(1+\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}^*_{\mathcal{A}}-\boldsymbol{\beta}_{\mathcal{A}}))-F_i(1)|P(Y_i=1)\\
&\quad +|-G_i(-1+\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}^*_{\mathcal{A}}-\boldsymbol{\beta}_{\mathcal{A}})+G_i(-1))|P(Y_i=-1)\Big]\\
&\le \sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}\frac{u_{\mathcal{A}^c}}{n}\sum_{i=1}^n\Big[|f_i(\xi_i^1(\boldsymbol{\beta}_{\mathcal{A}}))\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}_{\mathcal{A}}-\boldsymbol{\beta}^*_{\mathcal{A}})|+|g_i(\xi_i^2(\boldsymbol{\beta}_{\mathcal{A}}))\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}_{\mathcal{A}}-\boldsymbol{\beta}^*_{\mathcal{A}})|\Big],
\end{aligned}
\tag{3.83}
$$

where $\xi_i^1(\boldsymbol{\beta}_{\mathcal{A}})$ lies between 1 and $1+\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}^*_{\mathcal{A}}-\boldsymbol{\beta}_{\mathcal{A}})$ and $\xi_i^2(\boldsymbol{\beta}_{\mathcal{A}})$ is between $-1+\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}^*_{\mathcal{A}}-\boldsymbol{\beta}_{\mathcal{A}})$ and -1.

Since $\sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}|\boldsymbol{X}^T_{\mathcal{A},i}(\boldsymbol{\beta}_{\mathcal{A}}-\boldsymbol{\beta}^*_{\mathcal{A}})|+2\le\|\boldsymbol{X}_{\mathcal{A},i}\|_{\ell_2}\|\boldsymbol{\beta}_{\mathcal{A}}-\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\ell_2}+2\le(qU_{\mathcal{A}})^{1/2}r_1+2=3qU_{\mathcal{A}}\sqrt{\frac{\log n}{n}}+2\le u_1$, then we have $\sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}|f_i(\xi_i^1(\boldsymbol{\beta}_{\mathcal{A}}))|\le f_{\max}$ and $\sup_{\boldsymbol{\beta}_{\mathcal{A}}\in B^*}|g_i(\xi_i(\boldsymbol{\beta}_{\mathcal{A}}))|\le g_{\max}$, and this indicates that when $\hat{\boldsymbol{\beta}}_A^{oracle}\in B^*$,

$$
\max_{j\in\mathcal{A}^c}|D^2_{1,j}|\le\frac{u_{\mathcal{A}^c}}{n}(f_{\max}+g_{\max})(qM_{\mathcal{A}})^{1/2}r_1\le\frac{a_1}{4}\lambda.
$$

Therefore the second term can be bounded by

$$\begin{aligned} P\Big(\max_{j\in\mathcal{A}^c}|D^2_{1,j}|>\frac{a_1}{4}\lambda\Big) &= 1-P\Big(\max_{j\in\mathcal{A}^c}|D^2_{1,j}|\le\frac{a_1}{4}\lambda\Big) \\ &\le 1-P(\hat{\boldsymbol{\beta}}_A^{oracle}\in B^*)=P(\hat{\boldsymbol{\beta}}_A^{oracle}\notin B^*). \end{aligned} \tag{3.84}$$

Using the results in (3.75), we have

$$P(\hat{\boldsymbol{\beta}}_A^{oracle}\notin B^*)\le P(\|\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle}-\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\ell_2}>r_1)\le 4\exp(-\frac{1}{2}\lambda^2_{\min}\kappa^2\log n). \tag{3.85}$$

Putting (3.81), (3.85) and (3.80), we have that

$$\begin{aligned} P_1 &\le P\Big(\max_{j\in\mathcal{A}^c}|D^1_{1,j}|\ge\frac{1}{4}\lambda\Big)+P\Big(\hat{\boldsymbol{\beta}}_A^{oracle}\notin B^*\Big)+P\Big(\max_{j\in\mathcal{A}^c}|D^3_{1,j}|\ge\frac{a_1}{4}\lambda\Big) \\ &\le C_1\exp\Big[\log(p)+4q\log(n)-C_2n\lambda\Big] \\ &\quad+4\exp(-\frac{1}{2}\lambda^2_{\min}\kappa^2\log n)+p\exp(-C_3n\lambda^2). \end{aligned} \tag{3.86}$$

□

3.4.5 Algorithms for the Nonconvex Penalized Quantile Regression

In this part, we present the algorithm for nonconvex quantile regression in (3.8) and (3.9).

3.4.6 Algorithms for the Nonconvex Penalized SVM

Given $\boldsymbol{\beta}^{k+1}$, variables are updated as follows,

$$\boldsymbol{\omega}_i^{k+\frac{1}{2}}=\frac{1}{K+1}(\mathbf{1}_n-\boldsymbol{y}\beta_0^{k+1}-\boldsymbol{z}^k+(K+1)\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1}-\sum_{i=1}^{K}\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1}) \tag{3.87}$$

Algorithm 3.8 ADMM-CD for Nonconvex Penalized Quantile Regression

Initialization: $\tilde{\boldsymbol{\beta}}^{(0)}, \lambda, \upsilon, \tilde{\boldsymbol{z}}^{(0)}, \tilde{\boldsymbol{\gamma}}^{(0)}, \tilde{\boldsymbol{\omega}}_i^{(0)}$, and $\phi > 0, \theta = 1.618, k = 0$.

while the stopping criterion is not satisfied, **do**

Update $\tilde{\boldsymbol{\beta}}^{(k+1)}$ by

$$\tilde{\boldsymbol{\beta}}_1^{k+1} = \underset{\boldsymbol{\beta}_1 \in \mathcal{R}^{p_1}}{\operatorname{argmin}} n\upsilon\lambda\|\boldsymbol{\beta}_1\|_1 + \frac{\phi}{2}\|\boldsymbol{X}_1\boldsymbol{\beta}_1 + \sum_{i=1}^{K}\boldsymbol{\omega}_i^k + \boldsymbol{z}^k - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\|_2^2,$$

$$\tilde{\boldsymbol{\beta}}_i^{k+1} = \underset{\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}}{\operatorname{argmin}} n\upsilon\lambda\|\boldsymbol{\beta}_i\|_1 + \frac{\phi}{2}\|\boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2, \quad i = 2, \cdots, K$$

Compute $\tilde{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.16), then update $\tilde{\boldsymbol{z}}^{(k+1)}$ by (3.17).

Update $\tilde{\boldsymbol{\omega}}^{(k+1)}$ by (3.19), then $\tilde{\boldsymbol{\gamma}}^{(k+1)}$ by (3.20) and (3.21).

end while The solution is denoted as $\hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{\ell_1}$

Initialization: $\hat{\boldsymbol{\beta}}^{(0)} = \hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{(0)} = \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{(0)} = \hat{\boldsymbol{\omega}}^{\ell_1}$ and $\phi > 0, \theta = 1.618, k = 0$.

Compute $\alpha_j = \lambda^{-1}P'_\lambda(|\hat{\beta}_j^{(0)}|)$ for $j = 1, \cdots, p$.

while the stopping criterion is not satisfied, **do**

Update $\hat{\boldsymbol{\beta}}^{(k+1)}$ by (3.10) and (3.11).

Compute $\hat{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.16), then update $\hat{\boldsymbol{z}}^{(k+1)}$ by (3.17).

Update $\hat{\boldsymbol{\omega}}^{(k+1)}$ by (3.19), then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by (3.20) and (3.21).

end while

$$\boldsymbol{z}^{k+1} = \Big(-\frac{1}{n\phi} - \boldsymbol{y}\beta_0^{k+1} - \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} + \mathbf{1}_n - \frac{\boldsymbol{\gamma}_0^k}{\phi}\Big)_+ \tag{3.88}$$

$$+ \Big(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \frac{\boldsymbol{\gamma}_0^k}{\phi}\Big)_+ \tag{3.89}$$

$$\boldsymbol{\omega}_i^{k+1} = \frac{1}{K+1}(\mathbf{1}_n - \boldsymbol{y}\beta_0^{k+1} - \boldsymbol{z}^{k+1} + (K+1)\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \sum_{i=1}^{K}\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1}) \tag{3.90}$$

$$\boldsymbol{\gamma}_0^{k+1} = \boldsymbol{\gamma}_0^k + \theta\phi(\boldsymbol{y}\beta_0^{k+1} + \boldsymbol{z}^{k+1} + \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+1} - \mathbf{1}_n) \tag{3.91}$$

$$\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 1, \cdots, K. \tag{3.92}$$

Algorithm 3.9 ADMM-prox for Nonconvex Penalized Quantile Regression

Initialization: $\tilde{\boldsymbol{\beta}}^{(0)}, \lambda, \upsilon, \tilde{\boldsymbol{z}}^{(0)}, \tilde{\boldsymbol{\gamma}}^{(0)}, \tilde{\boldsymbol{\omega}}_i^{(0)}$, and $\phi > 0, \theta = 1.618, k = 0$.

while the stopping criterion is not satisfied, **do**

Update $\tilde{\boldsymbol{\beta}}^{(k+1)}$ by

$$\tilde{\boldsymbol{\beta}}_1^{k+1} = \text{Shrink}\Big(\boldsymbol{\beta}_{1j}^k - \frac{\phi}{\eta_1}\boldsymbol{X}_{1j}^T(\boldsymbol{X}_1\boldsymbol{\beta}_1^k + \sum_{i=2}^{K}\boldsymbol{\omega}_i^k + \boldsymbol{z}^k - \boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}), \frac{n\upsilon\lambda}{\eta_1}\Big)_{j=1,\cdots,p_1}$$

$$\tilde{\boldsymbol{\beta}}_i^{k+1} = \text{Shrink}\Big(\boldsymbol{\beta}_{ij}^k - \frac{\phi}{\eta_i}\boldsymbol{X}_{ij}^T(\boldsymbol{X}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}), \frac{n\upsilon\lambda}{\eta_i}\Big)_{j=1,\cdots,p_i}$$

Compute $\tilde{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.16), then update $\tilde{\boldsymbol{z}}^{(k+1)}$ by (3.17).

Update $\tilde{\boldsymbol{\omega}}^{(k+1)}$ by (3.19), then $\tilde{\boldsymbol{\gamma}}^{(k+1)}$ by (3.20) and (3.21).

end while denote the solution as $\hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{\ell_1}$

Initialization: $\hat{\boldsymbol{\beta}}^{(0)} = \hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{(0)} = \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{(0)} = \hat{\boldsymbol{\omega}}^{\ell_1}$ and $\phi > 0, \theta = 1.618, k = 0$.

Compute $\alpha_j = \lambda^{-1}P_\lambda'(|\hat{\beta}_j^{(0)}|)$ for $j = 1, \cdots, p$.

while the stopping criterion is not satisfied, **do**

Update $\hat{\boldsymbol{\beta}}^{(k+1)}$ by (3.14) and (3.15).

Compute $\hat{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.16), then update $\hat{\boldsymbol{z}}^{(k+1)}$ by (3.17).

Update $\hat{\boldsymbol{\omega}}^{(k+1)}$ by (3.19), then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by (3.20) and (3.21).

end while

Algorithm 3.10 ADMM-CD for Nonconvex Penalized Support Vector Machine

Initialization: $\tilde{\boldsymbol{\beta}}^{(0)}, \lambda, \upsilon, \tilde{\boldsymbol{z}}^{(0)}, \tilde{\boldsymbol{\gamma}}^{(0)}, \tilde{\boldsymbol{\omega}}_i^{(0)}$, and $\phi > 0, \theta = 1.618, k = 0$.

while the stopping criterion is not satisfied, **do**

Update $\tilde{\boldsymbol{\beta}}^{(k+1)}$ by

$$\tilde{\boldsymbol{\beta}}_0^{k+1} = \frac{1}{\|\boldsymbol{y}\|^2}\boldsymbol{y}^T(\mathbf{1}_n - \boldsymbol{z}^k - \sum_{i=1}^{K}\boldsymbol{\omega}_i^k - \frac{\boldsymbol{\gamma}_0^k}{\phi}),$$

$$\tilde{\boldsymbol{\beta}}_i^{k+1} = \underset{\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}}{\operatorname{argmin}}\, \upsilon\lambda\|\boldsymbol{\beta}_i\|_1 + \frac{\phi}{2}\|\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2, \quad i = 1, \cdots, K$$

Compute $\tilde{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.87), then update $\tilde{\boldsymbol{z}}^{(k+1)}$ by (3.88).

Update $\tilde{\boldsymbol{\omega}}^{(k+1)}$ by (3.90), then $\tilde{\boldsymbol{\gamma}}^{(k+1)}$ by (3.91) and (3.92).

end while The solution is denoted as $\hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{\ell_1}$

Initialization: $\hat{\boldsymbol{\beta}}^{(0)} = \hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{(0)} = \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{(0)} = \hat{\boldsymbol{\omega}}^{\ell_1}$ and $\phi > 0, \theta = 1.618, k = 0$.

while the stopping criterion is not satisfied, **do**

Compute $\alpha_j^{k+1} = \lambda^{-1}P_\lambda'(|\hat{\beta}_j^{(k)}|)$ for $j = 1, \cdots, p$.

Update $\hat{\boldsymbol{\beta}}^{(k+1)}$ by (3.27) and (3.28).

Compute $\hat{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.87), then update $\hat{\boldsymbol{z}}^{(k+1)}$ by (3.88).

Update $\hat{\boldsymbol{\omega}}^{(k+1)}$ by (3.90), then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by (3.91) and (3.92).

end while

Algorithm 3.11 ADMM-prox for Nonconvex Penalized Support Vector Machine

Initialization: $\tilde{\boldsymbol{\beta}}^{(0)}, \lambda, \upsilon, \tilde{\boldsymbol{z}}^{(0)}, \tilde{\boldsymbol{\gamma}}^{(0)}, \tilde{\boldsymbol{\omega}}_i^{(0)}$, and $\phi > 0, \theta = 1.618, k = 0$.

while the stopping criterion is not satisfied, **do**

Update $\tilde{\boldsymbol{\beta}}^{(k+1)}$ by

$$\tilde{\boldsymbol{\beta}}_0^{k+1} = \frac{1}{\|\boldsymbol{y}\|^2}\boldsymbol{y}^T(\mathbf{1}_n - \boldsymbol{z}^k - \sum_{i=1}^{K}\boldsymbol{\omega}_i^k - \frac{\boldsymbol{\gamma}_0^k}{\phi})$$

$$\tilde{\boldsymbol{\beta}}_i^{k+1} = \text{Shrink}\Big(\boldsymbol{\beta}_{ij}^k - \frac{1}{\eta_i}\boldsymbol{A}_{i(j)}^T(\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}), \frac{\upsilon\lambda}{\phi\eta_i}\Big)_{j=1,\cdots,p_i}.$$

Compute $\tilde{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.87), then update $\tilde{\boldsymbol{z}}^{(k+1)}$ by (3.88).

Update $\tilde{\boldsymbol{\omega}}^{(k+1)}$ by (3.90), then $\tilde{\boldsymbol{\gamma}}^{(k+1)}$ by (3.91) and (3.92).

end while The solution is denoted as $\hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{\ell_1}$

Initialization: $\hat{\boldsymbol{\beta}}^{(0)} = \hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{(0)} = \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{(0)} = \hat{\boldsymbol{\omega}}^{\ell_1}$ and $\phi > 0, \theta = 1.618, k = 0$.

while the stopping criterion is not satisfied, **do**

Compute $\alpha_j^{k+1} = \lambda^{-1}P_\lambda'(|\hat{\beta}_j^{(k)}|)$ for $j = 1, \cdots, p$.

Update $\hat{\boldsymbol{\beta}}^{(k+1)}$ by (3.27) and (3.28).

Compute $\hat{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (3.87), then update $\hat{\boldsymbol{z}}^{(k+1)}$ by (3.88).

Update $\hat{\boldsymbol{\omega}}^{(k+1)}$ by (3.90), then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by (3.91) and (3.92).

end while

Chapter 4

Multi-block ADMM for the Nonconvex Dantzig Selector and Linear Programming Discriminant Rule

Dantzig selector was first proposed in [70] as a variable selection procedure for linear regression model,

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{4.1}$$

where $\boldsymbol{y} = (y_1, \cdots, y_n)^T$ is the sample response vector, $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^T = (\boldsymbol{X}_1, \cdots, \boldsymbol{X}_p)$ is an $n \times p$ design matrix and $\boldsymbol{\varepsilon} = (\varepsilon_1, \cdots, \varepsilon_n)^T$ is the error term. In modeling ultra-high dimensional data where $\log(p) = \mathcal{O}(n^c)$ for $c \in (0, 1)$, we assume that the solution is sparse and the number of nonzero coefficients $q \ll p$. While both minimizing the ℓ_1-norm, Lasso controls the mean squared error and the Dantzig selector bounds the correlation between residuals with predictors, which

is also the maximum component of the gradient of the squared error. The Dantzig selector is the solution to the following problem,

$$\begin{aligned} \min_{\boldsymbol{\beta}\in\mathcal{R}^p} \quad & \|\boldsymbol{\beta}\|_1 \\ \text{s.t.} \quad & \|\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{y})\|_\infty \le \lambda, \end{aligned} \tag{4.2}$$

where λ is a tuning parameter and $\|\cdot\|_\infty$ is the infinity norm. [70] provides a bound on the ℓ_2 risk of the Dantzig selector under the restricted isometry property condition. [71] establishes the oracle inequalities and bounds on the ℓ_p estimation and prediction loss under a weaker restricted eigenvalue assumption. In this chapter, we propose a multi-block ADMM algorithm for solving the Dantzig selector with a general class of nonconvex penalties satisfying the following properties:

1. $P_\lambda(t)$ is non-decreasing and concave for $t \in [0,\infty)$ with $P_\lambda(0) = 0$,
2. $P_\lambda(t)$ is differentiable in $(0,\infty)$ and $P'_\lambda(0) := P'_\lambda(0+) \ge a_1\lambda$,
3. $P'_\lambda(t) \ge a_1\lambda$ for $t \in (0, a_2\lambda]$; $P'_\lambda(t) = 0$ for $t \in [a\lambda,\infty)$ with $a > a_2$,

where a_1 and a_2 are two constants associated with each penalty. For example, the SCAD penalty is a folded concave penalty with $a_1 = a_2 = 1$ and its derivative is defined as

$$P'_\lambda(|\beta|) = \lambda\left\{I(|\beta|\le\lambda) + \frac{(a\lambda-|\beta|)_+}{(a-1)\lambda}I(|\beta|>\lambda)\right\}, \quad \text{for some} \quad a>2, \tag{4.3}$$

and $P_\lambda'(0+) = \lambda$. Let $\rho(t, \lambda) = \lambda^{-1} P_\lambda(t)$, and write it as $\rho(t)$ for simplicity, then the nonconvex regularized Dantzig selector is the minimizer of the following problem,

$$\begin{aligned} \min_{\boldsymbol{\beta} \in \mathcal{R}^p} \quad & \rho(|\boldsymbol{\beta}|) \\ \text{s.t} \quad & |\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})| \leq n\lambda\rho'(|\hat{\beta}_j|), \ j = 1, \cdots, p. \end{aligned} \tag{4.4}$$

Folded concave penalty has better performance than Lasso since it carries no estimation bias and achieves model selection consistency without requiring the irrepresentable condition. Nonetheless, the nonconvexity of the penalty term poses great challenges for the computation as there are multiple local optimums. [14] proposes a feasible solution through local linear approximation (LLA). It approximates a nonconvex penalized problem with a series of adaptive Lasso problems. Specifically, the local linear approximation to the penalty is given by

$$P_\lambda(|\beta_j|) \approx P_\lambda(|\beta_j^{(0)}|) + P_\lambda'(|\beta_j^{(0)}|)(|\beta_j| - |\beta_j^{(0)}|), \text{ for } \beta_j \approx \beta_j^{(0)}, \tag{4.5}$$

where $\beta_j^{(0)}$ is a given initial value. Consider a folded concave penalized estimation problem,

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) + P_\lambda(|\boldsymbol{\beta}|), \tag{4.6}$$

then LLA algorithm iteratively updates $\boldsymbol{\beta}^{(k+1)}$ as

$$\boldsymbol{\beta}^{(k+1)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) + n \sum_{j=1}^{p} P_\lambda'(|\beta_j^{(k)}|)|\beta_j| \right\}, \tag{4.7}$$

where $L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X})$ is a convex loss function. LLA shares nice properties of Lasso in terms of computational efficiency. [14] proposes to use the properly initialized one-step LLA estimator instead of a computationally more intensive fully converged

estimator, and proves that the one-step LLA sparse estimator enjoys the oracle property when $\frac{1}{n}\boldsymbol{X}^T\boldsymbol{X} \to \boldsymbol{C}$ with a positive definite $\boldsymbol{C}$. [50] generalizes this study to high dimensional settings and shows that if the LLA algorithm is initialized by Lasso solution, it produces an oracle solution with overwhelming probability.

We will establish the strong oracle property of the one-step LLA estimator obtained by solving the following problem,

$$\begin{aligned} \min_{\boldsymbol{\beta}\in R^p} \quad & \sum_{j=1,\cdots,p} \rho'(|\hat{\beta}_j^{(0)}|)|\beta_j| \\ \text{s.t} \quad & |\boldsymbol{X}_j^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})| \le n\lambda \cdot \rho'(|\hat{\beta}_j^{(0)}|),\ j=1,\cdots,p. \end{aligned} \tag{4.8}$$

While one-step LLA alleviates the computational cost induced by the nonconvexity of the objective function, the high dimensionality of the data is also problematic for efficient computation. A standard approach to solving (4.2) is linear programming (LP), which is attractable when compared to quadratic programs for solving Lasso. One LP formulation for Dantzig selector is as follows,

$$\begin{aligned} \min_{\boldsymbol{u}_1,\boldsymbol{u}_2,\boldsymbol{u}} \quad & \mathbf{1}^T(\boldsymbol{u}_1+\boldsymbol{u}_2) \\ \text{s.t.} \quad & \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}_1 - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}_2 = \boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{u} \\ & \boldsymbol{u}_1, \boldsymbol{u}_2 \ge \mathbf{0}, \|\boldsymbol{u}\|_\infty \le \lambda. \end{aligned} \tag{4.9}$$

Any mature linear programming solver can be applied to solve (4.9). In our study, we compare with one popular CRAN package *lpSolve* [72]. As is shown, the main computational duties consist of handling a $p \times p$ matrix $\boldsymbol{X}^T\boldsymbol{X}$. Thus the computation time will increase rapidly with larger constraint matrix $\boldsymbol{X}^T\boldsymbol{X}$. While LP solvers work well with moderate data dimension, their efficiency is compromised by increasing data dimension. To address the numerical challenge of large

p, we propose an alternative approach for solving (4.4) through alternating direction method of multiplier (ADMM). In particular, we adopt multi-block ADMM algorithm, which is compatible with feature space split and parallel computing.

To facilitate the presentation, the following notation is used. For $\boldsymbol{M} = (m_{ij})_{s\times t}$, $\|\boldsymbol{M}\|_{\max} = \max_{(i,j)} |m_{ij}|$ is the entry-wise maximum absolute value, and $\|\boldsymbol{M}\|_{\min} = \min_{(i,j)} |m_{ij}|$ is the entry-wise minimum absolute value. Let $\boldsymbol{e}_j \in R^p$ be the standard orthonormal basis where all entries are zero except for position j. Denote $\lambda_{\min}(\boldsymbol{M})$ and $\lambda_{\max}(\boldsymbol{M})$ to be the smallest and largest eigenvalues of $\boldsymbol{M}$, respectively. Denote $\boldsymbol{\Sigma}_{S,T}$ as the submatrix of $\boldsymbol{\Sigma}$ with rows and columns indexed by S and T respectively. Similarly, we denote $\boldsymbol{\Sigma}_{\cdot A}$ the submatrix containing the columns of $\boldsymbol{\Sigma}$ indexed by the set A. Define the sign map as $\text{sign}(\cdot) : R^p \to \{-1, 0, 1\}^p, \forall p \geq 0$ such that $\text{sign}(\boldsymbol{\beta}) = (\text{sign}(\beta_1), \cdots, \text{sign}(\beta_p))$.

4.1 Nonconvex Dantzig Selector

In this section, we verify the strong oracle property of the estimator obtained by one-step LLA algorithm applied to problem (4.8). Denote the initial estimator as $\hat{\boldsymbol{\beta}}^{(0)} = (\hat{\beta}_0^{(0)}, \cdots, \hat{\beta}_p^{(0)})^T$. The oracle estimator for linear regression is defined as

$$\hat{\boldsymbol{\beta}}^{oracle} = \underset{\boldsymbol{\beta}:\boldsymbol{\beta}_{\mathcal{A}^c}=\boldsymbol{0}}{\text{argmin}}\ L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}), \tag{4.10}$$

where $L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X})$ is the least squares loss and without confusion we write it as $L(\boldsymbol{\beta})$ for brevity. Throughout the analysis, we assume that (4.10) is regular such that the oracle solution is unique,

$$\nabla_j L(\hat{\boldsymbol{\beta}}^{oracle}) = \frac{1}{n}\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}^{oracle}) = 0,\ \forall j \in \mathcal{A}. \tag{4.11}$$

Without loss of generality, we assume the data is normalized in the following analysis, namely,

$$\mathbf{1}_n^T \boldsymbol{X}_j = 0,\ \frac{1}{n}\boldsymbol{X}_j^T \boldsymbol{X}_j = 1,\ \text{for } j = 1, \cdots, p. \tag{4.12}$$

We impose the following regularity condition on the linear regression model (4.1):

(A1) The error term $\boldsymbol{\varepsilon} = (\varepsilon_1, \cdots, \varepsilon_n)^T$ are i.i.d sub-Gaussian variables for a fixed constant σ, i.e., $E(e^{\lambda \epsilon_i^2}) \leq \exp(\frac{\lambda^2 \sigma^2}{2}), \forall \lambda \in R$.

(A2) $\lambda_{\min} = \lambda_{\min}(\frac{1}{n}\boldsymbol{X}_{\mathcal{A}}^T \boldsymbol{X}_{\mathcal{A}}) > C$ for some positive constant C.

Next we provide conditions for the one-step LLA algorithm to find the oracle estimator in the nonconvex penalized Dantzig selector and derive the non-asymptotic bounds of $P_i, i = 1, 2, 3$. To bound P_1, we will use the standard Dantzig selector $\hat{\boldsymbol{\beta}}^D$ to initialize LLA and assume that the restricted eigenvalue condition introduced in [71] holds, namely, $\kappa_D = \min_{J_0 \subset \{1,\cdots,Q\};\, |J_0| \leq q} \min_{\boldsymbol{\delta} \neq 0;\, \|\boldsymbol{\delta}_{J_0^c}\|_1 \leq \|\boldsymbol{\delta}_{J_0}\|_1} \frac{\|\boldsymbol{X}\boldsymbol{\delta}\|_2^2}{n\|\boldsymbol{\delta}_{J_{01}}\|_2^2} > 0$. Then it follows that with probability $1 - c \cdot \exp(-\frac{n\lambda_D^2}{2\sigma^2})$,

$$\|\hat{\boldsymbol{\beta}}^D - \boldsymbol{\beta}^*\|_2^2 \leq \frac{16}{\kappa_D^2} q \lambda_D^2, \tag{4.13}$$

where λ_D is the regularization parameter used in the standard Dantzig Selector. With (4.13) serving as an upper bound of $\|\hat{\boldsymbol{\beta}}^D - \boldsymbol{\beta}^*\|_{\max}$, we can obtain the following theorem. The proof of Theorem 4.1 is given in the Appendix 4.5.

Theorem 4.1. *Consider the linear regression model* (4.1) *with* $\|\boldsymbol{\beta}_{\mathcal{A}}^*\|_{\min} > (a+1)\lambda$. *Under conditions (A1) and (A2), the LLA algorithm initialized by* $\hat{\boldsymbol{\beta}}^D$ *converges to* $\hat{\boldsymbol{\beta}}^{oracle}$ *after one iteration with probability* $1 - P_1 - P_2 - P_3$, *where* P_i *are the probabilities of events* $E_i, i = 1, 2, 3$ *and each* E_i *is defined as follows:*

- $E_1 = \{\|\hat{\boldsymbol{\beta}}^D - \boldsymbol{\beta}^*\|_{\max} > a_0\lambda\}$, $a_0 = \min\{1, a_2\}$;
- $E_2 = \{\|\hat{\boldsymbol{\beta}}^{oracle}_{\mathcal{A}}\|_{\min} \leq a\lambda\}$;
- $E_3 = \{\|\nabla_{\mathcal{A}^c} L(\hat{\boldsymbol{\beta}}^{oracle})\|_{\max} > a_1\lambda\}$.

Furthermore, if $\log(p) = o(n\lambda^2)$ *and* $\lambda > \frac{4}{a_0\kappa_D} q^{1/2}\lambda_D$, *then we have* $P_i \to 0, i = 1, 2, 3$ *as* $n \to \infty$.

LLA algorithm addresses the computational challenge coming from the non-convex penalty. It remains to resolve another challenge inherited from the high dimensional data. The classic LP solver needs to handle matrix $\boldsymbol{X}^T\boldsymbol{X} \in R^{p\times p}$, which demands intensive memory when p is relatively large. In the next section, we propose an efficient algorithm based on feature-split. The proposed algorithm not only reduces the memory usage but facilitates a parallel implementation of the computational framework.

4.2 3-block ADMM Algorithm

Assisted by one-step LLA, finding nonconvex Dantzig selector boils down to the computation of weighted Dantzig selector,

$$\begin{aligned} \min_{\boldsymbol{\beta}\in\mathcal{R}^p} \quad & \|\boldsymbol{\alpha}\circ\boldsymbol{\beta}\|_1 \\ \text{s.t.} \quad & |\boldsymbol{X}_j^T(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})| \leq n\cdot\lambda\alpha_j, \quad j = 1, \cdots, p, \end{aligned} \tag{4.14}$$

where $\alpha_j = \rho'(|\hat{\beta}_j^{(0)}|)$ and $\boldsymbol{\alpha}\circ\boldsymbol{\beta} = \sum_{j=1}^p \alpha_j\beta_j$ is the Hadamard product. In this section, we briefly review the alternating direction method of multiplier (ADMM) and derive the algorithm for finding the weighted Dantzig selector.

4.2.1 Review of ADMM

The research of ADMM dates back to [15] and [16]. Recently, it receives enormous attention thanks to its application to massive optimization problems. ADMM has a wide range of applications in many statistical and machine learning problems. The classic 2-block ADMM solves the problem of the following form,

$$\begin{aligned} &\min_{\boldsymbol{v}_1,\boldsymbol{v}_2} f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) \\ &\text{s.t. } \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 = \boldsymbol{c}, \end{aligned} \tag{4.15}$$

with parameters $\boldsymbol{v}_1 \in \mathcal{R}^{p_1}$ and $\boldsymbol{v}_2 \in \mathcal{R}^{p_2}$, where $\boldsymbol{A}_1 \in \mathcal{R}^{n\times p_1}$, $\boldsymbol{A}_2 \in \mathcal{R}^{n\times p_2}$, $\boldsymbol{c} \in \mathcal{R}^n$ and $f_i : \mathcal{R}^{p_i} \to \mathcal{R}, i = 1, 2$ are convex functions. For primal variables, ADMM iteratively minimizes the augmented Lagrangian function defined as

$$\mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{\gamma}) = f_1(\boldsymbol{v}_1) + f_2(\boldsymbol{v}_2) + \langle \boldsymbol{\gamma}, \boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\rangle + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2 - \boldsymbol{c}\|_2^2,$$

where $\boldsymbol{\gamma} \in \mathcal{R}^n$ is the dual variable and $\phi > 0$ is the parameter for the quadratic penalty. Given an initial point $(\boldsymbol{v}_1^0, \boldsymbol{v}_2^0, \boldsymbol{\gamma}^0)$, the classic iterative scheme for solving (4.15) is

$$\begin{aligned} \boldsymbol{v}_1^{k+1} &= \operatorname*{argmin}_{\boldsymbol{v}_1} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2^k; \boldsymbol{\gamma}^k) \\ \boldsymbol{v}_2^{k+1} &= \operatorname*{argmin}_{\boldsymbol{v}_2} \mathcal{L}_\phi(\boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2; \boldsymbol{\gamma}^k) \\ \boldsymbol{\gamma}^{k+1} &= \boldsymbol{\gamma}^k + \theta\phi(\boldsymbol{A}_1\boldsymbol{v}_1^{k+1} + \boldsymbol{A}_2\boldsymbol{v}_2^{k+1} - \boldsymbol{c}), \end{aligned} \tag{4.16}$$

where θ is a parameter controlling the step size. Convergence theory has been established when θ is restricted to $(0, \frac{1+\sqrt{5}}{2})$. Since a larger θ leads to faster convergence, a common practice is to set $\theta = \frac{1+\sqrt{5}}{2}$. The advantage of including the quadratic penalty term is that the algorithm can convergence under far more

general conditions, e.g., when f_1 or f_2 takes $+\infty$ or is not strictly convex.

The success of ADMM algorithm heavily depends on the solvability of subproblems, i.e. either subproblems have analytic solutions or can be solved efficiently by numerical algorithms. In the latter case, the generated sequences still converge if the numerical solutions are sufficiently accurate. In literature, proximal version of ADMM has been used widely to make the subproblems easier to solve. The proximal version of ADMM adds a proximal term to the subproblem. For instance, in the $\boldsymbol{v}_1$ update,

$$\begin{aligned} \boldsymbol{v}_1^{k+1} &= \underset{\boldsymbol{v}_1}{\operatorname{argmin}} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2^k; \boldsymbol{\gamma}^k) \\ &= \underset{\boldsymbol{v}_1}{\operatorname{argmin}} f_1(\boldsymbol{v}_1) + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{v}_1 + \boldsymbol{A}_2\boldsymbol{v}_2^k - \boldsymbol{c} + \frac{\boldsymbol{\gamma}^k}{\phi}\|_2^2, \end{aligned} \tag{4.17}$$

if matrix $\boldsymbol{A}_1$ is singular, then $\boldsymbol{v}_1^{k+1}$ has no analytic solution. To facilitate computation, we regularize (4.17) by adding a proximal term,

$$\boldsymbol{v}_1^{k+1} = \underset{\boldsymbol{v}_1}{\operatorname{argmin}} \mathcal{L}_\phi(\boldsymbol{v}_1, \boldsymbol{v}_2^k; \boldsymbol{\gamma}^k) + \frac{1}{2}\|\boldsymbol{v}_1 - \boldsymbol{v}_1^k\|_{\boldsymbol{S}}^2, \tag{4.18}$$

where $\boldsymbol{S}$ is a positive semidefinite matrix. $\boldsymbol{S}$ is chosen to be as small as possible while the subproblem (4.18) is relatively easy to solve. A reasonable choice of $\boldsymbol{S}$ is $\eta\boldsymbol{I} - \phi\boldsymbol{A}_1^T\boldsymbol{A}_1$, which essentially linearizes the subproblem as

$$\boldsymbol{v}_1^{k+1} = \underset{\boldsymbol{v}_1}{\operatorname{argmin}} f_1(\boldsymbol{v}_1) + \frac{\eta}{2}\|\boldsymbol{v}_1 - \boldsymbol{v}_1^k + \frac{\phi}{\eta}\boldsymbol{A}_1^T\Big(\boldsymbol{A}_1\boldsymbol{v}_1^k + \boldsymbol{A}_2\boldsymbol{v}_2^k - \boldsymbol{c} + \frac{\boldsymbol{\gamma}^k}{\phi}\Big)\|_2^2. \tag{4.19}$$

The convergence of proximal ADMM requires that $\boldsymbol{S}$ to be positive definite, which indicates that η should be greater than the largest eigenvalue of $\phi\boldsymbol{A}_1^T\boldsymbol{A}_1$. Large η can cause over-regularization and reduce the step size of the update, so we take

$\eta = \lambda_{\max}(\phi \boldsymbol{A}_1^T \boldsymbol{A}_1)$ in our experiments. To accelerate the linearized ADMM, [39] relaxes the positive definiteness requirement and establishes the global convergence of the new linearized ADMM when η is replaced by $\tau\eta$ with τ being as small as 0.75. Nevertheless, this relaxation is at the cost of restricting $\theta = 1$, which limits its improvement on the convergence speed.

Multi-block optimization problem is a natural extension of (4.15),

$$\min_{\boldsymbol{v}_1,\cdots,\boldsymbol{v}_b} \left\{ \sum_{i=1}^{b} f_i(\boldsymbol{v}_i) \mid \sum_{i=1}^{b} \boldsymbol{A}_i \boldsymbol{v}_i = \boldsymbol{c} \right\}, \quad b \geq 3. \tag{4.20}$$

While the convergence of the classic 2-block ADMM has been well studied in literature, it was shown by [29] that the direct extension of the procedure (4.16) to multi-block ADMM is not necessarily convergent. To resolve this issue, a number of studies have been conducted mainly in two strands. One strand of literature establishes convergence theory by imposing additional conditions on f_i, and the other strand explores various modifications on the iterative scheme to ensure convergence.

4.2.2 Proposed Algorithm

The computational competitiveness of the proposed algorithm leans on its compatibility with feature partition and distributed computing. Suppose that we can partition the regression parameters $\boldsymbol{\beta}$ into K blocks, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \cdots, \boldsymbol{\beta}_K^T)^T$ with $\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}$, where $\sum_{i=1}^{K} p_i = p$. Then correspondingly partition the gram matrix $\boldsymbol{A} = \boldsymbol{X}^T \boldsymbol{X}$ as $\boldsymbol{A} = (\boldsymbol{A}_1, \cdots \boldsymbol{A}_K)$ with $\boldsymbol{A}_i \in \mathcal{R}^{p \times p_i}$. Now (4.14) is equivalent to the

following 2-block ADMM problem,

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{z}} \sum_{i=1}^{K} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \delta_{\mathcal{Z}_0}(\boldsymbol{z}) \\ &\text{s.t. } \sum_{i=1}^{K} \boldsymbol{A}_i \boldsymbol{\beta}_i - \boldsymbol{z} = \boldsymbol{X}^T \boldsymbol{y}, \end{aligned} \tag{4.21}$$

where $\boldsymbol{z}$ is an auxiliary variable, $\mathcal{Z}_0 = \{\boldsymbol{z} : \|\boldsymbol{z}\|_\infty \leq n \cdot \lambda \alpha_j\}$ and

$$\delta_{\mathcal{Z}_0}(\boldsymbol{z}) = \begin{cases} 0, & \text{if } \boldsymbol{z} \in \mathcal{Z}_0 \\ +\infty, & \text{otherwise.} \end{cases}$$

(4.21) can be recast into a 3-block optimization problem by introducing the third block variable $\boldsymbol{\omega} = (\boldsymbol{\omega}_2, \cdots \boldsymbol{\omega}_K), \boldsymbol{\omega}_i \in \mathcal{R}^p, i = 2, \cdots, K$ into the formulation (4.21).

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{z},\boldsymbol{\omega}} \sum_{i=1}^{K} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \delta_{\mathcal{Z}_0}(\boldsymbol{z}) \\ &\text{s.t. } \boldsymbol{A}_1 \boldsymbol{\beta}_1 + \boldsymbol{\omega}_2 + \cdots + \boldsymbol{\omega}_K - \boldsymbol{z} = \boldsymbol{X}^T \boldsymbol{y}, \\ &\quad \boldsymbol{\omega}_i = \boldsymbol{A}_i \boldsymbol{\beta}_i, \ i = 2, \cdots, K. \end{aligned} \tag{4.22}$$

To solve problem (4.22), rather than applying the directly extended multi-block ADMM, we adopt the following procefure as recommended by [37],

$$\begin{cases} \boldsymbol{\beta}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}, \boldsymbol{z}^k, \boldsymbol{\omega}^k; \gamma^k), \\ \boldsymbol{\omega}^{k+\frac{1}{2}} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\omega}; \gamma^k), \\ \boldsymbol{z}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}, \boldsymbol{\omega}^{k+\frac{1}{2}}; \gamma^k), \\ \boldsymbol{\omega}^{k+1} & = \operatorname{argmin} \mathcal{L}_\phi(\boldsymbol{\beta}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{\omega}; \gamma^k), \\ \boldsymbol{\gamma}_1^{k+1} & = \boldsymbol{\gamma}_1^k + \theta\phi(\boldsymbol{X}_1\boldsymbol{\beta}_1^{k+1} + \boldsymbol{z}^k + \sum_{i=2}^K \boldsymbol{\omega}_i^{k+1} - \boldsymbol{y}), \\ \boldsymbol{\gamma}_i^{k+1} & = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{X}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 2, \cdots, K. \end{cases} \tag{4.23}$$

Clearly, (4.23) differs from the classic ADMM update scheme in that it updates the third block $\boldsymbol{\omega}$ twice before and after the update of $\boldsymbol{z}$. Next we show how to update each variable step by step. The $\boldsymbol{\beta}$-subproblems in (4.23) are given by

$$\begin{aligned} \boldsymbol{\beta}_1^{k+1} = \underset{\boldsymbol{\beta}_1}{\operatorname{argmin}} \quad & \Big\{ \|\boldsymbol{\alpha}_1 \circ \boldsymbol{\beta}_1\|_1 + \langle \boldsymbol{\gamma}_1, \boldsymbol{A}_1\boldsymbol{\beta}_1^k + \sum_{i=2}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \boldsymbol{X}^T\boldsymbol{y} \rangle \\ & \qquad + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{\beta}_1 + \sum_{i=2}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \boldsymbol{X}^T\boldsymbol{y}\|_2^2 \Big\} \\ \boldsymbol{\beta}_i^{k+1} = \underset{\boldsymbol{\beta}_i}{\operatorname{argmin}} \quad & \Big\{ \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \langle \boldsymbol{\gamma}_1^k, \boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k \rangle + \frac{\phi}{2}\|\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k\|_2^2 \Big\}, \end{aligned}$$

which are equivalent to

$$\boldsymbol{\beta}_1^{k+1} = \underset{\boldsymbol{\beta}_1 \in \mathcal{R}^{p_1}}{\operatorname{argmin}} \|\boldsymbol{\alpha}_1 \circ \boldsymbol{\beta}_1\|_1 + \frac{\phi}{2}\|\boldsymbol{A}_1\boldsymbol{\beta}_1 + \sum_{i=1}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \boldsymbol{X}^T\boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\|_2^2, \tag{4.24}$$

$$\boldsymbol{\beta}_i^{k+1} = \underset{\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}}{\operatorname{argmin}} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{\phi}{2}\|\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2, \ i = 2, \cdots, K. \tag{4.25}$$

Since (4.24) and (4.25) are essentially ℓ_1-penalized least squares problems, they can be efficiently solved by coordinate descent. We call the resulting algorithm *ADMM-CD*. On the other hand, for high dimensional data, $\boldsymbol{A}_i^T\boldsymbol{A}_i, i = 1, \cdots, K$ are not necessarily positive definite, which will inhibit the theoretical convergence analysis discussed later. Like Chapter 3, we add a proximal term $\boldsymbol{\mathcal{T}_i} = \eta_i \boldsymbol{I}_{p_i} - \phi \boldsymbol{A}_i^T\boldsymbol{A}_i$ with $\eta_i > \phi\lambda_{\max}(\boldsymbol{A}_i^T\boldsymbol{A}_i)$ to $\boldsymbol{\beta}$ subproblems, which leads to the following updates,

$$\begin{aligned}\boldsymbol{\beta}_1^{k+1} &= \underset{\boldsymbol{\beta}_1\in\mathcal{R}^{p_1}}{\operatorname{argmin}} \quad \|\boldsymbol{\alpha}_1 \circ \boldsymbol{\beta}_1\|_1 + \frac{\eta_1}{2}\Big\|\boldsymbol{\beta}_1 - \boldsymbol{\beta}_1^k \\ &\qquad + \frac{\phi\boldsymbol{A}_1^T\big(\boldsymbol{A}_1\boldsymbol{\beta}_1 + \sum_{i=1}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \boldsymbol{X}^T\boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\big)}{\eta_1}\Big\|_2^2 \\ &= \text{Shrink} \quad \Big(\boldsymbol{\beta}_{1j}^k - \frac{\phi}{\eta_1}\boldsymbol{A}_{1j}^T(\boldsymbol{A}_1\boldsymbol{\beta}_1 + \sum_{i=1}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \boldsymbol{X}^T\boldsymbol{y} + \frac{\boldsymbol{\gamma}_1^k}{\phi}), \frac{\boldsymbol{\alpha}_1}{\eta_1}\Big)_{j=1,\cdots,p_1}\end{aligned} \tag{4.26}$$

$$\begin{aligned}\boldsymbol{\beta}_i^{k+1} &= \underset{\boldsymbol{\beta}_i\in\mathcal{R}^{p_i}}{\operatorname{argmin}} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \frac{\eta_i}{2}\|\boldsymbol{\beta}_i - \frac{\eta_i\boldsymbol{\beta}_i^k - \phi\boldsymbol{A}_i^T(\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi})}{\eta_i})\|_2^2 \\ &= \text{Shrink}\Big(\boldsymbol{\beta}_{ij}^k - \frac{\phi}{\eta_i}\boldsymbol{A}_{ij}^T(\boldsymbol{A}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}), \frac{\boldsymbol{\alpha}_i}{\eta_i}\Big)_{j=1,\cdots,p_i},\end{aligned} \tag{4.27}$$

where $\text{Shrink}(x, t) = \text{sign}(x)(|x| - t)I(|x| > t)$ is the soft thresholding function. The positive definiteness of $\boldsymbol{\mathcal{T}_i}$ will make $\{\boldsymbol{\beta}^k\}$ automatically well defined. One benefit of splitting feature space is that η_i are relatively small compared to the "un-splitted" η, which satisfies $\eta > \phi\lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})$. With a large η, the step size for the update (i.e., $\frac{1}{\eta}$) is relatively small and slows down the algorithm convergence. Indeed, η can be rather large for high dimensional data. We use *ADMM-prox* to denote the algorithm with added proximal term. The efficiency of the proposed algorithm also relies on the extra cost incurred by updating $\boldsymbol{\omega}$, which is inconsid-

erable in this instance as $\boldsymbol{\omega}$-subproblem endows an easy update as follows,

$$\boldsymbol{\omega}_i^{k+\frac{1}{2}} = \frac{1}{K}(\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{z}^k + K\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{A}\boldsymbol{\beta}^{k+1}),\ i = 2, \cdots, K, \tag{4.28}$$

$$\boldsymbol{\omega}_i^{k+1} = \frac{1}{K}(\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{z}^{k+1} + K\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{A}\boldsymbol{\beta}^{k+1}),\ i = 2, \cdots, K. \tag{4.29}$$

Moreover, the $\boldsymbol{z}$ subproblem also admits a closed-form solution,

$$\boldsymbol{z}^{k+1} = \min\left(\max\left(\boldsymbol{A}_1\boldsymbol{\beta}_1^{k+1} + \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \boldsymbol{X}^T\boldsymbol{y} - \frac{\boldsymbol{\gamma}_1^k}{\phi}, -n\lambda\boldsymbol{\alpha}\right), n\lambda\boldsymbol{\alpha}\right). \tag{4.30}$$

Finally, $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_i$ are updated via gradient ascent,

$$\boldsymbol{\gamma}_1^{k+1} = \boldsymbol{\gamma}_1^k + \theta\phi(\boldsymbol{A}_1\boldsymbol{\beta}_1^{k+1} + \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+1} - \boldsymbol{z}^{k+1} - \boldsymbol{X}^T\boldsymbol{y}), \tag{4.31}$$

$$\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\boldsymbol{A}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}),\ i = 2, \cdots, K. \tag{4.32}$$

We can see that by introducing slack variables $\boldsymbol{\omega}$, the K sub-blocks of $\boldsymbol{\beta}$ are separable in the augmented Lagrangian function, which allows a natural parallelization for updates of $\boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_K$. Since $\boldsymbol{\beta}$-subproblems often play a dominant role in regard to the computational cost, distributed computing of $\boldsymbol{\beta}$ can significantly accelerate the computation. We summarize ADMM-CD and ADMM-prox in Algorithm 4.12 and Algorithm 4.13, respectively.

Algorithm 4.12 ADMM-CD for weighted Dantzig selector

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.
  while the stopping criterion is not satisfied, **do**
    Compute $\boldsymbol{\beta}^{k+1}$ by (4.24) and (4.25) through coordinate descent.
    Compute $\boldsymbol{\omega}_i^{k+\frac{1}{2}}$ by (4.29), $\boldsymbol{z}^{k+1}$ by (4.30) and then $\boldsymbol{\omega}_i^{k+1}$ by (4.29).
    Compute $\boldsymbol{\gamma}^{k+1}$ by (4.31) and (4.32)
  end while

Algorithm 4.13 ADMM-prox for weighted Dantzig selector

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.
  while the stopping criterion is not satisfied, **do**
    Compute $\boldsymbol{\beta}^{k+1}$ by (4.26) and (4.27) through coordinate descent.
    Compute $\boldsymbol{\omega}_i^{k+\frac{1}{2}}$ by (4.29), $\boldsymbol{z}^{k+1}$ by (4.30) and then $\boldsymbol{\omega}_i^{k+1}$ by (4.29).
    Compute $\boldsymbol{\gamma}^{k+1}$ by (4.31) and (4.32)
  end while

The algorithm for nonconvex Dantzig Selector can be derived based on Algorithm 4.12 and 4.13 and we summarize it in Algorithm 4.14.

Algorithm 4.14 Algorithm for Nonconvex Dantzig Selector

Initialization: $\tilde{\boldsymbol{\beta}}^{(0)}, \lambda, \upsilon, \tilde{\boldsymbol{z}}^{(0)}, \tilde{\boldsymbol{\gamma}}^{(0)}, \tilde{\boldsymbol{\omega}}_i^{(0)}$, and $\phi > 0, \theta = 1.618, k = 0$.
  Set $\alpha_j = 1$ for $j = 1, \cdots, p$
  while the stopping criterion is not satisfied, **do**
    Compute $\tilde{\boldsymbol{\beta}}^{(k+1)}$ by (4.24) and (4.25) or by (4.26) and (4.27), then compute $\tilde{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (4.28).
    Compute $\tilde{\boldsymbol{z}}^{(k+1)}$ by

$$\tilde{\boldsymbol{z}}^{k+1} = \min\left(\max\left(\boldsymbol{A}_1\boldsymbol{\beta}_1^{k+1} + \sum_{i=1}^{K}\boldsymbol{\omega}_i^{k+\frac{1}{2}} - \boldsymbol{X}^T\boldsymbol{y} - \frac{\boldsymbol{\gamma}_1^k}{\phi}, -n\upsilon\lambda\right), n\upsilon\lambda\right).$$

    Update $\tilde{\boldsymbol{\omega}}^{(k+1)}$ by (4.29), and $\tilde{\boldsymbol{\gamma}}^{(k+1)}$ by (4.31) and (4.32).
  end while The solution is denoted as $\hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{\ell_1}$
Initialization: $\hat{\boldsymbol{\beta}}^{(0)} = \hat{\boldsymbol{\beta}}^{\ell_1}, \hat{\boldsymbol{z}}^{(0)} = \hat{\boldsymbol{z}}^{\ell_1}, \hat{\boldsymbol{\omega}}^{(0)} = \hat{\boldsymbol{\omega}}^{\ell_1}$ and $\phi > 0, \theta = 1.618, k = 0$.
  Set $\alpha_j = \lambda^{-1}P_\lambda'(|\hat{\beta}_j^{\ell_1}|)$ for $j = 1, \cdots, p$.
  while the stopping criterion is not satisfied, **do**
    Compute $\tilde{\boldsymbol{\beta}}^{(k+1)}$ by (4.24) and (4.25) or by (4.26) and (4.27), then compute $\tilde{\boldsymbol{\omega}}^{(k+\frac{1}{2})}$ by (4.28).
    Update $\hat{\boldsymbol{z}}^{(k+1)}$ by (4.30).
    Update $\hat{\boldsymbol{\omega}}^{(k+1)}$ by (4.29), then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by (4.31) and (4.32).
  end while

4.2.3 Linear Rate of Convergence

In this section, we present the rate of convergence analysis of the proposed algorithm for solving weighted Dantzig selector. Note that since a nontrivial proximal term is necessary in establishing the theory, we only study the convergence of Algorithm 4.13. Throughout the analysis, we make the following assumption,

(C1) The feasible set $M = \{\boldsymbol{\beta} : |\boldsymbol{X}_j^T(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})| \leq n \cdot \lambda\alpha_j, \quad j = 1, \cdots, p\}$ is not empty and problem (4.14) has one unique solution.

Intuitively, (C1) will hold given that the feasible set M is not parallel to the closed L_1 unit ball of $\|\boldsymbol{\alpha} \circ \boldsymbol{\beta}\|_1$ and λ is properly chosen. We present the result in Theorem 4.2. The proof for Theorem 4.2 is a direct application of Lemma 3.4, 3.5 and 3.6.

Theorem 4.2. *Suppose that condition (C1) holds. If $\theta \in (0, (1+\sqrt{5})/2)$, then sequence $(\boldsymbol{\beta}^k, \boldsymbol{z}^k, \boldsymbol{\omega}^k, \boldsymbol{\gamma}^k)$ generated by Algorithm 4.13 converges in probability to a unique limit $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\gamma}})$ such that $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\omega}})$ is the primal optimal of (4.22) and $\bar{\boldsymbol{\gamma}}$ is the dual optimal. Moreover, there exists a constant $\mu \in (0, 1)$ such that*

$$Dist^{k+1} \leq \mu Dist^k,$$

where the distance function $Dist^k$ at iteration k is defined as

$$\begin{aligned} Dist^k = &\sum_{i=1}^{K} \|\boldsymbol{A}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i)\|_2^2 - \frac{1}{K}\|\sum_{i=1}^{K} \boldsymbol{A}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i)\|_2^2 \\ &+ \sum_{i=1}^{K} \|\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i\|_{\boldsymbol{\tau}_i}^2 + \|\boldsymbol{z}^k - \bar{\boldsymbol{z}}\|_2^2 + \frac{K-1}{K}\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|_2^2 \\ &+ \frac{m_1}{K}\|\sum_{i=1}^{K} \boldsymbol{A}_i(\boldsymbol{\beta}_i^k - \bar{\boldsymbol{\beta}}_i) + (\boldsymbol{z}^k - \bar{\boldsymbol{z}})\|_2^2, \end{aligned}$$

where $m_1 = 1 + d_1 - d_1\theta - (1 - d_1)\min\{\theta, \frac{1}{\theta}\}$ and $d_1 \in (0, \frac{1}{2})$.

4.3 Application to the Linear Programming Discriminant Rule

In this section, we apply the proposed procedure to the Linear Programming Discriminant (LPD) Rule. LPD is an estimation procedure for linear discriminant analysis (LDA) proposed in [73]. Consider two p-dimensional normal distributions $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ (class 1) and $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ (class 2) with a shared covariance matrix $\boldsymbol{\Sigma}$. Let $\{(\boldsymbol{x}_i, y_i), i = 1, \cdots, n\}$ be n samples from a joint distribution of $(\boldsymbol{X}, Y) \in \mathcal{R}^p \times \{0, 1\}$, where $\boldsymbol{X}$ is distributed as class 1 if $Y = 0$, and as class 2 if $Y = 1$. The problem of interest is to determine the associated value of Y given a data $\boldsymbol{x} \in \mathcal{R}^p$. Denote $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ to be the precision matrix, $\boldsymbol{\mu} = (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2$ and $\boldsymbol{\delta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$. Assuming equal prior probabilities of the two class, the Fisher's linear discriminant rule

$$\zeta(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = I\{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Omega} \boldsymbol{\delta} \geq 0\} \tag{4.33}$$

classifies $Y = 0$ if and only if $\zeta(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = 1$. Given n_1 samples of $(\boldsymbol{X}|Y = 0)$ and n_2 samples of $(\boldsymbol{X}|Y = 1)$ respectively, a natural way to estimate $\boldsymbol{\Omega}$ and $\boldsymbol{\delta}$ is to use sample inverse covariance matrix $\hat{\boldsymbol{\Sigma}}^{-1}$ and sample means $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2$. Plugging these estimates into (4.33) leads to an empirical classifier $\hat{\zeta}(x; \hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2, \hat{\boldsymbol{\Sigma}})$. However, when $p \gg n$, the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ is singular and its inverse is not well-defined. To ensure the consistent estimation on $\boldsymbol{\Omega}$ and $\boldsymbol{\delta}$, one often needs to assume that both of them are sparse. An alternative solution is provided by LPD, which

estimates the product $\boldsymbol{\Omega\delta}$ directly through constrained ℓ_1 minimization,

$$\begin{aligned} &\min_{\boldsymbol{\beta}\in\mathcal{R}^p} \|\boldsymbol{\beta}\|_1 \\ &\text{s.t. } \|\hat{\boldsymbol{\Sigma}}_n\boldsymbol{\beta} - (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)\|_\infty \le \lambda_n, \end{aligned} \tag{4.34}$$

where λ_n is a tuning parameter. In this formulation, the sparsity assumption is imposed on the product $\boldsymbol{\beta} := \boldsymbol{\Omega\delta}$ instead of on $\boldsymbol{\delta}$ and $\boldsymbol{\Omega}$ individually.

We generalize problem (4.34) to the case where $\|\boldsymbol{\beta}\|_1$ is replaced by nonconvex penalty $P_\lambda(|\boldsymbol{\beta}|)$, and obtain the nonconvex LPD as follows,

$$\begin{aligned} &\min_{\boldsymbol{\beta}\in\mathcal{R}^p} \rho(|\boldsymbol{\beta}|) \\ &\text{s.t. } \ |\hat{\boldsymbol{\Sigma}}_j^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le n\cdot\lambda\rho'(|\boldsymbol{\beta}_j|), \quad j = 1, \cdots, p, \end{aligned} \tag{4.35}$$

where $\hat{\boldsymbol{\Sigma}}_j$ is the j-th row of $\hat{\boldsymbol{\Sigma}}_n$ and $\hat{\mu}_{ij}$ is the j-th element of $\hat{\boldsymbol{\mu}}_i$. As discussed, LLA algorithm can redirect problem (4.35) to an adaptive version of problem (4.34), namely,

$$\begin{aligned} &\min_{\boldsymbol{\beta}\in\mathcal{R}^p} \sum_{j=1}^{p} \rho'(|\hat{\beta}_j^{(0)}|)|\beta_j| \\ &\text{s.t. } \ |\hat{\boldsymbol{\Sigma}}_j^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le n\lambda\rho'(|\hat{\beta}_j^{(0)}|), \quad j = 1, \cdots, p. \end{aligned} \tag{4.36}$$

To use LLA, we first verify its strong oracle property. Denote the between-class scatter matrix as $S_B = \hat{\boldsymbol{\delta}}\hat{\boldsymbol{\delta}}^T$, then we can express the oracle Fisher's rule as

$$\hat{\boldsymbol{\beta}}^{oracle} = \underset{\boldsymbol{\beta}:\boldsymbol{\beta}_{\mathcal{A}^c}=\mathbf{0}}{\operatorname{argmin}} L(\boldsymbol{\beta}) = \underset{\boldsymbol{\beta}:\boldsymbol{\beta}_{\mathcal{A}^c}=\mathbf{0}}{\operatorname{argmin}} -\frac{\boldsymbol{\beta}^T S_B\boldsymbol{\beta}}{\boldsymbol{\beta}^T\hat{\boldsymbol{\Sigma}}\boldsymbol{\beta}}. \tag{4.37}$$

In our analysis, we always assume that $\hat{\boldsymbol{\Sigma}}_{\mathcal{A}}^T\boldsymbol{\beta} - \hat{\boldsymbol{\delta}}_{\mathcal{A}} = 0$ is regular with one unique solution. [74] establishes the sign consistency of the standard LPD rule $\hat{\boldsymbol{\beta}}^{LPD}$ under sparsity assumption, and we use $\hat{\boldsymbol{\beta}}^{LPD}$ to initialize the LLA algorithm.

Theorem 4.3. *Under the assumptions of Theorem 2 in [74] and assume that* $\|\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\min} > (a+1)\lambda$, *then the LLA algorithm initialized by* $\hat{\boldsymbol{\beta}}^{LPD}$ *converges to* $\hat{\boldsymbol{\beta}}^{oracle}$ *after one iteration with probability* $1 - P_1 - P_2 - P_3$, *where* P_i *are the probabilities of events* $E_i, i = 1, 2, 3$ *and each* E_i *is defined as follows:*

- $E_1 = \{\|\hat{\boldsymbol{\beta}}^{LPD} - \boldsymbol{\beta}^*\|_{\max} > a_0\lambda\}$;
- $E_2 = \{\|\hat{\boldsymbol{\beta}}^{oracle}_{\mathcal{A}}\|_{\min} \leq a\lambda\}$;
- $E_3 = \{\|\hat{\boldsymbol{\Sigma}}^T_{\mathcal{A}^c}\hat{\boldsymbol{\beta}}^{oracle} - \hat{\boldsymbol{\delta}}_{\mathcal{A}^c}\|_{\max} > a_1\lambda\}$.

Furthermore, if $\log(p) = o(n\lambda^2)$, *then* $P_i \to 0, i = 1, 2, 3$ *as* $n \to \infty$.

The proof of Theorem 4.3 is given in the Appendix 4.5.

The derivation of algorithms for problem (4.36) is similar to that of Dantzig Selector and we summarize the algorithms in Algorithm 4.15 and 4.16.

Algorithm 4.15 ADMM-cd for LPD

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.

while the stopping criterion is not satisfied, **do**

Compute $\boldsymbol{\beta}^{k+1}$ by

$$\boldsymbol{\beta}_1^{k+1} = \text{argmin}_{\boldsymbol{\beta}_1 \in \mathcal{R}^{p_1}} \|\boldsymbol{\alpha}_1 \circ \boldsymbol{\beta}_1\|_1 + \tfrac{\phi}{2}\|\hat{\boldsymbol{\Sigma}}_1\boldsymbol{\beta}_1 + \textstyle\sum_{i=1}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \hat{\boldsymbol{\delta}} + \frac{\boldsymbol{\gamma}_1^k}{\phi}\|_2^2,$$

$$\boldsymbol{\beta}_i^{k+1} = \text{argmin}_{\boldsymbol{\beta}_i \in \mathcal{R}^{p_i}} \|\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i\|_1 + \tfrac{\phi}{2}\|\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \frac{\boldsymbol{\gamma}_i^k}{\phi}\|_2^2, \quad i = 2, \cdots, K.$$

Compute $\boldsymbol{\omega}_i^{k+\frac{1}{2}}$ by $\frac{1}{K}(\hat{\boldsymbol{\delta}} + \boldsymbol{z}^k + K\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i^{k+1} - \hat{\boldsymbol{\Sigma}}\boldsymbol{\beta}^{k+1}),\ i = 2, \cdots, K.$

Compute $\boldsymbol{z}^{k+1}$ by

$$\boldsymbol{z}^{k+1} = \min\left(\max\left(\hat{\boldsymbol{\Sigma}}_1\boldsymbol{\beta}_1^{k+1} + \sum_{i=1}^K \boldsymbol{\omega}_i^{k+\frac{1}{2}} - \hat{\boldsymbol{\delta}} - \frac{\boldsymbol{\gamma}_1^k}{\phi}, -\lambda\boldsymbol{\alpha}\right), \lambda\boldsymbol{\alpha}\right).$$

Compute $\boldsymbol{\omega}_i^{k+1}$ by $\frac{1}{K}(\hat{\boldsymbol{\delta}} + \boldsymbol{z}^{k+1} + K\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i^{k+1} - \hat{\boldsymbol{\Sigma}}\boldsymbol{\beta}^{k+1}),\ i = 2, \cdots, K.$

Compute $\boldsymbol{\gamma}^{k+1}$ by

$$\boldsymbol{\gamma}_1^{k+1} = \boldsymbol{\gamma}_1^k + \theta\phi(\hat{\boldsymbol{\Sigma}}_1\boldsymbol{\beta}_1^{k+1} + \textstyle\sum_{i=1}^K \boldsymbol{\omega}_i^{k+1} - \boldsymbol{z}^{k+1} - \hat{\boldsymbol{\delta}}),$$

$$\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 2, \cdots, K.$$

end while

Algorithm 4.16 ADMM-prox for LPD

Initialization: $\boldsymbol{\beta}^0, \boldsymbol{\omega}^0, \boldsymbol{z}^0, \boldsymbol{\gamma}^0$, and $\phi > 0, \theta > 0$ are given.

while the stopping criterion is not satisfied, **do**

Compute $\boldsymbol{\beta}^{k+1}$ by

$$\boldsymbol{\beta}_1^{k+1} = \text{Shrink}\Big(\boldsymbol{\beta}_{1j}^k - \tfrac{\phi}{\eta_1}\hat{\boldsymbol{\Sigma}}_{1j}^T(\hat{\boldsymbol{\Sigma}}_1\boldsymbol{\beta}_1 + \textstyle\sum_{i=1}^K \boldsymbol{\omega}_i^k - \boldsymbol{z}^k - \hat{\boldsymbol{\delta}} + \tfrac{\boldsymbol{\gamma}_1^k}{\phi}), \tfrac{\boldsymbol{\alpha}_1}{\eta_1}\Big)_{j=1,\cdots,p_1},$$

$$\boldsymbol{\beta}_i^{k+1} = \text{Shrink}\Big(\boldsymbol{\beta}_{ij}^k - \tfrac{\phi}{\eta_i}\hat{\boldsymbol{\Sigma}}_{ij}^T(\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i - \boldsymbol{\omega}_i^k + \tfrac{\boldsymbol{\gamma}_i^k}{\phi}), \tfrac{\boldsymbol{\alpha}_i}{\eta_i}\Big)_{j=1,\cdots,p_i}.$$

Compute $\boldsymbol{\omega}_i^{k+\frac{1}{2}}$ by $\frac{1}{K}(\hat{\boldsymbol{\delta}} + \boldsymbol{z}^k + K\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i^{k+1} - \hat{\boldsymbol{\Sigma}}\boldsymbol{\beta}^{k+1}),\ i = 2, \cdots, K,$

Compute $\boldsymbol{z}^{k+1}$ by

$$\boldsymbol{z}^{k+1} = \min\left(\max\left(\hat{\boldsymbol{\Sigma}}_1\boldsymbol{\beta}_1^{k+1} + \sum_{i=1}^K \boldsymbol{\omega}_i^{k+\frac{1}{2}} - \hat{\boldsymbol{\delta}} - \frac{\boldsymbol{\gamma}_1^k}{\phi}, -\lambda\boldsymbol{\alpha}\right), \lambda\boldsymbol{\alpha}\right).$$

Compute $\boldsymbol{\omega}_i^{k+1}$ by $\frac{1}{K}(\hat{\boldsymbol{\delta}} + \boldsymbol{z}^{k+1} + K\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i^{k+1} - \hat{\boldsymbol{\Sigma}}\boldsymbol{\beta}^{k+1}),\ i = 2, \cdots, K.$

Compute $\boldsymbol{\gamma}^{k+1}$ by

$$\boldsymbol{\gamma}_1^{k+1} = \boldsymbol{\gamma}_1^k + \theta\phi(\hat{\boldsymbol{\Sigma}}_1\boldsymbol{\beta}_1^{k+1} + \textstyle\sum_{i=1}^K \boldsymbol{\omega}_i^{k+1} - \boldsymbol{z}^{k+1} - \hat{\boldsymbol{\delta}}),$$

$$\boldsymbol{\gamma}_i^{k+1} = \boldsymbol{\gamma}_i^k + \theta\phi(\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\beta}_i^{k+1} - \boldsymbol{\omega}_i^{k+1}), \quad i = 2, \cdots, K.$$

end while

4.4 Numerical Experiments

In this part, we numerically evaluate the performance of the proposed algorithms. Standard method to compute Dantzig selector and LPD is via linear programming, so we use the R package *lpSolve* as a benchmark.

4.4.1 Simulation Study for Dantzig Selector

We generate the design matrix $\boldsymbol{X}$ with each row following a multivariate normal distribution with mean zero and the covariance matrix $\boldsymbol{\Sigma} = (\rho_{ij})_{p\times p}$. We consider three cases with $\rho_{ij} = 0.25^{|i-j|}, 0.5^{|i-j|}$ and $0.75^{|i-j|}$, respectively.

We randomly choose a set $S \subset \{1, \cdots, p\}$ with cardinality equal to 8. $\boldsymbol{\beta}$ is generated by randomly assign $3, 1.5, 10, 4, 2, 5, 2.5, 4.5$ without replacement to $\boldsymbol{\beta}_i$ for $i \in S$. At last, $\boldsymbol{y}$ is generated by $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$. To implement

Algorithm 4.12, 4.13 and 4.14, we set $\theta = 1.618$ for a faster convergence rate and the tuning parameter $\lambda = 10\sqrt{2\log(p)}$. We compare the performance with *lpSolve* in terms of $\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\|_2^2$, false positives (FP), false negatives (FN) over 500 replications. Table 4.1 reports the numerical performance of three methods. We can observe that SCAD-Dantzig Selector outperforms the standard Dantzig Selector by a large margin in both estimation accuracy and selection consistency. When $p = 1000$, ADMM-CD and ADMM-prox have similar performances to that of *lpSolve*, except that ADMM-CD and ADMM-prox result in fewer false positives. When $p = 10000$, *lpSolve* no longer works due to excessive memory consumption while ADMM-based methods still maintain satisfactory performance.

4.4.2 Simulation Study for LPD

We now compare the numerical performance of the proposed methods with *lpSolve* package on a simulated dataset. We fix the sample sizes $n_1 = n_2 = 200$, and consider cases $p = 2000, 10000$. Assume that $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\mu}_2 = (1, \cdots, 1, 0, \cdots, 0)^T$ with 10 nonzero components. The covariance matrix $\boldsymbol{\Sigma} = (\rho_{ij})_{p\times p}$ with $\rho_{ij} = 0.8^{|i-j|}$ for $1 \le i, j \le p$. We generate n_1 samples from $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and n_2 samples from $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ as training samples, and the test samples are generated with the same size. The tuning parameter λ is chosen by 5-fold cross validation. The estimation error $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2^2$, the average misclassification rates (MCR) on the test samples and the standard deviations over 100 replications are summarized in Table 4.2. We can see that SCAD-LPD performs better than standard LPD in terms of both estimation accuracy and misclassification rate. When $p = 2000$, ADMM and *lpSolve* lead to similar model performances. When $p = 10000$, *lpSolve* fails due to excessive memory usage.

Table 4.1. Numerical comparisons of ADMM and *lpSolve* of sparse Dantzig selector on the simulated dataset.

$n = 500, p = 1000$	ρ	$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2^2$	FP	FN
ℓ_1-ADMM-CD	0.25	0.121 (0.001)	52.16 (0.31)	0(0)
	0.5	0.125 (0.001)	47.69 (0.28)	0(0)
	0.75	0.179 (0.003)	38.28 (0.28)	0 (0)
ℓ_1-ADMM-prox	0.25	0.119 (0.001)	32.37 (0.26)	0 (0)
	0.5	0.123 (0.001)	30.56 (0.27)	0 (0)
	0.75	0.141 (0.003)	34.49 (0.54)	0 (0)
lpSolve	0.25	0.121 (0.001)	63.51 (0.34)	0 (0)
	0.5	0.125 (0.001)	59.12 (0.32)	0 (0)
	0.75	0.171 (0.003)	47.07 (0.29)	0 (0)
SCAD-ADMM-CD	0.25	0.017 (0.001)	0.47 (0.03)	0 (0)
	0.5	0.017 (0.001)	0.61 (0.03)	0 (0)
	0.75	0.022 (0.001)	0.33 (0.03)	0 (0)
SCAD-ADMM-prox	0.25	0.018 (0.000)	0.62 (0.09)	0 (0)
	0.5	0.018 (0.000)	0.61 (0.09)	0 (0)
	0.75	0.021 (0.001)	0.81 (0.17)	0 (0)

$n = 500, p = 10000$	τ	$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2^2$	FP	FN
ℓ_1-ADMM-CD	0.25	0.189 (0.002)	17.32 (0.30)	0 (0)
	0.5	0.192 (0.002)	17.95 (0.30)	0 (0)
	0.75	0.224 (0.003)	19.97 (0.27)	0 (0)
ℓ_1-ADMM-prox	0.25	0.150 (0.002)	66.56 (0.53)	0 (0)
	0.5	0.226 (0.003)	24.99 (0.83)	0 (0)
	0.75	0.214 (0.003)	33.14 (1.12)	0 (0)
lpSolve	0.25		✗	
	0.5		✗	
	0.75		✗	
SCAD-ADMM-CD	0.25	0.036 (0.001)	0.00 (0.00)	0 (0)
	0.5	0.025 (0.001)	0.01 (0.00)	0 (0)
	0.75	0.114 (0.002)	0.00 (0.00)	0 (0)
SCAD-ADMM-prox	0.25	0.036 (0.001)	0.00 (0.00)	0 (0)
	0.5	0.112 (0.001)	0.00 (0.00)	0 (0)
	0.75	0.116 (0.002)	0.00 (0.00)	0 (0)

Table 4.2. Numerical comparisons of ADMM and *lpSolve* of sparse LPD on the simulated dataset.

$p = 2000$	$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2^2$	MCR
ℓ_1-ADMM-CD	4.47 (0.10)	19.59 (0.25)
ℓ_1-ADMM-prox	4.30 (0.09)	19.05 (0.20)
lpSolve	4.50 (0.12)	19.13 (0.22)
SCAD-ADMM-CD	2.05 (0.12)	17.94 (0.21)
SCAD-ADMM-prox	1.78 (0.11)	18.16 (0.20)

$p = 10000$	$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|_2^2$	MCR
ℓ_1-ADMM-CD	10.43 (0.02)	28.10 (0.26)
ℓ_1-ADMM-prox	10.12 (0.05)	24.90 (0.24)
lpSolve	✗	✗
SCAD-ADMM-CD	5.68 (0.37)	21.86 (0.41)
SCAD-ADMM-prox	8.52 (0.11)	24.82 (0.32)

4.4.3 Microarray Data for Affymetrix's HGU133a Platform

We test the performance of proposed algorithms on a microarray dataset in [75]. This dataset contains 13182 publicly available microarray samples for Affymetrix's HGU133a platform. There are 2717 tissue types in the dataset (e.g., lung cancers, breast cancers, etc.), and each array sample contains 12704 genes. In the dataset, 55 array samples are identified as lung normal. We select other 148 array samples diagnosed with lung cancers, and then randomly choose 40 healthy samples and 100 unhealthy samples to be our training set, and use the rest samples for testing. Among all genes, 9130 genes have p-values smaller than 0.05 in the t-test, and gene *FAM107A* has the largest absolute value of t statistics. Our goal is to find an appropriate model for *FAM107A*, using 9130 statistically significant genes as potential predictors. We repeat the experiment 100 times and report the average prediction error and cardinality of selected sets (see table 4.3). While *lpSolve* runs

out of memory under such dimensionality, Algorithm 4.12 and 4.13 achieve similar prediction error but the latter identifies a much larger pool of influential genes. SCAD-Dantzig selects fewer features into the model with comparable performance achieved.

Table 4.3. Performances of ADMM and *lpSolve* of sparse Dantzig selector on the microarray dataset.

	Prediction Error	Cardinality
ℓ_1-ADMM-CD	0.055 (0.001)	315.8 (1.9)
ℓ_1-ADMM-prox	0.061 (0.001)	1562.9 (8.9)
lpSolve	✗	✗
SCAD-ADMM-CD	0.057 (0.001)	45.8 (0.5)
SCAD-ADMM-prox	0.056 (0.001)	399.2 (2.4)

4.5 Appendix

4.5.1 Proof of Theorem 4.1

Proof. Assume that none of the E_1, E_2 and E_3 is true, which is at the probability $1 - P_1 - P_2 - P_3$. Then since $\|\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\min} > (a+1)\lambda$, we have that

$$\begin{aligned} &|\hat{\beta}^D_j| \le \|\hat{\boldsymbol{\beta}}^D - \boldsymbol{\beta}^*\|_{\max} \le a_0\lambda \le a_2\lambda, \quad \text{for } j \in \mathcal{A}^c, \\ &|\hat{\beta}^D_j| \ge \|\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\min} - \|\hat{\boldsymbol{\beta}}^D - \boldsymbol{\beta}^*\|_{\max} > a\lambda, \quad \text{for } j \in \mathcal{A}. \end{aligned} \tag{4.38}$$

Consider the properties of folded concave penalty, it follows that

$$\begin{aligned} &\rho'(|\hat{\beta}^D_j|) \ge a_1, \quad \text{for } j \in \mathcal{A}^c, \\ &\rho'(|\hat{\beta}^D_j|) = 0 \quad \text{for } j \in \mathcal{A}. \end{aligned} \tag{4.39}$$

We first show that $\hat{\boldsymbol{\beta}}^{oracle} = (\hat{\boldsymbol{\beta}}_{\mathcal{A}}, \mathbf{0})$ is the unique solution to the following problem,

$$\begin{aligned} \min_{\boldsymbol{\beta} \in R^p} \quad & \sum_j \rho'(|\hat{\beta}_j^D|)|\beta_j| \\ \text{s.t} \quad & |\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})| \leq n \cdot \lambda \rho'(|\hat{\beta}_j^D|), j = 1, \cdots, p. \end{aligned} \tag{4.40}$$

Since $\rho'(|\hat{\beta}_j^D|) = 0$ for $j \in \mathcal{A}$, $\hat{\boldsymbol{\beta}}_{\mathcal{A}^c} = \mathbf{0}$ leads to optimal objective value. It suffices to show that $\hat{\boldsymbol{\beta}}_{\mathcal{A}}$ uniquely satisfies the following conditions:

(B1) $\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}}) = 0$, for $j \in \mathcal{A}$;

(B2) $|\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}})| \leq n\lambda\rho'(|\hat{\beta}_j^D|)$, for $j \in \mathcal{A}^c$.

For $j \in \mathcal{A}^c$, we have

$$\begin{aligned} & \big\{|\nabla_j L(\hat{\boldsymbol{\beta}}^{oracle})| \leq a_1\lambda\big\} \\ = \; & \big\{|\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\hat{\boldsymbol{\beta}}_{\mathcal{A}})| \leq na_1\lambda\big\} \\ \subseteq \; & \big\{|\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}}\hat{\boldsymbol{\beta}}_{\mathcal{A}})| \leq n\lambda\rho'(|\hat{\beta}_j^D|)\big\}. \end{aligned} \tag{4.41}$$

Therefore under the assumption that $\{\|\nabla_{\mathcal{A}^c} L(\hat{\boldsymbol{\beta}}^{oracle})\|_{\max} \leq a_1\lambda\}$, $\hat{\boldsymbol{\beta}}^{oracle}$ satisfies (B2). In addition, under regularity condition of (4.11), $\hat{\boldsymbol{\beta}}_{\mathcal{A}}$ is the (unique) solution to (B1). Therefore $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{oracle}$ is the unique solution to (4.40) and thus LLA algorithm finds the oracle estimator after one iteration.

Denote the solution of the next iteration of the LLA algorithm as $\hat{\boldsymbol{\beta}}^{(2)}$. Since $\hat{\boldsymbol{\beta}}_{\mathcal{A}^c}^{oracle} = \mathbf{0}$, we have $\rho'(|\hat{\beta}_j^{oracle}|) = \rho'(0) > a_1$ for $j \in \mathcal{A}^c$. $\rho'(|\hat{\beta}_j^{oracle}|) = 0$ for $j \in \mathcal{A}$ under the event $\{\|\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle}\|_{\min} > a\lambda\}$. Therefore, $\hat{\boldsymbol{\beta}}^{(2)} = \hat{\boldsymbol{\beta}}^{(oracle)}$ is the minimizer

of the following problem

$$\begin{aligned} \min_{\boldsymbol{\beta}\in R^p} \quad & \sum_j \rho'(|\hat{\beta}_j^{oracle}|)|\beta_j| \\ \text{s.t} \quad & |\boldsymbol{X}_j^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})| \le n\cdot\lambda\rho'(|\hat{\beta}_j^{oracle}|). \end{aligned} \tag{4.42}$$

Hence, the LLA algorithm finds the oracle estimator again and stays there.

Next we prove that $P_i \to 0, i = 1, 2, 3$. Since $\lambda > \frac{4}{a_0\kappa_D} q^{1/2}\lambda_D$, from the result in [71], we have $P_1 \to 0$, as $n \to +\infty$. [50] provides a bound for P_2 of the penalized linear regression using the Chernoff bound on the sub-gaussian random variables,

$$P_2 \le 2q\exp\Big(-\frac{n\lambda_{\min}}{2\sigma^2}\lambda^2\Big) \to 0,$$

as $n \to +\infty$. Using the Chernoff bound, we have

$$\begin{aligned} P_3 &\le \sum_{j\in\mathcal{A}^c} P\big(|\boldsymbol{X}_j^T(\boldsymbol{I}-\boldsymbol{X}_{\mathcal{A}}^T(\boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}})^{-1}\boldsymbol{X}_{\mathcal{A}}^T)\boldsymbol{\varepsilon}| > n\cdot a_1\lambda\big) \\ &\le \sum_{j\in\mathcal{A}^c} \exp\Big(-\frac{a_1^2n^2\lambda^2}{2\sigma^2\cdot\|\boldsymbol{X}_j^T(\boldsymbol{I}-\boldsymbol{X}_{\mathcal{A}}^T(\boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}})^{-1}\boldsymbol{X}_{\mathcal{A}}^T)\|_2^2}\Big) \\ &\le 2(p-q)\exp(-\frac{a_1^2n\lambda^2}{2\sigma^2}) \to 0, \end{aligned} \tag{4.43}$$

as $n \to +\infty$ by the assumption that $\log(p) = o(n\lambda^2)$. The last inequality comes from the fact that $\boldsymbol{X}$ is standardized and thus

$$\|\boldsymbol{X}_j^T(\boldsymbol{I}-\boldsymbol{X}_{\mathcal{A}}^T(\boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}})^{-1}\boldsymbol{X}_{\mathcal{A}}^T)\|_2^2 = \boldsymbol{X}_j^T(\boldsymbol{I}-\boldsymbol{X}_{\mathcal{A}}^T(\boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}})^{-1}\boldsymbol{X}_{\mathcal{A}}^T)\boldsymbol{X}_j \le n.$$

□

4.5.2 Proof of Theorem 4.3

Proof. Assume that none of the E_1, E_2 and E_3 is true, which is at the probability $1 - P_1 - P_2 - P_3$.

By the same argument in the proof of Theorem 4.1, we have

$$\begin{aligned} &\rho'(|\hat{\beta}_j^{LPD}|) \geq a_1, \quad \text{for } j \in \mathcal{A}^c, \\ &\rho'(|\hat{\beta}_j^{LPD}|) = 0 \quad \text{for } j \in \mathcal{A}. \end{aligned} \tag{4.44}$$

We derive $L(\boldsymbol{\beta})$ for $j \in \mathcal{A}$,

$$\begin{aligned} \nabla_j L(\boldsymbol{\beta}) &= [\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}]^{-2} \big\{ - [\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}] \frac{d[\boldsymbol{\beta}^T S_B \boldsymbol{\beta}]}{d\boldsymbol{\beta}_j} + [\boldsymbol{\beta}^T S_B \boldsymbol{\beta}] \frac{d[\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}]}{d\boldsymbol{\beta}_j} \big\} \\ &= [\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}]^{-2} \big\{ - [\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}] 2 \hat{\delta}_j \hat{\boldsymbol{\delta}}^T \boldsymbol{\beta} + [\boldsymbol{\beta}^T S_B \boldsymbol{\beta}] 2 \hat{\boldsymbol{\Sigma}}_j^T \boldsymbol{\beta} \big\}. \end{aligned} \tag{4.45}$$

Then we equate $\nabla_j L(\boldsymbol{\beta})$ to zero and obtain

$$\begin{aligned} &\nabla_j L(\boldsymbol{\beta}) = 0 \\ \iff &[\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}] \hat{\delta}_j \hat{\boldsymbol{\delta}}^T \boldsymbol{\beta} - [\boldsymbol{\beta}^T S_B \boldsymbol{\beta}] \hat{\boldsymbol{\Sigma}}_j^T \boldsymbol{\beta} = 0 \\ \iff &\hat{\delta}_j \hat{\boldsymbol{\delta}}^T \boldsymbol{\beta} - \frac{[\boldsymbol{\beta}^T S_B \boldsymbol{\beta}]}{[\boldsymbol{\beta}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{\beta}]} \hat{\boldsymbol{\Sigma}}_j^T \boldsymbol{\beta} = 0 \\ \iff &\hat{\delta}_j \hat{\boldsymbol{\delta}}^T \boldsymbol{\beta} + L(\boldsymbol{\beta}) \hat{\boldsymbol{\Sigma}}_j^T \boldsymbol{\beta} = 0. \end{aligned} \tag{4.46}$$

Therefore $\hat{\boldsymbol{\beta}}^{oracle} = (\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle}, \mathbf{0})$ is the solution to

$$\begin{aligned} &\min_{\boldsymbol{\beta}: \boldsymbol{\beta}_{\mathcal{A}^c} = \mathbf{0}} L(\boldsymbol{\beta}) \\ &\text{s.t.} \quad \hat{\boldsymbol{\delta}}_{\mathcal{A}} \hat{\boldsymbol{\delta}}^T \boldsymbol{\beta} + L(\boldsymbol{\beta}) \hat{\boldsymbol{\Sigma}}_{\mathcal{A}}^T \boldsymbol{\beta} = \mathbf{0}, \end{aligned} \tag{4.47}$$

where $\hat{\boldsymbol{\Sigma}}_{\mathcal{A}}$ denotes the sub-matrix of $\hat{\boldsymbol{\Sigma}}$ whose column indices are in $\mathcal{A}$. This is a

generalized eigenvalue problem and the solution is given by

$$\hat{\boldsymbol{\beta}}_{\mathcal{A}} = \hat{\boldsymbol{\Sigma}}_{\mathcal{A},\mathcal{A}}^{-1}\hat{\boldsymbol{\delta}}_{\mathcal{A}}, \quad \hat{\boldsymbol{\beta}}_{\mathcal{A}^c} = \mathbf{0}, \tag{4.48}$$

where $\hat{\boldsymbol{\Sigma}}_{\mathcal{A},\mathcal{A}} = (\hat{\Sigma}_{i,j})_{i\in\mathcal{A},j\in\mathcal{A}}$. Consider the LLA problem,

$$\begin{aligned} \min_{\boldsymbol{\beta}\in R^p} \quad & \sum_j \rho'(|\hat{\beta}_j^0|)|\beta_j| \\ \text{s.t} \quad & |\hat{\boldsymbol{\Sigma}}_j^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le \lambda\rho'(|\hat{\beta}_j^0|), \quad j = 1, \cdots, p. \end{aligned} \tag{4.49}$$

Given that (4.44) holds, it remains to prove that $\hat{\boldsymbol{\beta}}^{oracle} = (\hat{\boldsymbol{\beta}}_{\mathcal{A}}, \mathbf{0})$ satisfies the following conditions:

(B1') $\hat{\boldsymbol{\Sigma}}_{\mathcal{A}}^T\boldsymbol{\beta} - \hat{\boldsymbol{\delta}}_{\mathcal{A}} = 0$;

(B2') $|\hat{\boldsymbol{\Sigma}}_j^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le \lambda\rho'(|\hat{\beta}_j^{(0)}|)$, for $j \in \mathcal{A}^c$.

For $j \in \mathcal{A}^c$, since $\rho'(|\hat{\beta}_j^{(0)}|) \ge a_1$, we have

$$\begin{aligned} &\big\{|\hat{\boldsymbol{\Sigma}}^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le a_1\lambda\big\} \\ \subseteq &\big\{|\hat{\boldsymbol{\Sigma}}_j^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le \lambda\rho'(|\hat{\beta}_j^{(0)}|)\big\}. \end{aligned} \tag{4.50}$$

Therefore when E_3 is not true, $\hat{\boldsymbol{\beta}}^{oracle}$ satisfies (B2'). When $\hat{\boldsymbol{\beta}}_{\mathcal{A}^c} = \mathbf{0}$,

$$\hat{\boldsymbol{\Sigma}}_{\mathcal{A}}^T\boldsymbol{\beta} - \hat{\boldsymbol{\delta}}_{\mathcal{A}} = 0 \Longleftrightarrow \hat{\boldsymbol{\Sigma}}_{\mathcal{A},\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}} - \hat{\boldsymbol{\delta}}_{\mathcal{A}} = \mathbf{0}. \tag{4.51}$$

By (4.48), we have $\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle} = \hat{\boldsymbol{\Sigma}}_{\mathcal{A},\mathcal{A}}^{-1}\hat{\boldsymbol{\delta}}_{\mathcal{A}}$. Therefore $\hat{\boldsymbol{\beta}}^{oracle}$ satisfies (B1').

Denote the solution of the next iteration of the LLA algorithm as $\hat{\boldsymbol{\beta}}^{(2)}$. Since $\hat{\boldsymbol{\beta}}_{\mathcal{A}^c}^{oracle} = \mathbf{0}$, we have $\rho'(|\hat{\beta}_j^{oracle}|) = \rho'(0) > a_1$ for $j \in \mathcal{A}^c$. $\rho'(|\hat{\beta}_j^{oracle}|) = 0$ for $j \in \mathcal{A}$ under the event $\{\|\hat{\boldsymbol{\beta}}_{\mathcal{A}}^{oracle}\|_{\min} > a\lambda\}$. Therefore, we have $\hat{\boldsymbol{\beta}}^{(2)} = \hat{\boldsymbol{\beta}}^{(oracle)}$ is the

unique minimizer of the following problem,

$$\begin{aligned} \min_{\boldsymbol{\beta}\in R^p} \quad & \sum_j \rho'(|\hat{\beta}_j^{oracle}|)|\beta_j| \\ \text{s.t} \quad & |\hat{\boldsymbol{\Sigma}}_j^T\boldsymbol{\beta} - (\hat{\mu}_{1j} - \hat{\mu}_{2j})| \le \lambda\rho'(|\hat{\beta}_j^{oracle}|), \quad j = 1, \cdots, p. \end{aligned} \tag{4.52}$$

Hence, the LLA algorithm finds the oracle estimator again and stays there. Next we show that $P_i \to 0, i = 1, 2, 3$ as $n \to \infty$. According to the sign consistency result from Theorem 2 in [74], it is straightforward to derive that when $\log(p) = o(n\lambda^2)$, $P_1 \to 0$ as $n \to \infty$. Since $|\hat{\beta}_j^{oracle}| \ge \|\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\min} - \|\hat{\boldsymbol{\beta}}^{oracle} - \boldsymbol{\beta}^*\|_{\max}$ for $j \in \mathcal{A}$,

$$\begin{aligned} P_2 &= P(\|\hat{\boldsymbol{\beta}}^{oracle}_{\mathcal{A}}\|_{\min} \le a\lambda) \\ &\le P(\|\boldsymbol{\beta}^*_{\mathcal{A}}\|_{\min} - \|\hat{\boldsymbol{\beta}}^{oracle} - \boldsymbol{\beta}^*\|_{\max} \le a\lambda) \\ &\le P(\|\hat{\boldsymbol{\beta}}^{oracle} - \boldsymbol{\beta}^*\|_{\max} \ge \lambda). \end{aligned} \tag{4.53}$$

From the Lemma 15 of [76], we have that $\|\hat{\boldsymbol{\beta}}^{oracle} - \boldsymbol{\beta}^*\|_{\max} = \mathcal{O}\big(\sqrt{\frac{\log(q\log n)}{n}}\big)$ with probability at least $1 - \mathcal{O}(\log^{-1}(n))$. Thus by the assumption that $\log(p) = o(n\lambda^2)$, we have that $P_2 \to 0$ as $n \to \infty$.

An intermediate result of Theorem 2 in [74] implies that $\|\hat{\boldsymbol{\Sigma}}^T_{\mathcal{A}^c}\hat{\boldsymbol{\beta}}^{oracle} - \hat{\boldsymbol{\delta}}_{\mathcal{A}^c}\|_{\max} = \mathcal{O}\Big(\sqrt{\frac{\log\big((p-q)\log n\big)}{n}}\Big)$ with probability at least $1 - \mathcal{O}(\log^{-1}(n))$, therefore we also have

$$P_3 = P(\|\hat{\boldsymbol{\Sigma}}^T_{\mathcal{A}^c}\hat{\boldsymbol{\beta}}^{oracle} - \hat{\boldsymbol{\delta}}_{\mathcal{A}^c}\|_{\max} > a_1\lambda) \to 0, \tag{4.54}$$

as $n \to 0$. □

Chapter 5

Conclusion

This dissertation focuses on the computational challenge associated with high-dimensional non-smooth optimization. In Chapter 2, we give a systematic review on the relevant research in ADMM algorithm.

In Chapter 3, we propose an efficient and parallelizable for weighted ℓ_1 and SCAD penalized quantile regression and support vector machine based on alternating direction method of multiplier. In particular, to alleviate the intense memory usage in high dimensional data modeling, we adopt a 3-block ADMM that is compatible with feature space split and parallel computing. The proposed algorithm is computationally advantageous in high dimensional settings and enjoys the linear rate of convergence. The numerical experiments suggest that the proposed method has comparable performances compared with standard solvers in moderate dimensions and maintains satisfactory performances when the dimension is huge. The computational framework can also be generalized to a wide range of statistical model fitting problems. In Chapter 4, we introduce the application of the proposed 3-block ADMM method to solve the non-convex Dantzig selector and LPD for ultra-high dimensional data. We justify the strong oracle property of

the estimator obtained from the one-step LLA algorithm. We conduct simulation studies and real data analysis to demonstrate the favorable empirical performance of the proposed computational framework in ultra-high dimensions.

Bibliography

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

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

[3] Xiangyu Wang, Peichao Peng, and David B Dunson. Median selection subset aggregation for parallel inference. In *Advances in Neural Information Processing Systems*, pages 2195–2203, 2014.

[4] Michael I Jordan, Jason D Lee, and Yun Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, pages 1–14, 2018.

[5] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

[6] Shao-Bo Lin, Xin Guo, and Ding-Xuan Zhou. Distributed learning with regularized least squares. *The Journal of Machine Learning Research*, 18(1):3202–3232, 2017.

[7] Heather Battey, Jianqing Fan, Han Liu, Junwei Lu, and Ziwei Zhu. Distributed estimation and inference with statistical guarantees. *arXiv preprint arXiv:1509.05457*, 2015.

[8] Junhong Lin and Volkan Cevher. Optimal distributed learning with multi-pass stochastic gradient methods. In *Proceedings of the 35th International Conference on Machine Learning*, number CONF, 2018.

[9] Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Efficient distributed learning with sparsity. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3636–3645. JMLR. org, 2017.

[10] Jason D Lee, Qiang Liu, Yuekai Sun, and Jonathan E Taylor. Communication-efficient sparse regression. *The Journal of Machine Learning Research*, 18(1):115–144, 2017.

[11] Qifan Song and Faming Liang. A split-and-merge bayesian variable selection approach for ultrahigh dimensional regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(5):947–972, 2015.

[12] Xiangyu Wang, David B Dunson, and Chenlei Leng. Decorrelated feature space partitioning for distributed sparse regression. In *Advances in Neural Information Processing Systems*, pages 802–810, 2016.

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

[14] Hui Zou and Runze Li. One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of Statistics*, 36(4):1509, 2008.

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

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

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

[18] Shixiang Chen, Shiqian Ma, Lingzhou Xue, and Hui Zou. An alternating manifold proximal gradient method for sparse principal component analysis and sparse canonical correlation analysis. *Informs Journal on Optimization*, pages ijoo–2019, 2020.

[19] Lingzhou Xue, Shiqian Ma, and Hui Zou. Positive-definite ℓ_1-penalized estimation of large covariance matrices. *Journal of the American Statistical Association*, 107(500):1480–1491, 2012.

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

[21] Shiqian Ma, Lingzhou Xue, and Hui Zou. Alternating direction methods for latent variable gaussian graphical model selection. *Neural computation*, 25(8):2172–2198, 2013.

[22] Cheng Wang and Binyan Jiang. An efficient admm algorithm for high dimensional precision matrix estimation via penalized quadratic loss. *Computational Statistics & Data Analysis*, 142:106812, 2020.

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

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

[25] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3):889–916, 2016.

[26] Michel Fortin and Roland Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*, volume 15. Elsevier, 2000.

[27] Jim Douglas and Henry H Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[28] Jonathan Eckstein, Dimitri P Bertsekas, et al. An alternating direction method for linear programming. 1990.

[29] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016.

[30] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, pages 1–35, 2012.

[31] Tian-Yi Lin, Shi-Qian Ma, and Shu-Zhong Zhang. On the sublinear convergence rate of multi-block admm. *Journal of the Operations Research Society of China*, 3(3):251–274, 2015.

[32] Tianyi Lin, Shiqian Ma, and Shuzhong Zhang. On the global linear convergence of the admm with multiblock variables. *SIAM Journal on Optimization*, 25(3):1478–1497, 2015.

[33] Bingsheng He, Min Tao, and Xiaoming Yuan. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, 22(2):313–340, 2012.

[34] Bingsheng He, Min Tao, and Xiaoming Yuan. Convergence rate and iteration complexity on the alternating direction method of multipliers with a substitution procedure for separable convex programming. *Math. Oper. Res., under revision*, 2:000–000, 2012.

[35] Bingsheng He, Liusheng Hou, and Xiaoming Yuan. On full jacobian decomposition of the augmented lagrangian method for separable convex programming. *SIAM Journal on Optimization*, 25(4):2274–2312, 2015.

[36] Wei Deng, Ming-Jun Lai, Zhimin Peng, and Wotao Yin. Parallel multi-block admm with o (1/k) convergence. *Journal of Scientific Computing*, pages 1–25, 2014.

[37] Defeng Sun, Kim-Chuan Toh, and Liuqin Yang. A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints. *SIAM journal on Optimization*, 25(2):882–915, 2015.

[38] Deren Han and Xiaoming Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.

[39] Bingsheng He, Feng Ma, and Xiaoming Yuan. Optimally linearizing the alternating direction method of multipliers for convex programming. *Computational Optimization and Applications*, 75(2):361–388, 2020.

[40] Maryam Fazel, Ting Kei Pong, Defeng Sun, and Paul Tseng. Hankel matrix rank minimization with applications to system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34(3):946–977, 2013.

[41] Jie Sun. On monotropic piecewise quadratic programming (network, algorithm, convex programming, decomposition method). 1986.

[42] Min Li, Defeng Sun, and Kim-Chuan Toh. A convergent 3-block semi-proximal admm for convex minimization problems with one strongly convex block. *Asia-Pacific Journal of Operational Research*, 32(04):1550024, 2015.

[43] Xudong Li, Defeng Sun, and Kim-Chuan Toh. A schur complement based semi-proximal admm for convex quadratic conic programming and extensions. *Mathematical Programming*, 155(1-2):333–373, 2016.

[44] Xudong Li. *A Two-Phase Augmented Lagrangian Method For Convex Composite Quadratic Programming*. PhD thesis, 2015.

[45] Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

[46] Alexandre Belloni, Victor Chernozhukov, et al. l1-penalized quantile regression in high-dimensional sparse models. *The Annals of Statistics*, 39(1):82–130, 2011.

[47] Ben Sherwood and Adam Maidman. *rqPen: Penalized Quantile Regression*, 2017. R package version 2.0.

[48] Roger Koenker and Ivan Mizera. Convex optimization, shape constraints, compound decisions, and empirical bayes rules. *Journal of the American Statistical Association*, 109(506):674–685, 2014.

[49] Bo Peng and Lan Wang. An iterative coordinate descent algorithm for high-dimensional nonconvex penalized quantile regression. *Journal of Computational and Graphical Statistics*, 24(3):676–694, 2015.

[50] Jianqing Fan, Lingzhou Xue, and Hui Zou. Strong oracle optimality of folded concave penalized estimation. *The Annals of Statistics*, 42(3):819, 2014.

[51] Jianqing Fan, Lingzhou Xue, and Hui Zou. Multitask quantile regression under the transnormal model. *Journal of the American Statistical Association*, 111(516):1726–1735, 2016.

[52] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[53] Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor J Hastie. 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56, 2004.

[54] Donald Goldfarb and Shiqian Ma. Fast multiple-splitting algorithms for convex optimization. *SIAM Journal on Optimization*, 22(2):533–556, 2012.

[55] Yuwen Gu, Jun Fan, Lingchen Kong, Shiqian Ma, and Hui Zou. Admm for high-dimensional sparse penalized quantile regression. *Technometrics*, 60(3):319–331, 2018.

[56] Roger Koenker. Quantile regression: 40 years on. *Annual Review of Economics*, 9:155–176, 2017.

[57] Lan Wang, Yongdai Kim, and Runze Li. Calibrating non-convex penalized regression in ultra-high dimension. *The Annals of Statistics*, 41(5):2505, 2013.

[58] Xiang Zhang, Yichao Wu, Lan Wang, and Runze Li. Variable selection for support vector machines in moderately high dimensions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(1):53–76, 2016.

[59] Bo Peng, Lan Wang, and Yichao Wu. An error bound for l1-norm support vector machine coefficients in ultra-high dimension. *The Journal of Machine Learning Research*, 17(1):8279–8304, 2016.

[60] Jerome Friedman, Trevor Hastie, Holger Höfling, Robert Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[61] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

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

[63] Xiang Zhang, Yichao Wu, Lan Wang, and Runze Li. A consistent information criterion for support vector machines in diverging model spaces. *The Journal of Machine Learning Research*, 17(1):466–491, 2016.

[64] R Dennis Cook, Bing Li, and Francesca Chiaromonte. Dimension reduction in regression without matrix inversion. *Biometrika*, 94(3):569–584, 2007.

[65] Hansheng Wang. Forward regression for ultra-high dimensional variable screening. *Journal of the American Statistical Association*, 104(488):1512–1524, 2009.

[66] Deren Han, Defeng Sun, and Liwei Zhang. Linear rate convergence of the alternating direction method of multipliers for convex composite programming. *Mathematics of Operations Research*, 43(2):622–637, 2018.

[67] Nicolai Meinshausen and Bin Yu. Lasso-type recovery of sparse representations for high-dimensional data. *The Annals of Statistics*, 37(1):246–270, 2009.

[68] Ja-Yong Koo, Yoonkyung Lee, Yuwon Kim, and Changyi Park. A bahadur representation of the linear support vector machine. *Journal of Machine Learning Research*, 9(Jul):1343–1368, 2008.

[69] Keith Knight. Limiting distributions for l_1 regression estimators under general conditions. *The Annals of Statistics*, pages 755–770, 1998.

[70] Emmanuel Candes and Terence Tao. The dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

[71] Peter J Bickel, Ya'acov Ritov, Alexandre B Tsybakov, et al. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[72] Michel Berkelaar and others. *lpSolve: Interface to 'Lp_solve' v. 5.5 to Solve Linear/Integer Programs*, 2015. R package version 5.6.13.

[73] Tony Cai and Weidong Liu. A direct estimation approach to sparse linear discriminant analysis. *Journal of the American Statistical Association*, 106(496):1566–1577, 2011.

[74] Zhen Zhang, Shengzheng Wang, and Wei Bian. Sign consistency for the linear programming discriminant rule. *Pattern Recognition*, 100:107083, 2020.

[75] Matthew N. McCall, Benjamin M. Bolstad, and Rafael A. Irizarry. Frozen robust multiarray analysis (frma). *Biostatistics*, 11(2):242, 2010.

[76] Mladen Kolar and Han Liu. Optimal feature selection in high-dimensional discriminant analysis. *IEEE transactions on information theory*, 61(2):1063–1083, 2014.

JIAWEI WEN

EDUCATION

Ph.D. in Statistics, The Pennsylvania State University, University Park, USA 2015 - 2020

B.S. in Statistics, Shandong University, Jinan, China 2011 - 2015

EXPERIENCES

Research Scientist, Facebook, Sept 2020 - present

Research Intern, IBM Research, Aug 2019 - Nov 2019

Data Scientist Intern, Airbnb, May 2019 - Aug 2019

Data Scientist Intern, Stitch Fix, May 2018 - Aug 2018

Graduate Teaching Assistant, Pennsylvania State University, Aug 2015 - May 2020

PUBLICATIONS&PROJECTS

ET-Lasso: A New Efficient Tuning of Lasso-type Regularization for High-Dimensional Data. *in Proceedings of the 2019 ACM SIGKDD* (Oral)

Prioritizing genetic variants in GWAS using permutation-assisted lasso tuning. *Bioinformatics*

Revenue Maximization of Airbnb Marketplace using Search Results. *arXiv*

Multi-block ADMM Algorithms for High-Dimensional Sparse Estimation. *In submission*

Distributed Algorithms for Nonconvex Dantzig Selector and Linear Programming Discriminant Rule. *In submission*