Efficient Computation for Sparse Estimation Problems

Ryosuke Shimmura

MAY 2024

Efficient Computation for Sparse Estimation Problems

A dissertation submitted to
THE GRADUATE SCHOOL OF ENGINEERING SCIENCE
OSAKA UNIVERSITY
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN ENGINEERING

By

Ryosuke Shimmura

MAY 2024

Abstract

This thesis discusses efficient solutions for convex optimization problems in sparse estimation. Sparse estimation is a method that allows for simultaneous parameter estimation and variable selection by adding a regularization term to the loss function. It is widely used in various fields, including machine learning and a signal processing. The least absolute shrinkage and selection operator (lasso) is one of the most famous sparse estimation method. It involves a convex objective function, which allows for efficient solution finding through coordinate descent methods. However, for complex problems like convex clustering, there are no efficient solution methods, often resulting in significant computation time.

In sparse estimation, such as fused lasso and convex clustering, we apply either the alternating direction method of multipliers (ADMM) or the proximal gradient method to solve the problem. It takes time to include matrix division in the former case, while an efficient method such as FISTA(fast iterative shrinkage-thresholding algorithm) has been developed in the latter case. In the first part (Chapter 3), We propose a general method for converting the ADMM to the proximal gradient method, assuming that the derivative of the loss function is Lipschitz continuous. Then, we apply it to sparse estimation problems, such as sparse convex clustering and trend filtering, and we show by numerical experiments that we can obtain a significant improvement in terms of efficiency.

In the second part (Chapter 4), under the assumption that the loss function is strongly convex, we propose Newton and quasi-Newton methods that utilize proximal gradient steps. The quasi-Newton method can avoid the computation of the Hessian matrix, leading to improved efficiency. Furthermore, we prove that both proposed methods converge rapidly to the solution. These methods are particularly efficient for problems with L1 regularization or group regularization, as they perform variable selection in each update. Through numerical experiments, we show that these methods efficiently obtain solutions for sparse estimation problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Statistical and machine learning techniques are widely used in various fields such as chemistry, engineering, and business for analyzing large datasets and extracting meaningful insights. A common problem in these fields is identifying more important variables for predictors, enabling more accurate predictions and better understanding of underlying relationships. With the advent of digitalization leading to large-scale, high-dimensional data, efficiently identifying important variables becomes necessary, and sparse modeling is employed.

In sparse modeling, the L_0-norm, i.e., the number of non-zero elements, is a measure of sparsity. The Akaike Information Criterion (AIC)[1] and the Bayesian Information Criterion (BIC)[2] are nortable examples of methods employing the L_0-norm. However, as the L_0-norm is not a convex function, brute force search is the only way to minimize these criteria, requiring estimation for 2^n combinations of n variables, which is inefficient.

This thesis focuses on the efficiency of sparse estimation, exemplified by least absolute shrinkage and selection operator (lasso)[3], which performs L_1-regularization in linear regression. The use of the L_1-norm for regularization results in a convex optimization problem, making lasso popular because efficient procedures exist for finding solutions. Lasso extensions include logistic regression, Poisson regression, Cox regression, etc.[4]. Further, methods like group lasso[5], fused lasso[6], joint graphical lasso[7], and convex clustering[8, 9, 10] replace the regularization term with structured regularization. All these extensions are formulated as convex optimization problems, allowing efficient solution finding and model selection by estimating insignificant parameters as zero.

In Chapter 3, we propose converting the alternating direction method of multipliers (ADMM) into the proximal gradient method. While ADMM is more gen-

erally applicable than the proximal gradient method, it often requires inefficient inverse matrix calculations. In contrast, the proximal gradient method are simpler to implement and have a narrower application range. This chapter shows that ADMM procedures can generally be converted into proximal gradient methods, allowing for the application of acceleration techniques like the fast iterative shrinkage-thresholding algorithm (FISTA)[11] for efficient solution finding.

In Chapter 4, we propose Newton and quasi-Newton method using proximal gradient steps. While the proximal Newton method are commonly used for rapid convergence, the coordinate descent method[12] efficiently updates L_1-regularization problems. However, for issues like group regularization, partial problems must be solved using proximal gradient methods, not fully leveraging the speed of convergence. The proposed method, capitalizing on the sparsity of the solution, allows for computational savings and is efficient for problems like group lasso.

1.2 Overview

The rest of this thesis is organized as follows. Chapter 2 introduces theories of convex optimization and existing methods such as the proximal gradient method. Then, Chapter 3 proposes a general method for converting the ADMM procedures into the proximal gradient method, applying them to real sparse estimation problems like sparse convex clustering and trend filtering, confirming the speed of obtaining solutions. Chapter 4 proposes Newton-type methods that can omit the computation of components estimated as zero, applying it to group logistic regression and confirming rapid solution finding. Finally, Chapter 5 summarizes the proposed methods and points out future research directions.

Chapter 2

Background

2.1 Convex function and its subdifferential

In this section, we provide background information to understand the results in the subsequent sections. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \tag{2.1}$$

for any $x, y \in \mathbb{R}^n$ and $0 \leq \lambda \leq 1$. In particular, we say that the convex function f is *closed* if $\{x \in \mathbb{R}^n | f(x) \leq \alpha\}$ is a closed set for each $\alpha \in \mathbb{R}$. Moreover, we say that f is μ*-strongly convex* if

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) - \frac{\mu}{2}\lambda(1-\lambda)\|x-y\|_2^2 \tag{2.2}$$

for any $x, y \in \mathbb{R}^n$ and $0 \leq \lambda \leq 1$. When f is twice differentiable, f is μ*-strongly convex* if and only if $\nabla^2 f(x) - \mu I$ is positive semidefinite for any $x \in \mathbb{R}^n$ [13].

For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, we define the *subdifferential* of f at $x_0 \in \mathbb{R}^n$ by the set of $z \in \mathbb{R}^n$ such that

$$f(x) \geq f(x_0) + \langle z, x - x_0 \rangle \tag{2.3}$$

for any $x \in \mathbb{R}^n$ and denote it as $\partial f(x_0)$. For example, the subdifferential of $f(x) = |x|, x \in \mathbb{R}$ at $x = 0$ is the set of z such that $|x| \geq zx, x \in \mathbb{R}$, and we write $\partial f(0) = \{z \in \mathbb{R} \mid |z| \leq 1\}$.

2.2 Sparse estimation

Sparse estimation is used in signal processing and machine learning, and it allows for the identification of a small number of essential variables from a large set. This

approach is particularly useful when the number of observed variables exceeds the sample size, a situation that is often challenging in traditional statistics. Sparse estimation is frequently employed in high-dimensional data analysis.

2.2.1 Lasso

Lasso is one of the well-known models for sparse estimation, involving the addition of a regularization term to the model's loss function. In the case of linear regression, lasso solves the following problem:

$$\min_{\beta \in \mathbb{R}^n} \frac{1}{2m} \|y - X\beta\|_2^2 + \gamma \|\beta\|_1 \tag{2.4}$$

Here, $X \in \mathbb{R}^{m \times n}$ is the data matrix, $y \in \mathbb{R}^m$ is the target variable, and $\gamma > 0$ is the regularization parameter. By adding the L_1-norm, the coefficient vector β is shrunk towards zero. In particular, as the regularization parameter increases, more elements of the estimated β become zero. The graph of $\|\beta\|_1$ in two dimensions is shown in Figure 2.1. The L_1-norm is non-differentiable when each component is zero and has sharp corners, making it prone to shrinking towards zero. Therefore, using lasso can result in sparse solutions. Additionally, the problem (2.4) can be efficiently solved using the coordinate descent method [12].

2.2.2 Group lasso

In the field of life sciences, it is known that groups of genes may share the same biological functions. In cases where several variables are pre-assigned into groups and one wishes to collectively select or discard these groups, group lasso is used. Group lasso is formulated as follows:

$$\min_{\beta \in \mathbb{R}^n} \frac{1}{2m} \|y - X\beta\|_2^2 + \gamma \sum_{j=1}^{J} \|\beta_{I_j}\|_2 \tag{2.5}$$

Here, I_j are index sets such that $I_j \cap I_k = \emptyset$ $(j \neq k)$. The L_2-norm is non-differentiable only when all components of β are zero, as illustrated in Figure 2.2. Therefore, β is shrunk such that all components within a group become zero simultaneously, allowing for group-wise variable selection.

2.3 Proximal gradient method

The proximal gradient method finds the minimum solution of an objective function expressed by the sum of convex functions $f, g : \mathbb{R}^n \to \mathbb{R}$ such that f is

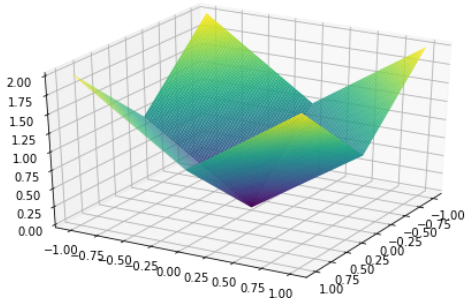

Figure 2.1: L_1-norm

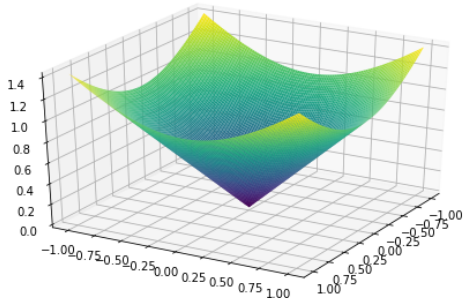

Figure 2.2: L_2-norm

differentiable and g, which is not necessarily differentiable. We define the functions

$$Q_\eta(x, y) := f(y) + \langle x - y, \nabla f(y)\rangle + \frac{1}{2\eta}\|x - y\|_2^2 + g(x) \tag{2.6}$$

$$p_\eta(y) := \underset{x}{\operatorname{argmin}} \quad Q_\eta(x, y) \tag{2.7}$$

for $\eta > 0$, and generate the sequence $\{x_k\}$ via

$$x_{k+1} \leftarrow p_\eta(x_k) \tag{2.8}$$

from the initial value x_0 until convergence to obtain the solution. If we define the *proximal map* w.r.t. $g : \mathbb{R}^n \to \mathbb{R}$ by

$$\operatorname{prox}_g(y) = \underset{x}{\operatorname{argmin}} \left\{ g(x) + \frac{1}{2}\|y - x\|_2^2 \right\} \tag{2.9}$$

then (2.7) can be expressed by

$$\begin{aligned} p_\eta(y) &= \underset{x}{\operatorname{argmin}}\, Q_\eta(x, y) \\ &= \underset{x}{\operatorname{argmin}} \left\{ \langle x - y, \nabla f(y)\rangle + \frac{1}{2\eta}\|x - y\|_2^2 + g(x) \right\} \\ &= \underset{x}{\operatorname{argmin}} \left\{ g(x) + \frac{1}{2\eta}\|x - y - \eta\nabla f(y)\|_2^2 \right\} \\ &= \operatorname{prox}_{\eta h}(y - \eta\nabla f(y)) \end{aligned} \tag{2.10}$$

In each iteration, the proximal gradient seeks x that minimizes the sum of the quadratic approximation of $g(x)$ around x_k and $h(x)$. The ISTA (iterative shrinkage-thresholding algorithm) procedure obtains $O(k^{-1})$ accuracy for the number of updates k [11]. We may replace ISTA by the faster procedure below: using the

sequence $\{\alpha_k\}$ such that $\alpha_0 = 1, \alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_i^2}}{2}$, generates $\{x_k\}$ and $\{y_k\}$ via the equations

$$x_k = p_\eta(y_k)$$
$$y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(x_k - x_{k-1})$$

from the initial value $y_1 = x_0$ until convergence to obtain the solution. Note that the quantity $\frac{\alpha_k - 1}{\alpha_{k+1}}$ is zero when $k = 1$, increases with k, and converges to one as $k \to \infty$. It behaves similarly to ISTA when k is small, and accelerates the updates when k increases to gain efficiency. The FISTA (fast iterative shrinkage-thresholding algorithm) procedure obtains $O(k^{-2})$ accuracy for the number of updates k [11]. This paper mainly uses the FISTA.

Even if it is updated using the formulas given by ISTA and FISTA, they do not necessarily converge to x, which minimizes the objective function unless we choose an appropriate parameter η. In the following, we assume that ∇f is Lipschitz continuous, which means that there exists $L_f > 0$ such that for arbitrary x, y,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L_f \|x - y\|_2. \tag{2.11}$$

It is known that ISTA and FISTA converge to x that minimizes $F(x)$ if we choose $\eta > 0$ as $0 < \eta \leq \frac{1}{L_f}$ [11].

2.4 Proximal Newton method

The proximal gradient method finds the minimum solution of an objective function expressed by the sum of convex functions $f, g : \mathbb{R}^n \to \mathbb{R}$ such that f is twice differentiable and g, which is not necessarily differentiable. In each iteration k, the proximal Newton method approximates $f(x)$ as

$$f^{(k)}(x) := f(x^{(k)}) + \langle x - x^{(k)}, \nabla f(x^{(k)}) \rangle + \frac{1}{2} \left(x - x^{(k)}\right)^T \nabla^2 f(x^{(k)}) \left(x - x^{(k)}\right) \tag{2.12}$$

and generates the sequence $\{x^{(k)}\}$ via

$$\tilde{x}^{(k)} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f^{(k)}(x) + g(x) \tag{2.13}$$
$$x^{(k+1)} = x^{(k)} + \eta_k(\tilde{x}^{(k)} - x^{(k)}), \quad 0 < \eta_k \leq 1 \tag{2.14}$$

from the initial value $x^{(0)}$ until convergence to obtain the solution. In this thesis, we consider only the case where $\eta_k = 1$. If $g(x) = \lambda \|x\|_1$ $(\lambda \geq 0)$, subproblem

(2.13) is equivalent to the lasso problem [3], and x can be updated efficiently via the coordinate descent method.

However, if $g(x) = \lambda \sum_{j=1}^{J} \|x_{I_j}\|_2$ where I_j $(j = 1, \ldots, J)$ are index sets, the subproblem (2.13) is equivalent to a group lasso problem [5], which is solved by using the proximal gradient method; hence, the high convergence speed of the proximal Newton method cannot be realized.

2.5 ADMM and its generalization

Let $A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}, c \in \mathbb{R}^p$, and $f, g : \mathbb{R}^n \to \mathbb{R}, h : \mathbb{R}^m \to \mathbb{R}$ be convex. We consider the convex optimization

$$\begin{aligned} &\min_{x,y} \quad f(x) + g(x) + h(y) \\ &\text{subject to} \quad Ax + By = c\,. \end{aligned} \tag{2.15}$$

When we apply the ADMM, for the convex optimization formulated as in (2.15), we define the augmented Lagrangian

$$L_\nu(x, y, \lambda) = f(x) + g(x) + h(y) + \langle \lambda, Ax + By - c\rangle + \frac{\nu}{2}\|Ax + By - c\|_2^2 \tag{2.16}$$

for $\nu > 0$, and repeatedly update via the equations

$$\begin{aligned} x^{(k+1)} &= \operatorname*{argmin}_x \; L_\nu(x, y^{(k)}, \lambda^{(k)}) \\ y^{(k+1)} &= \operatorname*{argmin}_y \; L_\nu(x^{(k+1)}, y, \lambda^{(k)}) \\ \lambda^{(k+1)} &= \lambda^{(k)} + \nu(Ax^{(k+1)} + By^{(k+1)} - c) \end{aligned}$$

from the initial values $y^{(0)}$ and $\lambda^{(0)}$ until convergence to obtain the solution.

An extension of the ADMM method modifies the update formula with positive semidefinite matrices $Q \succeq \mathcal{O}, P \succeq \mathcal{O}$ as follows:

$$\begin{aligned} x^{(k+1)} &= \operatorname*{argmin}_{x\in\mathbb{R}^n} \left\{ L_\nu(x, y^{(k)}, \lambda^{(k)}) + \frac{1}{2}(x - x^{(k)})^T Q(x - x^{(k)}) \right\}, \\ y^{(k+1)} &= \operatorname*{argmin}_{y\in\mathbb{R}^m} \left\{ L_\nu(x^{(k+1)}, y, \lambda^{(k)}) + \frac{1}{2}(y - y^{(k)})^T P(y - y^{(k)}) \right\}, \\ \lambda^{(k+1)} &= \lambda^{(k)} + \nu(Ax^{(k+1)} + By^{(k+1)} - c). \end{aligned}$$

The advantage of this extension is that, by using $\eta_1 \geq \nu\lambda_{\max}(A^TA), \eta_2 \geq \nu\lambda_{\max}(B^TB)$ and setting $Q = \eta_1 I - \nu A^TA \succeq \mathcal{O}, P = \eta_2 I - \nu B^TB \succeq \mathcal{O}$, the updates for x, y

become diagonalized, resulting in

$$x^{(k+1)} = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ f(x) + g(x) \right.$$
$$\left. + \frac{\eta_1}{2} \left\| x - \left(x^{(k)} - \frac{\nu}{\eta_1} A^T (Ax^{(k)} + By^{(k)} - c + \lambda^{(k)}/\nu) \right) \right\|_2^2 \right\},$$

$$y^{(k+1)} = \underset{y\in\mathbb{R}^m}{\operatorname{argmin}} \left\{ h(y) \right.$$
$$\left. + \frac{\eta_2}{2} \left\| y - \left(y^{(k)} - \frac{\nu}{\eta_2} B^T (Ax^{(k+1)} + By^{(k)} - c + \lambda^{(k)}/\nu) \right) \right\|_2^2 \right\}.$$

This makes it unnecessary to consider the diagonal components of the quadratic forms, simplifying the solution of many problems. Particularly, the methods for updating x, y as in the above equations are referred to as the Alternating Proximal Gradient Method (APGM)[14].

2.6 Alternating minimization algorithm

The Alternating Minimization Algorithm (AMA) [15] assumes that either f or g is strongly convex and slightly modifies the x-update in ADMM. Specifically, the update for x involves setting a positive optimization parameter to 0. The algorithm is repeated as follows:

$$x^{(k+1)} = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ L_0(x, y^{(k)}, \lambda^{(k)}) \right\},$$
$$y^{(k+1)} = \underset{y\in\mathbb{R}^m}{\operatorname{argmin}} \left\{ L_\nu(x^{(k+1)}, y, \lambda^{(k)}) \right\},$$
$$\lambda^{(k+1)} = \lambda^{(k)} + \nu(Ax^{(k+1)} + By^{(k+1)} - c).$$

Although it is necessary for either f or g to be strongly convex, AMA has the potential to solve problems faster than ADMM. In fact, for convex clustering, it can solve significantly faster than ADMM[16]. This method is equivalent to the proximal gradient method for the dual problem, so if ν is too large, it may not converge, necessitating appropriate optimization parameter settings.

2.7 Iterative methods and convergence rates

Many optimization algorithms for sparse estimation problems are expressed as iterative methods. Here, an iterative method involves generating a sequence con-

verging to the optimal solution by updating from an initial value $x^{(0)}$ to $x^{(0)}, \ldots, x^{(k)}$. It is necessary to choose a method for selecting the initial value $x^{(0)}$ and for updating the sequence so that the generated sequence $\{x^{(k)}\}$ converges to the optimal solution as quickly as possible. In this thesis, we define measures for evaluating the convergence speed as follow:

Definition 1. *Let $\{x^{(k)}\} \subset \mathbb{R}^n$ converge to x^*, and let $x^{(k)} \neq x^*$ for all k. Then we say that $x^{(k)}$ converges to x^**

1. *linearly if*

$$\limsup_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|_2}{\|x^{(k)} - x^*\|_2} < 1 \tag{2.17}$$

2. *superlinearly if*

$$\lim_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|_2}{\|x^{(k)} - x^*\|_2} = 0 \tag{2.18}$$

3. *quadratically if*

$$\limsup_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|_2}{\|x^{(k)} - x^*\|_2^2} < \infty \tag{2.19}$$

If $\{x^{(k)}\}$ converges to x^* linearly, then there exists constants $C > 0, r \in (0, 1)$ such that

$$\|x^{(k)} - x^*\|_2 < Cr^k$$

For superlinear convergence, there exists a sequence $\{r^{(k)}\}$ with $r^{(k)} \searrow 0$ as $k \to \infty$ such that

$$\|x^{(k)} - x^*\|_2 < C \prod_{j=1}^{k} r^{(j)}$$

Furthermore, for quadratic convergence, there exists constants $C > 0, r \in (0, 1)$ such that

$$\|x^{(k)} - x^*\|_2 < Cr^{2^k}$$

Thus, it is evident that quadratic convergence is faster than linear convergence.

The optimization algorithms used in sparse estimation vary in convergence speed depending on the problem setting, but can be broadly classified into first-order methods, which converge linearly, and second-order methods, which converge superlinearly or quadratically, as shown in Table 2.1. The characteristic of first-order methods is that they have a low computational cost per iteration and

can solve some problems quickly. However, they require many iterations due to slow convergence towards the solution, and their speed greatly varies depending on the optimization parameters. On the other hand, second-order methods have a high computational cost per iteration, but converge quickly, allowing for high-precision solutions with fewer iterations. This doctoral thesis discusses first-order methods in Chapter 3 and second-order methods in Chapter 4.

Table 2.1: Classification of optimization methods

First-Order Methods	Second-Order Methods
Proximal Gradient Method	Proximal Newton Method
Coordinate Descent Method	Proximal Quasi-Newton Method
ADMM	Interior Point Method
Chapter 3	Chapter 4

Chapter 3

Converting ADMM to the proximal gradient method

3.1 Introduction

In this chapter, we focus on the efficiency of sparse estimation procedures. In particular, we are motivated by the following observation. Two main approaches to finding the solution of lasso are the proximal gradient method and ADMM (alternating direction method of multipliers)[17, 18]. For detailed descriptions of the two procedures. We wonder why some procedures, such as fused lasso and graphical lasso, use ADMM, while others, such as group lasso and convex clustering, use the proximal gradient method. It seems that the proximal gradient method is more efficient than ADMM because efficient modifications such as the fast iterative shrinkage-thresholding algorithm (FISTA)[11] can be easily used for the former whereas inverse matrix computation is inevitable for the latter. The main contribution of this chapter is the following claim:

> The sparse estimation procedure that is realized by ADMM can be transformed to a sparse estimation procedure that is realized by the proximal gradient method as long as its Lipschitz constant exists.

This implies that sparse estimation will be improved if the proximal gradient-based procedure with a Lipschitz constant is more efficient than the ADMM-based procedure.

The Lipschitz condition is satisfied by adding a regularization term such as the L_1, L_2-norm to the loss function of linear regression, such as Lasso, Sparse group lasso, Sparse convex clustering, and trend filtering. The Lipschitz condition is also satisfied by adding a regularization term to the loss function of logistic loss or cox regression, which is used when the target variable is binary or multilevel.

The remainder of this chapter is organized as follows. Section 3.2 presents work that is related to the results in this chapter. Section 3.3 derives a general method for converting a problem that is solved by ADMM to a problem that is solved by the proximal gradient method. Sections 3.4 and 3.5 apply it to sparse convex clustering[19] and trend filtering[20] to evaluate its performance. Finally, Section 3.6 summarizes the results of this chapter.

3.2 Related work

The optimization problem that is considered in this chapter is

$$\min_{x} \ f(x) + g(x) + h(Ax) \ . \tag{3.1}$$

for convex $f, g : \mathbb{R}^n \to \mathbb{R}, h : \mathbb{R}^m \to \mathbb{R}$ and $A \in \mathbb{R}^{m \times n}$, where f is differentiable, ∇f is Lipschitz continuous with parameter $L_f > 0$, and h is a closed convex function. For example, in sparse convex clustering[19], f is the loss function, and g, h are regularization terms (constraints). The dual problem for (3.1) is

$$\min_{y \in \mathbb{R}^m} \ (f + g)^*(-A^T y) + h^*(y) \ , \tag{3.2}$$

where $(f + g)^*, h^*$ are the conjugate functions of $f + g, h$.

For optimization problems such as (3.1), sparse estimation often uses the proximal gradient method when, for example, $h \equiv 0$, and ADMM otherwise. Although the implementation of ADMM is simple and it can be applied to various problems, it is often computationally expensive. For example, we often need to compute the inverse matrix to solve the optimization problem. To simplify the computation, there are generalized ADMM [14, 21] which apply the proximal gradient method to ADMM. Nevertheless, the convergence becomes slow when n is large.

In addition, we may use the alternating minimization algorithm (AMA)[15, 22], which slightly modifies the ADMM steps. We can regard it as an application of the proximal gradient method to the dual problem (3.2). However, this requires either f or g to be strongly convex and have narrow applicability. In addition, when row m of A is large, it becomes a proximal gradient method with many dimensions, and convergence becomes slow.

In this chapter, we apply the proximal gradient method to method of multiplier [23] to convert the ADMM problem to a proximal gradient method problem and solve it. Since the proposed method applies the proximal gradient method to the main problem, it can solve the problem quickly when, for example, m is large.

3.3 The proposed method

(3.1) is equivalent to the following:

$$\begin{aligned} \min_{x,y} \ & f(x) + g(x) + h(y) \\ & \text{subject to } Ax = y\,. \end{aligned} \tag{3.3}$$

If we apply ADMM, then the augmented Lagrangian (2.16) for (3.3) is

$$L_\nu(x, y, \lambda) = f(x) + g(x) + h(y) + \langle \lambda, Ax - y\rangle + \frac{\nu}{2}\|Ax - y\|_2^2 \tag{3.4}$$

for $\nu > 0$.

In the proposed method, we update x, y simultaneously via the

$$\begin{aligned} (x^{(k+1)}, y^{(k+1)}) &= \operatorname*{argmin}_{x,y}\{L_{\nu^{(k)}}(x, y, \lambda^{(k)})\} \\ \lambda^{(k+1)} &= \lambda^{(k)} + (Ax^{(k+1)} - y^{(k+1)}) \end{aligned} \tag{3.5}$$

from the initial value $\lambda^{(0)}$. Although, in general, changing the value ν for each k may improve the performance, we set ν to be constant to proceed with the derivation, making the notation simple.

To update via (3.5), we consider the minimization of

$$\begin{aligned} \phi(x) :&= \min_y L_\nu(x, y, \lambda^{(k)}) \\ &= f(x) + g(x) + \min_y\{h(y) + \langle \lambda^{(k)}, Ax - y\rangle + \frac{\nu}{2}\|Ax - y\|_2^2\} \end{aligned}$$

w.r.t. y.

Theorem 1. *If we define* $\phi_1(x) := f(x) + \min_y\{h(y) + \langle \lambda^{(k)}, Ax - y\rangle + \frac{\nu}{2}\|Ax - y\|_2^2\}$*, then* ϕ_1 *is differentiable and we have*

$$\nabla\phi_1(x) = \nabla f(x) + A^T(\mathrm{prox}_{\nu h^*}(\nu Ax + \lambda^{(k)})). \tag{3.6}$$

Proof. we define the function $\psi(x)$ obtained by removing $f(x), g(x)$ from $\phi(x)$:

$$\begin{aligned} \psi(x) :&= \min_y\{h(y) + \langle \lambda^{(k)}, Ax - y\rangle + \frac{\nu}{2}\|Ax - y\|_2^2\} \\ &= \min_y\{h(y) + \frac{\nu}{2}\|y\|_2^2 - \langle y, \nu Ax + \lambda^{(k)}\rangle\} + \langle \lambda^{(k)}, Ax\rangle + \frac{\nu}{2}\|Ax\|_2^2 \quad (3.7) \\ &= -\max_y\{\langle y, \nu Ax + \lambda^{(k)}\rangle - h(y) - \frac{\nu}{2}\|y\|_2^2\} + \langle \lambda^{(k)}, Ax\rangle + \frac{\nu}{2}\|Ax\|_2^2 \\ &= -r^*(\nu Ax + \lambda^{(k)}) + \langle \lambda^{(k)}, Ax\rangle + \frac{\nu}{2}\|Ax\|_2^2\,, \quad (3.8) \end{aligned}$$

where $r(u) := h(u) + \frac{\nu}{2}\|u\|_2^2$ and $r^*(v) := \sup_u\{\langle u, v\rangle - r(u)\}$. Because the first term of (3.7) can be written as

$$\min_y\{h(y) + \frac{\nu}{2}\|y - (Ax + \frac{\lambda^{(k)}}{\nu})\|_2^2 - \frac{\nu}{2}\|Ax + \frac{\lambda^{(k)}}{\nu}\|_2^2\} \ ,$$

the quantity $h(y) + \langle\lambda^{(k)}, Ax - y\rangle + \frac{\nu}{2}\|Ax - y\|_2^2$ is minimized when

$$y^*(x) = \mathrm{prox}_{\nu^{-1}h}(Ax + \nu^{-1}\lambda^{(k)}) \tag{3.9}$$

where $\mathrm{prox}_{\nu^{-1}h}(\cdot)$ is the proximal map defined in (2.9). Then, we notice the following lemma:

Lemma 1 ([24] Theorem 26.3)**.** *Assume that $s : \mathbb{R}^m \to \mathbb{R}$ is closed and strongly convex. Then, conjugate function s^* is differentiable and $\nabla s^*(v) = \underset{u\in\mathbb{R}^m}{\mathrm{argmax}}\{\langle u, v\rangle - s(u)\}$ for $v \in \mathbb{R}^m$.*

From Lemma1, we have

$$\begin{aligned}
\nabla r^*(v) &= \underset{u}{\mathrm{argmax}}\{\langle u, v\rangle - r(u)\} = \underset{u}{\mathrm{argmax}}\{\langle u, v\rangle - h(u) - \frac{\nu}{2}\|u\|_2^2\} \\
&= \underset{u}{\mathrm{argmin}}\{\frac{1}{2}\|u\|_2^2 + \frac{1}{\nu}h(u) - \langle\frac{v}{\nu}, u\rangle\} \\
&= \underset{u}{\mathrm{argmin}}\{\frac{1}{2}\|u - \frac{v}{\nu}\|_2^2 + \frac{1}{\nu}h(u)\} \\
&= \mathrm{prox}_{h/\nu}(\frac{v}{\nu}) \ .
\end{aligned} \tag{3.10}$$

If we substitute $v = \nu Ax + \lambda^{(k)}$ into (3.10), we have

$$\nabla r^*(\nu Ax + \lambda^{(k)}) = \nu A^T\mathrm{prox}_{h/\nu}(Ax + \nu^{-1}\lambda^{(k)}) \ . \tag{3.11}$$

Moreover, we notice another lemma:

Lemma 2 ([25])**.** *If the function $s : \mathbb{R}^m \to \mathbb{R}$ is convex, then for any $z \in \mathbb{R}^m$ and $\gamma > 0$, we have*

$$\mathrm{prox}_{\gamma s}(z) + \gamma\mathrm{prox}_{s^*/\gamma}(\gamma^{-1}z) = z$$

.

From Lemma 2, (3.8), and (3.11), we have

$$\begin{aligned}
\nabla\phi_1(x) &= \nabla f(x) + A^T\lambda^{(k)} + \nu A^TAx - \nu A^T\mathrm{prox}_{h/\nu}(Ax + \nu^{-1}\lambda^{(k)}) \\
&= \nabla f(x) + A^T(\mathrm{prox}_{\nu h^*}(\nu Ax + \lambda^{(k)})) \ .
\end{aligned}$$

□

Since $\phi(x) = f(x) + g(x) + \psi(x) = \phi_1(x) + g(x)$ can be expressed by the sum of differentiable $\phi_1(x)$ and nondifferentiable $g(x)$, the minimization can be solved via the proximal gradient: update each time via □

$$x^{(l+1)} = \mathrm{prox}_{\eta g}(x^{(l)} - \eta \nabla \phi_1(x^{(l)})) \tag{3.12}$$

(see Section 2.3), where parameter $\eta > 0$ is $\eta \leq \frac{1}{L}$ for $L > 0$ such that

$$\|\nabla \phi_1(x_1) - \nabla \phi_1(x_2)\|_2 \leq L\|x_1 - x_2\|_2$$

Then, convergence is guaranteed.

Lemma 3. *If the function $h : \mathbb{R}^m \to \mathbb{R}$ is convex, then*

$$\|\mathrm{prox}_h(x) - \mathrm{prox}_h(y)\|_2 \leq \|x - y\|_2 \quad \text{for} \quad \forall x, y \in \mathbb{R}^m \, .$$

Proof. Let $u := \mathrm{prox}_h(x), v := \mathrm{prox}_h(y)$. Then, because u minimizes $h(u) + \frac{1}{2}\|x - u\|_2^2$, if we subdifferentiate it by u and equate it to be zero, there exists $s \in \partial h(u)$ such that $s + u - x = 0$. Similarly, we have $t := y - v$ is in $\partial h(v)$. Because the convexity of h means $\langle u - v, s - t \rangle \geq 0$, we have

$$\begin{aligned} \|x - y\|_2^2 &= \|u + s - (v + t)\|_2^2 \\ &= \|u - v\|_2^2 + 2\langle u - v, s - t\rangle + \|s - t\|_2^2 \geq \|u - v\|_2^2 \, . \end{aligned}$$

□

Theorem 2. *If ∇f is Lipschitz continuous with parameter L_f, then for any $x_1, x_2 \in \mathbb{R}^n$, we have*

$$\|\nabla \phi_1(x_1) - \nabla \phi_1(x_2)\|_2 \leq (L_f + \nu \lambda_{\max}(A^T A))\|x_1 - x_2\|_2. \tag{3.13}$$

Proof. From Lemma 3, for $x_1, x_2 \in \mathbb{R}^n$, we have

$$\begin{aligned} &\|A^T(\mathrm{prox}_{\nu h^*}(\nu A x_1 + \lambda^{(k)})) - A^T(\mathrm{prox}_{\nu h^*}(\nu A x_2 + \lambda^{(k)}))\|_2 \\ &\leq \sqrt{\lambda_{\max}(A^T A)} \times \|\mathrm{prox}_{\nu h^*}(\nu A x_1 + \lambda^{(k)}) - \mathrm{prox}_{\nu h^*}(\nu A x_2 + \lambda^{(k)})\|_2 \\ &\leq \sqrt{\lambda_{\max}(A^T A)} \times \|\nu A x_1 + \lambda^{(k)} - (\nu A x_2 + \lambda^{(k)})\|_2 \\ &\leq \nu \lambda_{\max}(A^T A)\|x_1 - x_2\|_2. \end{aligned} \tag{3.14}$$

Thus, when ∇f is Lipschitz continuous with parameter L_f, we have

$$\|\nabla \phi_1(x_1) - \nabla \phi_1(x_2)\|_2 \leq (L_f + \nu \lambda_{\max}(A^T A))\|x_1 - x_2\|_2, \tag{3.15}$$

□

Algorithm 1 (FISTA for $\min \phi(x)$)
Input :$z^{(0)}$, output :$z^{(\infty)}$

1. Initialize $u^{(1)} = z^{(0)}, \eta \in (0,1), \alpha_1 = 1$.

   For $j = 1, 2, \ldots$

2. (Update z)
$$z^{(j)} = \mathrm{prox}_{\eta g}(u^{(j)} - \eta \nabla \phi_1(u^{(j)}))$$

3. (Update α and u)
$$\alpha_{j+1} = \frac{1 + \sqrt{1 + 4\alpha_j^2}}{2}$$
$$u^{(j+1)} = z_j + \frac{\alpha_j - 1}{\alpha_{j+1}}(z_j - z_{j-1})$$

4. Repeat Steps 2-3 until convergence to obtain $z = z^{(\infty)}$.

In Theorem 2, the proximal gradient converges for $\eta := 1/(L_f + \nu\lambda_{\max}(A^T A))$. Hence, it is possible to solve (3.5) efficiently when ∇f is Lipschitz continuous.

The procedure (3.12) is not as efficient as FISTA [11]. We show the modification to FISTA in Algorithm 1.

Similarly, if we put $z = \nu^{-1}\lambda^{(k)} + Ax, \gamma = \nu^{-1}$ in Lemma 2, from (3.9), we obtain

$$\lambda^{(k)} + \nu(Ax - y^*(x)) = \mathrm{prox}_{\nu h^*}(\nu Ax + \lambda^{(k)}) \,. \tag{3.16}$$

Thus, the update of λ is

$$\lambda^{(k+1)} = \mathrm{prox}_{\nu h^*}(\lambda^{(k)} + \nu A x^{(k+1)}) \tag{3.17}$$

and we do not have to update the value of y because it is not required to update x, λ.

We show the actual procedure in Algorithm 2.

3.4 Application to sparse convex clustering

Let $X_{1\cdot}, X_{2\cdot}, \ldots, X_{n\cdot} \in \mathbb{R}^p$ be the n observations w.r.t. p variables. Let $U_{i\cdot}$ and u_j be the row and column vectors of a matrix $U \in \mathbb{R}^{n \times p}$.

Algorithm 2 (Proposed Method for solveing (3.3))
Input: $x^{(1)}, \lambda^{(1)}$, output: $x^{(\infty)}, \lambda^{(\infty)}$

1. Initialize $\nu > 0$.

   For $k = 1, 2, \ldots$

2. (Update x)

   Give $x^{(k)}$ as input to Algorithm 1 and take as output the value $x^{(k+1)}$.

3. (Update λ)
$$\lambda^{(k+1)} = \mathrm{prox}_{\nu h^*}(\lambda^{(k)} + \nu A x^{(k+1)})$$

4. Repeat Steps 2-3 until convergence to obtain $x = x^{(\infty)}$ and $\lambda = \lambda^{(\infty)}$.

The optimization of sparse convex clustering [19] is formulated as follows:

$$\min_U \ \frac{1}{2}\sum_{i=1}^{n}\|X_{i\cdot} - U_{i\cdot}\|_2^2 + \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{(i,j)}\|U_{i\cdot} - U_{j\cdot}\|_2 + \gamma_2 \sum_{j=1}^{p} r_j \|u_j\|_2 \ , \quad (3.18)$$

where $\gamma_1, \gamma_2 \geq 0$ are the regularized parameters, $w_{(i,j)}$ and $r_j \geq 0$ are nonnegative constants (weights), and $\mathcal{E} = \{(i,j); w_{ij} > 0, 1 \leq i < j \leq n\}$.

The objective function is the sum of the convex clustering's objective function and the group lasso regularization term. Since all the elements associated with u_j are expected to become zeros simultaneously when γ_2 is large, sparse convex clustering can choose relevant variables for clustering.

To apply the proposed method, we rewrite (3.18) as follows.

$$\begin{aligned} &\min_U \ \frac{1}{2}\sum_{i=1}^{n}\|X_{i\cdot} - U_{i\cdot}\|_2^2 + \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{(i,j)}\|v_{(i,j)}\|_2 + \gamma_2 \sum_{j=1}^{p} r_j \|u_j\|_2 \qquad (3.19)\\ &\text{subject} \quad \text{to} \quad U_{i\cdot} - U_{j\cdot} - v_{(i,j)} = 0 \quad ((i,j) \in \mathcal{E}) \end{aligned}$$

We note that the optimization with the constraints above is equivalent to the minimization of the augmented Lagrangian below:

$$\begin{aligned} L_\nu(U, V, \Lambda) =& \frac{1}{2}\sum_{i=1}^{n}\|X_{i\cdot} - U_{i\cdot}\|_2^2 + \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{(i,j)}\|v_{(i,j)}\|_2 + \gamma_2 \sum_{j=1}^{p} r_j \|u_j\|_2 \\ &+ \sum_{(i,j)\in\mathcal{E}} \langle \lambda_{(i,j)}, v_{(i,j)} - U_{i\cdot} + U_{j\cdot} \rangle + \frac{\nu}{2} \sum_{(i,j)\in\mathcal{E}} \|v_{(i,j)} - U_{i\cdot} + U_{j\cdot}\|_2^2 \end{aligned}$$

3.4.1 Application of the proposed method

In the following, we define $A_{\mathcal{E}}$ by $A_{\mathcal{E}}U = (u_{i,k} - u_{j,k})_{(i,j)\in\mathcal{E},k=1,\dots,p}$, and denote $\langle B, C\rangle = \mathrm{trace}(B^T C)$ for matrix B, C. If we define

$$f(U) := \frac{1}{2}\sum_{i=1}^{n}\|X_{i\cdot} - U_{i\cdot}\|_2^2 = \frac{1}{2}\|X - U\|_F^2 \tag{3.20}$$

$$g(U) := \gamma_2 \sum_{j=1}^{p} r_j \|u_j\|_2 \tag{3.21}$$

$$h(V) := \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{(i,j)}\|v_{(i,j)}\|_2 \ , \tag{3.22}$$

then we have

$$L_\nu(U, V, \Lambda) = f(U) + g(U) + h(V) + \langle \Lambda, V - A_{\mathcal{E}}U\rangle + \frac{\nu}{2}\|V - A_{\mathcal{E}}U\|_F^2 \ , \tag{3.23}$$

and can apply the proposed method.

Then, we consider the proximal gradient map of h^*. If we define $r(x) := C\|x\|_2$, then we have

$$r^*(y) = \begin{cases} 0 & \text{if} \quad \|y\|_2 \leq C \\ \infty & \text{otherwise} \end{cases} \ , \tag{3.24}$$

which means that for $Z = (z_{(i,j),k})_{(i,j)\in\mathcal{E},k=1,\dots,p}$, we have

$$h^*(Z) = \begin{cases} 0 & \text{if} \quad \|z_{(i,j)}\|_2 \leq \gamma_1 w_{(i,j)} \quad \text{for} \quad \forall (i,j) \in \mathcal{E} \\ \infty & \text{otherwise} \end{cases} \ . \tag{3.25}$$

Hence, if we map $P_C(Z)$ onto $C = \{Z \in \mathbb{R}^{\mathcal{E}\times p}; \|z_{(i,j)}\|_2 \leq \gamma_1 w_{(i,j)} \quad \text{for} \quad (i,j) \in \mathcal{E}\}$ of Z, we have

$$\mathrm{prox}_{\nu h^*}(\Lambda^{(k)} + \nu A_{\mathcal{E}}U^{(k+1)}) = P_C(\Lambda^{(k)} + \nu A_{\mathcal{E}}U^{(k+1)}) \ .$$

Finally, we consider the constant L such that $\|\nabla_U \phi_1(U_1) - \nabla_U \phi_1(U_2)\|_F \leq L\|U_1 - U_2\|_F$. From Lemma 3, we have

$$\begin{aligned} &\|A_{\mathcal{E}}^T(\mathrm{prox}_{\nu h^*}(\Lambda + \nu A_{\mathcal{E}}U_1) - \mathrm{prox}_{\nu h^*}(\Lambda + \nu A_{\mathcal{E}}U_2))\|_F \\ &\leq \sqrt{\lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}})} \times \|\mathrm{prox}_{\nu h^*}(\Lambda + \nu A_{\mathcal{E}}U_1) - \mathrm{prox}_{\nu h^*}(\Lambda + \nu A_{\mathcal{E}}U_2)\|_F \\ &\leq \sqrt{\lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}})} \times \|\Lambda + \nu A_{\mathcal{E}}U_1 - \Lambda - \nu A_{\mathcal{E}}U_2\|_F \\ &\leq \nu\lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}})\|U_1 - U_2\|_F \ . \end{aligned} \tag{3.26}$$

Since $\nabla_U f(U) = U - X$, we have

$$\begin{aligned}
&\|\nabla_U \phi_1(U_1) - \nabla_U \phi_1(U_2)\|_F \\
&\leq \|\nabla_U f(U_1) - \nabla_U f(U_2)\|_F + \|A_{\mathcal{E}}^T(\mathrm{prox}_{\nu h^*}(\Lambda + \nu A_{\mathcal{E}} U_1) - \mathrm{prox}_{\nu h^*}(\Lambda + \nu A_{\mathcal{E}} U_2))\|_F \\
&\leq \|U_1 - U_2\|_F + \nu \lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}}) \|U_1 - U_2\|_F \,, \qquad (3.27)
\end{aligned}$$

which means that the Lipshitz constant of $\nabla_U \phi_1(U)$ is upperbounded by $1 + \nu \lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}})$. For the derivation of $\lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}})$ and the setting of parameter η, see Appendix A.1.

3.4.2 Experiments

We constructed all the programs via Rcpp[1]. The AMA is an alternative to the ADMM such that the first step $x^{(k+1)} = \mathrm{argmin}_x L_\nu(x, y^{(k)}, \lambda^{(k)})$ is replaced by $x^{(k+1)} = \mathrm{argmin}_x L_0(x, y^{(k)}, \lambda^{(k)})$ in Section 2.2. While the differences between the two algorithms appear to be minor, complexity analysis and numerical experiments show AMA to be significantly more efficient [16].

In all experiments, parameter of proposed method is $\eta = \frac{1}{1+\nu_k \max_i G_{ii}}$ in Appendix A.1 and $\nu_1 = 1, \nu_{k+1} = 1.1\nu_k$.Furthermore, the number of features that affect the clusters was set to $p_{true} = 20$.

The data were set to $n = 1,000$ and $p = 500$, and 250 data points were generated independently from each of the Gaussian distributions with four different means. As parameters, w_{ij} used $\phi = \frac{0.5}{p}$, $k = 5$, and v_i was set to 1. Figure 3.1 shows the change in calculation time when we fix $\gamma_2 = 10$ and change γ_1. In Figure 1, we can see that the computation time of AMA changes significantly when γ_1 changes. In particular, the AMA takes up to 230 seconds when γ_1 is larger than 5, i.e., when the size of each cluster is large. However, the computation time of the proposed method is stable even when γ_1 changes. Furthermore, the maximum computation time is only about 10 seconds for all γ_1, indicating that the computation time can be reduced.

Figure 3.2 shows the change in the calculation time when we fix $\gamma_1 = 10$ and change γ_2 for the same data. We can see that when we change γ_2, the calculation time of AMA changes greatly depending on the value of γ_2, similar to the γ_1 case. In particular, the AMA takes a long time when γ_2 is small, i.e., when the result has few zeros and is not sparse, and the maximum time is about 350 seconds. When γ_2 is large and the solution is sparse, AMA takes less time to compute. In the proposed method, the fluctuation of the calculation time due to γ_2 is small, and the calculation time is shorter than that of AMA for all γ_1.

[1]The source code used in the experiments is available at `https://github.com/Theveni/SCC_TF`.

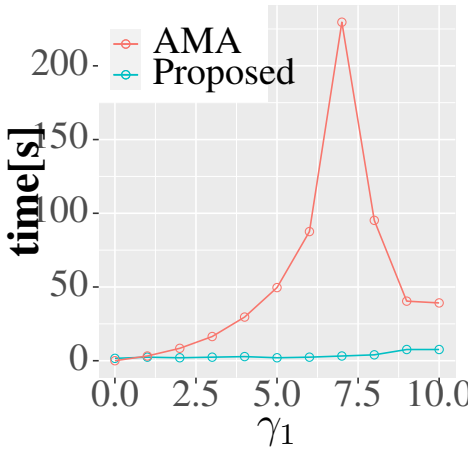

Figure 3.1: The changes in computation time due to γ_1

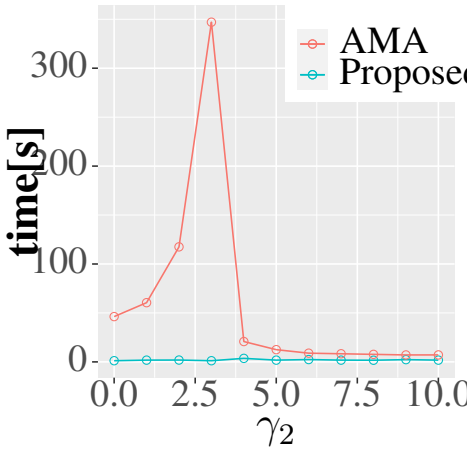

Figure 3.2: The changes in computational time due to γ_2

Moreover, we show in Figure 3.3 comparison of computation times for propothd method, generalized ADMM, AMA when we fix $p = 500, \gamma_1 = 10, \gamma_2 = 10$ and change the number n of data. AMA-FISTA is the calculation time when FISTA is applied to AMA. The data are generated by $\frac{n}{5}$ from a Gaussian distribution with five different means. In Figure 3.3, both AMA and AMA-FISTA show a large increase in computation time with respect to the increase in sample size, and the computation time is larger when the sample size is large than the other methods. The generalized ADMM takes the longest computation time when the sample size is small, but when the sample size becomes large, it can solve the problem more efficiently than AMA and AMA-FISTA. It can be seen that the proposed method has the smallest increase in computation time with increasing sample size, and the computation time is the shortest for all sample sizes.

Furthermore, we show in Figure 3.4 comparison of computation times for propothd method, generalized ADMM, AMA when we fix $n = 500, \gamma_1 = 5, \gamma_2 = 5$ and change the number p of variables. The data were generated by 100 each from a Gaussian distribution with five different means. Both AMA and AMA-FISTA have long computation times when the feature dimension is small, but they have the shortest computation time when the feature dimension is large and the solution is sparse. On the other hand, generalized ADMM has a short computation time when the feature dimension is small, but when the feature dimension is large, the computation time is larger than the other methods. The proposed method has the shortest computation time when the feature dimension is small, and the computation time is almost the same as that of AMA even when the feature dimension is large and sparse, indicating that it can solve the problem efficiently in all cases.

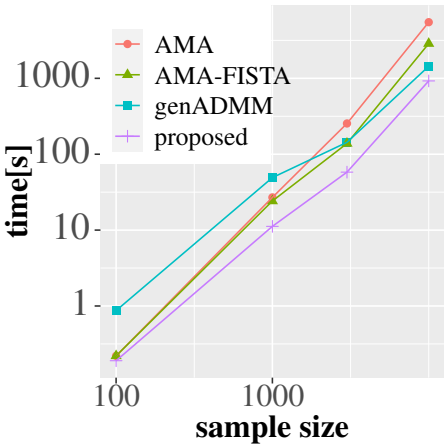

Figure 3.3: The changes in computational time due to the number n of variables

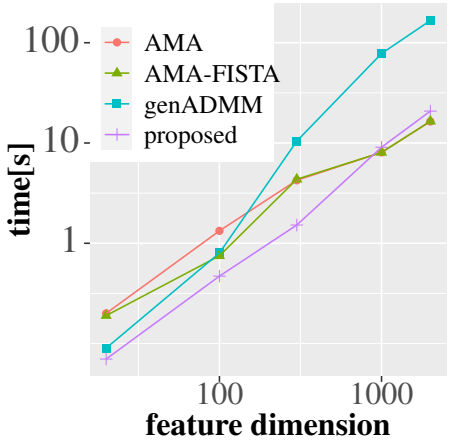

Figure 3.4: The changes in computational time due to the number p of data

3.5 Application to trend filtering

The trend filtering optimization problem is formulated as

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|y-x\|_2^2+\gamma\|D^{(k+1)}x\|_1 \tag{3.28}$$

for an integer $k \geq 0$ and the observed data $y = (y_1, \ldots, y_n)^T \in \mathbb{R}^n$, where $\gamma \geq 0$ is the tuning parameter and $D^{(k+1)}$ is the difference matrix of the order $k+1$ such that

$$D^{(1)} = \begin{pmatrix} -1 & 1 & & & 0 \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & -1 & 1 \end{pmatrix}$$

for $k = 0$, and

$$D^{(k+1)} = D^{(1)}D^{(k)} .$$

In Figures 3.5 and 3.6, we show an example applied to $\sin\theta (0 \leq \theta \leq 2\pi)$ when $k = 1$ and $k = 2$. The points are the observation data, and the solid lines are obtained by smoothing via trend filtering. We observe that the output becomes smoother as the degree k increases.

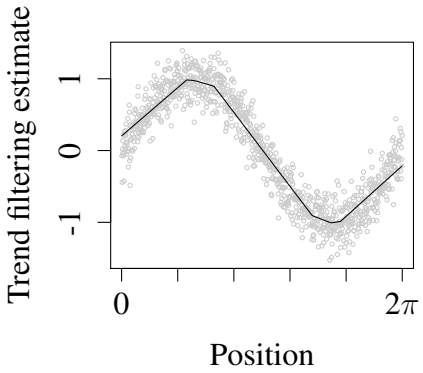

Figure 3.5: Trend filtering with order $k = 1$

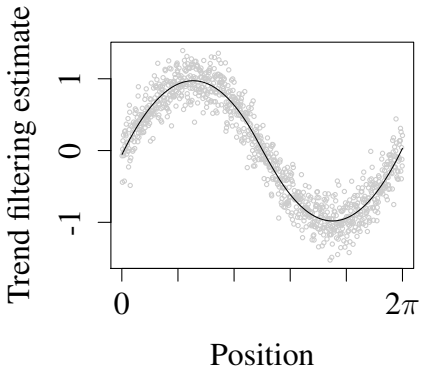

Figure 3.6: Trend filtering with order $k = 2$

3.5.1 Application of the proposed method

To apply the proposed method, we rewrite (3.28) as follows.

$$\begin{aligned} &\min_{x,z} \frac{1}{2}\|y - x\|_2^2 + \gamma\|z\|_1 \\ &\text{subject} \quad \text{to} \quad D^{(k+1)}x = z \end{aligned} \tag{3.29}$$

The augmented Lagrangian becomes

$$L_\nu(x, y, \lambda) = \frac{1}{2}\|y - x\|_2^2 + \gamma\|z\|_1 + \langle \lambda, z - D^{(k+1)}x \rangle + \frac{\nu}{2}\|z - D^{(k+1)}x\|_2^2 \,.$$

If we define

$$f(x) := \frac{1}{2}\|y - x\|_2^2 \tag{3.30}$$

$$g(x) := 0 \tag{3.31}$$

$$h(z) := \gamma\|z\|_1 \,, \tag{3.32}$$

then we have

$$L_\nu(x, z, \lambda) = f(x) + g(x) + h(z) + \langle \lambda, z - D^{(k+1)}x \rangle + \frac{\nu}{2}\|z - D^{(k+1)}x\|_2^2 \,. \tag{3.33}$$

For this case, we have $\mathrm{prox}_{\eta g}(x-\eta\nabla\phi_1(x)) = x-\eta\nabla\phi_1(x)$ due to $g(x) = 0$, and the update of (3.5) is the standard gradient method rather than the proximal gradient. The upper bound of the Lipshitz constant in $\nabla\phi_1(x)$ is $1+\nu\lambda_{\max}((D^{(k+1)})^T D^{(k+1)})$, which can be derived from a similar discussion in Section 3.4. For the evaluation of $\lambda_{\max}((D^{(k+1)})^T D^{(k+1)})$ and setting of parameter $\eta > 0$, see Appendix A.2.

3.5.2 Experiments

We constructed all the programs via Rcpp . Because the purpose of this chapter is to establish the theory of transformation from the ADMM to the proximal gradient we do not relate comparison with an ADMM procedure proposed in [26] that improves performance, considering an efficient computation of the difference matrix.

In all experiments, parameter of proposed method is $\eta = \frac{1}{1+\nu_k 4^{k+1}}$ in Appendix A.2 and $\nu_1 = 1, \nu_{k+1} = 1.1\nu_k$.

We generate $n = 1,000$ data by adding noise to $\sin\theta$, as shown in Figures 3.5, 3.6. Figure 3.7, 3.8 shows the change in calculation time when the value of γ is changed with respect to $k = 1, 2$. For $k = 1$, the computation time increases as γ increases for both ADMM and the proposed method. The computation time of the proposed method is shorter than that of ADMM for all γ, and for large γ, i.e., the computation time is about $\frac{1}{4}$ in the sparse case where $D^{(k)}\beta$ of the solution β has many 0.

In the case of $k = 2$, as in the case of $k = 1$, the computation time of both methods increases as γ increases. When γ is small, the ADMM and the proposed method have similar computation times, but when γ is large, the computation time is $\frac{1}{3}$. The results show that the proposed method is more efficient than ADMM in both cases of $k = 1, k = 2$.

3.6 Summary

In this chapter, we proposed a general method to convert the solution of the optimization problem by ADMM to the solution using the proximal gradient method. In addition, numerical experiments showed that it can be applied to sparse estimation problems such as sparse convex clustering and trend filtering, resulting in significant efficiency improvements.In particular, for both sparse convex clustering and trend filtering, the proposed method is much more efficient than existing methods such as ADMM when the regularization parameter is large such that the results are sparse. This suggests that the proposed method can perform efficient computation by making good use of the sparsity that the result becomes zero.

In applying the proposed method, it is premised that a Lipschtz constant or an

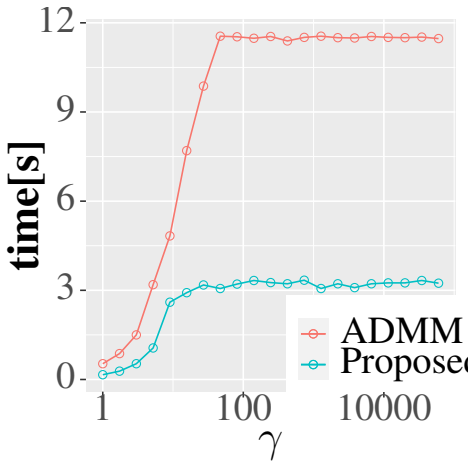

Figure 3.7: The changes in computational time due to γ when $k = 1$

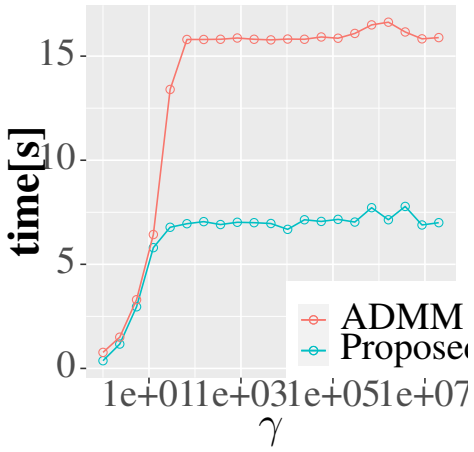

Figure 3.8: The changes in computational time due to γ when $k = 2$

upper bound is obtained. This method is expected to apply not only to existing sparse estimation problems but also to many problems of adding two regularization terms to the loss function. In that case, the problem of finding an efficient solution is reduced to the problem of finding the Lipschtz coefficient.

In this study, we focus on sparse estimation and its surrounding problems, however, it is necessary to actively apply it to optimization problems in general and further clarify its effectiveness.

Chapter 4

Newton-type methods with proximal gradient step

4.1 Introduction

In this chapter, we propose a new method to efficiently solve convex optimization problems encountered in statistics and machine learning, particularly those involving sparse estimation. We consider optimization problems of the form:

$$\min_{x \in \mathbb{R}^n} f(x) + g(x) \tag{4.1}$$

for convex $f : \mathbb{R}^n \to \mathbb{R},\ g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, where f is a loss function that is twice differentiable and μ-strongly convex ($\mu > 0$) and g is a regularization term that is closed convex. We define the strong convexity in Section 2.1. Most sparse estimation problems can be formulated as (4.1). For example, for $\lambda > 0$, $f(x) = \|Ax - b\|_2^2$ and $g(x) = \lambda\|x\|_1$ in lasso [3] and $g(x) = \lambda \sum_{j=1}^{J} \|x_{I_j}\|_2$ where $I_j, j = 1, \ldots, J$ are the index sets in group lasso [5], where $\|\cdot\|_2$ and $\|\cdot\|_1$ are the L_2-norm and L_1-norm, respectively. To solve this problem efficiently, we propose methods to find a fixed point of the proximal gradient method, which can be used more broadly than in sparse estimation.

The proximal gradient and proximal Newton methods are commonly used to solve similar optimization problems, but they have limitations. The proximal gradient method can perform each update quickly, but it converges slowly and requires many updates. On the other hand, the proximal Newton method converges rapidly, but the computational cost of each update becomes high. Moreover, there are some issues with the efficiency of the proximal Newton method, mainly when applied to group sparsity problems.

In recent research, a method to find fixed points of the proximal gradient method has been discussed in [27]. The semismooth Newton method can be used

to solve this problem, and is algorithmically equivalent to the approach proposed in [28, 29, 30]. In addition, stochastic methods have been suggested as an alternative [31, 32]. However, both methods require the Lipschitz constant for the first derivative of the loss function (∇f), and no reports have been made on their convergence when this constant is unknown or absent. In this study, we prove the convergence of the semismooth Newton method when the Lipschitz condition of ∇f is eliminated and extend the theory. Recently, a similar method has been proposed using the semismooth Newton method to find a fixed point of ADMM, which can efficiently obtain high-precision solutions [33, 34].

To overcome these limitations, we propose new methods that find the fixed point of the proximal gradient method efficiently, even when the Lipschitz constant is unknown. We also extend the theory to prove the convergence of the semismooth Newton method under such conditions. Additionally, we introduce a new quasi-Newton method that approximates only the second derivative of the loss function to avoid computing the Hessian matrix and improve efficiency.

The main contributions of this study are: (1) conducting a more detailed analysis of the semismooth Newton method under the assumption of strong convexity, (2) proposing a new quasi-Newton method that avoids computing the Hessian matrix and establishing its superlinear convergence, and (3) demonstrating the efficiency of the proposed methods in solving convex optimization problems encountered in statistics and machine learning through numerical experiments.

In general, our proposed methods offer a more efficient and effective way to solve convex optimization problems encountered in sparse estimation. Especially in sparse estimation techniques such as L_1 regularization and group regularization, our proposed method can efficiently find solutions by performing variable selection using the proximal gradient method with each update.

The remainder of this chapter is organized as follows. In Section 4.2, we provide background knowledge to understand this chapter. Section 4.3 presents works that are related to the results in this chapter. Section 4.4 presents the semismooth Newton method and proves its local convergence. Section 4.5 presents the new quasi-Newton method and proves its local convergence. In Section 4.6, we empirically evaluate the performance of the proposed methods. Finally, Section 4.7 summarizes the results of this chapter.

4.2 Optimality conditions and linear Newton approximations

4.2.1 Optimality conditions

Proposition 1 ([13]). *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a closed convex function. Then, the following are equivalent for all $\nu > 0$:*

$$x^* \in \operatorname*{argmin}_{x \in \mathbb{R}^n} f(x) + g(x) \tag{4.2}$$

$$0 \in \nabla f(x^*) + \partial g(x^*) \tag{4.3}$$

$$x^* = \mathrm{prox}_{\nu g}(x^* - \nu \nabla f(x^*)) \tag{4.4}$$

We define the function $F_\nu : \mathbb{R}^n \to \mathbb{R}^n$ for $\nu > 0$ as

$$F_\nu(x) := x - \mathrm{prox}_{\nu g}(x - \nu \nabla f(x)). \tag{4.5}$$

From Proposition 1, x such that $F_\nu(x) = 0$ minimizes (4.1). Therefore, by solving the nonlinear equation $F_\nu(x) = 0$, we can find x that minimizes (4.1). In this chapter, we consider Newton and quasi-Newton methods that solve $F_\nu(x) = 0$ for all $\nu > 0$. In addition, considering the updating equation of the proximal gradient method

$$x^{(k+1)} = \mathrm{prox}_{\nu g}(x^{(k)} - \nu \nabla f(x^{(k)})), \tag{4.6}$$

F_ν can be interpreted as the difference $F_\nu(x^{(k)}) = x^{(k)} - x^{(k+1)}$ in the proximal gradient method.

4.2.2 Linear Newton approximations

Definition 2. *If $\mathcal{A}(x)$ is a subset of $\mathbb{R}^{n\times n}$ for each $x \in \mathbb{R}^n$, then $\mathcal{A}$ is called a set-valued function, and we write $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$. A set-valued function $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is upper-semicontinuous at $x \in \mathbb{R}^n$ if for any $\epsilon > 0$, there exists $\delta > 0$ such that for all $y \in \mathbb{R}^n$,*

$$\|x - y\|_2 < \delta \;\Rightarrow\; \mathcal{A}(y) \subset \mathcal{A}(x) + \mathbb{B}(O, \epsilon), \tag{4.7}$$

where O is a matrix with all elements zero, $\mathbb{B}(B, \delta) := \{A \in \mathbb{R}^{n\times n} \mid \|A - B\| < \delta\}$, and $\|\cdot\|$ denotes the operator norm.

We will apply the Newton method using the derivative with respect to F_ν defined in (4.5), but, in general, $\mathrm{prox}_{\nu g}$ is not differentiable. However, $\mathrm{prox}_{\nu g}$ is Lipschitz continuous with parameter 1, which means that $\forall x \in \mathbb{R}^n, \forall y \in \mathbb{R}^n, \|\mathrm{prox}_{\nu g}(x) - \mathrm{prox}_{\nu g}(y)\|_2 \leq \|x - y\|_2$. Thus, we define a B-subdifferential, which generalizes the derivative for Lipschitz continuous functions, as follows.

Definition 3. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be Lipschitz continuous. The B-subdifferential of F at $x \in \mathbb{R}^n$ is*

$$\partial_B F(x) = \left\{ V \in \mathbb{R}^{n\times n} \mid \exists \{x^{(k)}\} \subset \mathcal{D}_F, \textit{such that } \; x^{(k)} \to x, \;\; \nabla F(x^{(k)}) \to V \right\}, \tag{4.8}$$

where $\mathcal{D}_F$ is the subset of $\mathbb{R}^n$ for which F is differentiable, i.e., the B-subdifferential is the set of V such that there exists a sequence $\{x^{(k)}\}$ that satisfies the following three conditions: 1. F is differentiable for all $x^{(k)}$, 2. $x^{(k)} \to x$, and 3. $\nabla F(x^{(k)}) \to V$.

If $F : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous, then $\partial_B F(x)$ is a compact and non-empty subset of $\mathbb{R}^{n\times n}$, and the set-valued function $\partial_B F$ is upper-semicontinuous at every $x \in \mathbb{R}^n$ [35, proposition 2.2]. If F is differentiable at x, then $\partial_B F(x) = \{\nabla F(x)\}$. In particular, if $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at x, then $\partial_B (\nabla f(x)) = \{\nabla^2 f(x)\}$. In this chapter, we approximate F_ν defined in (4.5) using $\partial_B \mathrm{prox}_{\nu g}$, which is the B-subdifferential of $\mathrm{prox}_{\nu g}$. Thus, it is important to approximate F_ν, for which we define the following linear Newton approximation.

Definition 4 ([36], Definition 7.5.13)**.** *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be continuous. We say that a set-valued function $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is a linear Newton approximation (LNA) of F at $x \in \mathbb{R}^n$ if $\mathcal{A}$ has compact images and is upper-semicontinuous at x and*

$$\|F(x) - F(y) - A(x - y)\|_2 = o(\|x - y\|_2) \;\; as \;\; y \to x. \tag{4.9}$$

for $y \in \mathbb{R}^n$ and any $A \in \mathcal{A}(y)$. If instead

$$\|F(x) - F(y) - A(x - y)\|_2 = O(\|x - y\|_2^2) \;\; as \;\; y \to x \tag{4.10}$$

for $y \in \mathbb{R}^n$ and any $A \in \mathcal{A}(y)$, then we say that $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is a strong linear Newton approximation (strong LNA) of F at $x \in \mathbb{R}^n$.

If F has an LNA, then there exists a matrix A that can approximate $F(y) - F(x)$. For example, if $F(x) = x$ and $\mathcal{A}(x) = \{I\}$ for any x, then

$$\|x - y - I(x - y)\|_2 = 0$$

for any $x, y \in \mathbb{R}^n$ and $\mathcal{A}$ is a strong LNA of $F(x) = x$ for every $x \in \mathbb{R}^n$. In general, if $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at $x \in \mathbb{R}^n$, then by using $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$, which is the Hessian of f, and setting $\mathcal{B}(x) = \{\nabla^2 f(x)\}$ for any $x \in \mathbb{R}^n$, we find that $\mathcal{B}$ is an LNA of ∇f for every x and

$$\|\nabla f(x) - \nabla f(y) - \nabla^2 f(y)(x-y)\|_2 = o(\|x-y\|_2) \ \ as \ \ y \to x$$

holds. In particular, if $\nabla^2 f$ is Lipschitz continuous, i.e., there exists $L_f > 0$ such that $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L_f \|x - y\|_2$ for any $x, y \in \mathbb{R}^n$, then $\mathcal{B}$ is a strong LNA of ∇f for every x. Here, since I and $\nabla^2 f$ are continuous functions on $\mathbb{R}^n$, it is apparent that both $\mathcal{A}$ and $\mathcal{B}$ are upper semicontinuous. However, if F is not differentiable, we need to determine whether $\partial_B F$ is an LNA of F. In this chapter, we construct an LNA of $\mathrm{prox}_{\nu g}$ using $\partial_B \mathrm{prox}_{\nu g}$.

An LNA has properties similar to those of ordinary derivatives, and the linearity and chain rule can be expressed as follows.

Lemma 4 ([36], Corollaly 7.5.18)**.** *Suppose that set-valued functions $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}, \mathcal{B} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ are (strong) LNAs of $F : \mathbb{R}^n \to \mathbb{R}^n$ and $G : \mathbb{R}^n \to \mathbb{R}^n$, respectively, at $x \in \mathbb{R}^n$. Then,*

$$(\mathcal{A} + \mathcal{B})(y) := \{A + B \mid A \in \mathcal{A}(y), \ B \in \mathcal{B}(y)\}$$

is a (strong) LNA of $F + G$ at x.

Lemma 5 ([36], Theorem 7.5.17)**.** *Suppose that the set-valued function $\mathcal{B} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is a (strong) LNA of $G : \mathbb{R}^n \to \mathbb{R}^n$ at $x \in \mathbb{R}^n$ and that $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is a (strong) LNA of $F : \mathbb{R}^n \to \mathbb{R}^n$ at $G(x)$. Then,*

$$(\mathcal{A}\mathcal{B})(y) := \{AB \mid A \in \mathcal{A}(G(y)), \ B \in \mathcal{B}(y)\}$$

is a (strong) LNA of $F \circ G$ at x, where $F \circ G$ is the composition of the mappings $F \circ G(x) = F(G(x))$.

From Lemmas 4 and 5, as in ordinary differential calculus, when the function for which an LNA is to be obtained is expressed as a sum of multiple functions or their composite map, it is sufficient to consider an LNA of each function. For example, we suppose the set-valued functions $\mathcal{A}, \mathcal{B} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ are (strong) LNAs of $F, G : \mathbb{R}^n \to \mathbb{R}^n$ at $x \in \mathbb{R}^n$ and that $\mathcal{C} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ is a (strong) LNA of $H : \mathbb{R}^n \to \mathbb{R}^n$ at $F(x) + G(x)$. Then,

$$\mathcal{C}(\mathcal{A} + \mathcal{B})(y) := \{C(A + B) \mid C \in \mathcal{C}(F(y) + G(y)), \ A \in \mathcal{A}(y), \ B \in \mathcal{B}(y)\}$$

is a (strong) LNA of $H \circ (F + G)$ at x.

4.2.3 LNA of the proximal map

L_1-norm

If $g(x) = \|x\|_1$, the i-th component of $\mathrm{prox}_{\nu g}$ is

$$\mathrm{prox}_{\nu g}(x)_i = \left(1 - \frac{\nu}{|x_i|}\right)_+ x_i, \tag{4.11}$$

where $(s)_+ = \max\{0, s\}$ for $s \in \mathbb{R}$. (4.11) is differentiable at any $|x_i| \neq \nu$, and its derivative is 0 for $|x_i| < \nu$ and 1 for $|x_i| > \nu$. For the case of $|x_i| = \nu$, if $x_i^{(k)} \to x_i, |x_i^{(k)}| \downarrow \nu$ as $k \to \infty$, then $\nabla \mathrm{prox}_{\nu g}(x^{(k)})_{i,i} \to 1$. In contrast, if $x_i^{(k)} \to x_i, |x_i^{(k)}| \uparrow \nu$ as $k \to \infty$, then $\nabla \mathrm{prox}_{\nu g}(x^{(k)})_i \to 0$. Thus, $\partial_B \mathrm{prox}_{\nu g}(x)$ becomes the set of diagonal matrices for any $x \in \mathbb{R}^n$, and its (i,i)-th component is

$$\partial_B \mathrm{prox}_{\nu g}(x)_{i,i} = \begin{cases} \{0\} & |x_i| < \nu \\ \{1\} & |x_i| > \nu \\ \{0, 1\} & |x_i| = \nu \end{cases}. \tag{4.12}$$

L_2-norm

If $g(x) = \|x\|_2$, $\mathrm{prox}_{\nu g}$ is

$$\mathrm{prox}_{\nu g}(x) = \left(1 - \frac{\nu}{\|x\|_2}\right)_+ x. \tag{4.13}$$

(4.13) is differentiable at any $\|x\|_2 \neq \nu$, and its derivative is O for $\|x\|_2 < \nu$ and $\frac{\nu}{\|x\|_2}(\frac{xx^T}{\|x\|_2^2} - I) + I$ for $\|x\|_2 > \nu$. For the case of $\|x\|_2 = \nu$, if $x^{(k)} \to x, \|x^{(k)}\|_2 \downarrow \nu$ as $k \to \infty$, then $\nabla \mathrm{prox}_{\nu g}(x^{(k)}) \to \frac{\nu}{\|x\|_2}(\frac{xx^T}{\|x\|_2^2} - I) + I$. In contrast, if $x^{(k)} \to x, \|x^{(k)}\|_2 \uparrow \nu$ as $k \to \infty$, then $\nabla \mathrm{prox}_{\nu g}(x^{(k)}) \to O$. Thus, $\partial_B \mathrm{prox}_{\nu g}(x)$ is the set of symmetric matrices for any $x \in \mathbb{R}^n$, and

$$\partial_B \mathrm{prox}_{\nu g}(x) = \begin{cases} \{O\} & \|x\|_2 < \nu \\ \left\{\frac{\nu}{\|x\|_2}(\frac{xx^T}{\|x\|_2^2} - I) + I\right\} & \|x\|_2 > \nu \\ \left\{O, \frac{xx^T}{\|x\|_2^2}\right\} & \|x\|_2 = \nu \end{cases}. \tag{4.14}$$

Lemma 6 ([37], Lemma 2.1)**.** *(4.12) and (4.14) are strong LNAs of* $\mathrm{prox}_{\nu\|\cdot\|_1}$ *and* $\mathrm{prox}_{\nu\|\cdot\|_2}$*, respectively, for any* $x \in \mathbb{R}^n$*.*

From Lemma 6, if g is either the L_1-norm or the L_2-norm, then the B-subdifferential is a strong LNA of $\mathrm{prox}_{\nu g}$. Furthermore, if $\mathrm{prox}_{\nu g}(x)_i$ is 0, i.e. x_i is inactive, then the corresponding component of $\partial_B \mathrm{prox}_{\nu g}$ is 0.

4.3 Related work

The basic idea of our proposed method is to find the fixed points of F_ν using the semismooth Newton method. The semismooth Newton method has been studied for a long time, and theoretical results [38, 39, 40] are well established. Additionally, there has been recent interest in considering the ADMM recursive relation as fixed point iteration and solving it using the semismooth Newton method [34, 33].

In recent research, a method to find the fixed points of the proximal gradient method using the semismooth Newton method has been proposed [27], and it has been extended to stochastic optimization [31, 32]. However, in the paper [27], the condition that f is μ-strongly convex is not required, but an assumption regarding the optimization parameter ν, specifically $\nu \leq 2L_f^{-1}$ for F_ν, is necessary. In fact, Proposition 2.3 (1) in [27] proves that when ∇f is β-cocoercive, then $0 < \nu \leq 2L_f^{-1}$ ensures that F_ν is a monotone operator. They leverage this monotonicity to prove global convergence and construct algorithms. Note that when ∇f is β-cocoercive, ∇f is Lipschitz continuous with a parameter of β^{-1}.

Furthermore, a method to solve optimization problem (4.1) by minimizing the forward-backward-envelope function:

$$\phi_\nu(x) = \min_{y \in \mathbb{R}^n} \{ f(x) + \langle \nabla f(x), y - x \rangle + g(u) + \frac{1}{2\nu} \|y - x\|_2^2 \}$$

has been proposed in [28, 29, 30]. As $\nabla \phi_\nu(x) = (I - \nu \nabla^2 f(x)) F_\nu(x)/\nu$, when $I - \nu \nabla^2 f(x)$ is nonsingular, this method essentially solves $F_\nu(x) = 0$ using the semismooth Newton method. However, the drawback of this method is that when $\nu > L_f^{-1}$, ϕ_ν may not be lower bounded, and furthermore, the assumption $\nu < L_f^{-1}$ is required because $I - \nu \nabla^2 f(x)$ needs to be nonsingular for optimality. Note that although Algorithm 1 in [30] adaptsively updates ν, the condition $\nu < L_f^{-1}$ remains essential for the proof of optimality.

In this chapter, we consider solving $F_\nu(x) = 0$ using the semismooth Newton method. We add the condition that f is μ-strongly convex, but we aim to provide a theoretical extension by removing the conditions related to the optimization parameter ν that were present in existing research. Additionally, we propose a new quasi-Newton method that avoids the computation of the Hessian matrix. This allows for solving the problem efficiently even when the dimension of the variables n is large. While the condition of μ-strong convexity is quite stringent, it can be mitigated by adding a ridge penalty term $\|\cdot\|_2^2$ to the objective function. Furthermore, our method efficiently handles group lasso, which was previously computationally inefficient with the widely used proximal Newton method in packages like glmnet.

4.4 Linear Newton method

Here, we consider the linear Newton method for solving $F_\nu(x) = 0$ for any $\nu > 0$. According to Proposition 1, x such that $F_\nu(x) = 0$ minimizes (4.1). First, we consider an LNA of F_ν to execute the linear Newton method. From Lemmas 4 and 5, since the LNA is linear and satisfies the chain rule, we define the set-valued function $\partial F_\nu : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ as

$$\partial F_\nu(x) = \left\{ I - V(I - \nu\nabla^2 f(x)) \mid V \in \partial_B \mathrm{prox}_{\nu g}(x - \nu\nabla f(x)) \right\}, \qquad (4.15)$$

which is an LNA of $F_\nu(x) = x - \mathrm{prox}_{\nu g}(x - \nu\nabla f(x))$. We show that (4.15) is a (strong) LNA of F_ν as follows.

Proposition 2. *Let $x \in \mathbb{R}^n$. If $\partial_B \mathrm{prox}_{\nu g}$ is an LNA of $\mathrm{prox}_{\nu g}$ at $x - \nu\nabla f(x)$, then ∂F_ν is an LNA of F_ν at x. Furthermore, if $\partial_B \mathrm{prox}_{\nu g}$ is a strong LNA of $\mathrm{prox}_{\nu g}$ at $x - \nu\nabla f(x)$ and $\nabla^2 f$ is Lipschitz continuous, then ∂F_ν is a strong LNA of F_ν at x.*

Proof. $\mathcal{A} : \mathbb{R}^n \ni y \mapsto \{I\}$ is an LNA of $F : \mathbb{R}^n \ni y \mapsto y \in \mathbb{R}^n$ at $x \in \mathbb{R}^n$. Since ∇f is differentiable, $\mathcal{B} : \mathbb{R}^n \ni y \mapsto \{-\nu\nabla^2 f(y)\}$ is an LNA of $G : \mathbb{R}^n \ni y \mapsto -\nu\nabla f(y) \in \mathbb{R}^n$ at x. Thus, from Lemma 4, $\mathcal{A} + \mathcal{B} = \{I - \nu\nabla^2 f\}$ is an LNA of

$$F + G : \mathbb{R}^n \ni y \mapsto y - \nu\nabla f(y) \in \mathbb{R}^n$$

at x. By assumption, $\mathcal{C} : \mathbb{R}^n \ni y \mapsto \partial_B \mathrm{prox}_{\nu g}(y)$ is an LNA of $H : \mathbb{R}^n \ni y \mapsto \mathrm{prox}_{\nu g}(y)$ at $x - \nu\nabla f(x)$. From Lemma 5, if we define the set-valued function $\partial P_\nu : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ as

$$\partial P_\nu(x) := \mathcal{C}(\mathcal{A} + \mathcal{B})(x) = \left\{ V(I - \nu\nabla^2 f(x) \mid V \in \partial_B \mathrm{prox}_{\nu g}(x - \nu\nabla f(x)) \right\},$$

then ∂P_ν is an LNA of $H \circ (F + G) : \mathbb{R}^n \ni y \mapsto \mathrm{prox}_{\nu g}(y - \nu\nabla f(y)) \in \mathbb{R}^n$ at x. Furthermore, applying Lemma 4, we find that ∂F_ν is an LNA of F_ν, which proves the first claim. If $\nabla^2 f$ is Lipschitz continuous, then $\mathcal{B}$ is a strong LNA of G at x, so we conclude that ∂F_ν is a strong LNA of F_ν at x. □

From Proposition 2, if $\partial_B \mathrm{prox}_{\nu g}$ is a (strong) LNA of $\mathrm{prox}_{\nu g}$, since g is L_1 regularization or group regularization, then ∂F_ν is an LNA of F_ν and can approximate F_ν. Thus, we conclude that $\partial_B \mathrm{prox}_{\nu g}$ is an LNA of $\mathrm{prox}_{\nu g}$, which is important for the discussion below. Next, we consider the linear Newton method based on ∂F_ν.

4.4.1 Procedure

From Proposition 2, which states that ∂F_ν is an LNA of F_ν, we can approximate F_ν using ∂F_ν. Specifically, we approximate

$$F_\nu(x) \approx F_\nu^{(k)}(x) := F_\nu(x^{(k)}) + U^{(k)}(x - x^{(k)}), \quad U^{(k)} \in \partial F_\nu(x^{(k)}) \tag{4.16}$$

for $k = 0, 1, 2, \ldots$ with the initial value $x^{(0)} \in \mathbb{R}^n$, and we update $x^{(k+1)}$ such that $F_\nu^{(k)}(x^{(k+1)}) = 0$. We present the procedure of the linear Newton method in Algorithm 3.

Algorithm 3 Linear Newton method

Input: $x^{(0)}$
Output: $x^{(k)}$

1: Initialization :$\nu > 0, k \leftarrow 0$
2: **while** not converged **do**
3: Select $V^{(k)} \in \partial_B \mathrm{prox}_{\nu g}(x^{(k)} - \nu \nabla f(x^{(k)}))$
4: (Obtaining d)

$$d^{(k)} \leftarrow -\left(I - V^{(k)}\left(I - \nu \nabla^2 f(x^{(k)})\right)\right)^{-1} F_\nu(x^{(k)}). \tag{4.17}$$

5: $x^{(k+1)} \leftarrow x^{(k)} + d^{(k)}$
6: $k \leftarrow k + 1$
7: **end while**

In this chapter, we prove the following proposition, namely, Proposition 3, to guarantee that the inverse matrix of $I - V^{(k)}\left(I - \nu \nabla^2 f(x^{(k)})\right)$ always exists and that the update of 4.17 is always possible. The proof is presented in detail in the appendix.

Proposition 3. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are μ-strongly convex and closed convex, respectively. Then, $I - V\left(I - \nu \nabla^2 f(x)\right)$ is a nonsingular matrix for any $x \in \mathbb{R}^n, \nu > 0$ or $V \in \partial_B \mathrm{prox}_{\nu g}(x - \nu \nabla f(x))$, and all eigenvalues are positive real numbers that are greater than or equal to* $\min\{\nu\mu, 1\}$.

It was shown in [27] that if $\nu \leq 2L_f^{-1}$, then all the eigenvalues of $I - V\left(I - \nu \nabla^2 f(x)\right)$ are nonnegative real numbers, but it was not shown whether they are nonsingular or singular. The proposition 3 shows that for general $\nu > 0$, $I - V\left(I - \nu \nabla^2 f(x)\right)$ is always nonsingular for any $x \in \mathbb{R}^n$ if f is μ-strongly convex. Thus, since any element of $\partial F_\nu(x^{(k)})$ is nonsingular for each iteration k, there exists $d^{(k)}$ such that (4.17) is satisfied and the update of Algorithm 3 is always possible.

4.4.2 L_1 regularization

Based on (4.12), we define the diagonal matrix $V^{(k)} \in \partial_B \mathrm{prox}_{\nu g}(x^{(k)} - \nu \nabla f(x^{(k)}))$ as

$$V_{i,i}^{(k)} = \begin{cases} 0, & |x_i^{(k)} - \nu \nabla f(x^{(k)})| \leq \nu\lambda \\ 1, & |x_i^{(k)} - \nu \nabla f(x^{(k)})| > \nu\lambda \end{cases} \tag{4.18}$$

and obtain $I - V^{(k)} \left(I - \nu \nabla^2 f(x^{(k)}\right) \in \partial F_\nu(x^{(k)})$. We define the index sets $\mathcal{I}^{(k)}, \mathcal{O}^{(k)}$ as

$$\mathcal{I}^{(k)} = \{i | V_{ii}^{(k)} = 1\}$$
$$\mathcal{O}^{(k)} = \{i | V_{ii}^{(k)} = 0\}.$$

Then, we can express the matrix as

$$I - V^{(k)}(I - \nu \nabla^2 f(x^{(k)})) = \begin{pmatrix} \nu \nabla^2 f(x^{(k)})_{\mathcal{I}^{(k)}, \mathcal{I}^{(k)}} & \nu \nabla^2 f(x^{(k)})_{\mathcal{I}^{(k)}, \mathcal{O}^{(k)}} \\ O & I \end{pmatrix}, \tag{4.19}$$

where $\nabla^2 f(x^{(k)})_{\mathcal{I}^{(k)}, \mathcal{I}^{(k)}}$ and $\nabla^2 f(x^{(k)})_{\mathcal{I}^{(k)}, \mathcal{O}^{(k)}}$ are the elements of the matrix in $(\mathcal{I}^{(k)}, \mathcal{I}^{(k)})$ and $(\mathcal{I}^{(k)}, \mathcal{O}^{(k)})$, respectively. Therefore, we can update efficiently by eliminating the calculation for the components i such that $V_{i,i}^{(k)} = 0$, i.e., $\mathrm{prox}_{\nu g}(x^{(k)} - \nu \nabla f(x^{(k)})) = 0$.

4.4.3 Convergence

We consider the convergence properties of Algorithm 3.

Theorem 3. *Suppose ∂F_ν is an LNA of F_ν at the optimal solution x^* and that all elements of $\partial F_\nu(x)$ are nonsingular for any $x \in \mathbb{R}^n$. Then, the sequence $\{x^{(k)}\}_{k=1}^\infty$ generated by Algorithm 3 converges locally superlinearly to x^* such that $F_\nu(x^*) = 0$. Moreover, if ∂F_ν is a strong LNA of F_ν at the optimal solution x^*, then the sequence $\{x^{(k)}\}_{i=1}^\infty$ generated by Algorithm 3 converges locally quadratically to x^*.*

Proof. By assumption, since ∂F_ν is a (strong) LNA of F_ν at the optimal solution x^* and A is a nonsingular matrix for any $x \in \mathbb{R}^n$ and $A \in \partial F_\nu(x)$, the proof of this theorem follows from Theorem 2.11 in reference [41] and Theorem 7.5.15 in reference [36]. □

From Proposition 3, if f is μ-strongly convex, all elements of $\partial F_\nu(x)$ are nonsingular for any $x \in \mathbb{R}^n$. Thus, in the cases of L_1 regularization, group regularization, etc., local quadratic convergence is achieved when updating with Algorithm 3 due to Theorem 3. Here, Theorem 3 suggests that the parameter ν of F_ν is arbitrary as long as $\nu > 0$. Therefore, Theorem 3 verifies the convergence of Algorithm 3 in the general case without requiring $\nu \leq 2L_f^{-1}$, as in [27], thus extending the previous results.

4.5 Hybrid linear quasi-Newton method

Algorithm 3 becomes less efficient, especially in dealing with high-dimensional variables, due to the time-consuming nature of computing $\nabla^2 f$. Thus, we consider approximating $\partial F_\nu(x^{(k)})$ in each iteration k. In this chapter, we consider an approximation of $\nabla^2 f(x^{(k)})$ to make the computation of $\partial F_\nu(x^{(k)})$ feasible while maintaining the advantage of the linear Newton method in that the calculation can be omitted when x equals 0. Specifically, we define a new set-valued function $\hat{\partial}^{(k)} F_\nu : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ as

$$\hat{\partial}^{(k)} F_\nu(x^{(k)}) = \left\{ I - V(I - \nu B^{(k)}) \mid V \in \partial_B \mathrm{prox}_{\nu g}(x^{(k)} - \nu \nabla f(x^{(k)})) \right\}$$

By using an approximation matrix $B^{(k)}$ at each iteration and by approximating F_ν as in (4.16).

4.5.1 Procedure

The approximation $\hat{\partial}^{(k)} F_\nu(x^{(k)})$ successfully approximates F_ν if $B^{(k)}$ accurately represents $\nabla^2 f(x^{(k)})$ in each iteration. We start the process with an initial value $B^{(0)} \in \mathbb{R}^{n\times n}$ and update $B^{(k)}$ to satisfy the following secant condition:

$$B^{(k+1)} \left(x^{(k+1)} - x^{(k)} \right) = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}). \tag{4.20}$$

Various update strategies can satisfy this condition, including the Broyden method; however, we employ the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. This update ensures that if f is μ-strongly convex and $B^{(0)}$ is a positive definite symmetric matrix, then $B^{(k)} > O$ holds. Thus, similar to Proposition 3, it can be shown that all elements of $\hat{\partial}^{(k)} F_\nu(x^{(k)})$ are nonsingular for all k. Algorithm 4 outlines the hybrid linear quasi-Newton method procedure.

[27] proposed a method that uses the L-BFGS approach, which approximates ∂F_ν by using the L-BFGS method. The matrix updated with the L-BFGS method becomes symmetric, but ∂F_ν is generally not symmetric. Therefore, this approximation might not be accurate. Additionally, [27] does not provide proof of convergence speed and requires the condition $\nu \leq 2L_f^{-1}$. We have proven that our

quasi-Newton method achieves superlinear convergence, which is a significant advancement.

Algorithm 4 Hybrid linear quasi-Newton method

Input: $x^{(0)}$

Output: $x^{(k)}$

1: Initialization : $\nu > 0, B^{(0)} \in \mathbb{R}^{n\times n}, k \leftarrow 0$

2: **while** not converged **do**

3: Select $V^{(k)} \in \partial_B \mathrm{prox}_{\nu g}(x^{(k)} - \nu\nabla f(x^{(k)}))$

4: (Obtaining d)

$$d^{(k)} \leftarrow -\left(I - V^{(k)}\left(I - \nu B^{(k)}\right)\right)^{-1} F_\nu(x^{(k)}). \tag{4.21}$$

5: $x^{(k+1)} \leftarrow x^{(k)} + d^{(k)}$

6: $y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$

7: (Updating B)

$$B^{(k+1)} \leftarrow B^{(k)} - \frac{B^{(k)} d^{(k)} (d^{(k)})^T B^{(k)}}{(d^{(k)})^T B^{(k)} d^{(k)}} + \frac{y^{(k)} (y^{(k)})^T}{(y^{(k)})^T d^{(k)}}. \tag{4.22}$$

8: $k \leftarrow k + 1$

9: **end while**

4.5.2 Efficiency

In both Algorithms 3 and 4, solving (4.17) and (4.21) requires computing the inverse of an $n \times n$ matrix, which itself requires $O(n^3)$ computations. For large n, this approach is highly inefficient. Therefore, we consider a more efficient approach to find the search direction $d^{(k)}$ that satisfies the linear equation

$$\left(I - V^{(k)}\left(I - \nu B^{(k)}\right)\right) d^{(k)} = -F_\nu(x^{(k)}). \tag{4.23}$$

Among the standard Newton methods, the Newton-CG method is widely used; this method combines the conjugate gradient (CG) method to efficiently solve linear equations. However, the CG method is applicable only to linear equations when the matrix is symmetric. In this case,

$$I - V^{(k)}\left(I - \nu B^{(k)}\right)$$

is generally not symmetric, so the conjugate gradient method cannot be used. Thus, by using the generalized conjugate residual (GCR) method, we find $d^{(k)}$

such that

$$\left\|F_\nu(x^{(k)}) + \left(I - V^{(k)}\left(I - \nu B^{(k)}\right)\right) d^{(k)}\right\|_2 \leq \epsilon^{(k)} \|F_\nu(x^{(k)})\|_2, \tag{4.24}$$

Here, $\epsilon^{(k)} > 0$, such as $\epsilon^{(k)} = \frac{1}{k+1}$, is the tolerance defined by the user. The search direction $d^{(k)}$ is obtained by executing the GCR method until the approximation error is acceptable. Specifically, we can express step 4 of Algorithm 4 as follows.

1. $I - V^{(k)}(I - \nu B^{(k)}) \in \hat{\partial}^{(k)} F_\nu(x^{(k)})$ for $V^{(k)} \in \partial_B \mathrm{prox}_{\nu g}(x^{(k)} - \nu \nabla f(x^{(k)}))$ is chosen; then, the search direction $d^{(k)} \in \mathbb{R}^n$ that satisfies 4.24 is found by using the GCR method.

We specify the GCR method for finding x such that $Ax = b$ for $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$ in Algorithm 5.

Algorithm 5 Generalized conjugate residual method for solving $Ax = b$

Input: $x^{(0)}$
Output: $x^{(k)}$

Initialization : $k \leftarrow 0$
$r^{(0)} \leftarrow b - Ax^{(0)}$
$p^{(0)} \leftarrow r^{(0)}$
while not converged **do**
  $\alpha^{(k)} \leftarrow \frac{\langle Ap^{(k)}, r^{(k)} \rangle}{\langle Ap^{(k)}, Ap^{(k)} \rangle}$
  $x^{(k+1)} \leftarrow x^{(k)} + \alpha^{(k)} p^{(k)}$
  $r^{(k+1)} \leftarrow r^{(k)} - \alpha^{(k)} Ap^{(k)}$
  for $i = 0, \ldots, k$ **do**
    $\beta_{i,k} \leftarrow -\frac{\langle Ap^{(i)}, Ar^{(k+1)} \rangle}{\langle Ap^{(i)}, Ap^{(i)} \rangle}$
  end for
  $p^{(k+1)} \leftarrow r^{(k+1)} + \sum_{i=1}^{k} \beta_{i,k} Ap^{(i)}$
  $k \leftarrow k + 1$
end while

If the GCR method can find $d^{(k)}$ that satisfies (4.24) in a finite number of steps $O(1)$, it can be updated by multiplying a matrix and a vector only, and the GCR method can update by using only computations $O(n^2)$. We can obtain $d^{(k)}$ in fewer updates, which is more efficient than the $O(n^3)$ computations needed to calculate the inverse matrix. Furthermore, it is well known that the convergence of the GCR method tends to be slow when the matrix A is ill conditioned. In many cases, preconditioning is used to address this issue. However, in scenarios such as L_1 regularization and group regularization, where the components of the $\mathcal{O}^{(k)}$ row corresponding to inactive variables, as in Equation (4.19), are the identity matrix

I, applying the GCR method effectively reduces working with a $|\mathcal{I}^{(k)}|\times|\mathcal{I}^{(k)}|$ matrix where $\mathcal{I}^{(k)}$ is the index set of active variables in iteration k. Consequently, it can be computed efficiently. When a linear system is ill conditioned, the optimization problem we aim to solve also becomes ill conditioned. Consequently, because the inverse of the strong convexity parameter μ of the function f is involved, the convergence of proximal gradient methods, proximal Newton methods and even the proposed method can be slow. Therefore, even if we preprocess and speed up the generalized conjugate residual (GCR) method, the overall algorithm may not experience a substantial increase in speed. Moreover, in this case, the ill-conditioning can be mitigated by using a ridge penalty, so we have not considered it in this instance.

4.5.3 Convergence

We consider the convergence properties of Algorithm 4. If $\partial_B \mathrm{prox}_{\nu g}$ is an LNA of $\mathrm{prox}_{\nu g}$, then we can show local linear convergence as follows. In this chapter, we prove the following theorem in the same way as in [40] (see Appendix B.2 for the proof).

Theorem 4. *Suppose that $\partial_B \mathrm{prox}_{\nu g}$ is an LNA of $\mathrm{prox}_{\nu g}$ at $x^* - \nu\nabla f(x^*)$. There exist $\epsilon > 0$ and $\Delta > 0$ such that if $\|x^{(0)} - x^*\|_2 < \epsilon$ and $\|B^{(k)} - \nabla^2 f(x^{(k)})\| < \Delta$ for any $k = 1, 2, \ldots$, then the sequence generated by Algorithm 4 converges locally linearly to x^*.*

From Theorem 4, if $B^{(k)}$ sufficiently approximates $\nabla^2 f(x^{(k)})$ and the initial value $x^{(0)}$ is sufficiently close to the optimal value x^*, then Algorithm 4 exhibits first-order convergence.

Moreover, we can show that Algorithm 4 leads to faster than linear convergence, in addition to the condition with respect to $\nabla^2 f$. In this chapter, we prove the following theorem.

Theorem 5. *Suppose that $\partial_B \mathrm{prox}_{\nu g}$ is an LNA of $\mathrm{prox}_{\nu g}$ at $x^* - \nu\nabla f(x^*)$, $\nabla^2 f$ is Lipschitz continuous, and the sequence $\{x^{(k)}\}$ generated by Algorithm 4 satisfies $x^{(k)} \neq x^*$ for any k and $\lim_{k\to\infty} x^{(k)} = x^*$. Then, $\{x^{(k)}\}$ superlinearly converges to x^* if $B^{(k)}$ satisfies*

$$\lim_{k\to\infty} \frac{\|(B^{(k)} - \nabla^2 f(x^*))(x^{(k+1)} - x^{(k)})\|_2}{\|x^{(k+1)} - x^{(k)}\|_2} = 0. \tag{4.25}$$

The condition (4.25) is similar to the condition for superlinear convergence of the standard quasi-Newton method. The BFGS formula (4.22) used in this study satisfies (4.25); therefore, $\{x^{(k)}\}$ generated by Algorithm 4 converges superlinearly.

4.6 Numerical experiments

In this section, we evaluate the performance of the linear Newton and the hybrid linear quasi-Newton methods in sparse estimation problems by comparing them with the proximal gradient and proximal Newton methods. All the programs are implemented by using Rcpp, and all the tests are performed in R Studio on a Windows machine with Ryzen 9 3900X @ 3.8 GHz and 64 GB of memory.

4.6.1 Group logistic regression

Let $(y_i, x_i) \in \{-1, 1\} \times \mathbb{R}^n, i = 1, \ldots, m$, where m is the number of observations. The group logistic regression optimization is formulated as follows:

$$\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp\{-y_i(\beta_0 + x_i^T \beta)\}) + \lambda \sum_{j=1}^{J} \|\beta_{I_j}\|_2, \tag{4.26}$$

where λ is a regularization parameter and $I_j, j = 1, \ldots, J$ are the index sets that belong to the j-th group such that $I_j \cap I_k = \emptyset$ $(j \neq k)$.

Our initial experiment focused on a single-group scenario:

$$\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp\{-y_i(\beta_0 + x_i^T \beta)\}) + \lambda \|\beta\|_2. \tag{4.27}$$

While single-group group lasso is not commonly used in practice, this experiment is to examine both worst-case (all features active) and best-case (all features inactive) scenarios for our proposed methods. Moreover, in multigroup cases, V becomes a block diagonal matrix. Consequently, since the single-group case becomes less efficient, the proposed method serves as one benchmark for comparison. We choose

$$V = \begin{cases} O, & \|x\|_2 \leq \nu \\ \frac{\nu}{\|x\|_2}(\frac{xx^T}{\|x\|_2^2} - I) + I, & \|x\|_2 > \nu \end{cases}$$

as $V \in \partial_B \text{prox}_{\nu g}(x)$. Figures 4.1 and 4.2 compare the proximal gradient (PG) and proximal Newton (PN) methods with the proposed linear Newton (LN), hybrid linear quasi-Newton (HLQN), and hybrid linear quasi-Newton + GCR (HLQN-GCR) methods. Furthermore, for each experiment, the PG method was updated 10,000 times, the PN method continued until the change in iterations became minor, and the LN, HLQN and HLQN-GCR methods were updated until the norm $||F_1(\beta_0^{(k)}, \beta^{(k)})||_2$ was reduced to below 10^{-12}. In both figures, the vertical and horizontal axes correspond to the value of $\|F_1(\beta_0^{(k)}, \beta^{(k)})\|_2$ and the computation time, respectively, which indicate the convergence of the algorithms.

We generated random data with features $n = 2000$ and sample sizes $m = 4000$ and set the initial values $\beta_0^{(0)} = 0, \beta^{(0)} = 0, B^{(0)} = \nabla^2 f((\beta_0^{(0)}, \beta^{(0)}))$. Furthermore, $B^{(k)}$ was updated by using the BFGS formula (4.22) and $\epsilon^{(k)} = 0.001$. Figure 4.1 shows the graph when λ is small ($\lambda = 1$), where all features are active. The LN, HLQN, and HLQN-GCR methods rapidly converge, and the LN method achieves a highly accurate solution. Although quadratically convergent, the PN method converges more slowly due to its reliance on the slower-converging PG method; the convergence of the proximal gradient method is slow and stops halfway. Moreover, the GCR method enhances the speed of the quasi-Newton method, surpassing the HLQN. Further detailed numerical data for the case of $\lambda = 1$ are presented in Table 4.1. Table 4.1 shows the conditions when $\|F_1(\beta_0^{(k)}, \beta^{(k)})\|_2$ is less than 10^{-3}, as well as the final number of iterations, the computation time, and the value of $\|F_1(\beta_0^{(k)}, \beta^{(k)})\|_2$.

Table 4.1: Computation time and Iterations for random data($\lambda = 1$)

method	$\|F_1\|_2 \leq 10^{-3}$			last		
	iter	time[s]	$\|F_1\|_2$	iter	time[s]	$\|F_1\|_2$
PG	6299	26.49	1.000×10^{-3}	10000	42.74	7.686×10^{-4}
PN	4	48.65	2.347×10^{-4}	6	52.74	6.918×10^{-5}
LN	5	20.90	1.286×10^{-4}	7	29.30	9.902×10^{-15}
HQLN	9	23.11	4.333×10^{-4}	33	80.17	8.108×10^{-13}
HQLN-GCR	9	11.53	4.333×10^{-4}	33	36.90	8.108×10^{-13}

Figure 4.2 shows the graph when λ is large, where all features are inactive. Here, our proposed methods converge more rapidly due to sparsity, with HLQN-GCR being the fastest. For large λ, the convergence of the proximal Newton method is slower than that for the small λ case, and the convergence of the proximal gradient method stops in the middle of the convergence process. Additionally, the specific numerical data at the time of convergence are shown in Table 4.2. It is evident from the fact that most of the estimated values converge to zero that the proposed methods converge in a small number of iterations.

We present the results of applying our proposed method to real-world data. The datasets used are cod-RNA [42] and ijcnn1 [43], obtained from the LIBSVM website[1], with sample sizes of $m = 59935$ and 49990, respectively. To introduce a group structure into the data, second-order polynomial features were generated from the original features [44, 45]. The dimensions of the generated features are $n = 140$ and 1155, respectively, with $J = 28$ and 231. In this case, as the loss

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Table 4.2: Computation time and iterations for random data(λ is large)

method	$\|F_1\|_2 \leq 10^{-3}$			last		
	iter	time[s]	$\|F_1\|_2$	iter	time[s]	$\|F_1\|_2$
PG	-	-	-	10000	44.38	1.900×10^{-3}
PN	1	12.75	4.832×10^{-4}	3	23.98	1.369×10^{-4}
LN	1	4.22	3.600×10^{-8}	2	8.41	1.162×10^{-16}
HQLN	1	4.22	3.600×10^{-8}	2	6.59	8.641×10^{-13}
HQLN-GCR	1	2.84	3.600×10^{-8}	2	3.81	8.641×10^{-13}

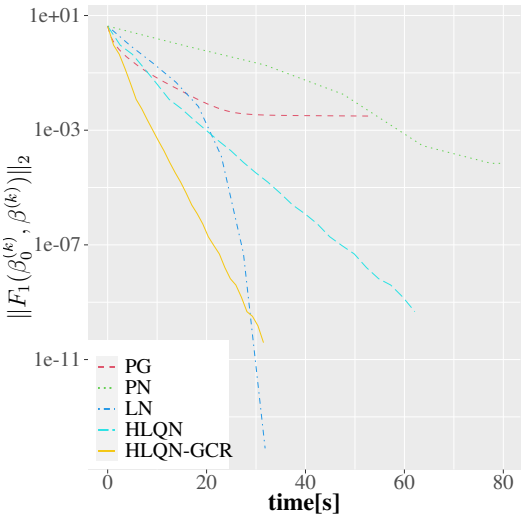

Figure 4.1: Changes in $F_1(x^{(k)})$ due to the computation time. ($\lambda = 1$)

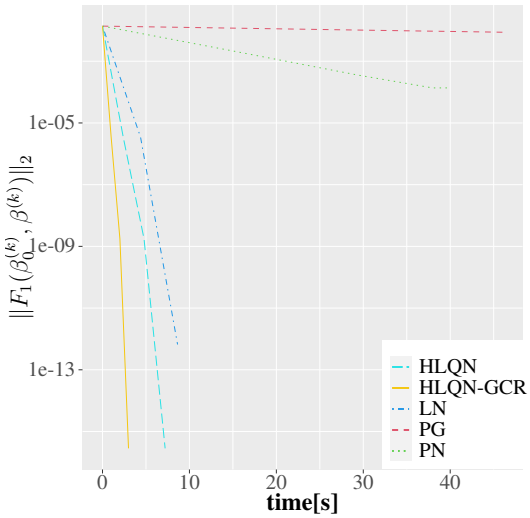

Figure 4.2: Changes in $F_1(x^{(k)})$ due to the computation time. (large λ)

function in (4.27) is not strongly convex, we add ridge regularization, resulting in the following optimization:

$$\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp\{-y_i(\beta_0 + x_i^T \beta)\}) + \frac{\epsilon}{2} \|\beta\|_2^2 + \lambda \sum_{j=1}^{J} \|\beta_{I_j}\|_2 \quad (4.28)$$

In this experiment, the ridge parameter was set to $\epsilon = 0.05$, and similar to the random data case, $B^{(0)} = \nabla^2 f((\beta_0^{(0)}, \beta^{(0)}))$ was used as the initial matrix and updated by using the BFGS formula. For each experiment, the PG method was updated 10,000 times, the PN method continued until the change in iterations became minor, and the LN, HLQN, and HLQN-GCR methods were updated until the norm $\|F_1(\beta_0^{(k)}, \beta^{(k)})\|_2$ was reduced below 10^{-10}.

Figures 4.3 and 4.4 show the computation times when applied to the cod-RNA dataset. The initial values $\beta_0^{(0)}$ and $\beta^{(0)}$ were set to -0.55 and 0, respectively, and -0.55 was obtained as the optimal β_0 when $\beta = 0$. Figure 4.3 and Table 4.3 represent the case in which $\lambda = 0.08$, the number of active groups is 9, and the dimension of the nonzero β is 45. Like for random data, the proposed LN, HLQN, and HLQN-GCR methods demonstrate fast convergence. However, the PN method shows slower convergence.

Table 4.3: Computation time and Iterations for cod-RNA dataset($\lambda = 0.08$)

method	$\|F_1\|_2 \leq 10^{-3}$			last		
	iter	time[s]	$\|F_1\|_2$	iter	time[s]	$\|F_1\|_2$
PG	-	-	-	10000	61.78	8.301×10^{-3}
PN	3	0.52	3.588×10^{-4}	10	1.24	3.766×10^{-8}
LN	4	0.67	5.758×10^{-6}	6	1.02	9.585×10^{-16}
HQLN	5	0.24	6.451×10^{-4}	12	0.30	3.833×10^{-11}
HQLN-GCR	5	0.21	5.762×10^{-4}	12	0.26	4.519×10^{-11}

Figure 4.4 and Table 4.4 show the results for the case with $\lambda = 0.28$, where the number of active groups is 1 and the nonzero dimension of β is 5. The quasi-Newton methods, HLQN and HLQN-GCR, converge rapidly and exhibit efficient performance. Although the proximal Newton method performs better than when λ is small, it stops converging midway. The LN method is slow at first, but eventually, the LN method rapidly converges. When n is small, there is not much difference between HLQN and HLQN-GCR.

Table 4.4: Computation time and iterations for cod-RNA dataset($\lambda = 0.28$)

method	$\|F_1\|_2 \leq 10^{-3}$			last		
	iter	time[s]	$\|F_1\|_2$	iter	time[s]	$\|F_1\|_2$
PG	2253	13.86	9.995×10^{-4}	10000	61.17	1.243×10^{-7}
PN	1	0.18	1.161×10^{-4}	4	0.69	4.5878×10^{-8}
LN	3	0.51	2.444×10^{-6}	4	0.68	7.722×10^{-12}
HQLN	3	0.19	5.737×10^{-6}	5	0.20	2.112×10^{-11}
HQLN-GCR	4	0.19	2.178×10^{-5}	7	0.21	1.966×10^{-12}

Figures 4.5 and 4.6 represent the computation times for the cod-RNA dataset. Furthermore, specific numerical data at the time of convergence are presented in Tables 4.5 and 4.6. The initial values were determined like those of the cod-RNA

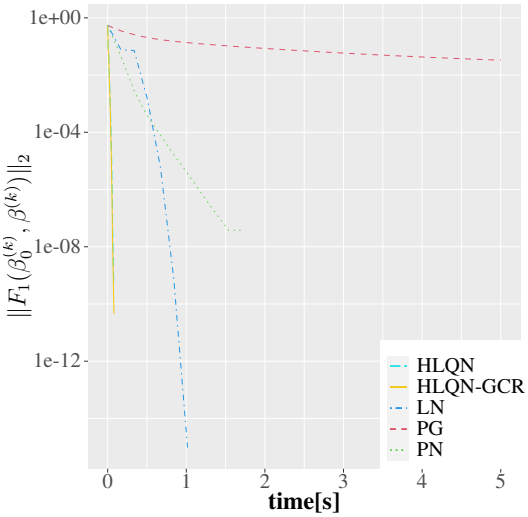

Figure 4.3: Change in computation time for the cod-RNA dataset. ($\lambda = 0.08$)

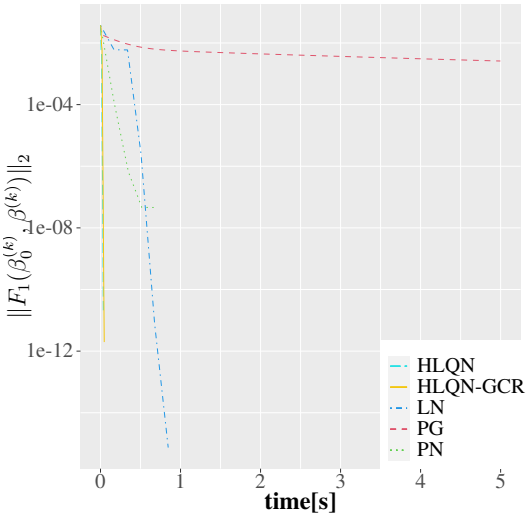

Figure 4.4: Change in computation time for the cod-RNA dataset. ($\lambda = 0.28$)

case, with $\beta_0^{(0)} = -1.1$ and $\beta^{(0)} = 0$. On the one hand, Figure 4.5 represents the case in which $\lambda = 0.08$, where the number of active groups is 49 and the nonzero dimension of β is 245. On the other hand, Figure 4.6 shows the results for the case with $\lambda = 0.12$, where the number of active groups is 5 and the nonzero dimension of β is 25. In both cases, the proposed HLQN and HLQN-GCR methods efficiently converge to the solutions, while LN, due to higher update costs, takes longer to approach optimality. As a result, the LN method cannot fully leverage its fast convergence, leading to a long computational time. The PG method is initially fast, but it eventually converges slowly. When n is large, HLQN-GCR requires more iterations than does HLQN, but due to its efficient updating, the computation time is shorter.

Table 4.5: Computation time and iterations for ijcnn dataset($\lambda = 0.08$)

method	$\|F_1\|_2 \leq 10^{-3}$			last		
	iter	time[s]	$\|F_1\|_2$	iter	time[s]	$\|F_1\|_2$
PG	-	-	-	10000	340.81	1.698×10^{-3}
PN	5	40.46	6.604×10^{-4}	26	209.15	1.203×10^{-7}
LN	9	75.25	9.874×10^{-7}	10	83.67	3.770×10^{-12}
HQLN	10	12.90	3.328×10^{-4}	20	17.94	5.443×10^{-11}
HQLN-GCR	11	10.48	3.492×10^{-4}	22	13.07	5.665×10^{-11}

Table 4.6: Computation time and iterations for ijcnn dataset($\lambda = 0.12$)

method	$\|F_1\|_2 \leq 10^{-3}$			last		
	iter	time[s]	$\|F_1\|_2$	iter	time[s]	$\|F_1\|_2$
PG	-	-	-	10000	344.06	1.727×10^{-2}
PN	6	48.44	6.535×10^{-4}	16	129.27	1.390×10^{-7}
LN	12	100.59	1.283×10^{-6}	13	108.98	9.196×10^{-12}
HQLN	9	12.56	2.535×10^{-4}	17	16.68	8.810×10^{-12}
HQLN-GCR	11	10.67	3.326×10^{-4}	19	12.56	1.012×10^{-11}

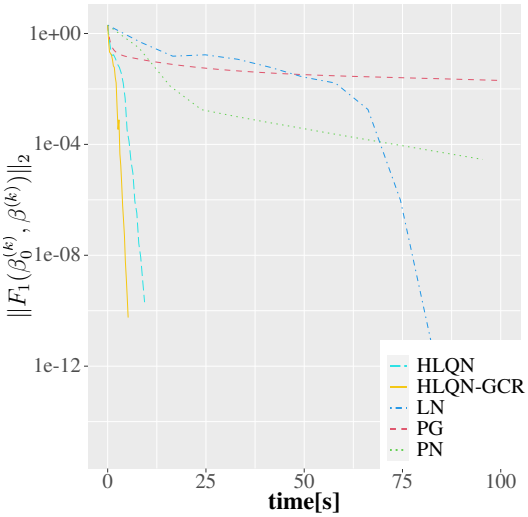

Figure 4.5: Change in computation time for the ijcnn1 dataset. ($\lambda = 0.08$)

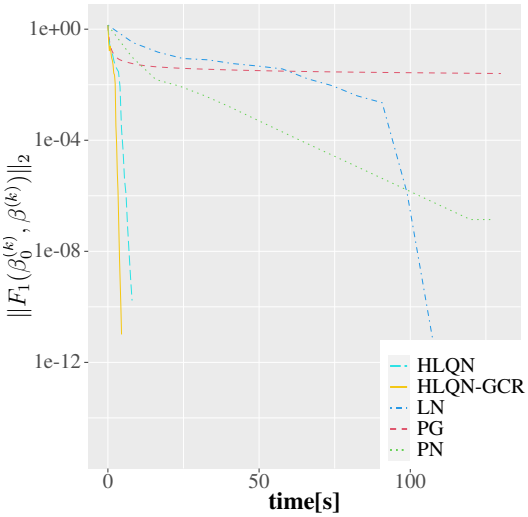

Figure 4.6: Change in computation time for the ijcnn1 dataset. ($\lambda = 0.12$)

4.7 Summary

In Chapter4, we tackled the optimization problem in sparse estimation, particularly focusing on scenarios where the loss function is strongly convex. We extended the semismooth Newton method, as proposed by [27], to a more general setting with $\nu > 0$. Our findings not only demonstrate the convergence of this method but also provide robust theoretical guarantees for its applicability, even in cases where the Lipschitz constant of ∇f is unknown. A significant contribution of this chapter is the development of the HLQN method, an innovative approach to approximate the second derivative $\nabla^2 f$. We theoretically established that the HLQN method is consistently updatable and achieves rapid convergence. This method is particularly beneficial when computing $\nabla^2 f$ is computationally chal-

lenging, such as in high-dimensional data scenarios. Our numerical experiments substantiate the computational efficiency of this method in applications such as group logistic regression. Although we assumed μ-strong convexity for f, we showed that this restriction can be relaxed, for example, by incorporating a ridge penalty, as shown in our numerical experiments (Section 4.6).

Chapter 5

Conclusions and future work

In Chapter 3, we presented a novel approach that converts ADMM to the proximal gradient method. The method was successfully applied to sparse estimation problems such as sparse convex clustering and trend filtering, demonstrating significant efficiency enhancements. Particularly for these two applications, our proposed method outperforms existing techniques like ADMM, especially when dealing with large regularization parameters that lead to sparsity in the results. This suggests that our method leverages sparsity effectively, resulting in more efficient computations that yield zero results. To apply our method effectively, it is essential to obtain a Lipschitz constant or an upper bound. We anticipate that this approach can extend beyond existing sparse estimation problems and be applied to various scenarios involving the addition of two regularization terms to the loss function. In such cases, the challenge of finding an efficient solution simplifies to determining the Lipschitz coefficient.

While our research primarily focuses on sparse estimation and related problems, we encourage the active exploration of its applicability to a wider range of optimization problems. Further investigation is needed to fully uncover and quantify its effectiveness in general optimization contexts.

In Chapter 4, we address the optimization problem in sparse estimation, particularly focusing on scenarios where the loss function is strongly convex. We extended the semismooth Newton method, as proposed by [27], to a more general setting with $\nu > 0$. Our findings not only demonstrate the convergence of this method but also provide robust theoretical guarantees for its applicability, even in cases where the Lipschitz constant ∇f is unknown. A significant contribution of this chapter is the development of the HLQN method, an innovative approach to approximate the second derivative $\nabla^2 f$. We theoretically established that the HLQN method is consistently updatable and achieves rapid convergence. This method is particularly beneficial in situations where computing $\nabla^2 f$ is computationally challenging, such as in high-dimensional data scenarios. Our numerical

experiments substantiate the computational efficiency of this method in applications such as group logistic regression. Although we assumed μ-strong convexity for f, we showed that this restriction can be relaxed, for example, by incorporating a ridge penalty, as shown in our numerical experiments (Section 4.6). Furthermore, if ∇f is Lipschitz continuous with parameter L_f, f^* becomes L_f^{-1}-strongly convex, where f^* is the conjugate function of f. Thus, considering the dual problem, our approach may be applicable to a wide range of problems. Our primary focus in this chapter was on group regularization. However, we acknowledge that not exploring preconditioning for the GCR method is a limitation, particularly for scenarios with many active variables or different regularization techniques, such as the fused lasso method. In future studies, the potential of our proposed methods under more general assumptions and in diverse regularization contexts should be investigated. Additionally, further research is warranted to explore global convergence properties by using strategies such as line search to broaden the applicability of our methods, especially in situations where the proximity of initial values to the optimal solution is uncertain.

Appendix A

The setting of the proximal gradient parameter η

A.1 Sparse convex clustering

Let $A_{\mathcal{E}} = (a_{(i,j),k})_{(i,j)\in\mathcal{E},k=1\ldots n} \in \mathbb{R}^{\mathcal{E}\times n}$. Then, we have

$$a_{(i,j),k} = \begin{cases} 1 & \text{if} \quad k = i \\ -1 & \text{if} \quad k = j \\ 0 & \text{otherwise} \end{cases}, \tag{A.1}$$

and the (i,j) element of G for $A_{\mathcal{E}}^T A_{\mathcal{E}} =: G \in \mathbb{R}^{n\times n}$ can be written as

$$G_{ij} = \begin{cases} -1 & \text{if} \quad (i,j) \in \mathcal{E} \\ \sum_{k\neq i}^{n} |G_{ik}| & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}. \tag{A.2}$$

Then, we notice the following lemma:

Lemma 7 (Gershgorin)**.** *Assume we have symmetric matrix $A \in \mathbb{R}^{n\times n}$.*

$$\lambda_{\max}(A) \leq \max_{i=1,\ldots,n} (a_{ii} + \sum_{j\neq i}^{n} |a_{ij}|) \tag{A.3}$$

From Lemma 7 because of

$$\lambda_{\max}(A_{\mathcal{E}}^T A_{\mathcal{E}}) \leq 2 \max_{i=1,\ldots,n} G_{ii} \tag{A.4}$$

it is appropriate to set $\eta > 0$ as

$$\eta = \frac{1}{1 + 2\nu \max_{i=1,\ldots,n} G_{ii}}. \tag{A.5}$$

A.2 Trend filtering

For $k \geq 0$, $D^{(k+1)} \in \mathbb{R}^{(n-k)\times n}$ can be written as

$$D^{(k+1)} = \begin{pmatrix} (-1)^{k+1}{}_{k+1}\mathrm{C}_0 & (-1)^{k+2}{}_{k+1}\mathrm{C}_1 & \cdots & {}_{k+1}\mathrm{C}_{k+1} & & \mathbf{0} \\ & (-1)^{k+1}{}_{k+1}\mathrm{C}_0 & (-1)^{k+2}{}_{k+1}\mathrm{C}_1 & \cdots & {}_{k+1}\mathrm{C}_{k+1} & \\ & & \ddots & \ddots & \ddots & \ddots \\ \mathbf{0} & & & (-1)^{k+1}{}_{k+1}\mathrm{C}_0 & (-1)^{k+2}{}_{k+1}\mathrm{C}_1 & \cdots & {}_{k+1}\mathrm{C}_{k+1} \end{pmatrix},$$

i.e., the (i, j) element of $D^{(k+1)}$ is

$$D^{(k+1)}_{ij} = \begin{cases} (-1)^{k+1+j-i}{}_{k+1}\mathrm{C}_{j-i} & \text{if} \quad 0 \leq j-i \leq k+1 \\ 0 & \text{otherwise} \end{cases} . \tag{A.6}$$

Thus, from Lemma 7, we have

$$\begin{aligned} \lambda_{\max}\left((D^{(k+1)})^T D^{(k+1)}\right) &\leq \max_{i=1,\ldots,n} \sum_{j=1}^{n} |\left((D^{(k+1)})^T D^{(k+1)}\right)_{ij}| \\ &= \max_{i=1,\ldots,n} |\sum_{j=1}^{n} \sum_{s=1}^{n} D_{si} D_{sj}| \\ &\leq \max_{i=1,\ldots,n} \sum_{j=1}^{n} \sum_{s=1}^{n} |D_{si} D_{sj}| \\ &\leq \sum_{j=0}^{k+1} \sum_{s=0}^{k+1} {}_{k+1}\mathrm{C}_j \times {}_{k+1}\mathrm{C}_s \\ &= (\sum_{j=0}^{k+1} {}_{k+1}\mathrm{C}_j)^2 = 4^{k+1} , \end{aligned} \tag{A.7}$$

and it is appropriate to set $\eta > 0$ as

$$\eta = \frac{1}{1 + \nu 4^{k+1}} .$$

Appendix B

Proof of thorems

B.1 Proof of Proposition 3

Lemma 8. *Suppose $A, B \in \mathbb{R}^{n\times n}$ are symmetric and A is positive semidefinite. Then, any eigenvalue λ of AB satisfies $\min\{\|A\|\lambda_{\min}(B), 0\} \leq \lambda \leq \max\{\|A\|\lambda_{\max}(B), 0\}$, where $\lambda_{\min}(B)$ and $\lambda_{\max}(B)$ are the minimum and maximum eigenvalues, respectively, of B.*

Proof. Since the eigenvalues of AB are equivalent to the eigenvalues of BA, we consider the eigenvalues of BA. We let $\lambda \in \mathbb{R}, x \in \mathbb{R}^n$ such that $BAx = \lambda x$ and $x \neq 0$. By multiplying $x^T A$ from the left, we obtain

$$x^T ABAx = \lambda x^T Ax.$$

If $x^T Ax = 0$, then $\lambda = 0$ since $Ax = 0$. Next, we consider the $x^T Ax > 0$ case. Since A is a symmetric positive semidefinite matrix, there exists $A^{\frac{1}{2}}$, and we obtain

$$\begin{aligned} \frac{x^T ABAx}{x^T Ax} &= \lambda \\ \frac{x^T A^{\frac{1}{2}} A^{\frac{1}{2}} BA^{\frac{1}{2}} A^{\frac{1}{2}} x}{x^T A^{\frac{1}{2}} A^{\frac{1}{2}} x} &= \lambda. \end{aligned} \tag{B.1}$$

We can rewrite (B.1) as

$$\frac{y^T A^{\frac{1}{2}} BA^{\frac{1}{2}} y}{y^T y} = \lambda,$$

where $y = A^{\frac{1}{2}} x \neq 0$. Thus, since $0 < \|A^{\frac{1}{2}} y\|_2 \leq \|A\|^{\frac{1}{2}} \|y\|_2$, we can obtain

$$\min\{\|A\|\lambda_{\min}(B), 0\} \leq \frac{y^T A^{\frac{1}{2}} BA^{\frac{1}{2}} y}{y^T y} \leq \max\{\|A\|\lambda_{\max}(B), 0\}.$$

Therefore, if $x^T Ax > 0$, then $\min\{\|A\|\lambda_{\min}(B), 0\} \leq \lambda \leq \|A\|\max\{\|A\|\lambda_{\max}(B), 0\}$. Using also the result when $x^T Ax = 0$, Lemma 8 holds. □

Theorem 6 ([30], Theorem 3.2)**.** *Suppose $g : \mathbb{R}^n \to (-\infty, +\infty]$ is a closed convex function. Every $V \in \partial_B \text{prox}_{\nu g}(x)$ is a symmetric positive semidefinite matrix that satisfies $\|V\| \leq 1$ for all $x \in \mathbb{R}^n$.*

Proof of Proposition 3. By assumption, since $\lambda_{\min}(\nabla^2 f(x)) \geq \mu$ for any $x \in \mathbb{R}^n$,

$$\lambda_{\max}(I - \nu\nabla^2 f(x)) \leq 1 - \nu\mu.$$

Since every $V \in \partial_B \text{prox}_{\nu g}(x)$ is a symmetric positive semidefinite matrix that satisfies $\|V\| \leq 1$ for all $x \in \mathbb{R}^n$ by Theorem 6, from Lemma 8,

$$\lambda_{\max}\left(V\left(I - \nu\nabla^2 f(x)\right)\right) \leq \max\{1 - \nu\mu, 0\}.$$

Thus, every eigenvalue of $I - V\left(I - \nu\nabla^2 f(x)\right)$ is a real number that is greater than or equal to $\min\{\nu\mu, 1\}$, and $I - V\left(I - \nu\nabla^2 f(x)\right)$ is a nonsingular matrix. □

B.2 Proof of Theorem 4

Lemma 9 ([46], Lemma 2.3.2)**.** *Let $A, C \in \mathbb{R}^{n\times n}$ and assume that A is invertible, with $\|A^{-1}\| \leq \alpha$. If $\|A - C\| \leq \beta$ and $\beta\alpha < 1$, then C is also invertible, and*

$$\|C^{-1}\| \leq \frac{\alpha}{(1 - \alpha\beta)}$$

Proof of Theorem 4. Let

$$\begin{aligned} V^{(k)} &\in \partial_B \text{prox}_{\nu g}(x^{(k)} - \nu\nabla f(x^{(k)})), \\ U^{(k)} &:= I - V^{(k)}\left(I - \nu\nabla^2 f(x^{(k)})\right) \in \partial F_\nu(x^{(k)}), \\ W^{(k)} &:= I - V^{(k)}\left(I - \nu B^{(k)}\right) \in \hat{\partial}^{(k)} F_\nu(x^{(k)}). \end{aligned}$$

From Proposition 3, every eigenvalue of $U^{(k)}$ is a real number that is greater than or equal to $\xi := \min\{\nu\mu, 1\}$, and

$$\left\|\left(U^{(k)}\right)^{-1}\right\| \leq \frac{\sqrt{n}}{\xi}.$$

Let $\Delta = \frac{\xi}{5\nu\sqrt{n}}$. Since ∂F_ν is the LNA of F_ν at x^*, there exists $\epsilon > 0$ such that

$$\|F_\nu(x) - F_\nu(x^*) - U(x - x^*)\|_2 \leq \nu\Delta\|x - x^*\|_2$$

for any $x \in B(x^*, \epsilon) := \{y \mid \|x^* - y\|_2 < \epsilon\}, U \in \partial F_\nu(x)$. Since $W^{(k)} - U^{(k)} = \nu V^{(k)} \left(B^{(k)} - \nabla^2 f(x^{(k)})\right)$ and $\|B^{(k)} - \nabla^2 f(x^{(k)})\| < \Delta$, we obtain $\|W^{(k)} - U^{(k)}\| \leq \nu\Delta$. By Lemma 9, $W^{(k)}$ is invertible and

$$\begin{aligned}\left\|\left(W^{(k)}\right)^{-1}\right\| &\leq \frac{\sqrt{n}/\xi}{1 - \sqrt{n}/\xi \times \nu\Delta} \\ &= \frac{5}{4}\frac{\sqrt{n}}{\xi}.\end{aligned}$$

Thus, if $\|x^{(k)} - x^*\|_2 < \epsilon$, then we have

$$\begin{aligned}\|x^{(k+1)} - x^*\|_2 &= \|x^{(k)} - (W^{(k)})^{-1} F_\nu(x^{(k)}) - x^*\|_2 \\ &\leq \|(W^{(k)})^{-1}\| \|F_\nu(x^{(k)}) - F_\nu(x^*) - W^{(k)}(x^{(k)} - x^*)\|_2 \\ &\leq \|(W^{(k)})^{-1}\| [\|F_\nu(x^{(k)}) - F_\nu(x^*) - U^{(k)}(x^{(k)} - x^*)\|_2 \\ &\quad + \|W^{(k)} - U^{(k)}\| \|x^{(k)} - x^*\|_2] \\ &\leq \frac{5}{4}\frac{\sqrt{n}}{\xi}(2\nu\Delta\|x^{(k)} - x^*\|_2) \\ &= \frac{1}{2}\|x^{(k)} - x^*\|_2\end{aligned}$$

Therefore, there exists ϵ, Δ such that the sequence generated by Algorithm 4 locally linearly converges to x^*. □

B.3 Proof of Theorem 5

Proof. Let $V^{(k)} \in \partial_B \mathrm{prox}_{\nu g}(x^{(k)} - \nu\nabla f(x^{(k)}))$, $U^{(k)} := I - V^{(k)}\left(I - \nu\nabla^2 f(x^{(k)})\right) \in \partial F_\nu(x^{(k)})$ and $W^{(k)} := I - V^{(k)}\left(I - \nu B^{(k)}\right) \in \hat{\partial}^{(k)} F_\nu(x^{(k)})$. We let $e^{(k)} = x^{(k)} - x^*, s^{(k)} = x^{(k+1)} - x^{(k)}$. We note that $s^{(k)} = e^{(k+1)} - e^{(k)}$ and $\{e^{(k)}\}$ and $\{s^{(k)}\}$ converge to 0 since $\{x^{(k)}\}$ converges to x^*. From the update rule of Algorithm 4, we have

$$\begin{aligned}F_\nu(x^*) &= \left[F_\nu(x^{(k)}) + W^{(k)} s^{(k)}\right] + \left[\left(U^{(k)} - W^{(k)}\right) s^{(k)}\right] \\ &\quad - \left[F_\nu(x^{(k)}) - F_\nu(x^*) - U^{(k)} e^{(k)}\right] - U^{(k)} e^{(k+1)} \\ &= \left[\left(U^{(k)} - W^{(k)}\right) s^{(k)}\right] - \left[F_\nu(x^{(k)}) - F_\nu(x^*) - U^{(k)} e^{(k)}\right] - U^{(k)} e^{(k+1)}.\end{aligned}$$

Since $F_\nu(x^*) = 0$ and $U^{(k)}$ is a nonsingular matrix,

$$\begin{aligned}U^{(k)} e^{(k+1)} &= \left[\left(U^{(k)} - W^{(k)}\right) s^{(k)}\right] - \left[F_\nu(x^{(k)}) - F_\nu(x^*) - U^{(k)} e^{(k)}\right] \\ e^{(k+1)} &= \left(U^{(k)}\right)^{-1} \left[\left(U^{(k)} - W^{(k)}\right) s^{(k)}\right] - \left(U^{(k)}\right)^{-1} \left[F_\nu(x^{(k)}) - F_\nu(x^*) - U^{(k)} e^{(k)}\right].\end{aligned}$$

By assumption, since $\|(W^{(k)} - U^{(k)})s^{(k)}\|_2 = \|\nu V^{(k)}(\nabla^2 f(x^{(k)}) - B^{(k)})s^{(k)}\|_2$ and $\|\nabla^2 f(x^*) - \nabla^2 f(x^{(k)})\| \to 0$ for $k \to \infty$, we have $\|(W^{(k)} - U^{(k)})s^{(k)}\|_2 = o(\|s^{(k)}\|_2)$. Therefore,

$$\|e^{(k+1)}\|_2 = o(\|s^{(k)}\|_2) + o(\|e^{(k)}\|_2) = o(\|e^{(k+1)}\|_2) + o(\|e^{(k)}\|_2).$$

Thus, we obtain $\|e^{(k+1)}\|_2 = o(\|e^{(k)}\|_2)$, and since $e^{(k)} = x^{(k)} - x^*$, the sequence $\{x^{(k)}\}$ generated by Algorithm 4 superlinearly converges to x^*.

□

List of Publications

[1] Shimmura, R., Suzuki, J. (2022). Converting ADMM to a proximal gradient for efficient sparse estimation. Japanese Journal of Statistics and Data Science, 5.2: 725-745.

[2] Shimmura, R., Suzuki, J. (2024) Newton-Type Methods with the Proximal Gradient Step for Sparse Estimation. In Operations Research Forum (Vol. 5, No. 2, p. 27). Cham: Springer International Publishing.

[3] Shimmura, R., Suzuki, J. (2023). Estimation of a Simple Structure in a Multidimensional IRT Model Using Structure Regularization. Entropy, 26(1), 44.

[4] Chen, J., Shimmura, R., Suzuki, J. (2021). Efficient proximal gradient algorithms for joint graphical lasso. Entropy, 23(12), 1623.

Acknowledgements

First, I would like to express my deepest gratitude to Prof. Joe Suzuki, my thesis supervisor, for their unwavering support and guidance throughout this journey. Your expertise and insights have been invaluable in shaping both my work and my professional growth.

Additionally, I am deeply grateful to my family for their constant love and encouragement, and for believing in my abilities. Your support has been the source of my strength. The bond we share as a family and the emotional support in my academic pursuits are of immeasurable value to me. Without you, I would not be where I am today.

References

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

[2] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

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

[4] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

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

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

[7] Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 76(2):373–397, 2014.

[8] Kristiaan Pelckmans, Joseph De Brabanter, Johan AK Suykens, and Bart De Moor. Convex clustering shrinkage. In *PASCAL workshop on statistics and optimization of clustering workshop*, 2005.

[9] Toby Dylan Hocking, Armand Joulin, Francis Bach, and Jean-Philippe Vert. Clusterpath an algorithm for clustering using convex fusion penalties. In *28th international conference on machine learning*, page 1, 2011.

[10] Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung. Clustering using sum-of-norms regularization: With application to particle filter output computation. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pages 201–204. IEEE, 2011.

[11] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

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

[13] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[14] Shiqian Ma. Alternating proximal gradient method for convex minimization. *Journal of Scientific Computing*, 68(2):546–572, 2016.

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

[16] Eric C Chi and Kenneth Lange. Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24(4):994–1013, 2015.

[17] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[18] 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.

[19] Binhuan Wang, Yilong Zhang, Will Wei Sun, and Yixin Fang. Sparse convex clustering. *Journal of Computational and Graphical Statistics*, 27(2):393–403, 2018.

[20] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. ℓ_1 trend filtering. *SIAM review*, 51(2):339–360, 2009.

[21] 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.

[22] Damek Davis and Wotao Yin. A three-operator splitting scheme and its optimization applications. *Set-valued and variational analysis*, 25(4):829–858, 2017.

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

[24] R Tyrrell Rockafellar. *Convex analysis*, volume 36. Princeton university press, 1970.

[25] Jean Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299, 1965.

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

[27] Xiantao Xiao, Yongfeng Li, Zaiwen Wen, and Liwei Zhang. A regularized semismooth Newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, 76(1):364–389, 2018.

[28] Panagiotis Patrinos and Alberto Bemporad. Proximal Newton methods for convex composite optimization. In *52nd IEEE Conference on Decision and Control*, pages 2358–2363. IEEE, 2013.

[29] Panagiotis Patrinos, Lorenzo Stella, and Alberto Bemporad. Forward-backward truncated Newton methods for convex composite optimization. *arXiv preprint arXiv:1402.6655*, 2014.

[30] Lorenzo Stella, Andreas Themelis, and Panagiotis Patrinos. Forward–backward quasi-Newton methods for nonsmooth optimization problems. *Computational Optimization and Applications*, 67(3):443–487, 2017.

[31] Andre Milzarek, Xiantao Xiao, Shicong Cen, Zaiwen Wen, and Michael Ulbrich. A stochastic semismooth newton method for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 29(4):2916–2948, 2019.

[32] Minghan Yang, Andre Milzarek, Zaiwen Wen, and Tong Zhang. A stochastic extra-step quasi-newton method for nonsmooth nonconvex optimization. *Mathematical Programming*, pages 1–47, 2021.

[33] Yongfeng Li, Zaiwen Wen, Chao Yang, and Ya xiang Yuan. A semi-smooth newton method for semidefinite programs and its applications in electronic

structure calculations. *SIAM Journal on Scientific Computing*, 40(6):A4131–A4157, 2018.

[34] Alnur Ali, Eric Wong, and J Zico Kolter. A semismooth Newton method for fast, generic convex programming. In *International Conference on Machine Learning*, pages 70–79. PMLR, 2017.

[35] Michael Ulbrich. *Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces*. SIAM, 2011.

[36] Francisco Facchinei and Jong Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer, 2003.

[37] Yangjing Zhang, Ning Zhang, Defeng Sun, and Kim Chuan Toh. An efficient Hessian based algorithm for solving large-scale sparse group lasso problems. *Mathematical Programming*, 179:223–263, 2020.

[38] Liqun Qi and Jie Sun. A nonsmooth version of newton's method. *Mathematical programming*, 58(1-3):353–367, 1993.

[39] Francisco Facchinei, Andreas Fischer, and Christian Kanzow. Inexact newton methods for semismooth equations with applications to variational inequality problems. *Nonlinear Optimization and Applications*, pages 125–139, 1996.

[40] Defeng Sun and Jiye Han. Newton and quasi-Newton methods for a class of nonsmooth equations and related problems. *SIAM Journal on Optimization*, 7(2):463–480, 1997.

[41] Michael Hintermüller. Semismooth Newton methods and applications. *Department of Mathematics, Humboldt-University of Berlin*, 2010.

[42] Andrew V Uzilov, Joshua M Keegan, and David H Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):1–30, 2006.

[43] Danil Prokhorov. Ijcnn 2001 neural network competition. *Slide presentation in IJCNN*, 1(97):38, 2001.

[44] Volker Roth and Bernd Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on Machine learning*, pages 848–855, 2008.

[45] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the fifth annual international conference on Computational biology*, pages 249–255, 2001.

[46] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.