이 학 박 사 학 위 논 문

MM algorithm for sparse regression model with the fused lasso penalty and its parallelization using GPU

Fused lasso 벌점을 가진 sparse 회귀모형에 대한 MM 알고리즘과 GPU를 이용한 병렬화

2013년 8월

서울대학교 대학원

통계학과

유 동 현

MM algorithm for sparse regression model with the fused lasso penalty and its parallelization using GPU

Fused lasso 벌점을 가진 sparse 회귀모형에 대한 MM 알고리즘과 GPU를 이용한 병렬화

지도교수 임 요 한

이 논문을 이학박사 학위논문으로 제출함

2013년 4월

서울대학교 대학원

통계학과

유 동 현

유동현의 이학박사 학위논문을 인준함

2013년 6월

위 원 장	오 희 석	(인)
부위원장	임 요 한	(인)
위 원	김 용 대	(인)
위 원	장 원 철	(인)
위 원	원 중 호	(인)

MM algorithm for sparse regression model with the fused lasso penalty and its parallelization using GPU

by

Dong Hyeon Yu

A Thesis
submitted in fulfillment of the requirement
for the degree of
Doctor of Philosophy
in
Statistics

The Department of Statistics
College of Natural Sciences
Seoul National University
August, 2013

Abstract

Dong Hyeon Yu
The Department of Statistics
The Graduate School
Seoul National University

In this paper, we propose a majorization-minimization (MM) algorithm for high-dimensional fused lasso regression (FLR) suitable for parallelization using graphics processing units (GPUs). The MM algorithm is stable and flexible as it can solve the FLR problems with various types of design matrices and penalty structures within a few tens of iterations. We also show that the convergence of the proposed algorithm is guaranteed. We conduct numerical studies to compare our algorithm with other existing algorithms, demonstrating that the proposed MM algorithm is competitive in many settings including the two-dimensional FLR with arbitrary design matrices. The merit of GPU parallelization is also exhibited.

keywords : *fused lasso regression, MM algorithm, parallel computation, graphics processing unit.*

Student Number : 2010-30083

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Sparse Regression model has been widely used for high-dimensional data analysis and variable selection. The *sparsity* in the regression model means that the small fraction of estimated coefficients are non-zero. Let $\mathbf{y} \in \mathbb{R}^n$ be a response vector and $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a design matrix. To obtain the sparsity, it is considered that the penalized least square problem

$$\min_{\beta} \frac{1}{2} \left\| \mathbf{y} - \mathbf{X}\beta \right\|_2^2 + \Omega_\lambda(\beta),$$

where $\Omega_\lambda(\beta)$ is a penalty function of β and λ is a regularization parameter.

Various kinds of penalties for obtaining the sparsity have been developed, for example the lasso, the SCAD, the elastic net, the fused lasso, the group lasso, and the OSCAR penalties (Tibshirani, 1996; Fan and Li, 2001; Zou and Hastie, 2005; Tibshirani et al., 2005; Yuan and Lin, 2006; Bondell and Reich, 2008). All the penalties include the ℓ_1-norm of coefficients that enforces some estimates of coefficients to be 0 with an appropriate regularization parameter

λ. Thus, the sparse regression models with these penalties can simultaneously achieve the variable selection and the estimation. In addition, the elastic net, the fused lasso, and the OSCAR penalties have a grouping property which means that the estimates of coefficients corresponding to highly correlated variables are forced to be same.

In this thesis, we consider the fused lasso regression (FLR) with the generalized fusion penalty, a special case of the ℓ_1-regularized regression. In FLR, not only that the sparsity among the coefficients is promoted, but also "adjacent" coefficients tend to have the same values, where adjacency is determined by the application. The **FLR with the generalized fusion penalty** minimizes

$$f(\beta) = \frac{1}{2}\big\|\mathbf{y} - \mathbf{X}\beta\big\|_2^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{(j,k)\in E, j<k} |\beta_j - \beta_k|, \tag{1.1}$$

where λ_1 and λ_2 are non-negative regularization parameters and E is the index set of the adjacent pairs of variables specified in the model.

The FLR with the generalized fusion penalty imposes sparsity on the differences of the coefficients of the specified adjacent pairs as well as the coefficients themselves. Tibshirani et al. (2005) introduce the "standard" FLR to find the sites of mass-over-charge ratio that distinguish between the case and the control groups in protein mass spectroscopy data. The **standard FLR** considers the set $E = \big\{(j-1, j)|\ j = 2, \ldots, p\big\}$ that represents a one-dimensional chain graph. Since Tibshirani et al. (2005), many variants of the FLR have been introduced, which consider a wider class of the index set E: the **two-dimensional FLR** uses a two-dimensional lattice $E_{2D} = \big\{\big(i(j-1), ij\big)|\ 1 \le i \le q, j = 2, 3, \ldots, q\big\} \cup \big\{\big((i-1)j, ij\big)|\ i =$

$2, 3, \ldots, q, 1 \leq j \leq q\}$ for coefficients $\left(\beta_{ij}\right)_{1 \leq i,j \leq q}$; other examples include the clustered lasso (She, 2010), the generalized lasso (Tibshirani and Taylor, 2011), and the pairwise fused lasso (Petry et al., 2011).

The FLR is basically a quadratic programming (QP) and can be solved in principle using general purpose QP solvers (Gill et al., 1997; Grant et al., 2008). However, the standard QP formulation introduces a large number of additional variables and constraints, and is not adequate for general purpose QP solvers if the dimension p increases.

To resolve this difficulty, many special algorithms for solving the FLR problem have been proposed. Some are based on the LARS (Efron et al., 2004)-flavored solution path methods (Friedman et al., 2007; Hoefling, 2010; Tibshirani and Taylor, 2011). Others work on a fixed set of the regularization parameters (λ_1, λ_2) (Liu et al., 2010; Ye and Xie, 2011; Lin et al., 2011; Chen et al., 2012). Although these algorithms are adequate for high-dimensional settings, restrictions may apply: some are only applicable for the special design matrix ($\mathbf{X} = \mathbf{I}$, Friedman et al. (2007); Hoefling (2010)); others require $\mathbf{X}$ to be full rank (Tibshirani and Taylor, 2011) or to have a special penalty structure (Liu et al., 2010). In addition, computational issues still pertain for ultra-high dimensional settings ($p \gg 1000$).

In this thesis, we apply the majorization-minimization (MM) algorithm (Lange et al., 2000) to solve the FLR problem. The MM algorithm iteratively finds and minimizes a surrogate function called the *majorizing function* that is generally convex and differentiable. It is well known that the efficiency of the MM algorithm depends largely on the choice of the majorizing function. Hence we begin the main body of the paper by proposing a majorizing

function suitable for the FLR problem.

Motivated by Hunter and Li (2005) on their work on the **classical lasso** (Tibshirani, 1996), we first introduce a perturbed version of the objective function (1.1) and propose a majorizing function for this perturbed objective. The MM algorithm we propose is based on these perturbed objective function and its majorizing function. The MM algorithm has several advantageous points over other existing algorithms. First, it has flexibility in the choice of both design matrix and penalty structure. We show that the proposed MM algorithm converges to the optimal objective (1.1) regardless of the rank of the design matrix. Furthermore, it can be applied to general penalty structures, while having comparable performance with state-of-the-art FLR solvers, some of which require a special penalty structure. Second, as numerically demonstrated in Chapter 5, our MM algorithm is stable in the number of iterations to converge regardless of the choice of design matrix and the penalty structure. Third and finally, an additional benefit of our MM formulation is that it opens up the possibility of solving the FLR problem in a massively parallel fashion using graphics processing units (GPUs).

This thesis is organized as follows. In Chapter 2, we review existing algorithms for solving the FLR. In Chapter 3, we propose an MM algorithm for the FLR with the generalized fusion penalty and prove the convergence of the algorithm. We also introduce the preconditioned conjugate gradient (PCG) method to accelerate the proposed MM algorithm. In Chapter 4, we explain how to parallelize the MM algorithm using GPUs. In Chapter 5, we conduct numerical studies to compare the proposed MM algorithm with the other existing algorithms. In Chapter 6, we conclude the paper with a brief

summary.

Chapter 2

Review of existing algorithms

Existing algorithms for solving the FLR problem can be classified into two categories. One is based on solution path methods, whose goal is to find the whole solutions for all values of the regularization parameters. The other is based on first-order methods, which attempt to solve a large scale problem given a fixed set of regularization parameters.

2.1. Solution path methods

These methods aim to provide all the solutions of interest with a small computational cost after solving the initial problems to find every change point of solution path. This convenience comes at a sacrifice: these methods can only solve the problems with a constraint on the design matrix $\mathbf{X}$ such as $\mathbf{X} = \mathbf{I}$ or $\mathbf{X}$ having full rank. (The FLR with $\mathbf{X} = \mathbf{I}$ is called the *fused lasso signal approximator* (**FLSA**).)

2.1.1. Path-wise optimization algorithm

Friedman et al. (2007) propose the path-wise optimization algorithm for the "standard" FLSA. Although this is not precisely a path algorithm, the path-wise algorithm provides an approximate solution path as the regularization parameter increases sequentially. The major challenge in the FLSA (and the FLR) as compared to the classical lasso is that its penalty function is non-separable as well as non-smooth. It is known that the coordinate descent (CD) algorithm fails to obtain the optimal solution for a regression problem with a non-separable penalty (Tseng, 2001). Hence, Friedman et al. (2007) propose a modified CD algorithm in which two coefficients move together if the coordinate-wise move fails to reduce the objective function.

The modified CD algorithm solves the standard FLSA

$$\min_{\beta} f(\beta) \equiv \frac{1}{2}\|\mathbf{y} - \beta\|_2^2 + +\lambda_1 \sum_{j=1}^{n} |\beta_j| + \lambda_2 \sum_{j=2}^{n} |\beta_j - \beta_{j-1}|,$$

where λ_1 and λ_2 are non-negative regularization parameters. Let $\widehat{\beta}(\lambda_1, \lambda_2)$ be the optimal solution for the standard FLSA at (λ_1, λ_2). In this case, it is known that $\widehat{\beta}_j(\lambda_1, \lambda_2)$ is obtained from $\widehat{\beta}_j(0, \lambda_2)$ by soft-thresholding

$$\widehat{\beta}_j(\lambda_1, \lambda_2) = \text{sign}\big(\widehat{\beta}_j(0, \lambda_2)\big)\big(|\widehat{\beta}_j(0, \lambda_2)| - \lambda_1\big)_+ \text{ for } j = 1, 2, \ldots, p, \quad (2.1)$$

where $(x)_+ = \max(x, 0)$. By the above property, the modified CD algorithm focuses on the standard FLSA with $\lambda_1 = 0$.

This modified CD algorithm is composed of three cycles (the smoothing cycle, the descent cycle, and the fusion cycle). The smoothing cycle is a strategy to obtain the optimal solution for a desired regularization parameter. In the smoothing cycle, λ_2 sequentially increases with small amount δ from

zero to the desired value. For each λ_2, the descent and the fusion cycles are iteratively applied until convergence occurs. Let $\tilde{\beta}$ be a solution at the previous iteration. In the descent cycle, the current solution $\widehat{\beta}_i$ is updated by the subdifferential of f with respect to β_i,

$$\frac{\partial f(\beta)}{\partial \beta_i} = -(y_i - \beta_i) - \lambda_2 \mathrm{sgn}(\tilde{\beta}_{i+1} - \beta_i) + \lambda_2 \mathrm{sgn}(\beta_i - \tilde{\beta}_{i-1}),$$

where $\mathrm{sgn}(x)$ is a sign of x if $x \neq 0$ and $\mathrm{sgn}(x) \in [-1, 1]$ if $x = 0$. Since the above subdifferential of f is a piecewise linear function with breaks at $\{\tilde{\beta}_{i-1}, \tilde{\beta}_{i+1}\}$, it has either an explicit solution or one of $\{\tilde{\beta}_{i-1}, \tilde{\beta}_{i+1}\}$. This descent cycle is exactly same to the CD algorithm, so it may fail to reduce the objective function and be stuck at a local optimal solution due to the non-separability. To solve it, the fusion cycle consider a descent cycle with a constraint $\beta_i = \beta_{i-1} = \gamma$. In the fusion cycle, the subdifferential of f with respect to γ is

$$\frac{\partial f(\beta)}{\partial \gamma} = -2\Big(\frac{y_{i-1} + y_i}{2} - \gamma\Big) - \lambda_2 \mathrm{sgn}(\tilde{\beta}_{i+1} - \gamma) + \lambda_2 \mathrm{sgn}(\gamma - \tilde{\beta}_{i-2}). \quad (2.2)$$

If the fusion cycle reduces the objective function, two adjacent variables are substituted with the average of them and the current estimates $\widehat{\beta}_i$ and $\widehat{\beta}_{i-1}$ are updated by the solution of (2.2).

The path-wise optimization algorithm is efficient to solve the high dimensional standard FLSA. In the FLR with general design matrix, however, the fusion of the solution path is not monotone in the sense that there exists λ_2^1 and λ_2^2 with $\lambda_2^1 < \lambda_2^2$ such that $\widehat{\beta}_k(\lambda_2^1) = \widehat{\beta}_k(\lambda_2^1)$ and $\widehat{\beta}_k(\lambda_2^2) \neq \widehat{\beta}_k(\lambda_2^2)$. This prevents the smoothing cycle from finding the optimal solution for the desired regularization parameter.

2.1.2. Path algorithm for the FLSA

Hoefling (2010) proposes a path algorithm for the FLSA with the generalized fusion penalty,

$$\min_{\beta} f(\beta) \equiv \frac{1}{2}\big\|\mathbf{y}-\beta\big\|_2^2 + \lambda_1 \sum_{i=1}^{n} |\beta_i| + \lambda_2 \sum_{(j,k)\in E, j<k} |\beta_j - \beta_k|, \tag{2.3}$$

where E is an index set of adjacent pairs of variables specified in the model. As stated in Section 2.1.1, $\widehat{\beta}(\lambda_1, \lambda_2)$ can be obtained from $\widehat{\beta}(0, \lambda_2)$ by soft thresholding in (2.1). For the sake of simplicity, the regularization parameter λ_1 is fixed as 0 and the estimate $\widehat{\beta}(0, \lambda_2)$ is denoted by $\widehat{\beta}(\lambda_2)$.

The function $f(\beta)$ in (2.3) can be represented a differentiable function by reparameterization based on sets of fused coefficients. Let $n_F(\lambda_2^0)$ be the number of sets of fused coefficients for a given λ_2^0. The sets of fused coefficients, $F_i(\lambda_2^0)$ for $i = 1, 2, \ldots, n_F(\lambda_2^0)$, are defined in Hoefling (2010) by the following conditions:

1. $\bigcup_{i=1}^{n_F(\lambda_2^0)} F_i(\lambda_2^0) = \{1, 2, \cdots, n\}$,
2. $F_i(\lambda_2^0) \bigcap F_j(\lambda_2^0) = \phi$ for $i \neq j$,
3. If $k, l \in F_i(\lambda_2^0)$, then $\widehat{\beta}_k(\lambda_2^0) = \widehat{\beta}_l(\lambda_2^0) \equiv \widehat{\beta}_{F_i}(\lambda_2^0)$,
4. If $F_i(\lambda_2^0)$ and $F_j(\lambda_2^0)$ for $i \neq j$, have a adjacent pair in E, then $\widehat{\beta}_k(\lambda_2) \neq \widehat{\beta}_l(\lambda_2)$ for $k \in F_i(\lambda_2)$, $l \in F_j(\lambda_2)$ and $^{\forall}\lambda_2 \in (\lambda_2^0, \lambda_2^0 + \epsilon)$, where ϵ is positive and sufficiently small.

With sets of fused coefficients, the problem (2.3) can be represented by

$$\min_{\beta} f^*(\beta) \equiv \frac{1}{2} \sum_{i=1}^{n_F(\lambda_2)} \Big(\sum_{j\in F_i(\lambda_2)} \big(y_j - \beta_{F_i}(\lambda_2)\big)^2 \Big) + \lambda_2 \sum_{i<j} n_{ij} |\beta_{F_i}(\lambda_2) - \beta_{F_j}(\lambda_2)|,$$

where $n_{ij} = |\{(k,l) \in E \mid k \in F_i(\lambda_2), l \in F_j(\lambda_2)\}|$. For given sets of fused coefficients, the function $f^*(\beta)$ is differentiable except for change-points of the solution path. (i.e., the fusion or the split occurs at the change-point.) If λ_2 is not a change-point of the solution path, the optimal solution based on the sets of fused coefficients is obtained from the derivative of $f^*(\beta)$ with respect to β_{F_i} for $i = 1, 2, \ldots, n_F(\lambda_2)$,

$$\frac{\partial f^*(\beta)}{\partial \beta_{F_i}} = |F_i(\lambda_2)|\beta_{F_i}(\lambda_2) - \sum_{j \in F_i(\lambda_2)} y_j + \lambda_2 \sum_{j \neq i} n_{ij} \text{sign}\big(\beta_{F_i}(\lambda_2) - \beta_{F_j}(\lambda_2)\big),$$

where $n_{ij} = |\{(k,l) \in E \mid k \in F_i(\lambda_2), l \in F_j(\lambda_2)\}|$.

The path algorithm starts at $\lambda_2 = 0$ with $\widehat{\beta}_{F_i}(0) = y_i$ and $F_i = \{i\}$ for $i = 1, 2, \ldots, n$. To identify the change-points of the solution path, he considers a *hitting time* and a *violation time*. A hitting time $h(\lambda_2^0)$ is a real value such that $h(\lambda_2^0) > \lambda_2^0$ and some two sets of fused coefficients are merged at $h(\lambda_2^0)$. A violation time $v(\lambda_2^0)$ is a real value such that $v(\lambda_2^0) > \lambda_2^0$ and one of the sets of fused coefficients at λ_2^0 is split at $v(\lambda_2^0)$.

A hitting time $h(\lambda_2^0)$ is derived from the definition of the sets of fused coefficients. By the definition, two sets $F_i(\lambda_2^0)$ and $F_j(\lambda_2^0)$ have to be fused if $\widehat{\beta}_{F_i}(\lambda_2^0) = \widehat{\beta}_{F_j}(\lambda_2^0)$ and there exist adjacent pairs between $F_i(\lambda_2^0)$ and $F_j(\lambda_2^0)$ in E. From the derivative of $f^*(\beta)$, $\widehat{\beta}_{F_i}(\lambda_2) = \widehat{\beta}_{F_j}(\lambda_2)$, and $\frac{\partial \widehat{\beta}_F}{\partial \lambda_2}$, the hitting time of two sets F_i and F_j at λ_2^0 is defined by

$$h_{ij}(\lambda_2^0) = \frac{\widehat{\beta}_{F_i}(\lambda_2^0) - \widehat{\beta}_{F_j}(\lambda_2^0)}{\frac{\partial \widehat{\beta}_{F_j}(\lambda_2^0)}{\partial \lambda_2} - \frac{\partial \widehat{\beta}_{F_i}(\lambda_2^0)}{\partial \lambda_2}} + \lambda_2^0$$

if $F_i(\lambda_2^0)$ and $F_j(\lambda_2^0)$ have adjacent pairs in E, and $h_{ij}(\lambda_2^0) = \infty$, otherwise. From the definition of two sets $F_i(\lambda_2^0)$ and $F_j(\lambda_2^0)$, $h_{ij}(\lambda_2^0) = \lambda_2^0$ can not occur. If $h_{ij}(\lambda_2^0) < \lambda_2^0$, $h_{ij}(\lambda_2^0)$ is not meaningful since two sets have been moving

apart already at λ_2^0. Hence, the hitting time at λ_2^0 is defined by

$$h(\lambda_2^0) = \min_i \min_{j:h_{ij}>\lambda_2^0} h_{ij}(\lambda_2^0).$$

A violation time $v(\lambda_2^0)$ is derived from the subdifferential of $f(\beta)$ in (2.3). Since its derivation is quite complicated, we briefly describe it here. The derivation of the violation time $v(\lambda_2^0)$ is composed of three steps. First, he derives the optimality conditions for the problem (2.3) with the subdifferential of $f(\beta)$ and expresses these conditions with sets of fused coefficients. Second, the optimality conditions are reformulated as *maximum-flow* (max-flow) problems in graph theory to find the smallest δ_i such that the optimality conditions are satisfied at $\lambda_2 \in [\lambda_2^0, \lambda_2^0 + \delta_i]$ for each $F_i(\lambda_2^0)$. Finally, a violation time and a set of fused coefficients which has to be split at the violation time, are defined.

In this algorithm, the FLSA with the generalized fusion penalty is reformulated as a max-flow problem to find the solution path. The computational cost of this path algorithm depends almost on that of solving the max-flow problem. The path algorithm for the FLSA with the generalized fusion penalty becomes inefficient as the dimension p increases due to the computational cost of a large scale max-flow problem. However, the path algorithm for the standard FLSA is efficient, because the solution path for the standard FLSA only has fusions and does not require solving max-flow problem.

2.1.3. Path algorithm for the generalized lasso

Tibshirani and Taylor (2011) propose a path algorithm for the generalized

lasso problem, which replaces the penalty terms in (1.1) with a generalized ℓ_1-norm penalty $\lambda\|\mathbf{D}\beta\|_1$,

$$\min_{\beta} f(\beta) \equiv \frac{1}{2}\big\|\mathbf{y} - \mathbf{X}\beta\big\|_2^2 + \lambda\big\|\mathbf{D}\beta\big\|_1, \tag{2.4}$$

where λ is a non-negative regularization parameter and $\mathbf{D}$ is a $m \times p$ dimensional matrix corresponding to the dependent structure of coefficients. Generalized lasso includes the other ℓ_1-regularized regression methods such as the linear trend filtering (Kim et al., 2009), and reduces to the FLR when the design matrix has full rank. They propose a path algorithm for the standard FLSA and extend this algorithm to the FLSA with the generalized fusion penalty and the FLR with a design matrix having full rank and the generalized fusion penalty.

First, they consider the Lagrangian of (2.4) with $\mathbf{X} = \mathbf{I}$ (standard FLSA) and a constraint $\mathbf{z} = \mathbf{D}\beta$,

$$L(\beta, \mathbf{z}, \mathbf{u}) = \frac{1}{2}\|\mathbf{y} - \beta\|_2^2 + \lambda\|\mathbf{z}\|_1 + \mathbf{u}^T(\mathbf{D}\beta - \mathbf{z}), \tag{2.5}$$

where $\mathbf{D}$ is a $(n-1) \times n$ dimensional matrix such that $(\mathbf{D}\beta)_j = \beta_{j+1} - \beta_j$ for $j = 1, 2, \ldots, n-1$. From the Lagrangian function (2.5), the dual problem is

$$\min_{\mathbf{u} \in \mathbb{R}^m} \frac{1}{2}\big\|\mathbf{y} - \mathbf{D}^T\mathbf{u}\big\|_2^2 \text{ subject to } \|\mathbf{u}\|_\infty \leq \lambda. \tag{2.6}$$

The dual solution $\widehat{u}_{\lambda,j}$s satisfy

$$\widehat{u}_{\lambda,j} = \begin{cases} \lambda & \text{if } (\mathbf{D}\widehat{\beta}_\lambda)_j > 0 \\ -\lambda & \text{if } (\mathbf{D}\widehat{\beta}_\lambda)_j < 0 \\ [-\lambda, \lambda] & \text{if } (\mathbf{D}\widehat{\beta}_\lambda)_j = 0 \end{cases},$$

where $\widehat{\beta}_\lambda = \mathbf{y} - \mathbf{D}^T\widehat{\mathbf{u}}_\lambda$ is the solution of the primal of (2.5) at λ. They prove the *boundary lemma* for the standard FLSA. The boundary lemma is that for $j = 1, 2, \ldots, n-1$, the solution $\widehat{u}_{\lambda,j}$ of (2.6) satisfies the following properties:

$$\begin{aligned} &\widehat{u}_{\lambda_0,j} = \lambda_0 \quad \Rightarrow \widehat{u}_{\lambda,j} = \lambda \ \text{ for all } \ 0 \le \lambda \le \lambda_0, \\ &\widehat{u}_{\lambda_0,j} = -\lambda_0 \Rightarrow \widehat{u}_{\lambda,j} = -\lambda \ \text{ for all } \ 0 \le \lambda \le \lambda_0. \end{aligned} \tag{2.7}$$

From the boundary lemma, the algorithm starts with $\lambda_0 = \infty$ and sequentially finds the change-points of the solution path where one of the dual solutions hits the boundary. The boundary set is defined by $\mathcal{B} = \{\ j \ \big|\ |u_j| = \lambda, j = 1, 2, \ldots, n-1\}$ and a corresponding sign vector is denoted by $\mathbf{s} = (s_1, \ldots, s_{|\mathcal{B}|})^T$ with $s_k = \text{sign}(u_{n_k})$ for $n_k \in \mathcal{B}$ and $k = 1, 2, \ldots, |\mathcal{B}|$. To be specific the path algorithm, they represent the dual problem (2.6) as

$$\begin{aligned} &\min_{\mathbf{u}\in\mathbb{R}^m} \ \frac{1}{2}\big\|\mathbf{y} - \lambda\mathbf{D}_{\mathcal{B}}^T\mathbf{s} - \mathbf{D}_{-\mathcal{B}}^T\mathbf{u}_{-\mathcal{B}}\big\|_2^2 \\ &\text{subject to } \|\mathbf{u}_{-\mathcal{B}}\|_\infty \le \lambda, \end{aligned} \tag{2.8}$$

where $-\mathcal{B}$ denotes a set of interior coordinates. (i.e., for $j \in -\mathcal{B}$, $u_j \in (-\lambda, \lambda)$.) From the above problem, they express the solution at λ_k as

$$\begin{aligned} &\widehat{\mathbf{u}}_{\lambda_k,\mathcal{B}} = \lambda_k\mathbf{s}, \\ &\widehat{\mathbf{u}}_{\lambda_k,-\mathcal{B}} = \big(\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{-\mathcal{B}})^T\big)^{-1}\mathbf{D}_{-\mathcal{B}}\big(\mathbf{y} - \lambda_k(\mathbf{D}_{\mathcal{B}})^T\mathbf{s}\big). \end{aligned} \tag{2.9}$$

For j-th interior coordinate, a hitting time $t_j^{(\text{hit})}$ at λ_k is defined by

$$t_j^{(\text{hit})} = \frac{\Big[\big(\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{-\mathcal{B}})^T\big)^{-1}\mathbf{D}_{-\mathcal{B}}\mathbf{y}\Big]_j}{\Big[\big(\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{-\mathcal{B}})^T\big)^{-1}\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{\mathcal{B}})^T\mathbf{s}\Big]_j \pm 1} \in [0, \lambda_k],$$

The next hitting time λ_{k+1} is defined by

$$\lambda_{k+1} = \max_j \ t_j^{(\text{hit})}.$$

It is shown that the boundary lemma still holds if $\mathbf{D}\mathbf{D}^T$ is diagonally dominant. This path algorithm can be applied to the FLSA with the generalized fusion penalty if $\mathbf{D}\mathbf{D}^T$ is diagonally dominant.

Second, they extend their path algorithm to the FLSA with a general matrix $\mathbf{D}$ when the boundary lemma fails. The FLSA with a general matrix $\mathbf{D}$ includes the FLSA with the generalized fusion penalty. Since the boundary lemma does not hold, some dual solutions on the boundary may leave as λ changes. For the current λ_k, they derive a next hitting time h_{k+1} and a next leaving time l_{k+1} from the Karush-Kuhn-Tucker (KKT) conditions for the dual problem (2.6) with a $m \times p$ dimensional matrix $\mathbf{D}$,

$$\begin{aligned}
\mathbf{D}\mathbf{D}^T\widehat{\mathbf{u}}_{\lambda_k} - \mathbf{D}\mathbf{y} + \alpha\gamma &= 0, \\
\|\widehat{\mathbf{u}}_{\lambda_k}\|_\infty &\leq \lambda_k, \\
\alpha &\geq 0, \\
\alpha(\|\widehat{\mathbf{u}}_{\lambda_k}\|_\infty - \lambda_k) &= 0, \\
\|\gamma\|_1 &\leq 1, \\
\gamma^T\widehat{\mathbf{u}}_{\lambda_k} &= \|\widehat{\mathbf{u}}_{\lambda_k}\|_\infty.
\end{aligned}$$

The dual solution $\widehat{\mathbf{u}}_{\lambda_k}$ at λ_k is represented by

$$\begin{aligned}
&\widehat{\mathbf{u}}_{\lambda_k,\mathcal{B}} = \lambda_k \mathbf{s}, \\
&\widehat{\mathbf{u}}_{\lambda_k,-\mathcal{B}} = \big(\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{-\mathcal{B}})^T\big)^+\mathbf{D}_{-\mathcal{B}}\big(\mathbf{y} - \lambda_k(\mathbf{D}_{\mathcal{B}})^T\mathbf{s}\big),
\end{aligned}$$

where $\mathbf{X}^+$ denotes a pseudoinverse of $\mathbf{X}$.

For the j-th interior coordinate, a hitting time at the current λ_k is defined as

$$t_j^{(\text{hit})} = \frac{\Big[\big(\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{-\mathcal{B}})^T\big)^+\mathbf{D}_{-\mathcal{B}}\mathbf{y}\Big]_j}{\Big[\big(\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{-\mathcal{B}})^T\big)^+\mathbf{D}_{-\mathcal{B}}(\mathbf{D}_{\mathcal{B}})^T\mathbf{s}\Big]_j \pm 1} \in [0, \lambda_k], \quad \text{and}$$

for the j-th boundary coordinate, a leaving time at the current λ_k is defined as

$$t_j^{(\text{leave})} = \begin{cases} \dfrac{s_j\Big[\mathbf{D}_{\mathcal{B}}\big[I - \mathbf{D}_{-\mathcal{B}}^T\big(\mathbf{D}_{-\mathcal{B}}\mathbf{D}_{-\mathcal{B}}^T\big)^+\mathbf{D}_{-\mathcal{B}}\big]\mathbf{y}\Big]_j}{s_j\Big[\mathbf{D}_{\mathcal{B}}\big[I - \mathbf{D}_{-\mathcal{B}}^T\big(\mathbf{D}_{-\mathcal{B}}\mathbf{D}_{-\mathcal{B}}^T\big)^+\mathbf{D}_{-\mathcal{B}}\big]\mathbf{D}_{\mathcal{B}}^T\mathbf{s}\Big]_j} \equiv \dfrac{c_j}{d_j} & \text{if } c_j, d_j < 0, \\ 0 \qquad \text{otherwise.} \end{cases}$$

Then, they define the next hitting time and the next leaving time as

$$h_{k+1} = \max_{j \in -\mathcal{B}} t_j^{(\text{hit})} \text{ and } l_{k+1} = \max_{j \in \mathcal{B}} t_j^{(\text{leave})}.$$

They obtain the primal solution from the dual solution by the primal-dual relationship,

$$\widehat{\beta}_\lambda = \mathbf{y} - \mathbf{D}^T\widehat{\mathbf{u}}_\lambda. \tag{2.10}$$

Finally, they extend their path algorithm to the FLR with a design matrix having full rank and the generalized fusion penalty. The dual of (2.4) can be represented by

$$\begin{aligned} &\min_{\mathbf{u} \in \mathbb{R}^m} \ \frac{1}{2}\big\|\tilde{\mathbf{y}} - \tilde{\mathbf{D}}^T\mathbf{u}\big\|_2^2 \\ &\text{subject to } \|\mathbf{u}\|_\infty \le \lambda, \ \mathbf{D}^T\mathbf{u} \in \text{row}(\mathbf{X}), \end{aligned} \tag{2.11}$$

where $\tilde{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, $\tilde{\mathbf{D}} = \mathbf{D}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, and row($\mathbf{X}$) is a row space of $\mathbf{X}$. They show that the constraint $\mathbf{D}^T\mathbf{u} \in \text{row}(\mathbf{X})$ can be removed when rank($\mathbf{X}$) = p. Thus, if the design matrix is of full rank, the problem (2.11) is equivalent to the dual of the FLSA with a general matrix $\mathbf{D}$ and can be solved by their path algorithm for the FLSA with the general matrix $\mathbf{D}$.

The path algorithm for the generalized lasso problem can solve various ℓ_1-regularization problems and obtains the exact solution path. After sequentially solving the dual, the solution path is obtained by the primal-dual

relationship. As stated in the path algorithm for the FLSA with the generalized fusion penalty (Hoefling, 2010), this path algorithm is efficient for solving the standard FLSA. However, it has several difficulties. If the rank of the design matrix $\mathbf{X}$ is not full rank, the dual problem has an additional constraint $\mathbf{D}^T\mathbf{u} \in \text{row}(\mathbf{X})$, where $\text{row}(\mathbf{A})$ denotes a row space of $\mathbf{A}$. It makes the dual problem difficult to solve. To resolve it, they suggest adding $\epsilon\|\beta\|_2^2$ penalty to the problem (2.4). The modified problem can be rewritten by substituting $\mathbf{y}$ and $\mathbf{X}$ as $\mathbf{y}^* = (\mathbf{y}^T, \mathbf{0}^T)^T$ and $\mathbf{X}^* = \left(\mathbf{X}^T, \sqrt{\epsilon}\,\mathbf{I}\right)^T$, respectively. This modification makes the design matrix has full rank, but the choice of ϵ affects the objective function value. If the small value of ϵ may lead ill-conditioned matrix $(\mathbf{X}^*)^T\mathbf{X}^*$. Moreover, we should put $\frac{\lambda_1}{\lambda}\mathbf{I}$ into the matrix $\mathbf{D}$ to solve the standard FLR with this path algorithm. It makes the path algorithm less efficient since the number of dual variables increases.

2.2. First-order methods

To avoid the restrictions in the solution path methods, several optimization algorithms, which target to solve the FLR problem with a fixed set of regularization parameters, have been developed. These algorithms can solve the FLR problems with the general design matrix $\mathbf{X}$ regardless of its rank. For scalability with the dimension p, they employ gradient descent-flavored first-order optimization methods.

2.2.1. Efficient fused lasso algorithm

Liu et al. (2010) propose the efficient fused lasso algorithm (EFLA) that solves the standard FLR

$$\begin{aligned} \min_\beta f(\beta) &= \frac{1}{2}\left\|\mathbf{y}-\mathbf{X}\beta\right\|_2^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}| \\ &\equiv \text{loss}(\beta) + \Omega_{\lambda_1,\lambda_2}(\beta). \end{aligned} \tag{2.12}$$

The EFLA iteratively solves the subproblem with the subgradient finding algorithm (SFA) and updates the solution by Nesterov's method (Nesterov, 2003, 2007) to accelerate the convergence of the algorithm. The subproblem is the minimization of the following function $f_{L,\widehat{\beta}^{(r)}}(\beta)$, which is derived from the first-order Taylor expansion of $\text{loss}(\beta)$ at $\widehat{\beta}^{(r)}$ and additional regularization term $\frac{L}{2}\|\beta - \widehat{\beta}^{(r)}\|_2^2$ for approximating $f(\beta)$ at $\widehat{\beta}^{(r)}$,

$$f_{L,\widehat{\beta}^{(r)}}(\beta) = \text{loss}(\widehat{\beta}^{(r)}) + \text{loss}'(\widehat{\beta}^{(r)})^T(\beta - \widehat{\beta}^{(r)}) + \Omega_{\lambda_1,\lambda_2}(\beta) + \frac{L}{2}\|\beta - \widehat{\beta}^{(r)}\|_2^2,$$

where $\widehat{\beta}^{(r)}$ is the solution of the EFLA in the r-th iteration and $L > 0$. The minimization of $f_{L,\widehat{\beta}^{(r)}}(\beta)$ is equivalent to

$$\min_\beta \frac{1}{2}\|\mathbf{v}-\beta\|_2^2 + \frac{\lambda_1}{L}\sum_{j=1}^{p}|\beta_j| + \frac{\lambda_2}{L}\sum_{j=2}^{p}|\beta_j - \beta_{j-1}|,$$

where $L > 0$ and $\mathbf{v} = \widehat{\beta}^{(r)} - (\mathbf{X}^T\mathbf{X}\widehat{\beta}^{(r)} + \mathbf{X}^T\mathbf{y})/L$. Since this problem is the standard FLSA, it can be efficiently solved by the existing algorithms (Friedman et al., 2007; Hoefling, 2010; Tibshirani and Taylor, 2011).

To accelerate the algorithm, the EFLA adopt the Nesterov's method (Nesterov, 2003, 2007). The Nesterovs's method is based on the sequence of search points. Let $\widehat{\beta}^{(r)}$ be the solution of the EFLA in the r-th iteration.

The sequence $\{\mathbf{s}^{(r)}\}_{r\geq 0}$ of search points is defined by

$$\mathbf{s}^{(0)} = \widehat{\beta}^{(0)} \text{ and } \mathbf{s}^{(r)} = \widehat{\beta}^{(r)} + \eta_r(\widehat{\beta}^{(r)} - \widehat{\beta}^{(r-1)}) \text{ for } r \geq 1,$$

where $\eta_r = \frac{\alpha_{r-1}-1}{\alpha_r}$, $\alpha_{r+1} = \frac{1+\sqrt{1+4\alpha_r^2}}{2}$, and $\alpha_0 = 1$. The sequence $\{\widehat{\beta}^{(r)}\}$ of solutions of the EFLA is obtained by

$$\widehat{\beta}^{(r+1)} = \underset{\beta}{\operatorname{argmin}}\, f_{L_r,\mathbf{s}^{(r)}}(\beta),$$

where L_r is the smallest value of $\{L_{r-1}, 2L_{r-1}, \ldots\}$ such that $f(\widehat{\beta}^{(r+1)}) \leq f_{L_r,\mathbf{s}^{(r)}}(\widehat{\beta}^{(r+1)})$ and L_0 is a given positive constant.

In the EFLA, the subproblem, which is equivalent to the standard FLSA, is solved by the SFA. For the sake of simplicity, we assume that the SFA aims to solve the standard FLSA with $\lambda_1 = 0$,

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{v} - \beta\|_2^2 + \lambda_2\|\mathbf{D}\beta\|_1,$$

where $\mathbf{D} \in \mathbb{R}^{(p-1)\times p}$ is defined in (2.5). As applied in Section 2.1.3, the dual of the standard FLSA can be represented as

$$\min_{\|\mathbf{u}\|_\infty \leq \lambda_2} \psi(\mathbf{u}) \equiv \frac{1}{2}\|\mathbf{D}^T\mathbf{u}\|_2^2 + \mathbf{v}^T\mathbf{D}^T\mathbf{u} \tag{2.13}$$

using the primal-dual relationship $\beta = \mathbf{v} - \mathrm{D}^T\mathbf{u}$. For sufficiently large λ_2, the dual problem (2.13) becomes the unconstrained minimization problem that is equivalent to the linear system $\mathbf{D}\mathbf{D}^T\mathbf{u} = \mathbf{D}\mathbf{v}$. Let $\lambda_2^{\max}$ be the smallest λ_2 such that the dual problem (2.13) is unconstrained and $\widehat{\mathbf{u}}$ be the solution of the linear system $\mathbf{D}\mathbf{D}^T\mathbf{u} = \mathbf{D}\mathbf{v}$. The solution $\widehat{\mathbf{u}}$ is the solution of the dual (2.13) for $\lambda_2 \geq \lambda_2^{\max}$ and satisfies $\|\widehat{\mathbf{u}}\|_\infty = \lambda_2^{\max}$.

For $0 \leq \lambda_2 < \lambda_2^{\max}$, the gradient descent method (Nesterov, 2003) is applied to solve the box-constrained optimization problem (2.13) and denoted

by $\mathrm{SFA_G}$. Let $\mathbf{h} \equiv \psi'(\mathbf{u})$ be the gradient of $\psi(\mathbf{u})$. The $\mathrm{SFA_G}$ iteratively updates $\widehat{\mathbf{u}}^{(r)}$ by

$$\widehat{\mathbf{u}}^{(r+1)} = \widehat{\mathbf{u}}^{(r)} - \mathbf{h}^{(r)}/K,$$

where $K = 2 - 2\cos(\pi(p-1)/p)$ is the largest eigenvalue of $\mathbf{D}\mathbf{D}^T$ and $\mathbf{h}^{(r)} = \psi'(\widehat{\mathbf{u}}^{(r)})$. This $\mathrm{SFA_G}$ has a linear convergence rate and can be accelerated by using the restart technique after each descent move.

The restart technique accelerates the $\mathrm{SFA_G}$ by updating the solution of the $\mathrm{SFA_G}$. The updated solution $\tilde{\mathbf{u}}$ is obtained from the linear system

$$\mathbf{D}\mathbf{D}^T\tilde{\mathbf{u}} = \mathbf{D}\big(\mathbf{v} - w(\widehat{\mathbf{u}})\big),$$

where $\widehat{\mathbf{u}}$ is the solution of the $\mathrm{SFA_G}$ and $w(\widehat{\mathbf{u}})$ is defined in (2.14). This updating rule is based on the support set $\mathbf{T}$ and the mapping $\beta = w(\mathbf{u})$. For any $\mathbf{u}$ such that $\|\mathbf{u}\|_\infty \le \lambda_2$, the support set $\mathbf{T}(\mathbf{u})$ is defined as

$$\mathbf{T}(\mathbf{u}) = \big\{t \in \{1, 2, \ldots, p-1\} \mid |u_t| = \lambda_2, u_t h_t < 0, \mathbf{h} = \psi'(\mathbf{u})\big\}.$$

Let $\{t_j\}$ be a sequence of the elements in $\mathbf{T}$ such that $t_j < t_{j+1}$ for $j = 1, 2, \ldots, |\mathbf{T}| - 1$. From the sequence $\{t_j\}$, the partition $\{G_j\}_{1 \le j \le |\mathbf{T}|+1}$ is defined as

$$G_j = \{i \mid t_{j-1} + 1 \le i \le t_j\} \quad \text{for } 1 \le j \le |\mathbf{T}| + 1,$$

where $t_0 = 0$ and $t_{|\mathbf{T}|+1} = p$. Let $\mathbf{e}$ be the p dimensional vector having all the elements as 1 and $\mathbf{e}_{G_j}$ and v_{G_j} be the j-th partition of $\mathbf{e}$ and $\mathbf{v}$ corresponding to G_j, respectively. The mapping $\beta = w(\mathbf{u})$ is defined as

$$\beta_i = \frac{\mathbf{e}_{G_j}^T \mathbf{v}_{G_j} - u_{t_{j-1}} + u_{t_j}}{|G_j|} \quad \text{for } i \in G_j, \tag{2.14}$$

where $j = 1, 2, \ldots, |\mathbf{T}| + 1$ and $u_0 = u_p = 0$.

The EFLA is efficient for the standard FLR with any design matrix but it has some limitations for the FLR with the generalized fusion penalty. In the SFA, the matrix $\mathbf{D}\mathbf{D}^T$ may not satisfy the positive definite condition, which is key condition of the SFA. Thus, the EFLA fails to obtain the optimal solution for the FLR with the generalized fusion penalty when the number of rows in $\mathbf{D}$ is greater than p. To avoid this problem, the existing path algorithms can be used instead of the SFA. This modification makes the EFLA obtain the optimal solution but it does not assure the efficiency of the algorithm, because the path algorithms always start from $\lambda_2 = 0$ (Hoefling, 2010) or $\lambda_2 = \infty$ (Tibshirani and Taylor, 2011).

2.2.2. Smoothing proximal gradient method

Chen et al. (2012) propose the smoothing proximal gradient (SPG) method that solves the regression problems with structured penalties including the generalized fusion penalty. Recall the FLR with generalized fusion penalty,

$$\begin{aligned}\min_{\beta} f(\beta) &\equiv \frac{1}{2}\left\|\mathbf{y} - \mathbf{X}\beta\right\|_2^2 + \lambda_2 \sum_{(j,k)\in E, j<k} |\beta_j - \beta_k| + \lambda_1 \sum_{j=1}^{p} |\beta_j| \\ &\equiv g(\beta) + \lambda_2 \|\mathbf{D}\beta\|_1 + \lambda_1 \|\beta\|_1,\end{aligned} \tag{2.15}$$

where λ_1 and λ_2 are non-negative regularization parameters, E is the index set of adjacent pairs of variables specified in the model, and $\mathbf{D}$ is an $m \times p$ dimensional matrix similarly defined in (2.5). The main idea of the SPG method is the smooth approximation of the non-smooth and non-separable penalty term $\|\mathbf{D}\beta\|_1$ that has a Lipschitz continuous gradient. From this property, the approximate problem can be efficiently solved by the fast iterative shrinkage-thresholding algorithm (FISTA) that solves the minimization

of the sum of a smooth convex function having a Lipschitz continuous gradient and a continuous convex function (Beck and Teboulle, 2009).

The SPG method consists of the smooth approximation and the smoothing proximal gradient descent steps. In the first step, the SPG method reformulates the penalty term $\|\mathbf{D}\beta\|_1$ into a maximization problem with the auxiliary variable α,

$$\|\mathbf{D}\beta\|_1 \equiv \max_{\|\alpha\|_\infty \leq 1} \alpha^T \mathbf{D}\beta \equiv \Omega(\beta),$$

where $\alpha \in \mathbb{R}^m$ and $\|\alpha\|_\infty$ is the largest value of α. By using the approximation technique due to Nesterov (2005), the smooth approximation to $\Omega(\beta)$ is

$$\tilde{\Omega}(\beta, \mu) \equiv \max_{\|\alpha\|_\infty \leq 1} \left(\alpha^T \mathbf{D}\beta - \mu \frac{1}{2}\|\alpha\|_2^2\right), \tag{2.16}$$

where μ is a positive smoothing parameter. The smooth approximation $\tilde{\Omega}(\beta, \mu)$ satisfies

$$\Omega(\beta) - \mu\frac{m}{2} \leq \tilde{\Omega}(\beta, \mu) \leq \Omega(\beta).$$

In the SPG method, the smoothing parameter μ is set to $\frac{\epsilon}{m}$ to obtain the desired accuracy for a given ϵ. It is shown that the gradient of $\tilde{\Omega}(\beta, \mu)$ is Lipschitz continuous with a Lipschitz constant $L_\mu = \frac{1}{\mu}\|\mathbf{D}\|_2^2$, where $\|\mathbf{D}\|_2$ is the spectral norm of $\mathbf{D}$ that is the square root of the largest eigenvalue of $\mathbf{D}^T\mathbf{D}$.

In the second step, the SPG solves the following approximate problem using the FISTA,

$$\begin{aligned} \min_\beta \tilde{f}(\beta) &\equiv g(\beta) + \lambda_2 \tilde{\Omega}(\beta, \mu) + \lambda_1 \|\beta\|_1 \\ &= h(\beta) + \lambda_1 \|\beta\|_1, \end{aligned} \tag{2.17}$$

where $h(\beta) = g(\beta)+\lambda_2\tilde{\Omega}(\beta,\mu)$. Note that $\tilde{f}(\beta)$ is the sum of a smooth convex function $h(\beta)$ with Lipschitz continuous gradient and $\lambda_1\|\beta\|_1$ is a continuous convex function. The gradient of $h(\beta)$ can be represented as

$$\nabla h(\beta) = \mathbf{X}^T(\mathbf{X}\beta - \mathbf{y}) + \lambda_2\mathbf{D}^T\alpha^*, \tag{2.18}$$

where $\alpha^* = \mathbf{S}(\mathbf{D}\beta/\mu)$ is the optimal solution in (2.16), $\mathbf{S}(\mathbf{x}) = (s(x_1),\ldots,s(x_m))^T$ for $\mathbf{x} \in \mathbb{R}^m$, and $s(x) = \text{sign}(x)\min(|x|,1)$. Thus, this gradient $\nabla h(\beta)$ is Lipschitz-continuous with the Lipschitz constant

$$L = \|\mathbf{X}\|_2^2 + \lambda_2 L_\mu = \|\mathbf{X}\|_2^2 + \frac{\lambda_2}{\mu}\|\mathbf{D}\|_2^2, \tag{2.19}$$

where $\|\mathbf{A}\|_2$ is the spectral norm of an $m \times p$ dimensional matrix $\mathbf{A}$ whose computational complexity is $O\big(\min(m^2p, mp^2)\big)$. With the gradient ∇h and the Lipschitz constant L, the FISTA iteratively solves the subproblem

$$\min_\beta Q_L(\beta, \mathbf{w}^{(r)}) = h(\mathbf{w}^{(r)}) + (\beta - \mathbf{w}^{(r)})^T\nabla h(\mathbf{w}^{(r)}) + \lambda_1\|\beta\|_1 + \frac{L}{2}\|\beta - \mathbf{w}^{(r)}\|_2^2, \tag{2.20}$$

where $\mathbf{w}^{(r)} = \widehat{\beta}^{(r)} + \frac{1-\theta_{r-1}}{\theta_{r-1}}\theta_r(\widehat{\beta}^{(r)} - \widehat{\beta}^{(r-1)})$, $\mathbf{w}^{(0)} = \widehat{\beta}^{(0)}$, $\widehat{\beta}^{(r)}$ is the solution of the FISTA at the r-th iteration, $\theta_r = \frac{2}{r+2}$, and $\theta_0 = 1$.

The computational cost of the Lipschitz constant L is heavy for large p. To reduce this computational cost, the SPG method uses a sequence $\{L_r\}_{r\geq 0}$ instead of a global constant L in (2.19). The sequence $\{L_r\}_{r\geq 0}$ is defined by $L_{r+1} = \tau^a L_r$ with a given scaling factor $\tau > 1$ and the smallest integer $a \in \{0,1,2,\ldots\}$ such that

$$\tilde{f}(\widehat{\beta}_{L_{r+1}}(\mathbf{w}^{(r)})) \leq Q_{L_{r+1}}(\widehat{\beta}_{L_{r+1}}(\mathbf{w}^{(r)}), \mathbf{w}^{(r)}), \tag{2.21}$$

where $\widehat{\beta}_{L_{r+1}}(\mathbf{w}^{(r)}) = \text{argmin}_\beta\, Q_{L_{r+1}}(\beta, \mathbf{w}^{(r)})$. But, it still has some problems when p or m increases: for every iteration, until finding L_{r+1} in (2.21), it

needs to calculate the approximate objective $\tilde{f}(\widehat{\beta}_{L_{r+1}}(\mathbf{w}^{(r)}))$ whose computational complexity is $O\big(\max(mp, np)\big)$, where n is the number of samples and m is the number of rows in $\mathbf{D}$.

2.2.3. Alternating directions methods

Split Bregman algorithm

Ye and Xie (2011) propose the split Bregman (SB) algorithm for the FLR with the generalized fusion penalty,

$$\min_{\beta} g(\beta) + \lambda_1\|\beta\|_1 + \lambda_2\|\mathbf{D}\beta\|_1, \tag{2.22}$$

where $g(\beta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ and $\mathbf{D}$ is an $m \times p$ dimensional matrix similarly defined in (2.5). The SB algorithm resolves the difficulty involved with non-differentiable and non-separable penalty by using the augmented Lagrangian method (Hestenes, 1969; Rockafellar, 1973). The augmeneted Lagrangian function of (2.22) is represented as

$$\begin{aligned} L(\beta, \mathbf{a}, \mathbf{b}, \mathbf{u}, \mathbf{v}) = g(\beta) + \lambda_1\|\mathbf{a}\|_1 + \lambda_2\|\mathbf{b}\|_1 + \mathbf{u}^T(\beta - \mathbf{a}) + \mathbf{v}^T(\mathbf{D}\beta - \mathbf{b}) \\ + \frac{\mu_1}{2}\|\beta - \mathbf{a}\|_2^2 + \frac{\mu_2}{2}\|\mathbf{D}\beta - \mathbf{b}\|_2^2, \end{aligned} \tag{2.23}$$

where λ_1 and λ_2 are non-negative regularization parameters, μ_1 and μ_2 are positive parameters, $\mathbf{a} \in \mathbb{R}^p$ and $\mathbf{b} \in \mathbb{R}^m$ are augmented variables, and $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^m$ are dual variables. With the augmented Lagrangian, the ℓ_1-norm penalties on the coefficients are removed and imposed on the newly introduced augmented variables. In addition, ℓ_2-norms on the differences between the coefficients and the augmented variables are included with the additional parameters μ_1 and μ_2.

The SB algorithm alternately solves the primal and the dual problems, which is derived from the augmented Lagrangian function in (2.23),

$$(\widehat{\beta}^{(r)}, \mathbf{a}^{(r)}, \mathbf{b}^{(r)}) = \underset{\beta,\mathbf{a},\mathbf{b}}{\operatorname{argmin}} \; L(\beta, \mathbf{a}, \mathbf{b}, \mathbf{u}^{(r-1)}, \mathbf{v}^{(r-1)}),$$
$$(\mathbf{u}^{(r)}, \mathbf{v}^{(r)}) = \underset{\mathbf{u},\mathbf{v}}{\operatorname{argmin}} \; L(\beta^{(r)}, \mathbf{a}^{(r)}, \mathbf{b}^{(r)}, \mathbf{u}, \mathbf{v}),$$

where $\widehat{\beta}^{(r)}$, $\mathbf{a}^{(r)}$, and $\mathbf{b}^{(r)}$ are the solutions of the primal and $\mathbf{u}^{(r)}$ and $\mathbf{v}^{(r)}$ are the solutions of the dual at the r-th iteration. Since the dual problem is a linear function of $\mathbf{u}$ and $\mathbf{v}$, the SB algorithm adopt a gradient ascent algorithm with step sizes δ_1 and δ_2,

$$\mathbf{u}^{(r)} = \mathbf{u}^{(r-1)} + \delta_1(\widehat{\beta}^{(r)} - \mathbf{a}^{(r)}),$$
$$\mathbf{v}^{(r)} = \mathbf{v}^{(r-1)} + \delta_2(\mathbf{D}\widehat{\beta}^{(r)} - \mathbf{b}^{(r)}).$$

To obtain the primal solution, three subproblems for the minimization of β, $\mathbf{a}$, and $\mathbf{b}$ are alternately solved. First, the minimization of β is expressed as

$$\widehat{\beta}^{(r)} = \underset{\beta}{\operatorname{argmin}} \; g(\beta) + (\mathbf{u}^{(r-1)})^T(\beta - \mathbf{a}^{(r-1)}) + (\mathbf{v}^{(r-1)})^T(\mathbf{D}\beta - \mathbf{b}^{(r-1)}) + \frac{\mu_1}{2}\|\beta - \mathbf{a}^{(r-1)}\|_2^2 + \frac{\mu_2}{2}\|\mathbf{D}\beta - \mathbf{b}^{(r-1)}\|_2^2.$$

This minimization is equivalent to the linear system

$$(\mathbf{X}^T\mathbf{X} + \mu_1\mathbf{I} + \mu_2\mathbf{D}^T\mathbf{D})\beta = \mathbf{X}^T\mathbf{y} + (\mu_1\mathbf{a}^{(r-1)} - \mathbf{u}^{(r-1)}) + \mathbf{D}^T(\mu_2\mathbf{b}^{(r-1)} - \mathbf{v}^{(r-1)}). \tag{2.24}$$

The SB algorithm solves this linear system to obtain the solution $\widehat{\beta}^{(r)}$. Although the complexity of solving this linear system is $O(p^3)$, it can be solved quickly by using the preconditioned conjugate gradient (PCG) algorithm in the standard FLR. The PCG algorithm is an iterative algorithm to solve

$\mathbf{Ax} = \mathbf{b}$ having the positive definite matrix $\mathbf{A}$ with a preconditioner $\mathbf{M}$ (Demmel, 1997). In this case, $\mu_1\mathbf{I} + \mu_2\mathbf{D}^T\mathbf{D}$ is used as a preconditioner since $\mathbf{D}^T\mathbf{D}$ is tridiagonal in the standard FLR. After updating $\widehat{\beta}^{(r)}$, last two subproblems for the minimization of $\mathbf{a}$ and $\mathbf{b}$,

$$\mathbf{a}^{(r)} = \underset{\mathbf{a}}{\operatorname{argmin}} \ \lambda_1\|\mathbf{a}\|_1 + (\widehat{\beta}^{(r)} - \mathbf{a})^T\mathbf{u}^{(r-1)} + \frac{\mu_1}{2}\|\widehat{\beta}^{(r)} - \mathbf{a}\|_2^2,$$
$$\mathbf{b}^{(r)} = \underset{\mathbf{b}}{\operatorname{argmin}} \ \lambda_1\|\mathbf{b}\|_1 + (\mathbf{D}\widehat{\beta}^{(r)} - \mathbf{b})^T\mathbf{v}^{(r-1)} + \frac{\mu_1}{2}\|\mathbf{D}\widehat{\beta}^{(r)} - \mathbf{b}\|_2^2,$$

can be solved easily by soft-thresholding,

$$a_j^{(r)} = \operatorname{sign}\left(\widehat{\beta}_j^{(r)} + \frac{u_j^{(r-1)}}{\mu_1}\right)\left(\left|\widehat{\beta}_j^{(r)} + \frac{u_j^{(r-1)}}{\mu_1}\right| - \frac{\lambda_1}{\mu_1}\right)_+ \text{ for } j = 1, \ldots, p,$$
$$b_j^{(r)} = \operatorname{sign}\left((\mathbf{D}\widehat{\beta}^{(r)})_j + \frac{v_i^{(r-1)}}{\mu_2}\right)\left(\left|(\mathbf{D}\widehat{\beta}^{(r)})_j + \frac{v_i^{(r-1)}}{\mu_2}\right| - \frac{\lambda_2}{\mu_2}\right)_+ \text{ for } j = 1, \ldots, m,$$

where $(x)_+ = \max(x, 0)$.

The SB algorithm needs to choose the additional parameters μ_1 and μ_2 and the step sizes δ_1 and δ_2. Ye and Xie (2011) show that the choices of these values do not affect the convergence of the SB algorithm. In the implementation of the SB algorithm, they use a real value μ such that $\mu = \mu_1 = \mu_2 = \delta_1 = \delta_2$ and consider pretrial procedure to obtain highest convergence rate. They suggest candidates of μ as $\{0.2, 0.4, 0.6, 0.8, 1\} \times \|\mathbf{y}\|_2$. They remark that the choice of μ can be improved but their suggestion empirically works well for all problem they tested. It is, however, known that the rate of convergence is very sensitive to the choice of the parameter μ (Ghadimi et al., 2012). This makes the SB algorithm to stall or to fail to reach the optimal solution. (We illustrate this problem in Appendix A.)

Alternating linearization algorithm

Lin et al. (2011) propose the alternating linearization (ALIN) algorithm for the structured regularization problem. The ALIN algorithm solves the minimization of the sum of two convex functions,

$$f(\beta) + h(\beta), \tag{2.25}$$

where $f(\beta)$ is a convex loss function and $h(\beta)$ is a convex penalty function. The ALIN algorithm achieves the minimum of (2.25) by alternately solving two subproblems, h-subproblem and f-subproblem which are derived by linearization of $h(\beta)$ and $f(\beta)$, respectively. Let $\{\widehat{\beta}^{(k)}\}_{k\geq 0}$ be a sequence of solutions of the problem (2.25) and $\{\tilde{\beta}_h^{(r)}\}_{r\geq 1}$ and $\{\tilde{\beta}_f^{(r)}\}_{r\geq 1}$ be sequences of solutions of h-subproblem and f-subproblem, respectively. After solving each subproblem, the ALIN checks the stopping criterion and the updating criterion. If the updating criterion is satisfied, the $\widehat{\beta}^{(k)}$ is updated by the solution of the subproblem. The ALIN algorithm goes on until the stopping criterion is satisfied.

In the h-subproblem, the solution $\tilde{\beta}_h^{(r)}$ is obtained by solving the approximation problem,

$$\min_{\beta} \tilde{f}(\beta) + h(\beta) + \frac{1}{2}\|\beta - \widehat{\beta}^{(k)}\|_{\mathbf{A}}^2,$$

where $\tilde{f}(\beta) = f(\tilde{\beta}_f^{(r-1)}) + \mathbf{s}_f^T(\beta - \tilde{\beta}_f^{(r-1)})$ is a linearized function of $f(\beta)$ at $\tilde{\beta}_f^{(r-1)}$, $\mathbf{s}_f \in \partial f(\tilde{\beta}_f^{(r-1)})$ is a subgradient of $f(\tilde{\beta}_f^{(r-1)})$, and $\|\beta - \widehat{\beta}\|_{\mathbf{A}}^2 = (\beta - \widehat{\beta})^T\mathbf{A}(\beta - \hat{\beta})$ with a diagonal matrix $\mathbf{A} = \mathrm{diag}(a_1, \ldots, a_p)$, $a_j > 0$, for $j = 1, 2, \ldots, p$. Then, the subgradient of $h(\tilde{\beta}_h^{(r)})$ is calculated by $\mathbf{s}_h = -\mathbf{s}_f - \mathbf{A}(\tilde{\beta}_h^{(r)} - \widehat{\beta}^{(k)}) \in \partial h(\tilde{\beta}_h^{(r)})$. The solution $\tilde{\beta}_f^{(r)}$ is similarly obtained by

solving the f-subproblem,

$$\min_{\beta} f(\beta) + \tilde{h}(\beta) + \frac{1}{2}\|\beta - \widehat{\beta}^{(r)}\|^2_{\mathbf{A}},$$

where $\tilde{h}(\beta) = h(\tilde{\beta}_h^{(r)}) + \mathbf{s}_h^T(\beta - \tilde{\beta}_h^{(r)})$ is a linearized function of $h(\beta)$ at $\tilde{\beta}_h^{(r)}$. The subgradient of $f(\tilde{\beta}_f^{(r)})$ is also calculated by $\mathbf{s}_f = -\mathbf{s}_h - \mathbf{A}(\tilde{\beta}_f^{(r)} - \widehat{\beta}^{(k)})$.

After solving each subproblem, the ALIN algorithm checks out each stopping criterion. The stopping criteria for the h-subproblem and the f-subproblem are defined as

$$h\text{−sub} \ : \ \tilde{f}(\tilde{\beta}_h) + h(\tilde{\beta}_h) \geq f(\widehat{\beta}) + h(\widehat{\beta}) - \epsilon,$$
$$f\text{−sub} \ : \ f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f) \geq f(\widehat{\beta}) + h(\widehat{\beta}) - \epsilon,$$

where ϵ is a tolerance of the algorithm. If the stopping criterion at each iteration is not satisfied, then the ALIN algorithm also checks out the updating criterion. The updating criteria for the h-subproblem and the f-subproblem are defined as

$$h\text{−sub} \ : \ f(\tilde{\beta}_h) + h(\tilde{\beta}_h) \leq (1-\gamma)[f(\widehat{\beta}) + h(\widehat{\beta})] + \gamma[\tilde{f}(\tilde{\beta}_h) + h(\tilde{\beta}_h)],$$
$$f\text{−sub} \ : \ f(\tilde{\beta}_f) + h(\tilde{\beta}_f) \leq (1-\gamma)[f(\widehat{\beta}) + h(\widehat{\beta})] + \gamma[f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f)],$$

where $\gamma \in (0, 1)$. If the updating criterion is satisfied, then $r = r + 1$ and the $\widehat{\beta}^{(r)}$ is updated as the solution of the subproblem; otherwise the ALIN algorithm goes on with the unchanged $\widehat{\beta}^{(r)}$.

The ALIN algorithm can solve the generalized lasso problem in (2.4). To be specific, recall that

$$f(\beta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2, \quad h(\beta) = \lambda\|\mathbf{D}\beta\|_1, \tag{2.26}$$

where $\mathbf{D}$ is an $m \times p$ dimensional matrix similarly defined in (2.5). The h-subproblem of the generalized lasso problem is defined as

$$\min_{\beta, \mathbf{z}} \mathbf{s}_f^T\beta + \lambda\|\mathbf{z}\|_1 + \frac{1}{2}\|\beta - \widehat{\beta}\|^2_{\mathbf{A}} \quad \text{subject to} \quad \mathbf{D}\beta = \mathbf{z}, \tag{2.27}$$

where $\mathbf{A} = \text{diag}\big(\mathbf{X}^T\mathbf{X}\big)$ and $\mathbf{s}_f = \mathbf{X}^T(\mathbf{X}\widehat{\beta} - \mathbf{y})$. To solve the h-subproblem, the approach in Tibshirani and Taylor (2011) is applied. The Lagrangian of the h-subproblem is,

$$L(\beta, \mathbf{z}, \mathbf{u}) = \mathbf{s}_f^T\beta + \lambda\|\mathbf{z}\|_1 + \mathbf{u}^T(\mathbf{D}\beta - \mathbf{z}) + \frac{1}{2}\|\beta - \widehat{\beta}\|_{\mathbf{A}}^2, \tag{2.28}$$

where $\mathbf{u}$ is a dual variable of h-subproblem. After minimizing over β and $\mathbf{z}$, the dual problem of the h-subproblem is

$$\begin{aligned} &\min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T\mathbf{D}\mathbf{A}^{-1}\mathbf{D}^T\mathbf{u} + \mathbf{u}^T\mathbf{D}(\widehat{\beta} - \mathbf{A}^{-1}\mathbf{s}_f) \\ &\text{subject to} \quad \|\mathbf{u}\|_\infty \leq \lambda. \end{aligned} \tag{2.29}$$

The solution of h-subproblem is obtained by the primal-dual relationship $\tilde{\beta}_h = \widehat{\beta} - \mathbf{A}^{-1}(\mathbf{s}_f + \mathbf{D}^T\mathbf{u}^*)$, where $\mathbf{u}^*$ is the optimal solution of (2.29). The dual problem is solved by the active-set box-constrained PCG algorithm with the diagonal of $\mathbf{D}\mathbf{A}^{-1}\mathbf{D}^T$ as a preconditioner. The f-subproblem of the generalized lasso problem is

$$\min_{\beta} \mathbf{s}_h^T\beta + \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{1}{2}\|\beta - \widehat{\beta}\|_{\mathbf{A}}^2, \tag{2.30}$$

where $\mathbf{A} = \text{diag}\big(\mathbf{X}^T\mathbf{X}\big)$. The solution of f-subproblem is obtained by solving the following linear system,

$$(\mathbf{X}^T\mathbf{X} + \mathbf{A})(\beta - \widehat{\beta}) = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\widehat{\beta}) - \mathbf{s}_h. \tag{2.31}$$

The PCG method is applied to solve the linear system (2.31) with a preconditioner $2\mathbf{A} = 2\text{diag}(\mathbf{X}^T\mathbf{X})$.

The ALIN algorithm has similar problem in Tibshirani and Taylor (2011). For solving the FLR in (1.1), we should put a p dimensional identity matrix into the matrix $\mathbf{D}$ to impose the penalty on the ℓ_1-norm of the coefficients.

It makes the dimension of the dual increase. Moreover, the matrix $\mathbf{D}\mathbf{A}^{-1}\mathbf{D}^T$ is not a positive definite matrix when the number of rows in $\mathbf{D}$ is greater than p. It violates the assumption of the PCG. Thus, the ALIN algorithm becomes inefficient when the number of rows in $\mathbf{D}$ is large.

2.3. Summary of the reviewed algorithms

Table 2.1 summarizes the existing algorithms explained in Chapter 2 and the MM algorithm proposed in Chapter 3 according to types of the design matrix $\mathbf{X}$ and penalty structure. The items marked with a filled circle ($\bullet$) denote the availability of the algorithm.

Table 2.1: Summary of algorithms to solve the FLSA and the FLR with the standard and the generalized fusion penalties.

Method	FLSA ($\mathbf{X} = \mathbf{I}$)		FLR (General $\mathbf{X}$)	
	STD	GEN	STD	GEN
Path-wise optimization	•			
Path algorithm for the FLSA	•	•		
Path algorithm for the generalized lasso	•	△	▲	▲
EFLA	•		•	
SPG algorithm	•	○	•	○
SB algorithm	•	•	•	•
ALIN algorithm	•	•	•	•
MM algorithm	•	•	•	•
MM algorithm with parallelizatioin	•		•	

STD denotes the standard fusion penalty $\sum_{j=2}^{p} |\beta_j - \beta_{j-1}|$.

GEN denotes the generalized fusion penalty $\sum_{(j,k)\in E} |\beta_j - \beta_k|$ for a given index set E.

• denotes that the method is applicable.

△ denotes that the dual solution path is computationally infeasible in large scale image denoising.

▲ denotes the method is only applicable when $\mathbf{X}$ has full rank.

○ denotes the method is not adequate since computation of spectral matrix norm when p is large.

Chapter 3

MM algorithm for fused lasso problem

3.1. MM algorithm for FLR

This section proposes an MM algorithm to solve the FLR (1.1). The MM algorithm works by iterating two steps, the majorization step and the minimization step. Given the current estimate $\widehat{\beta}^0$ of the optimal solution to (1.1), the majorization step finds the majorizing function $g\big(\beta\big|\widehat{\beta}^0\big)$ such that $f\big(\widehat{\beta}^0\big) = g\big(\widehat{\beta}^0\big|\widehat{\beta}^0\big)$ and $f\big(\beta\big) \leq g\big(\beta\big|\widehat{\beta}^0\big)$ for all $\beta \neq \widehat{\beta}^0$. The minimization step updates the estimate with

$$\widehat{\beta} = \underset{\beta}{\operatorname{argmin}}\ g\big(\beta\big|\widehat{\beta}^0\big).$$

It is known that the MM algorithm has a descent property in the sense that $f\big(\widehat{\beta}\big) \leq f\big(\widehat{\beta}^0\big)$. Furthermore, if $f\big(\beta\big)$ and $g\big(\beta\big|\widehat{\beta}^0\big)$ satisfy some regular-

ity conditions (for example, conditions in Vaida (2005)), the MM algorithm converges to the optimal solution of $f\big(\beta\big)$.

The MM algorithm is applied to the classical lasso problem by Hunter and Li (2005). They propose to minimize a perturbed version of the objective function of the classical lasso

$$f_{\lambda,\epsilon}(\beta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} \Big(|\beta_j| - \epsilon \log\big(1 + \frac{|\beta_j|}{\epsilon}\big)\Big)$$

instead of

$$f_{\lambda}(\beta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_1$$

to avoid occurrence of a zero denominator in the algorithm. The majorizing function for $f_{\lambda,\epsilon}(\beta)$ at $\widehat{\beta}^0$ is given by

$$g_{\lambda,\epsilon}(\beta|\widehat{\beta}^0) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 \ + \lambda \sum_{j=1}^{p} \left\{ |\widehat{\beta}_j^0| - \epsilon \log\left(1 + \frac{|\widehat{\beta}_j^0|}{\epsilon}\right) + \frac{\beta_j^2 - (\widehat{\beta}_j^0)^2}{2\Big(|\widehat{\beta}_j^0| + \epsilon\Big)} \right\}.$$

Hunter and Li (2005) show that the sequence $\{\widehat{\beta}^{(r)}\}_{r\geq 0}$ such that $\widehat{\beta}^{(r+1)} = g_{\lambda,\epsilon}\big(\beta\big|\widehat{\beta}^{(r)}\big)$ converges to the minimizer of $f_{\lambda,\epsilon}(\beta)$, and that $f_{\lambda,\epsilon}(\beta)$ converges to $f_{\lambda}(\beta)$ uniformly as ϵ approaches 0.

Motivated by the above development, we introduce a perturbed version $f_{\epsilon}(\beta)$ of the objective (1.1) and the majorizing function $g_{\epsilon}(\beta|\widehat{\beta}^0)$ for the FLR problem (1.1):

$$\begin{aligned} f_{\epsilon}(\beta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_1 \sum_{j=1}^{p} \left\{ |\beta_j| - \epsilon \log\Big(1 + \frac{|\beta_j|}{\epsilon}\Big) \right\} \\ +\lambda_2 \sum_{(j,k)\in E} \left\{ |\beta_j - \beta_k| - \epsilon \log\Big(1 + \frac{|\beta_j - \beta_k|}{\epsilon}\Big) \right\}, \end{aligned} \tag{3.1}$$

and

$$
\begin{aligned}
g_\epsilon(\beta|\widehat{\beta}^0) = \frac{1}{2}\|\mathbf{y}-\mathbf{X}\beta\|_2^2 & \\
+\lambda_1 \sum_{j=1}^{p} & \left\{ |\widehat{\beta}_j^0| - \epsilon \log\Big(1+\frac{|\widehat{\beta}_j^0|}{\epsilon}\Big) + \frac{\beta_j^2 - (\widehat{\beta}_j^0)^2}{2(|\widehat{\beta}_j^0|+\epsilon)} \right\} \\
+\lambda_2 \sum_{(j,k)\in E} & \left\{ \big|\widehat{\beta}_j^0 - \widehat{\beta}_k^0\big| - \epsilon \log\Big(1+\frac{|\widehat{\beta}_j^0 - \widehat{\beta}_k^0|}{\epsilon}\Big) \right. \\
& \left. + \frac{(\beta_j-\beta_k)^2 - (\widehat{\beta}_j^0 - \widehat{\beta}_k^0)^2}{2(|\widehat{\beta}_j^0 - \widehat{\beta}_k^0|+\epsilon)} \right\}.
\end{aligned} \tag{3.2}
$$

Below, we see that $g_\epsilon(\beta|\beta')$ majorizes $f_\epsilon(\beta)$ at β', and $g_\epsilon(\beta|\beta')$ has a unique global minimum for $\lambda_1 > 0$ regardless of the rank of $\mathbf{X}$.

Proposition 3.1. For $\epsilon > 0$ and $\lambda_1 > 0$, $g_\epsilon(\beta|\beta')$ majorizes $f_\epsilon(\beta)$ at β', and $g_\epsilon(\beta|\beta')$ has a unique global minimum for all β'.

Proof. For a given $\epsilon > 0$ and $x_0 \in \mathbb{R}$, let us consider the functions

$$
\begin{aligned}
p_\epsilon(x) &= |x| - \epsilon \log\big(1+\frac{|x|}{\epsilon}\big) \\
q_\epsilon(x|x^0) &= |x^0| - \epsilon \log\left(1+\frac{|x^0|}{\epsilon}\right) + \frac{x^2-(x^0)^2}{2(|x^0|+\epsilon)},
\end{aligned}
$$

and their difference $h(y|y^0) = q_\epsilon(x|x^0) - p_\epsilon(x)$ where $y = |x|$ and $y_0 = |x_0|$. The function $h(y|y^0)$ is twice differentiable and simple algebra shows that it is increasing for $y \geq y_0$ and decreasing for $y < y_0$. We have

$$
h\big(y\big|y^0\big) = q_\epsilon(x|x^0) - p_\epsilon(x) \geq h\big(y^0\big|y^0\big) = 0
$$

and the equality holds if and only if $y = y^0$ (equivalently, $|x| = |x^0|$). This shows that $q_\epsilon(x|x^0)$ majorizes $p_\epsilon(x)$ at $x = x^0$, and finally $g_\epsilon(\beta|\beta')$ majorizes $f_\epsilon(\beta)$ at $\beta = \beta'$ in terms of p_ϵ and q_ϵ.

The majorizing function $g_\epsilon(\beta|\beta')$ is strictly convex since

$$\frac{\partial^2 g_\epsilon(\beta|\beta')}{\partial \beta^2} = \mathbf{X}^T\mathbf{X} + \lambda_1 \mathbf{A}'_\epsilon + \lambda_2 \mathbf{B}'_\epsilon$$

is positive definite, where $\mathbf{A}'_\epsilon$ and $\mathbf{B}'_\epsilon$ are defined in (3.3). Thus, it has a unique global minimum point.

□

Let $\widehat{\beta}^{(r)}$ be the current estimate in r-th iteration. The minimization of the majorizing function $g_\epsilon(\beta|\widehat{\beta}^{(r)})$ can be performed by solving a linear system $\partial g_\epsilon(\beta|\widehat{\beta}^{(r)})/\partial\beta = 0$ for β. This linear system can be written as

$$\left(\mathbf{X}^T\mathbf{X} + \lambda_1 \mathbf{A}^{(r)} + \lambda_2 \mathbf{B}^{(r)}\right)\beta = \mathbf{X}^T\mathbf{y}, \tag{3.3}$$

where $\mathbf{A}^{(r)} = \mathrm{diag}(a_1^{(r)}, a_2^{(r)}, \ldots, a_p^{(r)})$ with $a_j^{(r)} = \frac{1}{|\widehat{\beta}_j^{(r)}|+\epsilon}$ for $1 \le j \le p$, and $\mathbf{B}^{(r)} = \left(b_{jk}^{(r)}\right)_{1\le j,k\le p}$ is a symmetric and positive semidefinite matrix with

$$\begin{aligned} b_{jj}^{(r)} &= \sum_{k:(j,k)\in E} \frac{1}{|\widehat{\beta}_j^{(r)} - \widehat{\beta}_k^{(r)}| + \epsilon}, \quad j = 1, \ldots, p, \\ b_{jk}^{(r)} &= b_{kj}^{(r)} = -\frac{1}{|\widehat{\beta}_j^{(r)} - \widehat{\beta}_k^{(r)}| + \epsilon}, \quad \forall (j,k) \in E. \end{aligned}$$

The procedure described so far is summarized in Algorithm 1 as the MM algorithm for the FLR with the generalized fusion penalty. Note that, once the matrix $\mathbf{X}^T\mathbf{X} + \lambda_1 \mathbf{A}^{(r)} + \lambda_2 \mathbf{B}^{(r)}$ is constructed, the linear system (3.3) can be efficiently solved. Since this matrix is symmetric and positive definite, hence Cholesky decomposition and back substitution can be applied, e.g., using `dpotrf` and `dpotri` functions in LAPACK (Anderson et al., 1995).

Furthermore, construction of the matrix $\lambda_1 \mathbf{A}^{(r)} + \lambda_2 \mathbf{B}^{(r)}$ is efficient in the MM algorithm. Note that the the generalized fusion penalty in (1.1) can be

written $\|\mathbf{D}\beta\|_1$ for a $m \times p$ dimensional matrix $\mathbf{D}$, where m is the number of penalty terms. In Algorithm 1, the matrix $\lambda_1\mathbf{A}^{(r)} + \lambda_2\mathbf{B}^{(r)}$ is constructed in $O(m)$ time, i.e., independent of the problem dimension p. This is in contrast to the other algorithms that allow the generalized fusion penalty reviewed in Chapter 2; the SPG algorithm requires to check a specific condition to guarantee its convergence; this takes at least $O\big(np\big)$ time for every iteration; the SB algorithm needs to construct the matrix $\mu\mathbf{I} + \mu\mathbf{D}^T\mathbf{D}$, where μ is the parameter discussed in (2.23), and it takes $O(mp^2)$ time; the ALIN and the path algorithm for generalized lasso need to compute the matrix $\mathbf{D}\mathbf{D}^T$ for solving the dual, hence requires $O(m^2p)$ time. The caveat with the MM algorithm is that the matrix $\lambda_1\mathbf{A}^{(r)} + \lambda_2\mathbf{B}^{(r)}$ needs to be constructed for each iteration. However, our experience is that the MM algorithm tends to converge within a few tens of iterations (See Section 5). Hence if p is more than a few tens, which is of our genuine interest, the MM algorithm is expected to run faster than the other algorithms.

When the matrix $\mathbf{M}^{(r)} = \lambda_1\mathbf{A}^{(r)} + \lambda_2\mathbf{B}^{(r)}$ is tridiagonal, as appears in the standard FLR or block tridiagonal as in the two-dimensional FLR, we can employ the preconditioned conjugate gradient (PCG) method by using the matrix $\mathbf{M}^{(r)}$ as the preconditioner, to efficiently solve the linear system (3.3). The PCG method is an iterative method for solving a linear system $\mathbf{Q}\mathbf{x} = \mathbf{c}$ having a symmetric positive definite matrix $\mathbf{Q}$ with a preconditioner $\mathbf{M}$ (Demmel, 1997). For example, we can use $\mathrm{diag}(\mathbf{X}^T\mathbf{X})$ as a preconditioner $\mathbf{M}$ if $\mathbf{Q} = \mathbf{X}^T\mathbf{X}$. The PCG method achieves the solution by iteratively

updating four sequences $\{\mathbf{x}_j\}_{j\geq 0}$, $\{\mathbf{r}_j\}_{j\geq 0}$, $\{\mathbf{z}_j\}_{j\geq 0}$, and $\{\mathbf{p}_j\}_{j\geq 1}$ using

$$\begin{aligned}\mathbf{x}_j &= \mathbf{x}_{j-1} + \nu_j \mathbf{p}_j,\\ \mathbf{r}_j &= \mathbf{r}_{j-1} + \nu_j \mathbf{Q}\mathbf{p}_j,\\ \mathbf{z}_j &= \mathbf{M}^{-1}\mathbf{r}_j,\\ \mathbf{p}_{j+1} &= \mathbf{z}_j + \gamma_j \mathbf{p}_j,\end{aligned}$$

where $\nu_j = (\mathbf{z}_{j-1}^T \mathbf{r}_{j-1})/(\mathbf{p}_j^T \mathbf{Q}\mathbf{p}_j)$, $\gamma_j = (\mathbf{z}_j^T \mathbf{r}_j)/(\mathbf{z}_{j-1}^T \mathbf{r}_{j-1})$, $\mathbf{x}_0 = 0$, $\mathbf{r}_0 = \mathbf{c}$, $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{c}$, and $\mathbf{p}_1 = \mathbf{z}_0$. For the standard FLR, the matrix $\mathbf{M}^{(r)}$ is tridiagonal and its Cholesky decomposition can be evaluated within $O\big(p\big)$ time, much faster than that of general positive definite matrices. In this case, we solve the linear system $\mathbf{M}^{(r)}\mathbf{z}_j = \mathbf{r}_j$ using `dpbtrf` and `dpbtrs` functions in LAPACK that perform Cholesky decomposition and linear system solving for symmetric positive definite band matrix, respectively. Algorithm 2 describes the proposed MM algorithm using the PCG method for the standard FLR and the two-dimensional FLR.

3.2. Convergence

This section shows the convergence of the proposed MM algorithm for the FLR. We first show that the minimizer of the perturbed version (3.1) of the objective (1.1), denoted by $\widehat{\beta}_\epsilon$, exhibits a minimal difference from the minimizer of the true objective (1.1), denoted by $\widehat{\beta}$, for small ϵ. We then prove that the sequence of solutions of the MM algorithm $\widehat{\beta}_\epsilon^{(r)}$, $r = 0, 1, 2, \ldots$, converges to $\widehat{\beta}_\epsilon$. In showing these results, we assume that $\{\beta \mid f_\epsilon(\beta) \leq c\}$ is a compact subset of $\mathbb{R}^p$ for any $\epsilon \geq 0$ and $c \geq 0$.

Theorem 3.1 below shows that $\widehat{\beta}_\epsilon$, the minimizer of $f_\epsilon(\beta)$, approaches to

Algorithm 1 MM algorithm for the FLR with the generalized fusion penalty

Input: $\mathbf{y}$, $\mathbf{X}$, λ_1, λ_2, convergence tolerance δ, perturbation constant ϵ.

1: **for** $j = 1, \cdots, p$ **do**

2: $\quad \widehat{\beta}_j^{(0)} \leftarrow \dfrac{X_j^T \mathbf{y}}{X_j^T X_j}$ ▷ initialization

3: **end for**

4: **repeat** $r = 0, 1, 2, \ldots$

5: $\quad \mathbf{M}^{(r)} \leftarrow \mathbf{X}^T\mathbf{X} + \lambda_1 \mathbf{A}^{(r)} + \lambda_2 \mathbf{B}^{(r)}$

6: $\quad \widehat{\beta}^{(r+1)} \leftarrow \left(\mathbf{M}^{(r)}\right)^{-1} \mathbf{X}^T \mathbf{y}$ ▷ using `dpotrf` and `dpotri` in LAPACK

7: **until** $\dfrac{|f(\widehat{\beta}^{(r+1)}) - f(\widehat{\beta}^{(r)})|}{|f(\widehat{\beta}^{(r)})|} \leq \delta$

Output: $\widehat{\beta}_\epsilon = \widehat{\beta}^{(r+1)}$

$\widehat{\beta}$, minimizer of $f(\beta)$, as $\epsilon \to 0$.

Theorem 3.1. Consider an arbitrary deceasing sequence $\{\epsilon_n, n = 1, 2, \ldots, \infty\}$ that converges to 0. Then, any limit point of $\widehat{\beta}_{\epsilon_n}$ is a minimizer of $f(\beta)$.

Proof. We first show that $f_\epsilon(\beta)$ approaches to $f(\beta)$ uniformly on $\beta \in \mathbf{C}$ as ϵ goes to 0, where $\mathbf{C}$ is a compact subset in $\mathbb{R}^p$. We recall that

$$f(\beta) - f_\epsilon(\beta) = \lambda_1 \sum_{j=1}^{p} \epsilon \log\left(1 + \frac{|\beta_j|}{\epsilon}\right) + \lambda_2 \sum_{(j,k)\in E, j<k} \epsilon \log\left(1 + \frac{|\beta_j - \beta_k|}{\epsilon}\right).$$

The uniform convergence of $f_\epsilon(\beta)$ is simply by noting that

$$\begin{aligned} \left|f(\beta) - f_\epsilon(\beta)\right| \leq \lambda_1 \sum_{j=1}^{p} \epsilon \log\left(1 + \frac{\max_j \left|\beta_j\right|}{\epsilon}\right) \\ + \lambda_2 \sum_{(j,k)\in E, j<k} \epsilon \log\left(1 + \frac{\max_{j,k} \left|\beta_j - \beta_k\right|}{\epsilon}\right), \end{aligned}$$

Algorithm 2

MM algorithm using the PCG method for the standard FLR and the two-dimensional FLR

Input: $\mathbf{y}$, $\mathbf{X}$, λ_1, λ_2, convergence tolerance δ, perturbation constant ϵ.

for $j = 1, \cdots, p$ **do**

  $\widehat{\beta}_j^{(0)} \leftarrow \dfrac{X_j^T \mathbf{y}}{X_j^T X_j}$ ▷ initialization

end for

repeat $r = 0, 1, 2, \ldots$

  $\mathbf{M}^{(r)} \leftarrow \lambda_1 \mathbf{A}^{(r)} + \lambda_2 \mathbf{B}^{(r)}$ ▷ preconditioner

  $\mathbf{x}_0 \leftarrow \widehat{\beta}^{(r)}$ ▷ begin the PCG method

  $\mathbf{r}_0 \leftarrow \mathbf{X}^T\mathbf{y} - (\mathbf{X}^T\mathbf{X} + \mathbf{M}^{(r)})\mathbf{x}_0$

  $\mathbf{z}_0 \leftarrow \left(\mathbf{M}^{(r)}\right)^{-1}\mathbf{r}_0$

  $\mathbf{p}_1 \leftarrow \mathbf{z}_0$

  repeat $j = 1, 2, \ldots$

    $\nu_j \leftarrow \dfrac{\mathbf{r}_{j-1}^T \mathbf{z}_{j-1}}{\mathbf{p}_j^T (\mathbf{X}^T\mathbf{X} + \mathbf{M}^{(r)})\mathbf{p}_j}$

    $\mathbf{x}_j \leftarrow \mathbf{x}_{j-1} + \nu_j \mathbf{p}_j$

    $\mathbf{r}_j \leftarrow \mathbf{r}_{j-1} + \nu_j (\mathbf{X}^T\mathbf{X} + \mathbf{M}^{(r)})\mathbf{p}_j$

    Solve $\mathbf{M}^{(r)}\mathbf{z}_j = \mathbf{r}_j$ ▷ using `dpbtrf` and `dpbtrs` in LAPACK

    $\gamma_j \leftarrow \dfrac{\mathbf{r}_j^T \mathbf{z}_j}{\mathbf{r}_{j-1}^T \mathbf{z}_{j-1}}$

    $\mathbf{p}_{j+1} \leftarrow \mathbf{z}_j + \gamma_j \mathbf{p}_j$ ▷ update conjugate gradient

  until $\dfrac{\|\mathbf{r}_j\|_2}{\|\mathbf{X}^T\mathbf{y}\|_2} \leq \delta$ ▷ end the PCG method

  $\widehat{\beta}^{(r+1)} \leftarrow \mathbf{x}_j$ ▷ update solution

until $\dfrac{|f(\widehat{\beta}^{(r+1)}) - f(\widehat{\beta}^{(r)})|}{|f(\widehat{\beta}^{(r)})|} \leq \delta$

Output: $\widehat{\beta}_\epsilon = \widehat{\beta}^{(r+1)}$

which monotonically decreases to 0 as ϵ goes to 0.

We now prove our main theorem in two steps. In the first step, we show that $\widehat{\beta}_\epsilon \in \mathbf{C}$, where $\mathbf{C}$ is compact set. In the second step, we show that $f(\widehat{\beta}_\epsilon) - f(\widehat{\beta})$ converges to 0.

First, simple algebra shows that the objective function

$$f_\epsilon(\beta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_1 \sum_{j=1}^{p} \left\{ |\beta_j| - \epsilon \log\left(1 + \frac{|\beta_j|}{\epsilon}\right) \right\} + \lambda_2 \sum_{(j,k)\in E, j<k} \left\{ |\beta_j - \beta_k| - \epsilon \log\left(1 + \frac{|\beta_j - \beta_k|}{\epsilon}\right) \right\},$$

is an infinitely differentiable and decreasing function in ϵ for $\epsilon > 0$. In addition, as shown at the beginning, as ϵ goes to 0, it approaches to $f(\beta)$ uniformly on $\beta \in \mathbf{C}$ for any compact set $\mathbf{C}$.

Define the level set

$$\Omega\big(\epsilon, c\big) = \big\{\beta \in \mathbb{R}^p \big| f_\epsilon\big(\beta\big) \le c\big\},$$

for every $\epsilon > 0$ and $c > 0$. Similarly, we define

$$\Omega\big(0, c\big) = \big\{\beta \in \mathbb{R}^p \big| f\big(\beta\big) \le c\big\}.$$

Then, $\Omega\big(\epsilon, c\big)$ is compact for every $\epsilon > 0$ and $c > 0$, and $\Omega\big(\epsilon, c\big)$ approaches $\Omega\big(0, c\big)$ almost surely.

By taking $\mathbf{C} = \Omega\big(\epsilon, f(\widehat{\beta})\big)$, for every $\alpha \in \Omega\big(\epsilon, \widehat{\beta}\big)$, $f_\epsilon(\alpha) \le f(\widehat{\beta})$. We also know that $f_\epsilon(\widehat{\beta}_\epsilon) \le f_\epsilon(\alpha)$ by definition of $\widehat{\beta}_\epsilon$. In summary,

$$\min_\gamma f_\epsilon(\gamma) = f_\epsilon(\widehat{\beta}_\epsilon) \le f_\epsilon(\alpha) \le f\big(\widehat{\beta}\big) = \min_\gamma f\big(\gamma\big).$$

This proves the first step $\widehat{\beta}_\epsilon \in \Omega\big(\epsilon, f(\widehat{\beta})\big)$.

Second, $f(\widehat{\beta}_\epsilon) \geq f(\widehat{\beta})$ is satisfied by the definition of $\widehat{\beta}_\epsilon$ and $\widehat{\beta}$, thus

$$\begin{aligned} 0 \leq f\big(\widehat{\beta}_\epsilon\big) - f\big(\widehat{\beta}\big) &\leq f\big(\widehat{\beta}_\epsilon\big) - f_\epsilon\big(\widehat{\beta}_\epsilon\big) + f_\epsilon\big(\widehat{\beta}\big) - f\big(\widehat{\beta}\big) \\ &\leq \big|f\big(\widehat{\beta}_\epsilon\big) - f_\epsilon\big(\widehat{\beta}_\epsilon\big)\big| + \big|f_\epsilon\big(\widehat{\beta}\big) - f\big(\widehat{\beta}\big)\big|. \end{aligned} \tag{3.4}$$

Since $\Omega\big(\epsilon, f(\widehat{\beta})\big)$ and $\Omega\big(0, f(\widehat{\beta})\big)$ are compact, the uniform convergence of $f_\epsilon(\beta)$ shows that both terms in (3.4) converge to 0 as ϵ decreases to 0. Thus, if β^* is a limit point of the sequence $\{\widehat{\beta}_{\epsilon_n}\}_{n\geq 1}$ for ϵ_n decreasing to 0, then, from the continuity of $f(\beta)$ in β,

$$\lim_{n\to\infty} f\big(\widehat{\beta}_{\epsilon_n}\big) = f\big(\beta^*\big) = f\big(\widehat{\beta}\big) = \min_\beta f\big(\beta\big).$$

□

Theorem 3.2 shows that the sequence $\{\widehat{\beta}^{(r)}\}_{r\geq 0}$ of solution generated by the MM algorithm converges to the minimizer of $f_\epsilon(\beta)$. We first need a fact that the updating rule for the MM algorithm is continuous (Lemma 3.1), and the result on the global convergence of this updating rule (Lemma 3.2) (Wu, 1983). The proof of Theorem 3.2 is similar to Vaida (2005).

Lemma 3.1. The function $M(\alpha) = \text{argmin}_\beta\, g_\epsilon(\beta|\alpha)$ is continuous in α, where $g_\epsilon(\beta|\alpha)$ is the proposed majorizing function of $f_\epsilon(\beta)$ at α.

Proof. Let us consider a sequence $\big\{\alpha_k, k = 0, 1, 2, \dots\big\}$ which converges to $\tilde{\alpha}$ with finite $f_\epsilon(\tilde{\alpha})$ and satisfies $f_\epsilon(\alpha_k) \leq f_\epsilon(\alpha_0) < \infty$ for $k \geq 1$. We then want to show that $M(\alpha_k)$ converges to $M(\tilde{\alpha})$.

We first show that the sequence $\big\{M(\alpha_k), k = 0, 1, 2, \dots\big\}$ is a subset of the level set

$$\Omega\big(\epsilon, f_\epsilon(\alpha_0)\big) = \big\{\beta \in \mathbb{R}^p \big| f_\epsilon(\beta) \leq f_\epsilon(\alpha_0)\big\}.$$

From the definitions of $g_\epsilon\big(\beta\big|\alpha\big)$ and $M(\alpha)$, for every k,

$$f_\epsilon\big(M(\alpha_k)\big) \le g_\epsilon\big(M(\alpha_k)\big|\alpha_k\big) \le g_\epsilon\big(\alpha_k\big|\alpha_k\big) = f_\epsilon\big(\alpha_k\big).$$

Thus $\big\{M(\alpha_k), k = 0, 1, 2, \dots\big\} \subset \Omega\big(\epsilon, f_\epsilon(\alpha_0)\big)$.

Since the level set $\Omega\big(\epsilon, f_\epsilon(\alpha_0)\big)$ is compact, the sequence $\big\{M\big(\alpha_k\big), k = 0, 1, 2, \dots\big\}$ is bounded. We can find a convergent subsequence $\big\{M(\alpha_{k_n}), n = 1, 2, \dots\big\}$; let its limit point be $\tilde{M}$. Then, by the continuity of $g_\epsilon\big(\beta\big|\alpha\big)$ in both β and α, for all β,

$$g_\epsilon\big(\tilde{M}\big|\tilde{\alpha}\big) = \overline{\lim}_{n\to\infty} g_\epsilon\big(M(\alpha_{k_n})\big|\alpha_{k_n}\big) \le \overline{\lim}_{n\to\infty} g_\epsilon\big(\beta\big|\alpha_{k_n}\big) = g_\epsilon\big(\beta\big|\tilde{\alpha}\big).$$

Thus $\tilde{M}$ minimizes $g_\epsilon\big(\beta\big|\tilde{\alpha}\big)$. Since the minimizer of $g_\epsilon\big(\beta\big|\alpha\big)$ is unique by Proposition 3.1, we have $\tilde{M} = M\big(\tilde{\alpha}\big)$ and $M\big(\alpha_{k_n}\big)$ converges to $M(\tilde{\alpha})$.

□

Lemma 3.2. (Global convergence theorem, Wu (1983)) A map M from points of X to subsets of X is called a point-to-set map on X. It is said to be closed at x if $x_k \to x$ for $x_k \in X$ and $y_k \to y$ for $y_k \in M(x_k)$ imply $y \in M(x)$. For point-to-point map, continuity implies closedness. Let the sequence $\{x_k\}_{k=0}^{\infty}$ be generated by $x_{k+1} \in M(x_k)$, where M is a point-to-set map on X. Let a solution set $\Gamma \subset X$ be given, and suppose that: (i) all points x_k are contained in a compact set $S \subset X$; (ii) M is closed over the complement of Γ; (iii) there is a continuous function u on X such that (a) if $x \notin \Gamma$, $u(y) > u(x)$ for all $y \in M(x)$, and (b) if $x \in \Gamma$, $u(y) \ge u(x)$ for all $y \in M(x)$. Then, all the limit points of $\{x_k\}$ are in the solution set Γ and $u(x_k)$ converges monotonically to $u(x)$ for some $x \in \Gamma$.

Theorem 3.2. Let $\mathbf{S}$ be a set of stationary points of $f_\epsilon(\beta)$ and $\{\widehat{\beta}^{(r)}\}_{r\geq 0}$ be a sequence of solutions of the proposed MM algorithm given by $\widehat{\beta}^{(r+1)} = M(\widehat{\beta}^{(r)}) = \operatorname{argmin}_\beta g_\epsilon(\beta|\widehat{\beta}^{(r)})$. Then limit points of $\{\widehat{\beta}^{(r)}\}_{r\geq 0}$ are stationary points of $f_\epsilon(\beta)$ and $f_\epsilon(\widehat{\beta}^{(r)})$ converges monotonically to $f_\epsilon(\beta^*)$ for some $\beta^* \in \mathbf{S}$, and they are minimizer of $f_\epsilon(\beta)$.

Proof. Apply Lemma 3.2 (global convergence theorem) with

$$M(\alpha) = \operatorname*{argmin}_\beta g_\epsilon(\beta|\alpha),$$
$$\Gamma = \left\{\beta \Big| \frac{\partial f_\epsilon(\beta)}{\partial \beta} = 0\right\},$$

and $u(\beta) = -f_\epsilon(\beta)$. Define the level set

$$\Omega\big(\epsilon, f_\epsilon(\widehat{\beta}^{(0)})\big) = \big\{\beta \in \mathbb{R}^p \big| f_\epsilon(\beta) \leq f_\epsilon(\widehat{\beta}^{(0)})\big\}.$$

Then, by the descent property of the MM algorithm, $\big\{\widehat{\beta}^{(r)}\big\}_{r\geq 0} \subset \Omega\big(\epsilon, f_\epsilon(\widehat{\beta}^{(0)})\big)$. From Lemma 3.1, $M(\alpha)$ is a point-to-point map and continuous in α. Thus, it is a closed map on any compact set of $\mathbb{R}^p$, and is also closed over the complement of Γ. The third condition of the Lemma 3.2 is from the strict convexity of $g_\epsilon(\beta\big|\alpha)$ in Proposition 3.1 and the definition of $M(\alpha)$. Thus, Lemma 3.2 shows that all the limit points of $\{\widehat{\beta}^{(r)}\}_{r\geq 0}$, are stationary points of $f_\epsilon(\beta)$, and $f_\epsilon(\beta^{(r)})$ converges monotonically to $f_\epsilon(\beta^*)$ fo some $\beta^* \in \Gamma$.

□

Combining the Theorems 3.1 and 3.2, we see that the solution of MM algorithm converges to the optimal solution of the FLR problem (1.1) as $\epsilon \to 0$.

Chapter 4

Parallelization of the MM algorithm with GPU

4.1. Parallelization tools

Graphics Processing Unit

Graphics processing units (GPUs) were originally developed as a display device conducting graphics operations but have evolved into a general-purpose computing workhorse for problems with abundant data-level parallelism (Owens et al., 2008). A GPU has a collection of thousands of light-weight computing cores, which makes it suitable for massively parallel applications. A GPU can run thousands of threads in parallel, in a single-instruction multiple-data (SIMD) fashion which means that many elements in the data set are processed by the same code in each step (Patterson and Hennessey, 1998).

GPUs are built up on a different architecture from traditional central processing units (CPUs), and their distinct memory hierarchy, illustrated in Figure 4.1, poses a challenge for programmers who want to unleash the full potential of GPUs. The history of evolution of GPU architectures and GPU programming models can be found in Owens et al. (2008).

Figure 4.1: Memory hierarchy and thread organization of compute unified device architecture.

Compute Unified Device Architecture

In the past, the graphics application program interfaces (APIs) should be used to program general-purpose parallel computations on GPU and needs the knowledge of graphics such as graphics pipeline. Recently, the compute unified device architecture (CUDA) (Kirk and Hwu, 2010; Farber, 2011) is designed for general-purpose computation by NVIDIA Corporation (Santa Clara, CA) and widely used as parallel-computing architecture. This architecture opens up the opportunity of the parallel computation using GPU without the knowledge of graphics. CUDA supports many programming languages such as C, Fortran, and Python with additional rules to access and control resources on GPU. In CUDA, every single instruction is expressed by a thread and a chosen number of threads is set to a block. A grid consists of these blocks and is corresponding to a graphics device. This thread organization also has memory hierarchy illustrated in Figure 4.1.

4.2. Parallel tridiagonal solvers

As explained in Chapter 3, the MM algorithm using the PCG method needs to solve the linear system $\mathbf{M}^{(r)}\mathbf{z}_j = \mathbf{r}_j$ in Algorithm 2 iteratively using the PCG method for each iteration. For the standard FLR, this linear system is a symmetric and positive definite tridiagonal linear system that can be efficiently solved by using the band Cholesky decomposition (Golub and Van Loan, 1996) followed by forward and backward substitution methods. This solution method, which we call Band Cholesky, requires $8p$ operations. Instead, we can reduce the computation time by using parallel algorithms that

solve a tridiagonal linear system. Some of these parallel algorithms can reduce the computation time by using many cores at once. In this Section, we introduce three parallel tridiagonal solvers and a hybrid algorithm thereof. In what follows, let $\mathbf{Ax} = \mathbf{b}$ be the tridiagonal linear system of interest, where $\mathbf{x} \in \mathbb{R}^p$,

$$\mathbf{A} = \begin{pmatrix} d_1 & u_1 & & & \mathbf{0} \\ l_2 & d_2 & u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & u_{p-1} \\ \mathbf{0} & & & l_p & d_p \end{pmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_p \end{pmatrix}.$$

Cyclic reduction

Cyclic reduction (CR), proposed by Hockney (1965), consists of the forward reduction step and the backward substitution step. In the forward reduction, the original system is repeatedly reduced to the smaller system having half the number of equations until only one or two equations remains. To be specific, all the even-indexed equations are updated in parallel at each step. Let the j-th equation, $l_j x_{j-1} + d_j x_j + u_j x_{j+1} = b_j$, be a even-indexed equation at the current step. This equation is updated with l'_j, d'_j, u'_j and b'_j by

$$\begin{aligned} l'_j &= -l_{j-1}\frac{l_j}{d_{j-1}} \\ d'_j &= d_j - u_{j-1}\frac{l_j}{d_{j-1}} - l_{j+1}\frac{u_j}{d_{j+1}} \\ u'_j &= -u_{j+1}\frac{u_j}{d_{j+1}} \\ b'_j &= b_j - b_{j-1}\frac{l_j}{d_{j-1}} - b_{j+1}\frac{u_j}{d_{j+1}}. \end{aligned} \tag{4.1}$$

For example, the equations 1,2, and 3 are reduced as the updated equation 2 by using the above equations (4.1) in Figure 4.2.

In the backward substitution, the solution of the original linear system is sequentially achieved by the equation in (4.2) after solving last two equations in the forward reduction. To be specific, let the j-th equation, $l'_j x_{j-1} + d'_j x_j + u'_j x_{j+1} = b'_j$, be an odd-indexed equation of the current step. Since x_{j-1} and x_{j+1} have been calculated already at previous step, the solution of x_j is obtained from

$$x_j = \frac{b'_j - l'_j x_{j-1} - u'_j x_{j+1}}{d'_j}. \tag{4.2}$$

For example, the solution of x_3 is obtained by (4.2) with the solutions of x_2 and x_4 in Figure 4.2.

The CR algorithm requires $17p$ operations. This is more costly than Band Cholesky if no parallelization is involved. But if sufficiently many cores are utilized at once, the CR algorithm achieves the solution in $2\log_2 p - 1$ steps while Band Cholesky does in $2p$ steps. The "step" means here that each reduction or substitution step. For example, a reduction step from the equations 1–9 to the equations 2, 4, 6, and 8 is a one step in Figure 4.2.

Parallel cyclic reduction

Parallel cyclic reduction (PCR), proposed by Hockney and Jesshope (1981), is similar to the CR algorithm but only has the forward reduction step. In the forward reduction, the PCR algorithm partitions each linear system at the current step to two smaller linear systems each one with even-indexed equations and the other with odd-indexed equations. This scheme is illustrated in Figure 4.3. Each equation is updated by the equation in (4.1) until all the

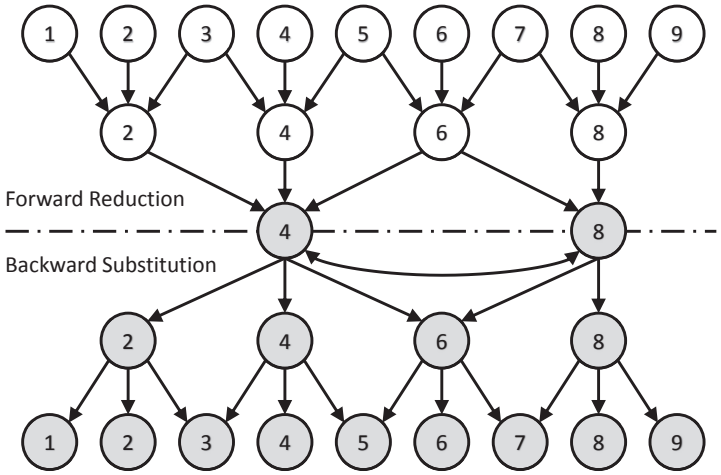


Figure 4.2: The CR algorithm algorithm for solving tridiagonal system with 9 variables. A gray circle means that a corresponding variable achieves the solution.

partitioned linear systems have one equation or two equations. Finally, the PCR solves all the linear systems in parallel.

The PCR algorithm requires $12p \log_2 p$ operations which is even more than the CR algorithm. But, the PCR achieves the solution with $\log_2 p$ steps if sufficiently many cores are used at once in each step. Thus, the PCR algorithm requires fewer steps than the CR algorithm although it requires more operations per step.

Recursive doubling

Recursive doubling (RD) is proposed by Stone (1973) and reformulated as the *scan form* by Eğecioğlu et al. (1989). The scan form of the RD algorithm

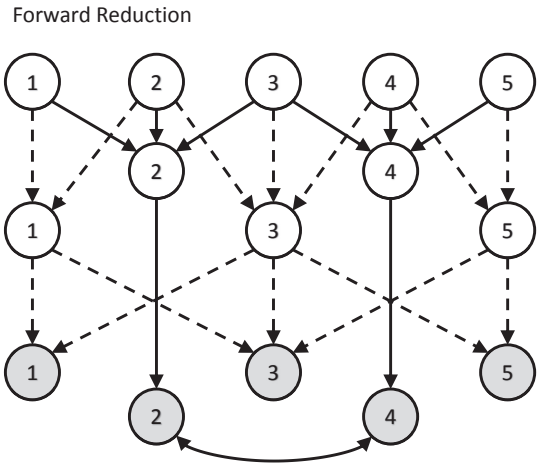


Figure 4.3: The PCR algorithm for solving tridiagonal system with 5 variables. A gray circle means that a corresponding variable achieves the solution.

is defined by

$$\begin{aligned}\text{scan}(\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_p) &\equiv [\mathbf{B}_1, \mathbf{B}_2\mathbf{B}_1, \ldots, \mathbf{B}_{p-1}\cdots\mathbf{B}_2\mathbf{B}_1, \mathbf{B}_p\cdots\mathbf{B}_2\mathbf{B}_1] \\ &\equiv [\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_{p-1}, \mathbf{C}_p],\end{aligned}$$

where $\mathbf{B}_j$ is a 3×3 dimensional matrix defined as

$$\mathbf{B}_j = \begin{pmatrix} -\dfrac{d_j}{u_j} & -\dfrac{l_j}{u_j} & \dfrac{b_j}{c_j} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Then the solution of the linear system is obtained from the matrices $\mathbf{C_1}, \ldots, \mathbf{C}_p$ by the following properties:

$$x_1 = -\frac{c_{13}}{c_{11}} \quad \text{and} \quad [\mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_p] \equiv [\mathbf{X}_1\mathbf{C}_2, \mathbf{X}_1\mathbf{C}_3, \ldots, \mathbf{X}_1\mathbf{C}_{p-1}],$$

where for $i = 1, 2, \ldots, p$, $\mathbf{C}_i = \prod_{j=1}^{i} \mathbf{B}_j$, $\mathbf{C}_p = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix}$, $\mathbf{X}_i = \begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}$,

and $x_0 = 0$.

This RD algorithm can be calculated by parallel fashion. The data parallel algorithm proposed by Hillis and Steele Jr (1986) is applied to calculate the matrices $\mathbf{C_1}, \ldots, \mathbf{C}_p$. This algorithm starts with $\mathbf{B_1}, \ldots, \mathbf{B}_p$, then the partial products, $\prod_{j=k}^{l} \mathbf{B}_j$, are iteratively calculated in parallel until $\mathbf{C_1}, \ldots, \mathbf{C}_p$ are obtained. Figure 4.4 shows the pattern of the partial products.

With the data parallel algorithm (Hillis and Steele Jr, 1986), the RD algorithm needs $20p \log_2 p$ operations and achieves the solution with $\log_2 p + 2$ steps. Although it has even more computational cost than the PCR, the required number of steps is similar to the PCR.

Hybrid algorithm

The hybrid algorithm, proposed by Zhang et al. (2010), combines the CR algorithm with the PCR or the RD algorithm. This hybrid algorithm is motivated by the two features of the existing parallel algorithms:

(i) The CR algorithm is the best parallel algorithm with respect to the computational complexity $\big(O(p)\big)$, but it loses the advantage of parallelization near the end of the forward reduction and the beginning of the backward substitution.

(ii) Although the PCR and the RD algorithms have high computational complexity $\big(O(p \log_2 p)\big)$ than the CR, they requires fewer steps than the CR algorithm.

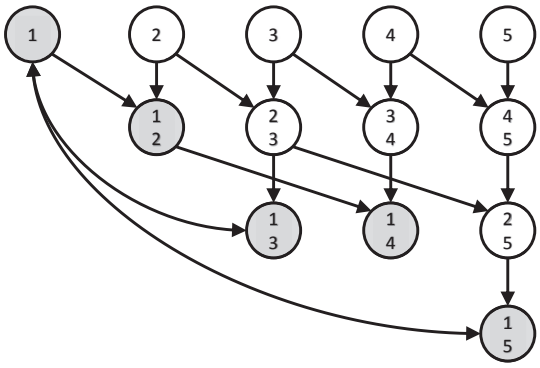


Figure 4.4: The pattern of the partial products in the RD algorithm for solving tridiagonal system with 5 variables. Let k and l be the first and the second number in a circle, respectively. A circle with k, l denotes $\prod_{j=k}^{l} \mathbf{B}_j$. A gray circle means that the desired partial product is achieved.

From this observation, the hybrid algorithm starts with the forward reduction of the CR algorithm. When the size of the reduced system in the forward reduction reaches a given threshold, the hybrid algorithm switches the CR algorithm to the PCR or the RD algorithm. After solving the reduced system, the hybrid algorithms apply the backward substitution of the CR algorithm with the solution of the reduced system. The scheme of the hybrid algorithms are illustrated in Figure 4.5.

The hybrid algorithm based on the CR and the PCR algorithms is denoted by CR-PCR. (CR-RD is similarly defined.) Let m be a given size of the reduced system in the forward reduction. The CR-PCR has a computational complexity as $17(p - m) + 12m \log_2 m$ and needs $2 \log_2 p - \log_2 m - 1$ steps

to achieve the solution of the original linear system. The CR-RD has a computational complexity as $17(p-m)+20m\log_2 m$ and takes $2\log_2 p - \log_2 m + 1$ steps to obtain the solution.

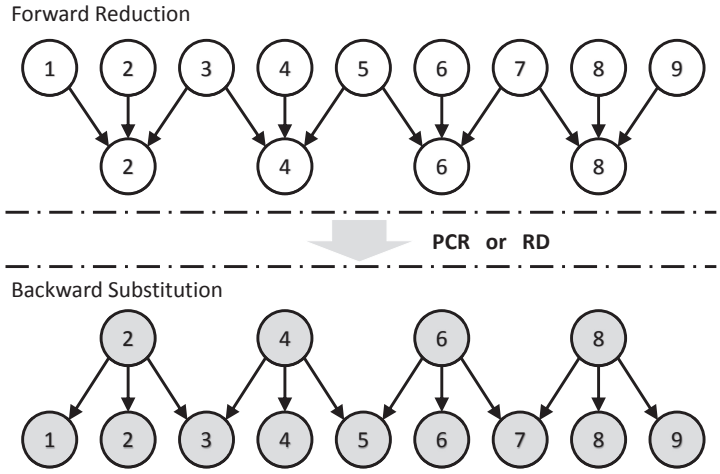


Figure 4.5: The hybrid algorithms for solving tridiagonal system with 9 variables. A gray circle means that a corresponding variable achieves the solution.

4.3. Parallelization of the MM algorithm

The parallel algorithms for solving a tridiagonal system, explained in the preceding section, are suitable for GPU implementation, as GPU has thousands of cores that can be utilized at once. Hence the inner iteration of Algorithm 2 for the standard FLR can be greatly accelerated by the GPU parallelization. Moreover, the hybrid algorithm CR-PCR is implemented in the CUDA sparse matrix (`cuSPARSE`) library as the function `cusparseDgtsv`.

Thus, we adopt the CR-PCR algorithm to accelerate the MM algorithm for the standard FLR since the CR-PCR is the best algorithm on GPUs.

In addition to the tridiagonal system, Algorithm 2 also contains a set of linear operations suitable for parallelization on the SIMD architecture of GPU. For instance, lines 7, 11–13, and 15–17 rely on basic linear operations such as matrix-matrix multiplication, matrix-vector multiplication, vector-vector addition, inner-product, and ℓ_2-norm computation. These operations involve identical arithmetics for each coordinate, hence the SIMD architecture of GPU is appropriate for parallelizing the algorithm. The CUDA basic linear algebra subroutines (`cuBLAS`) library provides an efficient implementation of such operations on CUDA, hence we use `cuBLAS` for parallelizing lines 7, 11–13, and 15–17 with GPU.

Unfortunately, our parallelization of the PCG method is limited to the standard FLR and at this moment is not extendable to the generalized fusion penalty. In this case, we recommend to solve the tridiagonal linear system (3.3) with Band Cholesky LAPACK functions `dpotrf` and `dpotri` using CPUs as in Algorithm 1. Currently this is more efficient than using GPUs.

Chapter 5

Numerical studies

In this section, we compare the performance of the MM algorithm with the other existing algorithms using several data sets that generally fit into the "large p, small n" setting. The general QP solvers are excluded in the comparison, since they are slower than other specialized algorithms for the FLR problems (Ye and Xie, 2011; Lin et al., 2011). These numerical studies include five scenarios for the FLR problem. In the first three scenarios, we consider the standard FLR problems with various sparsities of the true coefficients. Since the path algorithms have restrictions on solving the FLR with general design matrices, we only compare the proposed MM algorithms (with and without GPU acceleration, denoted by MMGPU and MM, respectively) with EFLA, SPG, and SB algorithms. In next two scenarios, we consider the two-dimensional FLR problem with a general design matrix and the image denoising problem. In the two-dimensional FLR with a general design matrix, The MM, SPG, and SB algorithms are only available. In the

image denoising problem equivalent to the two-dimensional FLSA problem ($\mathbf{X} = \mathbf{I}$), the path algorithms are also available. We additionally consider the path algorithm for FLSA with the generalized fusion penalty (**PathFLSA**).

The convergence of the algorithm is measured using a relative error of the objective function at each iteration. The relative error at the $r-$th iteration is defined by, as in Algorithms 1 and 2, $\mathrm{RE}(\widehat{\beta}^{(r)}) = \frac{|f(\widehat{\beta}^{(r)}) - f(\widehat{\beta}^{(r-1)})|}{f(\widehat{\beta}^{(r-1)})}$, where $\widehat{\beta}^{(r)}$ is the estimate in r-th iteration. In all algorithms, the iteration ends when the relative error becomes smaller than 10^{-5} or the number of iterations exceeds 10000. The SB algorithm and the proposed MM algorithm has additional parameters μ and ϵ, respectively. The details of choosing μ of the SB algorithm are given in Appendix B. In the MM algorithm, we set the perturbation $\epsilon = 10^{-8}$ to avoid machine precision error.[1]

The algorithms are compared in the computation the time and the number of iterations for aforementioned five scenarios. We also investigate the sensitivity of the SB algorithm and the MM algorithm to the choice of their additional parameters μ and ϵ. All algorithms are implemented in MATLAB except for PathFLSA, The computation times are measured in CPU time by using a desktop PC (Intel Core2 extreme X9650 CPU (3.00 GHz) and 8 GB RAM) with NVIDIA GeForce GTX 465 GPU.

[1] Although setting ϵ a constant does not exactly satisfy the condition for Theorem 3.1, this choice of constant is sufficiently small within the machine precision and to prevent divide by zero.

5.1. Standard FLR

We consider three scenarios for the standard FLR and check the efficiency and the stability of the five algorithms; MMGPU, MM, EFLA, SPG, and SB algorithms; with $(\lambda_1, \lambda_2) \in \big\{(0.1, 0.1), (0.1, 1), (1, 0.1), (1, 1)\big\}$.[2] We generate n samples $X_1^T, X_2^T, \ldots, X_n^T$ from a p-dimensional multivariate normal distribution $N(\mathbf{0}, \mathbf{I}_p)$. The response variable $\mathbf{y}$ is generated from the model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon, \tag{5.1}$$

where $\epsilon \sim N\big(0, \mathbf{I}_n\big)$ and $\mathbf{X} = (X_1^T, X_2^T, \ldots, X_n^T)^T$. To examine the performance with the dimension p, we change p for sample size $n = 1000$. The candidate dimensions are 200, 1000, 10000, and 20000. We generate 10 data sets according to the following scenarios (**C1**)–(**C3**).

(C1) Sparse case

Following Ye and Xie (2011), we set 41 coefficients of β nonzero:

$$\beta_j = \begin{cases} 2 & \text{for } \ j = 1, 2, \ldots, 20, 121, \ldots, 125 \\ 3 & \text{for } \ j = 41 \\ 1 & \text{for } \ j = 71, \ldots, 85 \\ 0 & \text{otherwise} \end{cases}$$

(C2) Moderately sparse case

[2]These choices of regularization parameters are similar in Liu et al. (2010); Lin et al. (2011); Ye and Xie (2011).

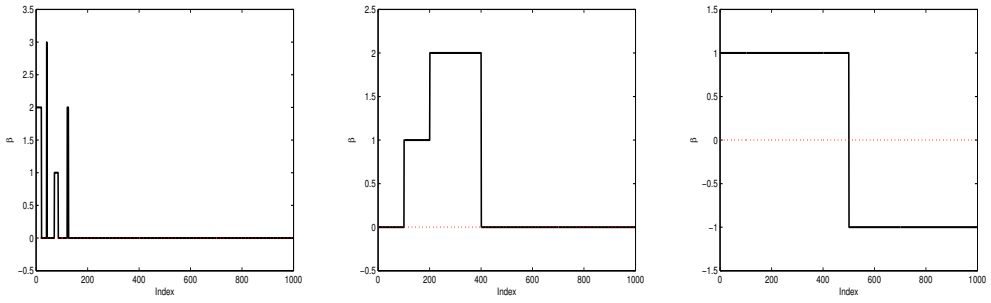
Following Lin et al. (2011), we set 30% of the coefficients nonzero:

$$\beta_j = \begin{cases} 1 & \text{for } \; j = p/10+1,\; p/10+2,\; \ldots,\; 2p/10 \\ 2 & \text{for } \; j = 2p/10+1,\; 2p/10+2,\; \ldots,\; 4p/10 \\ 0 & \text{otherwise} \end{cases}$$

(C3) Dense case

We set all the coefficients nonzero:

$$\beta_j = \begin{cases} 1 & \text{for } \; j = 1,\; 2,\; \ldots,\; 5p/10 \\ -1 & \text{for } \; j = 5p/10+1,\; 5p/10+2,\; \ldots,\; p \end{cases}$$



(a) Case 1 **(C1)** (b) Case 2 **(C2)** (c) Case 3 **(C3)**

Figure 5.1: Plots of the true coefficients in (**C1**)–(**C3**).

These true coefficient structures are illustrated in Figure 5.1.

We report the averages of computation times, the number of iterations, and the objective function values at the solution in Appendix B, Tables B.1–B.9 and summarize the average computation times in Figure 5.2. Excluding MMGPU (discussed below), EFLA is generally the fastest in most of the

scenarios considered. For relatively dense case (**(C2)** and **(C3)**) with a small penalty ($(\lambda_1, \lambda_2) = (0.1, 0.1)$) for large dimensions ($p = 10000, 20000$), MM is the fastest. We see that MM is $1.14 \sim 4.85$ times slower than EFLA, and is comparable to SPG and similar to SB for high dimensions ($p = 10000$ or 20000), each of which is the contender for the second place with MM in the respective dimensions. A detailed report on the computation time is given in Appendix B.

The benefit of parallelization is visible with large dimensions. MMGPU has within about 10% of the computation time of MM and the fastest among all the algorithms considered when $p = 10000$ and 20000. MMGPU is $2 - 63$ times faster than EFLA, which in turn is the fastest among MM, SPG, and SB. When the dimension is small ($p = 200$), however, MMGPU is the slowest among the five algorithms. This is due to the memory transfer overhead between CPU and GPU. When p is small, memory transfer occupies most of computation time. As the dimension increases, the relative portion of the memory transfer in computation time decreases, and MMGPU becomes efficient.

Focusing on the number of iterations, we see that MM converges within a few tens of iterations whereas other algorithms require up to a few hundreds of iterations, especially for large p. The number of iterations of MM is also insensitive to the sparsity structure and the choice of λ_1 and λ_2. (EFLA and SPG need more iterations as the coefficients become dense, together with SB, they also require more iterations for small λ_1 or λ_2.) This stability in the number of iterations contribute the performance gain in MMGPU. MMGPU speeds up each iteration, hence if the number of iteration does not

rely much on the input, we can consistently expect a gain by parallelization. Moreover, MM has smaller objective function values at the obtained solution than EFLA, SPG, and SB in three scenarios. This means that the solution of MM is closer to the global minimizer than other algorithms with same convergence tolerance.

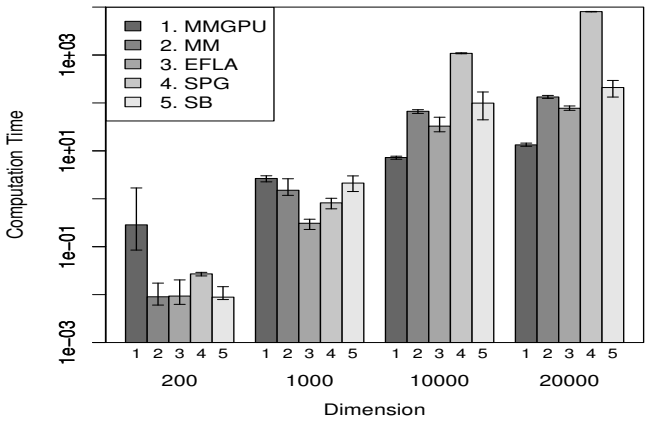
5.2. Two-dimensional FLR

We consider two scenarios for the two-dimensional FLR, where the coefficients are indexed on $q \times q$ lattice (hence $p = q^2$) and the penalty structure is imposed on the horizontally and vertically adjacent coefficients. The first is the two-dimensional FLR with general design matrix $\mathbf{X}$. The second is image denoising, which is equivalent to the two-dimensional FLSA ($\mathbf{X} = \mathbf{I}$). Note that EFLA and MMGPU are not applicable for these scenarios. Instead, the path algorithm for FLSA (PathFLSA, Hoefling (2010)) is considered for the second scenario, since it is known as an efficient algorithm for the two-dimensional FLSA.

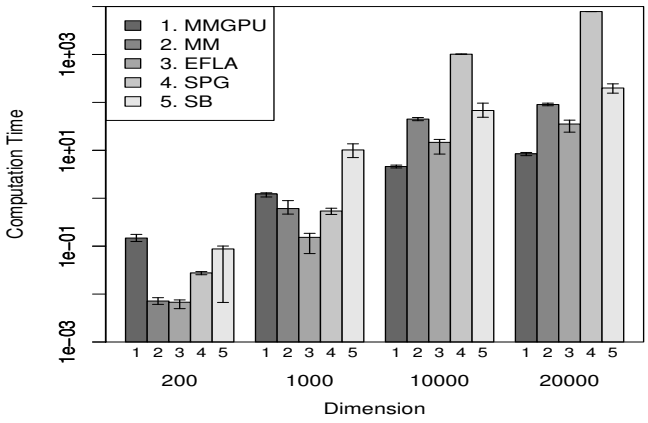
For the example of the two-dimensional FLR problem with general $\mathbf{X}$, we generate 10 data sets according to the following scenario.

(C4) Two-dimensional FLR with general design matrix, moderately sparse case.

We set the $q \times q$ dimensional matrix $B = (b_{ij})$ as follows.

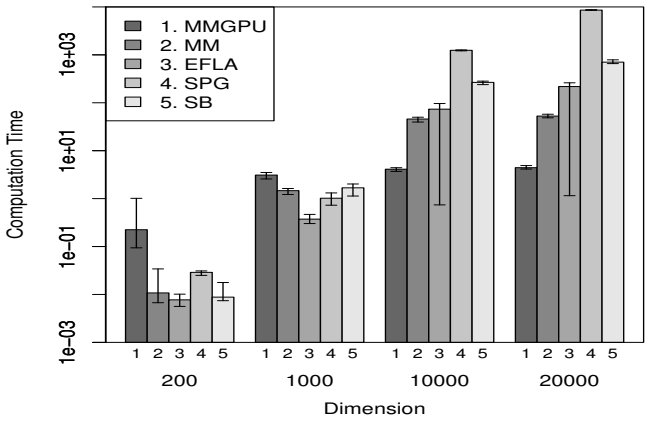


(a) $(\lambda_1, \lambda_2) = (0.1, 0.1)$

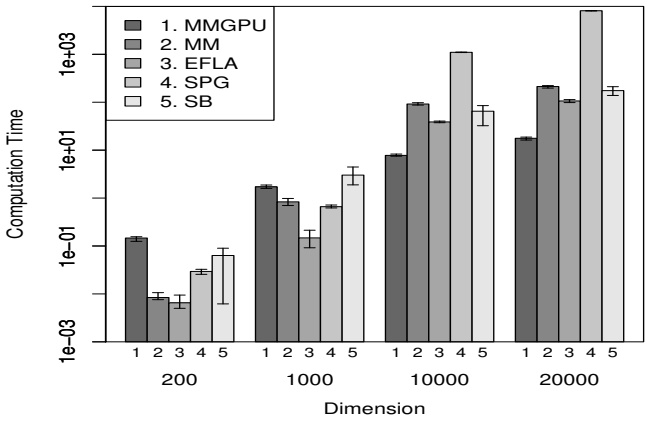


(b) $(\lambda_1, \lambda_2) = (1, 1)$

Figure 5.2: Summary of the results of **(C1)** for $n = 1000$. The vertical lines denote the range of computation times for 10 data sets.

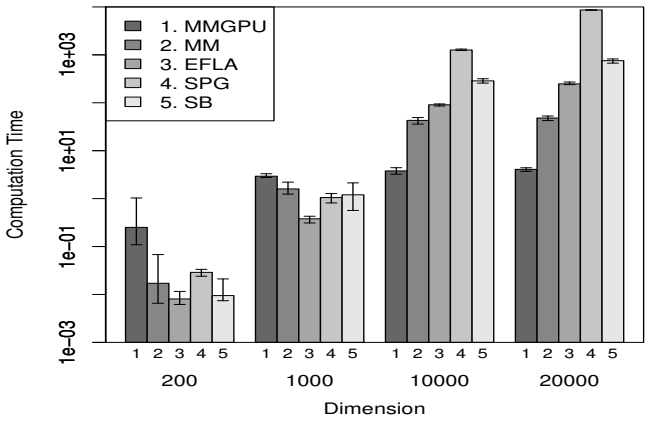


(a) $(\lambda_1, \lambda_2) = (0.1, 0.1)$

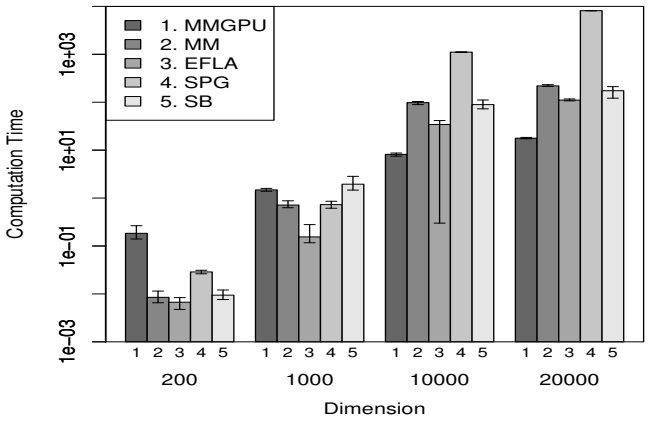


(b) $(\lambda_1, \lambda_2) = (1, 1)$

Figure 5.3: Summary of the results of **(C2)** for $n = 1000$. The vertical lines denote the range of computation times for 10 data sets. Let p denote the dimension of problem. For $(\lambda_1, \lambda_2) = (0.1, 0.1)$ and $p = 10000,\ 20000$, EFLA fails to converge the optimal solution at the lower bounds of vertical lines of EFLA (See Appendix B.)



(a) $(\lambda_1, \lambda_2) = (0.1, 0.1)$



(b) $(\lambda_1, \lambda_2) = (1, 1)$

Figure 5.4: Summary of the results of **(C3)** for $n = 1000$. The vertical lines denote the range of computation times for 10 data sets. Let p denote the dimension of problem. For $(\lambda_1, \lambda_2) = (1, 1)$ and $p = 10000$, EFLA fails to converge the optimal solution at the lower bounds of vertical lines of EFLA (See Appendix B.)

For $k = 0, 1, 2, 3$,

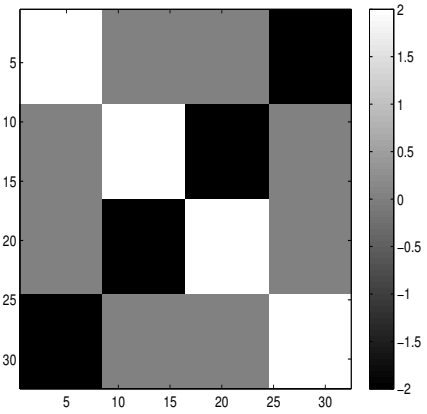
$$b_{ij} = \begin{cases} 2 & \text{for } \frac{q}{4}k + 1 \le i, j \le \frac{q}{4}(k+1) \\ -2 & \text{for } \frac{q}{4}k + 1 \le i \le \frac{q}{4}(k+1) \ \frac{q}{4}(3-k) + 1 \le j \le \frac{q}{4}(4-k) \\ 0 & \text{otherwise} \end{cases}$$

The true coefficient vector β is the vectorization of the matrix B denoted by $\beta = \text{vec}(B)$. The structure of the two-dimensional coefficient matrix B is shown in Figure 5.5 (a).

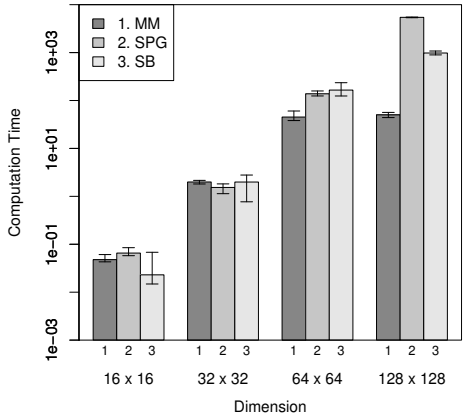
As the previous scenarios, we generate $\mathbf{X}$ and $\mathbf{y}$ from equation (5.1) with true coefficients defined in (**C4**). The candidate dimensions are $q = 16,\ 32,\ 64,\ 128$ and $n = 1000$. We also consider four sets of regularization parameters $(\lambda_1, \lambda_2) \in \{(0.1, 0.1), (0.1, 1), (1, 0.1), (1, 1)\}$ as in (**C1**) – (**C3**).

The computation times of (**C4**) are shown in Figure 5.5 (b) and (c). MM gains its competitiveness competitive as the dimension $p\ (= q^2)$ increases. MM is $1.6 \sim 19.5$ times faster than SB when $p > n$, has similar performance to SPG when $p = 10000$, and MM is $1.3 \sim 12.3$ times faster than SPG when $p = 20000$. A detailed report on the computation time can be found in Appendix B, Table B.10. It can be seen that the variation in the number of iterations for both dimensions and values of (λ_1, λ_2) is the smallest for MM. As shown in the results of (**C1**) – (**C3**), MM also has the smaller objective function values than SPG and SB in (**C4**). Hence MM is more stable than SPG and SB. This stability is consistent with the observations from (**C1**) – (**C3**).

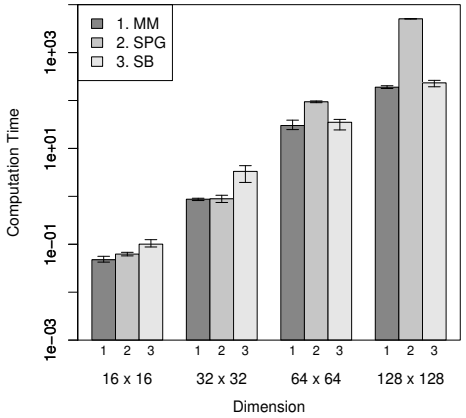
(C5) Image denoising ($\mathbf{X} = \mathbf{I}$)



(a) True Coefficients



(b) $(\lambda_1, \lambda_2) = (0.1, 0.1)$



(d) $(\lambda_1, \lambda_2) = (1, 1)$

Figure 5.5: Summary of Case 4 **(C4)** for sample size $n = 1000$.

We use a 256×256 gray scale image (i.e., $q = 256$, $p = q^2 = 65536$) of R.A. Fisher, as used in Friedman et al. (2007). After standardizing

pixel intensities, we add Gaussian noise with a standard variation 0.3. We set $\lambda_1 = 0$ because the application is image denoising.

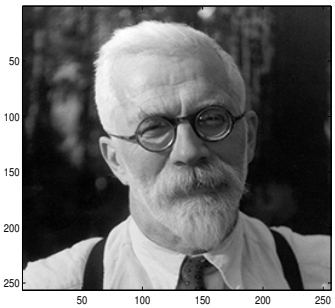
We consider two regularization parameters $\lambda_2 = 0.1, 1$. We report the average computation times and the average numbers of iterations in Table 5.1. We present the true image, the noisy image, and the denoising results at $\lambda_2 = 0.1$ in Figure 5.6. Since the path for all the solutions for two-dimensional FLSA is computationally infeasible, we terminate PathFLSA at the desired value of λ_2. For small λ_2 ($\lambda_2 = 0.1$), MM is slower than SPG and PathFLSA, but much faster than SB. As λ_2 increases, MM becomes faster than PathFLSA since PathFLSA always has to start from $\lambda_2 = 0$. In addition, SPG fails to obtain the solution at $\lambda_2 = 1$. (See the Figure 5.7.) MM also exhibits smaller variation in the number of iterations and the objective function value at the obtained solution than SPG and SB. Again, this stability is a consistent behavior of MM throughout $(\mathbf{C1}) - (\mathbf{C5})$.

Table 5.1: Summary of (**C5**) with computation times and the numbers of iterations of MM, SPG, SB, and PathFLSA. N/A denotes the method fails to obtain the optimal solution.

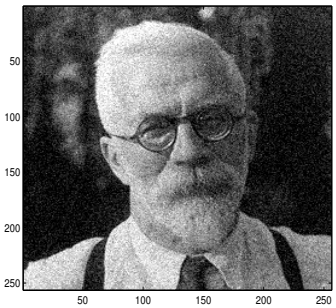
(λ_1, λ_2)	Method	Computation time (sec.)	# of iterations*	$f(\widehat{\beta})$
$(0, 0.1)$	MM	21.2082	26	2810.16
	SPG	1.9964	140	2817.31
	SB	217.2167	698	2822.40
	PathFLSA	3.2151	-	2809.96
$(0, 1)$	MM	48.7812	56	6920.43
	SPG$^{\triangle}$	N/A	N/A	N/A
	SB	635.8651	4431	7308.07
	PathFLSA	359.2660	-	6919.06

$*$ Since PathFLSA is a path algorithm, the number of iterations of PathFLSA is omitted.

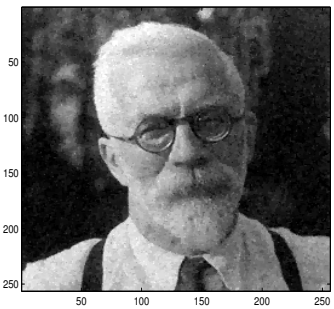
$\triangle$ SPG stops after 11 iterations with $f(\widehat{\beta}) = 35595.82$.



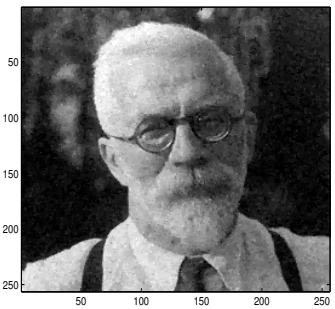
(a) True image



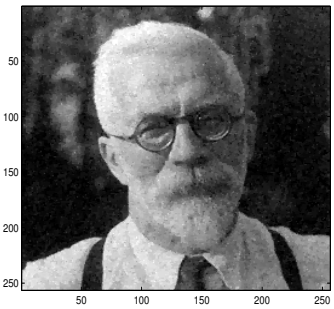
(b) Noisy image



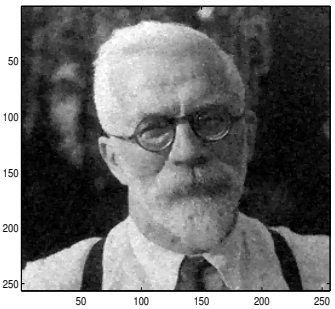
(c) MM



(d) SPG

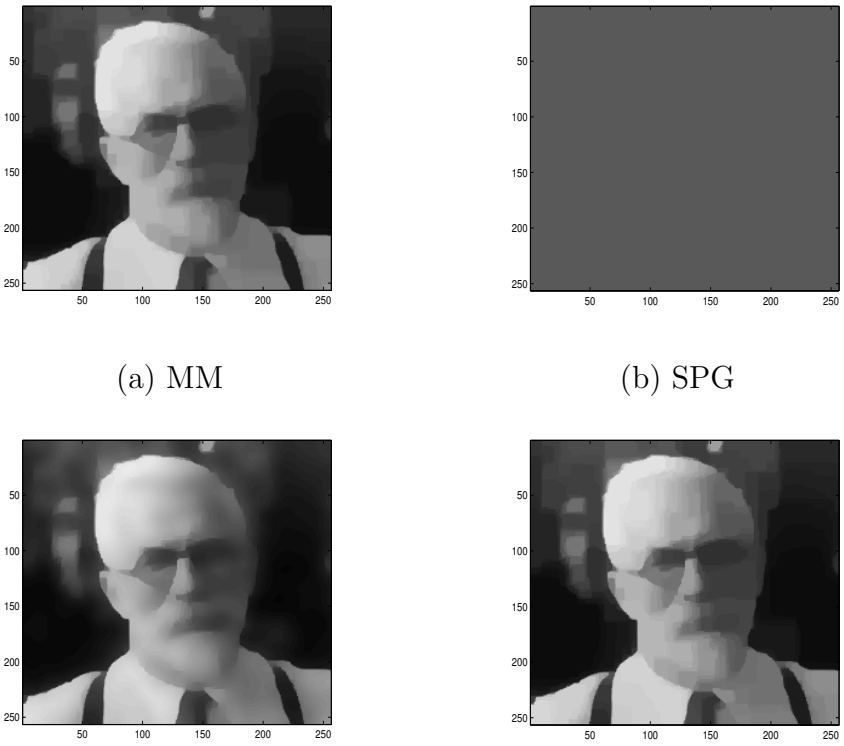


(e) SB



(f) PathFLSA

Figure 5.6: Image denoising with various algorithms for the FLR ($\lambda_2 = 0.1$)



(a) MM (b) SPG

(c) SB (d) PathFLSA

Figure 5.7: Image denoising with various algorithms for the FLR ($\lambda_2 = 1$). SPG fails to obtain the optimal solution.

Chapter 6

Conclusion

In this thesis, we have proposed an MM algorithm for the fused lasso regression (FLR) problem with the generalized fusion penalty. The proposed algorithm is flexible in the sense that it can be applied to the FLR with a wide class of design matrix and penalty structure. It is stable in the sense that the convergence of the algorithm is not sensitive to the dimension of the problem, the choice of the regularization parameters, and the sparsity of the true model. Even when a special structure on the design matrix or the penalty is imposed, the MM algorithm shows comparable performance to the algorithms tailored to the special structure, though it may not be the fastest. These features make the proposed algorithm an attractive choice as an off-the-shelf FLR algorithm. Moreover, the performance of the MM algorithm can be improved by parallelizing it with GPU, when the standard fused lasso penalty is imposed and the dimension increases. Extension of the GPU algorithm to the two-dimensional FLR problem is our future direction of research.

Bibliography

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. (1995), *LAPACK Users' Guide, Second Edition*. SIAM, Philadelphia, PA.

Beck, A. and Teboulle, M. (2009), A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, **2**, 183–202.

Bondell, H.D. and Reich, B.J. (2008). Simultaneous regression shrinkage, variable selection and clustering of predictors with OSCAR. *Biometrics*, **64**(1), 115–123.

Chen, X., Lin, Q., Kim, S., Carbonell, J.G., and Xing, E.P. (2012), Smoothing proximal gradient method for general structured sparse regression. *Annals of Applied Statistics*, **6**(2), 719–752.

Demmel, J.W. (1997), *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA.

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

Eğecioğlu, Ö., Koc, C.K., and Laub, A.J. (1989), A recursive doubling algorithm for solution of tridiagonal systems on hypercube multiprocessors. *Journal of Computational and Applied Mathematics*, **27**, 95–108.

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

Farber, R. (2011), *CUDA application design and development*. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007), Pathwise coordinate optimization. *The Annals of Applied Statistics*, **1**(2), 302–332.

Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M. (2012), On the optimal step-size selection for the alternating direction method of multipliers. In *Proceedings IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys)*.

Gill, P.E., Murray, W., and Saunders, M.A. (1997), User's guide for snopt 5.3: A fortran package for large-scale nonlinear programming. *Technical Report NA 97-4*. University of California, San Diego.

Golub, G.H. and Van Loan, C.F. (1996), *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD..

Grant, M., Boyd, S., and Ye, Y. (2008), *CVX: Matlab software for disciplined convex programming*. Web page and software, `http://stanford.edu/~boyd/cvx`.

Hestenes, M.R. (1969), Multiplier and gradient methods. *Journal Optimization Theory & Applications*, **4**, 303–320.

Hillis, W.D. and Steele Jr, G.L. (1986), Data parallel algorithms. *Communications of the ACM*, **29**(12), 1170–1183.

Hockney, R. (1965), A fast direct solution of Poisson's equation using Fourier analysis. *Journal of the ACM*, **12**(1), 95–113.

Hockney, R. and Jesshope, C.R. (1981), *Parallel Computers*. Adam Hilger, Bristol.

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

Hunter, D.R. and Li, R. (2005), Variable selection using mm algorithm. *The Annals of Statistics*, **33**(4), 1617–1642.

Kim, S.J., Koh, K., Boyd, S., and Gorinevsky, D. (2009), ℓ_1 trend filtering. *SIAM Review*, **51**(2), 339–360.

Kirk, D.B. and Hwu, W.W. (2010), *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Lange, K., Hunter, D.R., and Yang, I. (2000), Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, **9**(1), 1–20.

Lin, X., Pham, M., and Ruszczynski, A. (2011), Alternating linearization for structured regularization problems. Preprint. Available at : `http://arxiv.org/abs/1201306`

Liu, J., Yuan, L., and Ye, J. (2010), An efficient algorithm for a class of fused lasso problems. In *The ACM SIG Knowledge Discovery and Data Mining.* ACM, Washington, DC.

Nesterov, Y. (2003), *Introductory Lectures on Convex Optimization: A Basic course*. Kluwer Academic Publishers.

Nesterov, Y. (2005), Smooth minimization of non-smooth functions. *Mathematical Programming*, **103**, 127–152.

Nesterov, Y. (2007), Gradient methods for minimizing composite objective function. *CORE Discussion Paper*.

Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., and Phillips, J.C. (2008), GPU computing. *Proceedings of the IEEE*, **96**(5), 879–899.

Petry, S., Flexeder, C., and Tutz, G. (2011), Pairwise fused lasso. *Technical Reports, No.102*. University of Munich, Available at : `http://epub.ub.uni-muenchen.de/12164/1/petry_etal_TR102_2011.pdf`.

Patterson, D.A. and Hennessey, J.L. (1998), *Computer Organization and De-*

sign (2nd ed.): the hardware/software interface. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

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

She, Y. (2010), Sparse regression with exact clustering. *Electronic Journal of Statistics*, **4**, 1055–1096.

Stone, H.S. (1973), An efficient parallel algorithm for the solution of a tridiagonal linear system of equations. *Journal of the ACM*, **20**(1), 27–38.

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

Tibshirani, R.J. and Taylor, J. (2011), The solution path of the generalized lasso. *The Annals of Statistics*, **39**(3), 1335–1371.

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

Tseng, P. (2001), Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, **109**(3), 475–494.

Vaida, F. (2005), Parameter convergence for EM and MM algorithms. *Statistica Sinica*, **15**(3), 831–840.

Wu, C.F.J. (1983), On the convergence properties of the EM algorithm. *The Annals of Statistics*, **11**(1), 95–103.

Ye, G.B. and Xie, X. (2011), Split bregman method for large scale fused lasso. *Computational Statistics & Data Analysis*, **55**(4), 1552–1569.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, **68**(1), 49–67.

Zhang, Y., Cohen, J., and Owens, J.D. (2010), Fast tridiagonal solvers on the GPU. *SIGPLAN Not.*, **45**(5), 127–136.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, **67**(2), 301–320.

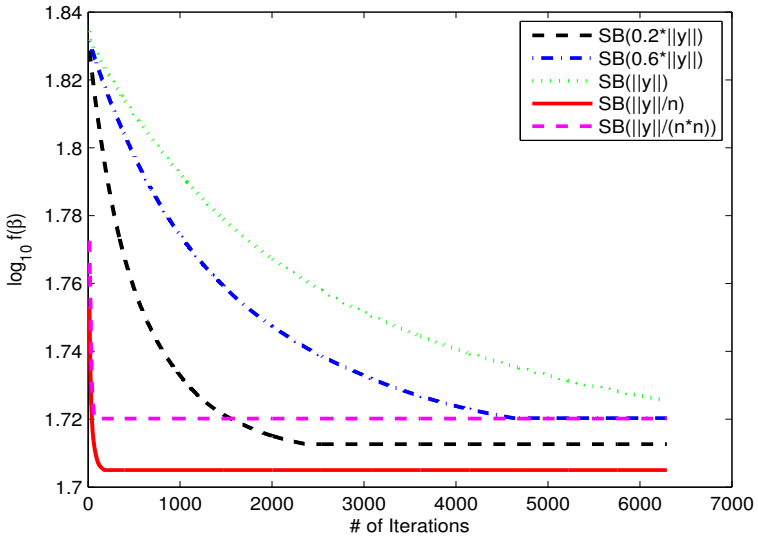
Appendix

A. Choice of μ in the SB algorithm

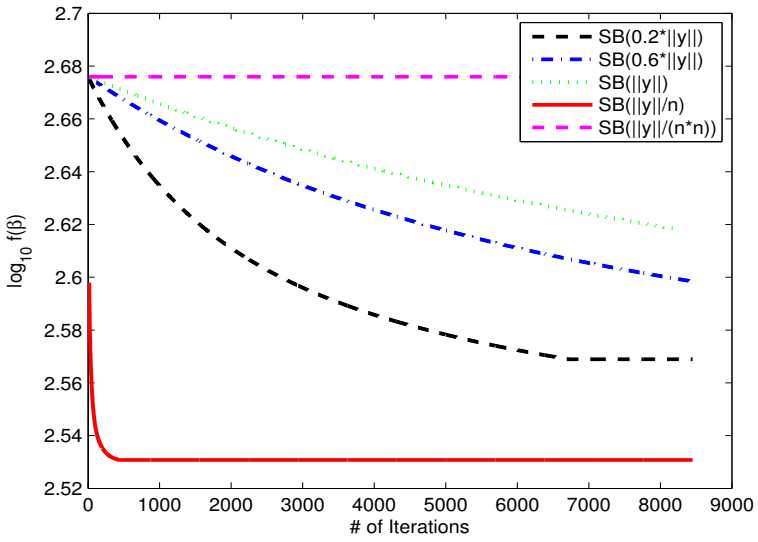
In the SB algorithm, the current solution $\widehat{\beta}^{(r+1)}$ is obtained from the solution of the linear system in (2.24),

$$\big(\mathbf{X}^T\mathbf{X} + \mu\mathbf{I} + \mu\mathbf{D}^T\mathbf{D}\big)\beta = \mathbf{c}^{(r)}(\mu),$$

where $\mathbf{c}^{(r)}(\mu) = \mathbf{X}^T\mathbf{y} + (\mu\mathbf{a}^{(r)} - \mathbf{u}^{(r)}) + \mathbf{D}^T(\mu\mathbf{b}^{(r)} - \mathbf{v}^{(r)})$. The additional parameter μ affects the convergence rate but there is no optimal rule for its choice in this problem (Ghadimi et al., 2012). Thus, Ye and Xie (2011) suggest a pretrial procedure to choose μ as one of $\{0.2, 0.4, 0.6, 0.8, 1\} \times \|\mathbf{y}\|_2$ by testing computation times for given (λ_1, λ_2). In addition to this recipe, we consider $\frac{1}{n}\|\mathbf{y}\|_2$ and $\frac{1}{n^2}\|\mathbf{y}\|_2$ as candidate values of μ. In our design of examples, we observe that the values other than $\frac{1}{n}\|\mathbf{y}\|_2$ makes the SB algorithm prematurely stop before reaching the optimal solution. The value $\frac{1}{n}\|\mathbf{y}\|_2$ also leads the best convergence rate from our pretrials in Figure A.1. Thus, we set $\mu = \frac{1}{n}\|\mathbf{y}\|_2$ in the numerical studies.



(a) $n = 200$, $p = 1000$



(b) $n = 1000$, $p = 10000$

Figure A.1: The number of iterations in SB algorithm as μ changes for $(\lambda_1, \lambda_2) = (0.1, 0.1)$

B. Tables of results for (C1)–(C4)

Table B.1: Summary of (**C1**) with computation times for $n = 1000$.

λ_1	λ_2	p	Computation time (sec.)				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	0.2858	0.0090	0.0093	0.0152	0.0089
		1000	2.6552	1.5070	0.3085	0.5799	2.1276
		10000	7.2544	66.9573	32.8320	102.5558	99.1599
		20000	13.3395	133.2365	77.4856	281.4834	209.6247
0.1	1	200	0.2207	0.0092	0.0067	0.0273	0.0656
		1000	2.2735	1.1956	0.1987	0.4140	10.0084
		10000	5.3980	51.1480	19.3404	58.0435	117.2644
		20000	9.8197	101.1593	49.4796	163.0641	238.4468
1	0.1	200	0.1469	0.0074	0.0069	0.0168	0.0150
		1000	1.6121	0.8684	0.2128	0.4359	9.2411
		10000	7.4584	75.5762	15.5698	48.1091	54.5518
		20000	14.9428	163.5872	44.3698	133.7532	152.5818
1	1	200	0.1477	0.0072	0.0068	0.0157	0.0873
		1000	1.2295	0.6092	0.1527	0.2986	10.1666
		10000	4.5901	44.8417	14.6557	32.6244	67.7480
		20000	8.4576	90.2439	35.3691	96.1971	199.6160

Table B.2: Summary of (**C1**) with the numbers of iterations for $n = 1000$.

λ_1	λ_2	p	Number of iterations				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	3.0	3.0	27.8	42.5	3.0
		1000	22.6	23.0	258.9	405.3	15.1
		10000	84.1	83.9	1067.8	1756.1	158.2
		20000	90.3	90.3	1287.3	2437.4	206.6
0.1	1	200	4.0	4.0	24.3	73.8	28.7
		1000	26.3	26.4	166.9	285.8	75.3
		10000	86.3	86.1	618.7	988.9	186.9
		20000	94.2	93.7	808.7	1411.3	234.9
1	0.1	200	3.2	3.2	26.1	43.0	5.8
		1000	23.2	23.0	166.5	266.7	69.7
		10000	82.9	82.9	506.3	822.1	87.7
		20000	92.8	92.8	737.6	1155.5	152.4
1	1	200	4.0	4.0	25.0	42.2	37.8
		1000	24.4	24.4	128.1	209.6	79.2
		10000	82.5	82.4	465.8	554.9	110.9
		20000	88.6	88.5	575.4	830.1	201.5

Table B.3: Summary of (**C1**) with the objective function values at $\widehat{\beta}$ for $n = 1000$.

λ_1	λ_2	p	$f(\widehat{\beta})$				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	409.15	409.15	409.20	409.15	409.15
		1000	39.39	39.39	39.48	39.42	39.41
		10000	12.13	12.13	13.32	12.81	12.54
		20000	11.67	11.67	13.25	12.63	12.26
0.1	1	200	428.52	428.52	428.65	428.53	428.63
		1000	118.08	118.08	118.14	118.10	118.77
		10000	36.20	36.20	38.90	37.63	43.40
		20000	33.48	33.48	36.20	36.34	42.03
1	0.1	200	474.32	474.32	474.38	474.32	474.42
		1000	151.18	151.18	151.28	151.19	151.79
		10000	86.51	86.51	90.32	86.62	98.49
		20000	84.12	84.12	87.14	84.27	98.36
1	1	200	493.49	493.49	493.55	493.49	493.69
		1000	211.97	211.97	212.48	211.99	213.04
		10000	120.30	120.30	123.01	121.28	137.94
		20000	115.92	115.92	120.71	117.81	136.49

Table B.4: Summary of (**C2**) with computation times for $n = 1000$. The gray cells denote that the EFLA fails to reach the optimal solution for one or two samples. We report the average computation times of the EFLA removed the failed cases.

λ_1	λ_2	p	Computation time (sec.)				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	0.2247	0.0108	0.0077	0.0170	0.0088
		1000	3.0984	1.4673	0.3732	0.7777	1.6792
		10000	4.1122	45.8289	91.7927	263.7253	266.5620
		20000	4.4272	52.7964	242.0749	821.5516	704.5284
0.1	1	200	0.2276	0.0115	0.0072	0.0168	0.0595
		1000	3.6282	1.5755	0.2211	0.5595	4.7977
		10000	10.3537	109.7741	47.8311	132.3191	107.4803
		20000	23.6943	271.3948	236.7397	503.1258	232.9924
1	0.1	200	0.1279	0.0086	0.0061	0.0171	0.0081
		1000	1.7817	0.9148	0.1901	0.5022	2.9606
		10000	10.8871	127.3424	51.5586	169.2509	142.5682
		20000	20.1332	244.0547	141.8118	526.4848	313.0687
1	1	200	0.1467	0.0084	0.0065	0.0176	0.0633
		1000	1.7161	0.8342	0.1476	0.4186	3.0164
		10000	7.7781	91.8040	38.7895	120.4112	64.9480
		20000	17.6899	212.5753	106.2446	339.2643	174.6900

Table B.5: Summary of (**C2**) with the numbers of iterations for $n = 1000$. The gray cells denote that the EFLA fails to reach the optimal solution for one or two samples. We report the average computation times of the EFLA removed the failed cases.

λ_1	λ_2	p	Number of iterations				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	3.0	3.0	25.0	46.8	3.0
		1000	22.2	21.8	311.8	508.8	19.9
		10000	36.3	35.5	2976.3	4527.5	466.7
		20000	27.2	27.6	4043.3	7108.9	776.4
0.1	1	200	4.2	4.2	25.4	41.9	26.8
		1000	28.1	28.2	180.5	359.1	57.7
		10000	53.9	53.9	1526.8	2272.4	188.4
		20000	63.9	63.7	3915.2	4347.2	256.3
1	0.1	200	3.1	3.1	21.8	44.6	2.7
		1000	20.3	20.3	160.7	325.7	34.9
		10000	91.6	91.5	1665.9	2910.1	249.8
		20000	88.7	89.7	2383.4	4556.9	344.7
1	1	200	4.2	4.2	23.4	44.6	28.8
		1000	22.5	22.5	122.6	269.4	36.0
		10000	92.1	92.2	1226.0	2077.8	114.0
		20000	102.3	102.3	1756.6	2929.0	192.6

Table B.6: Summary of (**C2**) with the objective function values at $\widehat{\beta}$ for $n = 1000$. The gray cells denote that the EFLA fails to reach the optimal solution for one or two samples. We report the average computation times of the EFLA removed the failed cases.

λ_1	λ_2	p	$f(\widehat{\beta})$				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	412.49	412.49	412.61	412.50	412.50
		1000	77.50	77.50	77.85	77.60	77.50
		10000	344.63	344.67	348.83	345.41	344.96
		20000	451.97	451.88	459.02	452.31	451.64
0.1	1	200	422.73	422.73	422.85	422.73	422.84
		1000	146.95	146.95	150.39	146.99	147.29
		10000	516.88	516.88	547.25	523.05	519.59
		20000	1010.81	1010.81	1015.63	1020.47	1023.09
1	0.1	200	506.02	506.02	506.33	506.02	506.13
		1000	564.19	564.19	567.06	564.68	564.56
		10000	1716.94	1716.94	1729.08	1717.67	1724.08
		20000	2146.89	2146.87	2169.95	2148.09	2159.64
1	1	200	516.09	516.09	516.31	516.09	516.32
		1000	620.66	620.66	628.12	620.69	621.47
		10000	3437.01	3437.00	3449.02	3437.55	3454.54
		20000	4489.60	4489.60	4519.21	4493.67	4509.41

Table B.7: Summary of (**C3**) with computation times for $n = 1000$. The gray cells denote that the EFLA fails to reach the optimal solution for one sample. We report the average computation times of the EFLA removed failed case.

λ_1	λ_2	p	Computation time (sec.)				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	0.2511	0.0171	0.0081	0.0170	0.0095
		1000	2.9360	1.5933	0.3793	0.8082	1.2007
		10000	3.7909	42.9165	90.8098	282.7641	286.2912
		20000	4.0806	48.1163	248.1919	871.9119	756.4445
0.1	1	200	0.3401	0.0133	0.0069	0.0174	0.0662
		1000	3.9890	1.8233	0.2031	0.5462	3.8157
		10000	15.0492	180.0133	54.7479	144.0007	108.6189
		20000	28.2846	343.3138	169.7143	505.0450	238.1494
1	0.1	200	0.0890	0.0058	0.0072	0.0178	0.0130
		1000	1.6862	0.8142	0.3392	0.8342	2.3927
		10000	10.7328	127.3884	50.5020	176.1017	127.5324
		20000	19.5732	239.8993	150.7861	520.9941	307.9786
1	1	200	0.1840	0.0085	0.0067	0.0169	0.0094
		1000	1.4819	0.7155	0.1548	0.4869	1.9491
		10000	8.1713	97.7493	38.0236	129.9449	89.1512
		20000	17.7725	219.5605	110.7899	365.3513	173.9217

Table B.8: Summary of (**C3**) with the numbers of iterations for $n = 1000$. The gray cells denote that the EFLA fails to reach the optimal solution for one sample. We report the average computation times of the EFLA removed failed case.

λ_1	λ_2	p	Number of iterations				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	3.0	3.0	26.7	45.2	3.0
		1000	21.3	21.9	327.4	511.3	13.2
		10000	34.7	34.0	3025.2	4789.1	498.3
		20000	26.0	25.8	4108.3	7553.7	824.2
0.1	1	200	4.2	4.2	25.3	44.9	27.4
		1000	27.4	27.3	170.8	352.3	43.5
		10000	91.8	91.9	1795.4	2446.6	189.3
		20000	96.8	97.0	2762.1	4376.6	259.7
1	0.1	200	3.4	3.4	26.5	46.3	5.0
		1000	23.8	23.8	287.1	555.2	24.8
		10000	88.6	88.6	1681.8	2985.3	222.3
		20000	85.4	85.8	2497.4	4511.6	335.6
1	1	200	4.4	4.4	24.9	44.3	3.4
		1000	19.7	19.6	124.3	308.7	23.0
		10000	95.4	95.5	1239.9	2206.9	155.5
		20000	101.9	101.9	1799.7	3156.2	189.4

Table B.9: Summary of (**C3**) with the objective function values at $\widehat{\beta}$ for $n = 1000$. The gray cells denote that the EFLA fails to reach the optimal solution for one sample. We report the average computation times of the EFLA removed failed case.

λ_1	λ_2	p	$f(\widehat{\beta})$				
			MMGPU	MM	EFLA	SPG	SB
0.1	0.1	200	423.50	423.50	423.57	423.51	423.50
		1000	122.55	122.55	122.82	122.64	122.57
		10000	368.19	368.23	371.84	368.75	368.73
		20000	486.41	486.46	493.75	486.07	486.14
0.1	1	200	432.11	432.11	432.23	432.11	432.21
		1000	197.44	197.44	201.83	197.48	197.96
		10000	995.36	995.36	1000.75	997.71	1002.65
		20000	1456.03	1456.03	1469.85	1460.41	1464.57
1	0.1	200	603.45	603.45	603.56	603.45	603.45
		1000	999.39	999.39	1000.21	999.38	999.41
		10000	1792.15	1792.15	1805.15	1792.77	1801.67
		20000	2290.35	2290.32	2313.64	2291.72	2306.86
1	1	200	612.06	612.06	612.17	612.06	612.39
		1000	1092.42	1092.43	1098.08	1092.45	1093.23
		10000	3670.35	3670.35	3682.73	3670.96	3681.24
		20000	4828.02	4828.02	4860.65	4831.53	48511

Table B.10: Summary of (**C4**) with computation times for $n = 1000$.

λ_1	λ_2	$q \times q$	Computation time (sec.)		
			MM	SPG	SB
0.1	0.1	16×16	0.0477	0.0252	0.0230
		32×32	1.9948	1.1403	1.9887
		64×64	45.1773	68.6246	164.9881
		128×128	50.2078	615.4028	979.8359
0.1	1	16×16	0.0549	0.0234	0.0971
		32×32	1.1902	0.5846	4.1854
		64×64	15.1176	23.0897	33.2731
		128×128	72.8389	208.2940	119.7563
1	0.1	16×16	0.0612	0.0232	0.0180
		32×32	1.4630	0.6096	2.3580
		64×64	48.5678	46.7797	92.2762
		128×128	241.6816	343.2788	657.3494
1	1	16×16	0.0476	0.0221	0.1006
		32×32	0.8718	0.4960	3.3169
		64×64	30.0718	24.6371	35.1351
		128×128	189.6904	237.8033	233.3495

Table B.11: Summary of (**C4**) with the numbers of iterations for $n = 1000$.

λ_1	λ_2	$q \times q$	Number of iterations		
			MM	SPG	SB
0.1	0.1	16×16	5.0	54.9	2.8
		32×32	21.3	603.6	16.5
		64×64	63.7	2745.9	456.0
		128×128	28.8	6414.6	1066.6
0.1	1	16×16	8.3	53.3	26.7
		32×32	23.0	294.8	37.9
		64×64	39.6	923.6	90.8
		128×128	49.8	2169.3	123.5
1	0.1	16×16	7.3	56.2	3.0
		32×32	20.6	326.3	21.5
		64×64	80.1	1876.4	254.2
		128×128	85.3	3579.7	714.9
1	1	16×16	9.5	51.5	27.3
		32×32	20.5	273.1	30.3
		64×64	63.5	978.6	95.9
		128×128	109.0	2476.8	248.1

Table B.12: Summary of (**C4**) with the objective function values at $\widehat{\beta}$ for $n = 1000$.

λ_1	λ_2	$q \times q$	$f(\widehat{\beta})$		
			MM	SPG	SB
0.1	0.1	16×16	423.47	423.48	423.48
		32×32	172.80	172.97	172.84
		64×64	489.57	491.57	489.75
		128×128	1110.11	1107.79	1111.67
0.1	1	16×16	609.83	609.83	609.96
		32×32	619.21	619.30	619.72
		64×64	1225.58	1226.96	1229.94
		128×128	3203.55	3219.16	3220.85
1	0.1	16×16	657.03	657.03	657.11
		32×32	1118.86	1120.38	1119.41
		64×64	2172.99	2173.41	2178.05
		128×128	3520.95	3526.15	3542.57
1	1	16×16	843.00	843.00	843.18
		32×32	1553.60	1553.64	1554.38
		64×64	4892.06	4895.37	4897.92
		128×128	11028.98	11030.06	11057.08

국 문 초 록

Fused lasso 벌점을 가진 sparse 회귀모형에 대한
MM 알고리즘과 GPU를 이용한 병렬화
MM algorithm for sparse regression model with
the fused lasso penalty and its parallelization using GPU

본 논문에서는 고차원의 fuse lasso 벌점을 지닌 회귀 모형에 대해 GPU를 이용하여 병렬화된 MM 알고리즘을 제안하였다. 본 논문에서 다루는 모형에 대한 MM 알고리즘의 수렴성이 보장됨을 보이고 다양한 형태의 설명 변수의 행렬과 회귀 계수의 구조에 대해서 유연성과 안정성을 가짐을 수치적으로 확인하였다. MM 알고리즘과 기존의 다른 방법들을 다양한 예제들로 비교하여 병렬화된 MM 알고리즘이 차원이 큰 표준의 fused lasso 벌점을 갖는 회귀 모형에서 더 빠르게 추정량을 제공함을 보였다.

주요어 : fused lasso 회귀모형, MM 알고리즘, 병렬계산, 그래픽 처리 장치.
학 번 : 2010-30083