Communications for Statistical Applications and Methods
2022, Vol. 29, No. 4, 441–451

https://doi.org/10.29220/CSAM.2022.29.4.441
Print ISSN 2287-7843 / Online ISSN 2383-4757

ADMM for least square problems with pairwise-difference penalties for coefficient grouping

Soohee Park[a], Seung Jun Shin[1,a]

[a]Department of Statistics, Korea University, Republic of Korea

Abstract

In the era of bigdata, scalability is a crucial issue in learning models. Among many others, the Alternating Direction of Multipliers (ADMM, Boyd *et al.*, 2011) algorithm has gained great popularity in solving large-scale problems efficiently. In this article, we propose applying the ADMM algorithm to solve the least square problem penalized by the pairwise-difference penalty, frequently used to identify group structures among coefficients. ADMM algorithm enables us to solve the high-dimensional problem efficiently in a unified fashion and thus allows us to employ several different types of penalty functions such as LASSO, Elastic Net, SCAD, and MCP for the penalized problem. Additionally, the ADMM algorithm naturally extends the algorithm to distributed computation and real-time updates, both desirable when dealing with large amounts of data.

Keywords: alternating direction of multipliers, grouping coefficients, real-time update, high-dimensional data

1. Introduction

Grouping coefficients in regression problems is an important but challenging task in many contemporary applications. In genomics, for example, grouping genes (i.e., variables) that play a similar role is often of interest in order to have a more precise understanding of how genes affect the human body.

The coefficients grouping can be regarded as the generalization of sparsity (Ke *et al.*, 2015). There have been many studies for grouping coefficients through regularization. A canonical example is fused LASSO (Tibshirani *et al.*, 2005) that penalizes the difference between adjacent coefficients. The fused LASSO requires the coefficients to be pre-ordered, but we may face many cases where we do not have prior information about ordering. Bondell and Reich (2008) applied pairwise L_∞ regulation, and Shen and Huang (2010) proposed regularization to the difference of the pairwise coefficients for data-driven grouping pursuit, both of which did not require the pre-ordering of the regression coefficients. Ke *et al.* (2015) proposed the CARDS algorithm based on a hybrid version of fused LASSO penalty and total variation penalty (Harchaoui and Leévy-Leduc, 2010) to detect homogeneous groups of regression coefficients.

This article focuses on the pairwise-difference penalty proposed by Shen and Huang (2010). For a given set of data $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$ and $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{R}^{n\times p}$, our goal is to solve

This work is partially funded by the National Research Foundation of Korea (NRF) grants 2018R1D1A1B07043034 and 2019R1A4A1028134.

[1] Corresponding author: Department of Statistics, Korea University, 145 Anam-Ro, Sungbuk-Gu, Seoul 02841, Korea. E-mail: sjshin@korea.ac.kr

Published 31 July 2022 / journal homepage: http://csam.or.kr

the following problem with by the pairwise-difference penalty in order to to identify the grouping structure of regression coefficients.

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_{i<j} p_\lambda\left(\left|\beta_i - \beta_j\right|\right), \tag{1.1}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \boldsymbol{\beta}_p)^T$ and $p_\lambda(\cdot)$ denotes a sparsity-pursing penalty function with a tuning parameter λ. There are various choices for the penalty function. We consider four popular penalties including LASSO (Tibshirani, 1996), elastic net (Zou and Hastie, 2005), smoothly clipped absolute deviance (SCAD) (Fan and Li, 2001), and minimax concave penalty (MCP) (Zhang, 2010).

We should point out that (1.1) does not require preordering because it penalizes all pairwise distances between coefficients. However, due to the larger number of penalty terms, its computation may not always be straightforward. To tackle this, we propose to apply the Alternating Direction of Multipliers (ADMM) (Boyd *et al.*, 2011) algorithm. The ADMM is an efficient algorithm, which can be summarized as the intersection of optimizing the primal problem and the dual problem based on the Lagrangian theory. In addition, the ADMM naturally allows us to extend to the distributed (or parallel) computation desirable for handling large-scale data. The ADMM algorithm has been applied in a wide variety of applications in penalized regressions. Wahlberg *et al.* (2012) proposed to solve the fused lasso problem with the ADMM. Zhu (2017) applied the augmented ADMM to solve the generalized lasso problems. Jeon *et al.* (2017) employed the ADMM to solve the grouping-coefficient-problem for the generalized linear model with non-convex penalty functions.

The paper is organized as follows. Section 2 provides a brief overview of the ADMM algorithm. In Section 3, we describe how the ADMM is applied to solve (1.1) with various penalty functions. The extension to distributed computation and real-time update for (1.1) for large data sets is described in Section 4. Section 5 illustrates the proposed algorithm to the simulated data and conclusions follow in Section 6.

2. Alternating direction method of multipliers

Consider the following linearly constrained optimization problem with a variable $\mathbf{x} \in \mathbb{R}^p$, and two convex functions f, g:

$$\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}). \tag{2.1}$$

Introducing an auxiliary variable $\mathbf{z} \in \mathbb{R}^p$, (2.1) can be expressed as follows:

$$p^* = \min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \quad \text{subject to } \mathbf{x} = \mathbf{z}. \tag{2.2}$$

The primal problem and its Lagrangian function of (2.2) is

$$L(\mathbf{x}, \mathbf{z}, \mathbf{a}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{a}^T(\mathbf{x} - \mathbf{z}), \tag{2.3}$$

where $\mathbf{a} \in \mathbb{R}^p$ denote Lagrangian multipliers. By Lagrangian theory (Boyd *et al.*, 2004), the dual problem of (2.1) is

$$d^* = \max_{\mathbf{a}} d(\mathbf{a}),$$

where $d(\mathbf{a}) = \inf_{\mathbf{x},\mathbf{z}} L(\mathbf{x}, \mathbf{z}, \mathbf{a})$ is referred to as the dual function of (2.2). It can be readily known that $d^* \leq p^*$. Under certain conditions, we have $d^* = p^*$ (strong duality) under which one can use the dual

problem to solve the primal problem (Boyd *et al.*, 2004). The dual problem is often much simpler than the primal problem since the dual function is always convex even if the primal function is not. Support vector machine is a well-known example.

One natural way to solve the dual problem of (2.2) is the gradient ascent algorithm that iteratively updates the primal function and the dual parameter based on the gradient. Yet, it is known to be unstable (Boyd *et al.*, 2011). As an alternative, the augmented Lagrangian introduces an additional L_2 regularization term to stabilize the optimization as follows.

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{a}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{a}^T(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|_2^2, \tag{2.4}$$

where $\rho > 0$ is a tuning parameter that controls the step size of the update based on the gradient. In addition, we equivalently rewrite (2.4) as the following rescaled form.

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z} + \mathbf{u}\|_2^2 - \frac{\rho}{2}\|\mathbf{u}\|_2^2, \tag{2.5}$$

where $\mathbf{u} = \mathbf{a}/\rho$ is the scaled dual variable.

Note that (2.5) is additively separable. The ADMM algorithm iteratively updates $\mathbf{x}, \mathbf{z}, \mathbf{u}$ in an alternating way, which explains its name. Namely, the ADMM algorithm consists of the following three steps of iterations:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \; f(\mathbf{x}) + \left(\frac{\rho}{2}\right)\left\|\mathbf{x} - \mathbf{z}^t + \mathbf{u}^t\right\|_2^2, \tag{2.6}$$

$$\mathbf{z}^{t+1} = \arg\min_{\mathbf{z}} \; g(\mathbf{z}) + \left(\frac{\rho}{2}\right)\left\|\mathbf{x}^{t+1} - \mathbf{z} + \mathbf{u}^t\right\|_2^2, \tag{2.7}$$

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \rho\left(\mathbf{x}^{t+1} - \mathbf{z}^{t+1}\right). \tag{2.8}$$

We remark that both $\mathbf{x}$ and $\mathbf{z}$ can be efficiently updated using the proximal operator (Parikh and Boyd, 2014). The proximal operator of f (scaled by η) denoted with $\text{prox}_{\eta f}$ is defined by

$$\text{prox}_{\eta f}(\mathbf{x}) = \arg\min_{\mathbf{v}}\left[f(\mathbf{v}) + \frac{1}{2\eta}\|\mathbf{v} - \mathbf{x}\|_2^2\right],$$

where $\eta > 0$ is a scale parameter. That is, $\text{prox}_f(\mathbf{x})$ can be interpreted as a minimizer of f near $\mathbf{x}$. Employing the proximal operators we can respectively rewrite (2.6) and (2.7) as

$$\mathbf{x}^{t+1} = \text{prox}_{\frac{f}{\rho}}\left(\mathbf{z}^t - \mathbf{u}^t\right) \quad \text{and} \quad \mathbf{z}^{t+1} = \text{prox}_{\frac{g}{\rho}}\left(\mathbf{x}^{t+1} - \mathbf{u}^t\right).$$

This explains why ADMM is efficient when the complexity of prox_f and prox_g is much smaller than that of prox_{f+g}.

Finally, we remark the convergence conditions for the ADMM algorithm. Boyd *et al.* (2011) showed that the ADMM algorithm converges under the following two assumptions:

A1. The functions $f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, convex.

A2. The unaugmented Lagrangian with $\rho = 0$ has a saddle point

The latter assumption A2 implies that the strong duality (Boyd *et al.*, 2011).

During the iterations, the ADMM yields two kinds of residuals: the primal residual and the dual residual at iteration $t+1$, which are respectively given by

$$\mathbf{r}^{t+1} = \mathbf{x}^{t+1} - \mathbf{z}^{t+1} \quad \text{and} \quad \mathbf{s}^{t+1} = -\rho\left(\mathbf{z}^{t+1} - \mathbf{z}^{t}\right).$$

Both residuals converge to zero as the iterations proceed and thus the ADMM algorithm is terminated if $\|\mathbf{r}^{t+1}\|_2^2$ and $\|\mathbf{s}^{t+1}\|_2^2$ are sufficiently small, i.e., $\|\mathbf{r}^{t+1}\|_2^2 < \epsilon_1$ and $\|\mathbf{s}^{t+1}\| < \epsilon_2$ for sufficiently small values of ϵ_1 and ϵ_2. In this article, we set

$$\epsilon_1 = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\left\{\left\|\mathbf{x}^k\right\|_2, \left\|\mathbf{z}^k\right\|_2\right\} \quad \text{and} \quad \epsilon_2 = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\left\|\mathbf{u}^k\right\|_2 \tag{2.9}$$

with $\epsilon^{\text{abs}} = 10^{-4}$, $\epsilon^{\text{rel}} = 10^{-2}$ as suggested by Boyd *et al.*, 2011.

3. Application of ADMM to the least square problem with the pairwise-difference penalty

In what follows, we apply the ADMM algorithm to solve (1.1) with four popular penalty functions including LASSO, elastic net, SCAD, and MCP. Toward this, (1.1) can be equivalently rewritten as follows

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_{i<j} p_\lambda\left([|\mathbf{D}\boldsymbol{\beta}|]_i\right), \tag{3.1}$$

where $\mathbf{D} \in \mathbb{R}^{p(p-1)/2 \times p}$ is a transformation matrix for the pairwise-difference such that $\mathbf{D}\boldsymbol{\beta} = (\beta_1 - \beta_2,\ \beta_1 - \beta_3,\ \ldots,\ \beta_{p-1} - \beta_p)^T \in \mathbb{R}^{p(p-1)/2}$, and $[\mathbf{v}]_i$ is the ith element of a vector $\mathbf{v}$. To apply ADMM algorithm, we introduce the auxiliary variable $\mathbf{z} \in \mathbb{R}^{p(p-1)/2}$ to write (3.1) as follows:

$$\min_{\boldsymbol{\beta},\mathbf{z}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_{i=1}^{p(p-1)/2} p_\lambda\left([|\mathbf{z}|]_i\right), \quad \text{subject to } \mathbf{D}\boldsymbol{\beta} = \mathbf{z}. \tag{3.2}$$

The scaled version of augmented Lagrangian of (3.2) is given by

$$\min_{\boldsymbol{\beta},\mathbf{z}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_{i=1}^{p(p-1)/2} p_\lambda\left(|\mathbf{z}_i|\right) + \frac{\rho}{2}\|\mathbf{D}\boldsymbol{\beta} - \mathbf{z} + \mathbf{u}\|_2^2 - \frac{\rho}{2}\|\mathbf{u}\|_2^2.$$

The corresponding ADMM updating equations for $\boldsymbol{\beta}, \mathbf{z}$, and $\mathbf{u}$ are

$$\boldsymbol{\beta}^{t+1} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \frac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\beta} - \mathbf{z}^t + \mathbf{u}^t\right\|_2^2, \tag{3.3}$$

$$\mathbf{z}^{t+1} = \arg\min_{\mathbf{z}} \sum_{i=1}^{p(p-1)/2} p_\lambda\left([|\mathbf{z}|]_i\right) + \frac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\beta}^{t+1} - \mathbf{z} + \mathbf{u}^t\right\|_2^2, \tag{3.4}$$

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \mathbf{D}\boldsymbol{\beta}^{t+1} - \mathbf{z}^{t+1}. \tag{3.5}$$

Since (3.5) is straightforward, it is crucial to solve (3.4) and (3.5). First, we note (3.3) for $\boldsymbol{\beta}$ is a ridge-regression and has a closed-form solution as follows.

$$\begin{aligned}\boldsymbol{\beta}^{t+1} &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \frac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\beta} - \mathbf{z}^t + \mathbf{u}^t\right\|_2^2 \\ &= \left(\mathbf{X}^T\mathbf{X} + \rho\mathbf{D}^T\mathbf{D}\right)^{-1}\left(\mathbf{X}^T\mathbf{y} + \rho\mathbf{D}^T\left(\mathbf{z}^t - \mathbf{u}^t\right)\right).\end{aligned} \tag{3.6}$$

Next, (3.4) for $\mathbf{z}$ involves the penalty function p_λ, and its solution differs according to the choice of penalty functions.

LASSO. For the LASSO penalty, i.e., $p_\lambda(\theta) = \lambda\theta$ for $\theta \geq 0$, (3.4) becomes

$$\begin{aligned}\mathbf{z}^{t+1} &= \arg\min_{\mathbf{z}} \ \lambda\|\mathbf{z}\|_1 + \frac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\beta}^{t+1} - \mathbf{z} + \mathbf{u}^t\right\|_2^2 \\ &= S_{\frac{\lambda}{\rho}}\left(\mathbf{D}\boldsymbol{\beta}^{t+1} + \mathbf{u}^t\right), \end{aligned} \tag{3.7}$$

where $S_{\lambda/\rho}(\mathbf{a}) = \{S_{\lambda/\rho}(a_i) = \text{sign}(a_i)[a_i - \lambda/\rho]_+, i = 1, 2, \ldots, \dim(\mathbf{a})\}$ denotes an element-wise soft-thresholding operator.

Elastic Net. The elastic net penalty is defined as $p_\lambda(\theta) = \lambda\{\alpha\theta + (1-\alpha)/2\theta^2\}$ for a given $\alpha \in [0, 1]$. Now, (3.4) with the elastic net is

$$\begin{aligned}\mathbf{z}^{t+1} &= \arg\min_{\mathbf{z}} \lambda_1\|\mathbf{z}\|_1 + \lambda_2\|\mathbf{z}\|_2^2 + \frac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\beta}^{t+1} - \mathbf{z} + \mathbf{u}^t\right\|_2^2 \\ &= S_{\frac{\lambda_1}{\rho}}\left(\frac{1}{1 - 2\lambda_2/\rho}\left(\mathbf{D}\boldsymbol{\beta}^{t+1} + \mathbf{u}^t\right)\right), \end{aligned} \tag{3.8}$$

where $\lambda_1 = \alpha\lambda$ and $\lambda_2 = \lambda(1-\alpha)/2$

SCAD. The SCAD penalty is defined through its derivative as

$$p'_\lambda(\theta) = \lambda\left\{I(\beta \leq \lambda) + \frac{[a\lambda - \theta]_+}{(a-1)\lambda} I(\theta > \lambda)\right\}, \quad \theta \geq 0 \tag{3.9}$$

for a given $a > 2$. A popular choice of a is 3.7 (Fan and Li, 2001). The SCAD penalty is non-convex and (3.4) with the SCAD penalty is not as simple as the previous two convex penalties. There are several ways to solve the SCAD-penalized least square problem. In this article, we propose to apply the local linear approximation (LLA) (Zou and Li, 2008). The LLA algorithm employs the first order Taylor expansion to approximate the SCAD penalty as follows.

$$p_\lambda(\theta) \approx p_\lambda(\theta_0) + p'_\lambda(\theta_0)(\theta - \theta_0), \quad \text{for } \theta \approx \theta_0. \tag{3.10}$$

The LLA algorithm using (3.10) is known to be best convex approximation of the SCAD penalty function, and one-step update is enough to have a sensible estimator (Zou and Li, 2008). Employing (3.10) to approximate the SCAD penalty $p_\lambda(|z_i|)$ at around $|z_i| \approx |z_i^t|$, (3.4) with SCAD penalty can be updated as follows.

$$\begin{aligned}\mathbf{z}^{t+1} &= \arg\min_{\mathbf{z}} \sum_{i=1}^{p(p-1)/2} p'_\lambda(|z_i^t|)|z_i| + \frac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\beta}^{t+1} - \mathbf{z} + \mathbf{u}^t\right\|_2^2 \\ &= S_{\frac{\mathbf{w}^t}{\rho}}\left(\mathbf{D}\boldsymbol{\beta}^{t+1} + \mathbf{u}^t\right), \end{aligned} \tag{3.11}$$

where $[S_{\mathbf{w}^t/\rho}(\mathbf{D}\boldsymbol{\beta}^{t+1} + \mathbf{u}^t)]_i = S_{w_i^t/\rho}([\mathbf{D}\boldsymbol{\beta}^{t+1}]_i + u_i^t)$ with $[\mathbf{w}^t]_i = w_i^t = p'_\lambda(|z_i^t|)$ for $i = 1, 2, \ldots, p(p-1)/2$.

MCP. MCP is another popular non-convex penalty and defined through the derivative as follows:

$$p'_\lambda(\theta) = \left[\lambda - \frac{\theta}{a}\right] I(0 \le \theta \le a\lambda), \quad \theta \ge 0 \tag{3.12}$$

LLA algorithm for (3.4) with MCP yields an identical solution (3.11), but with a differently defined $w_i^t = p'_\lambda(|z_i^t|)$ from (3.12).

We remark that (3.4) for updating $\mathbf{z}$ can be expressed as a proximal operator of the penalty function P, where $P(\mathbf{z}) = \sum_{i=1}^{p(p-1)/2} P([|\mathbf{z}|_i])$.

$$\mathbf{z}^{t+1} = \text{prox}_{\frac{P}{\rho}}\left(\mathbf{D}\boldsymbol{\beta}^{t+1} + \mathbf{u}^t\right).$$

Finally, Algorithm 1 provides a summary of the ADMMA algorithm to solve (1.1) with four popular penalty functions.

Algorithm 1 ADMM for solving (1.1)

initialize
  $\boldsymbol{\beta}^0 \leftarrow \mathbf{0}_p,\ \mathbf{z}^0 \leftarrow \mathbf{0}_{p(p-1)/2},\ \mathbf{u}^0 \leftarrow \mathbf{0}_{p(p-1)/2}$
repeat
  Update $\boldsymbol{\beta}^{t+1}$ from (3.6)
  Update $\mathbf{z}^{t+1}$ from
    – LASSO: (3.7)
    – Elastic Net: (3.8)
    – SCAD: (3.11) with w_i^t based on (3.9).
    – MCP: (3.11) with w_i^t based on (3.12).
  Update $\mathbf{u}^{t+1}$ from (3.5)
  $\mathbf{r}^{t+1} = \mathbf{D}\boldsymbol{\beta}^{t+1} - \mathbf{z}^{t+1},\ \mathbf{s}^{t+1} = -\rho\mathbf{D}^T(\mathbf{z}^{t+1} - \mathbf{z}^t)$
until $\|\mathbf{r}^{t+1}\|_2^2 \le \epsilon_1, \|\mathbf{s}^{t+1}\|_2^2 \le \epsilon_2$ for sufficiently small ϵ_1 and ϵ_2 given, as suggested in (2.9).

3.1. Grouping Coefficients

It is well-known that the ADMM algorithm does not produce an exact zero for the penalized estimator (Andrade *et al.*, 2021). That is, $\hat{\beta}_i - \hat{\beta}_j$ is not exactly zero for all $i < j$ in our setting, where $\hat{\beta}_i$ and $\hat{\beta}_j$ denote the solution obtained by the proposed ADMM algorithm. However, the estimated auxiliary variable $\hat{\mathbf{z}}$ is sparse, and therefore one can perform a coefficient grouping based on the sparsity structure of $\hat{\mathbf{z}}$ (Chi and Lange, 2015).

Figure 1 illustrates the grouping results based on the sparsity of $\hat{\mathbf{z}}$, with the LASSO penalty as λ varies, for the simulated data used in Section 5.1 (with $n = 1000$ and $r = 1$). The color indicates the true class. As the λ increases, the effect of the grouping penalty increases, and the number of resulting groups decreases. Note that the grouping result looks promising for a properly selected λ (= 2 in this example).

4. Extensions to large-scale data

This section describes two additional advantages of the use of the ADMM algorithm: distributed computation and real-time update, both of which are attractive features in the era of bigdata.

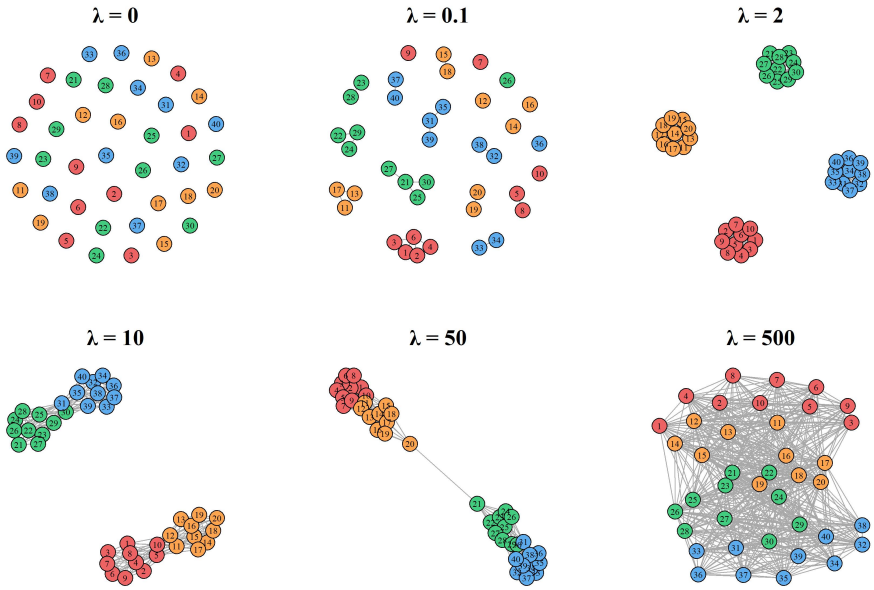


Figure 1: *Illustration of Coefficient-Grouping via* (3.1) *with the LASSO penalty for different values of* λ.

4.1. Consensus ADMM for Distributed Computing

For the distributed computation, we consider the *global consensus* version of (3.1). Suppose the data is divided into several, say K groups, i.e., $\mathbf{y} = (\mathbf{y}_1 : \mathbf{y}_2 : \cdots : \mathbf{y}_K)$ and $\mathbf{X} = (\mathbf{X}_1 : \mathbf{X}_2 : \cdots : \mathbf{X}_K)$. Then the global consensus problem of (3.1) is

$$\begin{aligned} \min_{\boldsymbol{\beta}_k, \mathbf{z}} \frac{1}{2} \sum_{k=1}^{K} \left\| \mathbf{y}_k - \mathbf{X}_k \boldsymbol{\beta}_k \right\|_2^2 + \sum_{i=1}^{p(p-1)/2} p_\lambda \left([|\mathbf{z}|]_i \right) \\ \text{subject to} \quad \mathbf{D}\boldsymbol{\beta}_k - \mathbf{z} = 0 \quad k = 1, \ldots, K \end{aligned} \tag{4.1}$$

where $\boldsymbol{\beta}_k \in \mathbb{R}^p$ is the local parameters for $(\mathbf{y}_k, \mathbf{X}_k), k = 1, 2, \ldots, K$ and $\mathbf{z}$ is a common global variable.

The ADMM algorithm is a natural choice to solve the global consensus problem (4.1). We call it consensus ADMM algorithm. The consensus ADMM for (4.1) consists of the following updating equations.

$$\boldsymbol{\beta}_k^{t+1} = \arg\min_{\boldsymbol{\beta}_k} \left\| \mathbf{y}_k - \mathbf{X}_k \boldsymbol{\beta}_k \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{D}\boldsymbol{\beta}_k - \mathbf{z}^t + \mathbf{u}_k^t \right\|_2^2, \quad k = 1, \ldots, K, \tag{4.2}$$

$$\mathbf{z}^{t+1} = \arg\min_{\mathbf{z}} \sum_{i=1}^{p(p-1)/2} p_\lambda \left([|\mathbf{z}|]_i \right) + \frac{K\rho}{2} \left\| \mathbf{D}\bar{\boldsymbol{\beta}}^{t+1} - \mathbf{z} + \bar{\mathbf{u}}^t \right\|_2^2, \tag{4.3}$$

$$\mathbf{u}_k^{t+1} = \mathbf{u}_k^t + \mathbf{D}\boldsymbol{\beta}_k^{t+1} - \mathbf{z}^{t+1}, \quad k = 1, \ldots, K, \tag{4.4}$$

where $\mathbf{u}_k$ denotes the scaled dual variable corresponding to $(\mathbf{y}_k, \mathbf{X}_y), k = 1, 2, \ldots, K$, and $\bar{\mathbf{u}} = \sum_{k=1}^{K} \mathbf{u}_k / K$ and $\bar{\boldsymbol{\beta}} = \sum_{k=1}^{K} \boldsymbol{\beta}_k / K$.

The consensus ADMM for (4.1) is a canonical example of the optimization problems with distributed objective and constraint terms across multiple processors or computers. Each processor only

has to deal with its own constraint optimization problem which can be separately updated at each iteration. However, the quadratic regularization term $\|\mathbf{D}\boldsymbol{\beta}_k - \mathbf{z} + \mathbf{u}_k\|_2^2$ forces the individual parameters $\boldsymbol{\beta}_k$ converging to a common value $\mathbf{z}$, which is the solution of the global problem.

The primal and dual residuals in the consensus ADMM are respectively given by

$$\mathbf{r}^{t+1} = \left(\mathbf{D}\boldsymbol{\beta}_1^{t+1} - \mathbf{z}^{t+1}, \ldots, \mathbf{D}\boldsymbol{\beta}_K^{t+1} - \mathbf{z}^{t+1}\right) \quad \text{and} \quad \mathbf{s}^{t+1} = -\rho\left(\mathbf{z}^{t+1} - \mathbf{z}^{t}, \ldots, \mathbf{z}^{t+1} - \mathbf{z}^{t}\right),$$

and the algorithm is terminated if both $\|\mathbf{r}^t\|_2^2$ and $\|\mathbf{s}^t\|_2^2$ are sufficiently small.

4.2. Real-time update for streamed data

Now, we assume stream data where we continuously collect data. For handling such stream data, it is desired to update the solution without using all data previously accumulated. This is called real-time update which becomes crucial technique to handle bigdata often collected in a continuous manner.

Suppose we have a set of old data $\mathbb{D}_0 = (\mathbf{y}_0, \mathbf{X}_0)$. Let $\hat{\boldsymbol{\beta}}_0, \mathbf{z}_0, \mathbf{u}_0$ denote the solution of (3.2) for the old data denoted by $\mathbb{D}_0$. Now, we have a new set of data denoted by $\mathbb{D}_\mathtt{N} = (\mathbf{y}_\mathtt{N}, \mathbf{X}_\mathtt{N})$. The goal of real-time update is to compute the solution of (3.1) for the whole data, $\mathbb{R}_\mathtt{W} = \mathbb{D}_0 \cup \mathbb{D}_\mathtt{N}$ without saving $\mathbb{D}_0$ whose sizes increases as more data accumulated.

We remark that ADMM algorithm naturally yields the real-time update for (3.1). This is because the ADMM algorithm updates loss and penalty terms separately. Note that we only require data for updating $\boldsymbol{\beta}$, (3.3) which can be expressed as

$$\begin{aligned}\boldsymbol{\beta}^{t+1} &= \left(\mathbf{X}_\mathtt{W}^T\mathbf{X}_\mathtt{W} + \rho\mathbf{D}^T\mathbf{D}\right)^{-1}\left(\mathbf{X}_\mathtt{W}{}^T\mathbf{y}_\mathtt{W} + \rho\mathbf{D}^T\left(\mathbf{z}^t - \mathbf{u}^t\right)\right)\\ &= \left(\mathbf{X}_0^T\mathbf{X}_0 + \mathbf{X}_\mathtt{N}^T\mathbf{X}_\mathtt{N} + \rho\mathbf{D}^T\mathbf{D}\right)^{-1}\left(\mathbf{X}_0^T\mathbf{y}_0 + \mathbf{X}_\mathtt{N}^T\mathbf{y}_\mathtt{N} + \rho\mathbf{D}^T\left(\mathbf{z}^t - \mathbf{u}^t\right)\right).\end{aligned}$$

We thus need to store $\mathbf{X}_0^T\mathbf{X}_0$ and $\mathbf{X}_0^T\mathbf{y}_0$ to update $\boldsymbol{\beta}^{t+1}$ for $\mathbb{D}_\mathtt{W}$. We remark that the size of both $\mathbf{X}_0^T\mathbf{X}_0$ and $\mathbf{X}_0^T\mathbf{y}_0$ depends only on p, not the sample size of $\mathbb{D}_0$. That is, the proposed algorithm is scalable.

5. Simulation

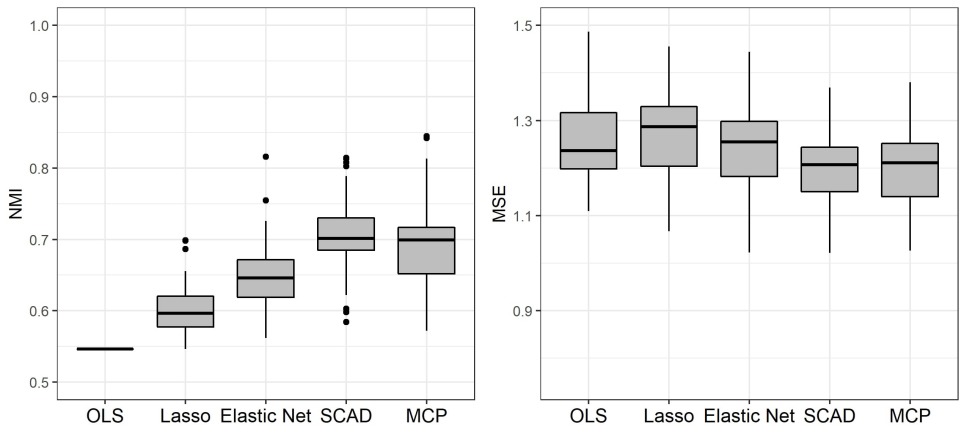
In this section, we conduct a simulation study to investigate the performance of the proposed method.

5.1. Estimation Performance

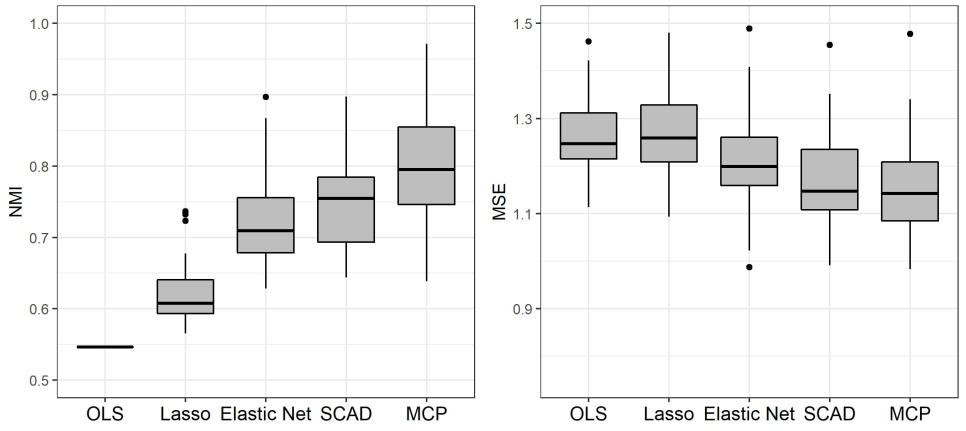
We first compare the performance of the different penalty function of the PD penalized problems. First, we generate data from the linear regression model

$$\mathbf{y}_i = \mathbf{x}_i^T\boldsymbol{\beta} + \epsilon_i, \quad i = 1, \ldots, n,$$

where $\mathbf{x}_i$ are generated from the mean-zero p-variate normal distribution with a first-order autoregressive structure, i.e. $\mathrm{Corr}(x_{ij}, x_{ik}) = 0.5^{|j-k|}$ $(1 \le j, k \le p)$, and ϵ_i are from the standard normal distribution. We consider $n = 200, 500, 1000$ with $p = 40$. The true regression coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{40})$ consist of four groups according to their size, each of size 10. We set (group 1) $\beta_j = -2r$ for $j = 1, 2, \ldots, 10$; (group 2) $\beta_j = -r$ for $j = 11, 12, \ldots, 20$; $\beta_j = r$ for (group 3) $j = 21, 21, \ldots, 30$; and (group 4) $\beta_j = 2r$ for $j = 31, 32, \ldots, 40$ where r controls the signal size. We set $r = 0.5, 1$ for weak and strong signal cases, respectively. We select tuning parameter λ that maximizes Bayesian information criterion (BIC).



(a) $r = 0.5$



(b) $r = 1$

Figure 2: *NMI and MSE values for selected models in experiments where n = 200 under independent 50 simulations.*

We measure the estimation performance of $\hat{\boldsymbol{\beta}}$ using mean squared error (MSE). For the grouping performance, we use the normalized mutual information (NMI) (Ke *et al.*, 2015) that measures the distance between the estimated and true group structure (i.e., corresponding group labels). NMI is a standardized measure taking a value between 0 and 1, and NMI = 1 indicates that the two grouping results are identical. See Section 5 of Ke *et al.* (2015) for the details about NMI.

Figure 2 shows the comparison results when $n = 200$. The upper panels show the case when $r = 0.5$ and the lower panels show the case when $r = 1$. The unpenalized OLS estimator does not perform coefficient grouping while rest estimators with the pairwise difference penalty can identify the true cluster of coefficients. One can observe that the two non-convex penalties, SCAD and MCP, outperform all others in terms of coefficient grouping measured by NMI. The elastic net penalty performs better than the LASSO in this simulation, which is not surprising since the predictors are correlated. The unpenalized OLS is not bad in terms of estimation error, but the two non-convex

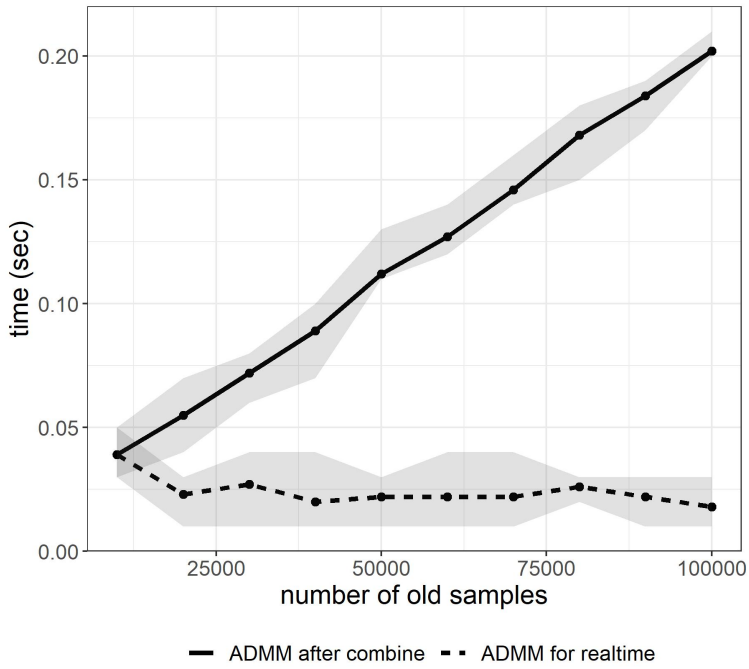


Figure 3: *Comparison of computing times between the original versus real-time ADMM algorithm. The range of computing times over ten repetitions is represented by shaded areas.*

penalties are still winners. Between the two, MCP seems slightly better than SCAD. The results for larger sample sizes $n = 500$ and 1000 are nearly identical and hence omitted to avoid redundancy.

5.2. Real-time update

In what follows, we illustrate the computational efficiency of the real-time update. Toward this, the data are generated in the same way as the previous simulation. To mimic the streamed-data scenario, we assume that the data arrives in batches. We generate 100 batches with 10000 observations in each batch. We then compare the computation times for the real-time ADMM introduced in Section 4.2 to the original ADMM algorithm applied to the whole data, $\mathbb{D}_{\mathtt{W}}$. Figure 3 depicts the empirical computing time averaged over ten repetitions for the two methods. As more data arrive, the original algorithm takes longer to estimate parameters while the proposed real-time algorithm exhibits stable computing times regardless of the total data size. What matters in the real-time algorithm is the batch size, which is constant in this streamed-data setting. In conclusion, the computational efficiency of the real-time ADMM becomes substantial as more data is accumulated.

6. Conclusion

To solve penalized regression problems with the pairwise-difference penalty for coefficient grouping, we propose using the ADMM algorithm. The proposed ADMM algorithm not only makes optimization much easier by updating parameters in an alternative way, but it is also general enough to handle a wide range of penalty functions. Furthermore, it naturally leads to the extension of distributed computing and real-time updates, both desirable for handling large amounts of data.

Although we focus on the conventional regression problem with the squared loss problem in this article, we can extend it to other popular problems such as generalized linear model and quantile regression. The ADMM algorithm is still powerful as long as updating the regression coefficient without penalty is simple.

References

Andrade, D., Fukumizu, K., and Okajima, Y. (2021) Convex covariate clustering for classification, *Pattern Recognition Letters* **151**, 193–199.

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

Boyd, S., Boyd, S.P. and Vandenberghe, L. (2004) *Convex optimization*, Cambridge university press.

Boyd, S., Parikh, N. and Chu, E. (2011) *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc.

Chi, E.C. and Lange, K. (2015) Splitting methods for convex clustering, *Journal of Computational and Graphical Statistics* **24**(4), 994–1013.

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.

Harchaoui, Z. and Lévy-Leduc, C. (2010) Multiple change-point estimation with a total variation penalty, *Journal of the American Statistical Association* **105** (492), 1480–1493.

Jeon, J.-J., Kwon, S., and Choi, H. (2017) Homogeneity detection for the high-dimensional generalized linear model, *Computational Statistics & Data Analysis* **114**, 61–74.

Ke, Z.T., Fan, J. and Wu, Y. (2015) Homogeneity pursuit, *Journal of the American Statistical Association* **110**(509), 175–194.

Parikh, N. and Boyd, S. (2014) Proximal algorithms, *Foundations and Trends in optimization* **1**(3), 127–239.

Shen, X. and Huang, H.-C. (2010) Grouping pursuit through a regularization solution surface, *Journal of the American Statistical Association* **105**(490), 727–739.

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

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 (Statistical Methodology)* **67**(1), 91–108.

Wahlberg, B., Boyd, S., Annergren, M., and Wang, Y. (2012) An ADMM algorithm for a class of total variation regularized estimation problems, *IFAC Proceedings Volumes* **45**(16), 83–88.

Zhang, C.-H. (2010) Nearly unbiased variable selection under minimax concave penalty, *The Annals of statistics*, **38**(2), 894–942.

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

Zou, H. and Hastie, T. (2005) Regularization and variable selection via the elastic net, *Journal of the royal statistical society: series B* **67**(2), 301–320.

Zou, H. and Li, R.(2008) One-step sparse estimates in nonconcave penalized likelihood models, *Annals of statistics* **36**(4), 1509–1533.

Received November 30, 2021; Revised January 3, 2022; Accepted January 20, 2022