A Fast Unified Algorithm for Solving Group-Lasso Penalized Learning Problems

Yi Yang and Hui Zou[*]

This version: July 2014

Abstract

This paper concerns a class of group-lasso learning problems where the objective function is the sum of an empirical loss and the group-lasso penalty. For a class of loss function satisfying a quadratic majorization condition, we derive a unified algorithm called groupwise-majorization-descent (GMD) for efficiently computing the solution paths of the corresponding group-lasso penalized learning problem. GMD allows for general design matrices, without requiring the predictors to be group-wise orthonormal. As illustration examples, we develop concrete algorithms for solving the group-lasso penalized least squares and several group-lasso penalized large margin classifiers. These group-lasso models have been implemented in an R package `gglasso` publicly available from the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/web/packages/gglasso`. On simulated and real data, `gglasso` consistently outperforms the existing software for computing the group-lasso that implements either the classical groupwise descent algorithm or Nesterov's method.

[*]School of Statistics, University of Minnesota, Email: zouxx019@umn.edu.

Keywords: Groupwise descent, Group Lasso, grplasso, Large margin classifiers, MM principle, SLEP.

1 Introduction

The lasso (Tibshirani, 1996) is a very popular technique for variable selection for high-dimensional data. Consider the classical linear regression problem where we have a continuous response $\mathbf{y} \in \mathbb{R}^n$ and an $n \times p$ design matrix $\mathbf{X}$. To remove the intercept that is not penalized, we can first center $\mathbf{y}$ and each column of $\mathbf{X}$, that is, all variables have mean zero. The lasso linear regression solves the following ℓ_1 penalized least squares:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \qquad \lambda > 0. \tag{1}$$

The group-lasso (Yuan and Lin, 2006) is a generalization of the lasso for doing group-wise variable selection. Yuan and Lin (2006) motivated the group-wise variable selection problem by two important examples. The first example concerns the multi-factor ANOVA problem where each factor is expressed through a set of dummy variables. In the ANOVA model, deleting an irrelevant factor is equivalent to deleting a group of dummy variables. The second example is the commonly used additive model in which each nonparametric component may be expressed as a linear combination of basis functions of the original variables. Removing a component in the additive model amounts to removing a group of coefficients of the basis functions. In general, suppose that the predictors are put into K non-overlapping groups such that $(1, 2, \ldots, p) = \bigcup_{k=1}^{K} I_k$ where the cardinality of index set I_k is p_k and $I_k \bigcap I_{k'} = \emptyset$ for $k \neq k'$. Consider the linear regression model again and the group-lasso linear regression model solves the following penalized least squares:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{k=1}^{K} \sqrt{p_k}\|\boldsymbol{\beta}^{(k)}\|_2, \qquad \lambda > 0, \tag{2}$$

where $\|\boldsymbol{\beta}^{(k)}\|_2 = \sqrt{\sum_{j\in I_k} \beta_j^2}$. The group-lasso idea has been used in penalized logistic regression (Meier et al., 2008).

The group-lasso is computationally more challenging than the lasso. The entire solution paths of the lasso penalized least squares can be efficiently computed by the *least angle regression* (LARS) algorithm (Efron et al., 2004). See also the homotopy algorithm of Osborne et al. (2000). However, the LARS-type algorithm is not applicable to the group-lasso penalized least squares, because its solution paths are not piecewise linear. Another efficient algorithm for solving the lasso problem is the *coordinate descent* algorithm (Tseng, 2001; Fu, 1998; Daubechies et al., 2004; Genkin et al., 2007; Wu and Lange, 2008; Friedman et al., 2010). Yuan and Lin (2006) implemented a block-wise descent algorithm for the group-lasso penalized least squares by following the idea of Fu (1998). However, their algorithm requires the group-wise orthonormal condition, i.e., $\mathbf{X}_{(k)}^{\intercal}\mathbf{X}_{(k)} = \mathbf{I}_{p_k}$ where $\mathbf{X}_{(k)} = [\cdots X_j \cdots]$, $j \in I_k$. Meier et al. (2008) also developed a block coordinate gradient descent algorithm BCGD for solving the group-lasso penalized logistic regression. Meier's algorithm is implemented in an R package `grplasso` available from the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/web/packages/grplasso`.

From an optimization viewpoint, it is more interesting to solve the group-lasso with a general design matrix. From a statistical perspective, the group-wise orthonormal condition should not be the basis of a good algorithm for solving the group-lasso problem, even though we can transform the predictors within each group to meet the group-wise orthonormal condition. The reason is that even when the group-wise orthonormal condition holds for the observed data, it can be easily violated when removing a fraction of the data or perturbing the dataset as in bootstrap or sub-sampling. In other words, we cannot perform cross-validation, bootstrap or sub-sampling analysis of the group-lasso, if the algorithm's validity depends on the group-wise orthonormal condition. In a popular MATLAB package `SLEP`, Liu et al. (2009) implemented Nesterov's method (Nesterov, 2004, 2007) for a variety of sparse learning problems. For the group-lasso case, `SLEP` provides functions for solving the group-lasso penalized least squares and logistic regression. Nesterov's method can handle general design matrices. The `SLEP` package is available at `http://www.public.asu.edu/~jye02/Software/SLEP`.

In this paper we consider a general formulation of the group-lasso penalized learning where the learning procedure is defined by minimizing the sum of an empirical loss and the group-lasso penalty. The aforementioned group-lasso penalized least squares and logistic regression are two examples of the general formulation. We propose a simple unified algorithm, *groupwise-majorization-descent* (GMD), for solving the general group-lasso learning problems under the condition that the loss function satisfies a quadratic majorization (QM) condition. GMD is remarkably simple and has provable numerical convergence properties. We show that the QM condition indeed holds for many popular loss functions used in regression and classification, including the squared error loss, the Huberized hinge loss, the squared hinge loss and the logistic regression loss. It is also important to point out that GMD works for general design matrices, without requiring the group-wise orthogonal assumption. We have implemented the proposed algorithm in an R package `gglasso` which contains the functions for fitting the group-lasso penalized least squares, logistic regression, Huberized SVM using the Huberized hinge loss and squared SVM using the squared hinge loss. The Huberized hinge loss and squared hinge loss are interesting loss functions for classification from machine learning viewpoint. In fact, there has been both theoretical and empirical evidence showing that the Huberized hinge loss is better than the hinge loss (Zhang, 2004; Wang et al., 2008). The group-lasso penalized Huberized SVM and squared SVM are not implemented in `grplasso` and `SLEP`.

Here we use breast cancer data (Graham et al., 2010) to demonstrate the speed advantage of `gglasso` over `grplasso` and `SLEP`. This is a binary classification problem where $n = 42$ and $p = 22,283$. We fit a sparse additive logistic regression model using the group-lasso. Each variable contributes an additive component that is expressed by five B-spline basis functions. The group-lasso penalty is imposed on the coefficients of five B-spline basis functions for each variable. Therefore, the corresponding group-lasso logistic regression model has $22,283$ groups and each group has 5 coefficients to be estimated. Displayed in Figure 1 are three solution path plots produced by `grplasso`, `SLEP` and `gglasso`. We computed the group-lasso solutions at 100 λ values on an Intel Xeon X5560 (Quad-core 2.8 GHz) processor. It took `SLEP` about 450 and `grplasso` about 360 seconds to compute the logistic regression paths, while `gglasso` used only about 10 seconds.

The rest of this article is organized as follows. In Section 2 we formulate the general group-lasso learning problem. We introduce the quadratic majorization (QM) condition and show that many popular loss functions for regression and classification satisfy the QM condition. In Section 3 we derive the GMD algorithm for solving the group-lasso model satisfying the QM condition and discuss some important implementation issues. Simulation and real data examples are presented in Section 4. We end the paper with a few concluding remarks in Section 5. We present technical proofs in an Appendix.

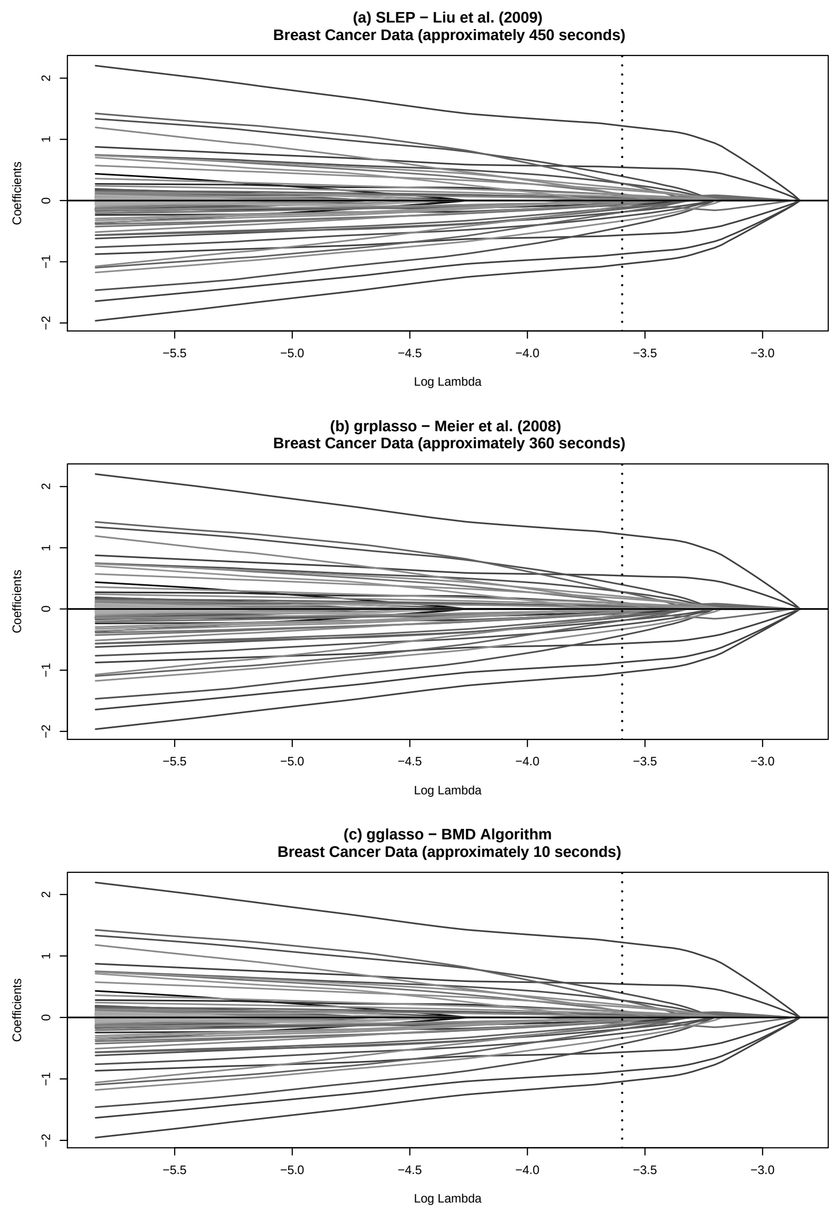


Figure 1: Fit a sparse additive logistic regression model using the group-lasso on the breast cancer data (Graham et al., 2010) with $n = 42$ patients and $22,283$ genes (groups). Each gene's contribution is modeled by 5 B-Spline basis functions. The solution paths are computed at 100 λ values. The vertical dotted lines indicate the selected λ ($\log\lambda = -3.73$), which selects 8 genes.

2 Group-Lasso Models and The QM Condition

2.1 Group-lasso penalized empirical loss

To define a general group-lasso model, we need to introduce some notation. Throughout this paper we use $\mathbf{x}$ to denote the generic predictors which are used to fit the group-lasso model. Note that $\mathbf{x}$ may not be the original variables in the raw data. For example, if we use the group-lasso to fit an additive regression model. The original predictors are $z_1, \dots, z_q$ but we generate $\mathbf{x}$ variables by using basis functions of $z_1, \dots, z_q$. For instance, $x_1 = z_1, x_2 = z_1^2, x_3 = z_1^3$, $x_4 = z_2, x_5 = z_2^2$, etc. We assume that the user has defined the $\mathbf{x}$ variables and we only focus on how to compute the group-lasso model defined in terms of the $\mathbf{x}$ variables.

Let $\mathbf{X}$ be the design matrix with n rows and p columns where n is the sample size of the raw data. If an intercept is used in the model, we let the first column of $\mathbf{X}$ be a vector of 1. Assume that the group membership is already defined such that $(1, 2, \dots, p) = \bigcup_{k=1}^{K} I_k$ and the cardinality of index set I_k is p_k, $I_k \bigcap I_{k'} = \emptyset$ for $k \neq k', 1 \leq k, k' \leq K$. Group k contains $x_j, j \in I_k$, for $1 \leq k \leq K$. If an intercept is included, then $I_1 = \{1\}$. Given the group partition, we use $\boldsymbol{\beta}^{(k)}$ to denote the segment of $\boldsymbol{\beta}$ corresponding to group k. This notation is used for any p-dimensional vector.

Suppose that the statistical model links the predictors to the response variable y via a linear function $f = \boldsymbol{\beta}^{\intercal}\mathbf{x}$. Let $\Phi(y, f)$ be the loss function used to fit the model. In this work we primarily focus on statistical methods for regression and binary classification, although our algorithms are developed for a general loss function. For regression, the loss function $\Phi(y, f)$ is often defined as $\Phi(y - f)$. For binary classification, we use $\{+1, -1\}$ to code the class label y and consider the large margin classifiers where the loss function $\Phi(y, f)$ is defined as $\Phi(yf)$. We obtained an estimate of $\boldsymbol{\beta}$ via the group-lasso penalized empirical loss formulation defined as follows:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \tau_i \Phi(y_i, \boldsymbol{\beta}^{\intercal} \mathbf{x}_i) + \lambda \sum_{k=1}^{K} w_k \|\boldsymbol{\beta}^{(k)}\|_2, \tag{3}$$

where $\tau_i \geq 0$ and $w_k \geq 0$ for all i, k.

Note that we have included two kinds of weights in the general group-lasso formulation. The observation weights τ_is are introduced in order to cover methods such as weighted regression and weighted large margin classification. The default choice for τ_i is 1 for all i. We have also included penalty weights w_ks in order to make a more flexible group-lasso model. The default choice for w_k is $\sqrt{p_k}$. If we do not want to penalize a group of predictors, simply let the corresponding weight be zero. For example, the intercept is typically not penalized so that $w_1 = 0$. Following the adaptive lasso idea (Zou, 2006), one could define the adaptively weighted group-lasso which often has better estimation and variable selection performance than the un-weighted group-lasso (Wang and Leng, 2008). Our algorithms can easily accommodate both observation and penalty weights.

2.2 The QM condition

For notation convenience, we use $\mathbf{D}$ to denote the working data $\{\mathbf{y}, \mathbf{X}\}$ and let $L(\boldsymbol{\beta}|\mathbf{D})$ be the empirical loss, i.e.,

$$L(\boldsymbol{\beta} \mid \mathbf{D}) = \frac{1}{n} \sum_{i=1}^{n} \tau_i \Phi(y_i, \boldsymbol{\beta}^{\intercal} \mathbf{x}_i).$$

Definition 1. *The loss function Φ is said to satisfy the quadratic majorization (QM) condition, if and only if the following two assumptions hold:*

(i). $L(\boldsymbol{\beta} \mid \mathbf{D})$ is differentiable as a function of $\boldsymbol{\beta}$, i.e., $\nabla L(\boldsymbol{\beta}|\mathbf{D})$ exists everywhere.

(ii). There exists a $p \times p$ matrix $\mathbf{H}$, which may only depend on the data $\mathbf{D}$, such that for all $\boldsymbol{\beta}, \boldsymbol{\beta}^*$,

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \leq L(\boldsymbol{\beta}^* \mid \mathbf{D}) + (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \nabla L(\boldsymbol{\beta}^* | \mathbf{D}) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \mathbf{H}(\boldsymbol{\beta} - \boldsymbol{\beta}^*). \quad (4)$$

The following lemma characterizes a class of loss functions that satisfies the QM condition.

Lemma 1. *Let τ_i, $1 \leq i \leq n$ be the observation weights. Let $\boldsymbol{\Gamma}$ be a diagonal matrix with $\boldsymbol{\Gamma}_{ii} = \tau_i$. Assume $\Phi(y, f)$ is differentiable with respect to f and write $\Phi'_f = \frac{\partial \Phi(y,f)}{\partial f}$. Then*

$$\nabla L(\boldsymbol{\beta} | \mathbf{D}) = \frac{1}{n} \sum_{i=1}^{n} \tau_i \Phi'_f(y_i, \mathbf{x}_i^\intercal \boldsymbol{\beta}) \mathbf{x}_i.$$

(1). If Φ'_f is Lipschitz continuous with constant C such that

$$|\Phi'_f(y, f_1) - \Phi'_f(y, f_2)| \leq C|f_1 - f_2| \quad \forall\ y, f_1, f_2,$$

then the QM condition holds for Φ and $\mathbf{H} = \frac{2C}{n} \mathbf{X}^\intercal \boldsymbol{\Gamma} \mathbf{X}$.

(2). If $\Phi''_f = \frac{\partial \Phi^2(y,f)}{\partial f^2}$ exits and

$$\Phi''_f \leq C_2 \quad \forall\ y, f,$$

then the QM condition holds for Φ and $\mathbf{H} = \frac{C_2}{n} \mathbf{X}^\intercal \boldsymbol{\Gamma} \mathbf{X}$.

In what follows we use Lemma 1 to verify that many popular loss functions indeed satisfy the QM condition. The results are summarized in Table 1.

We begin with the classical squared error loss for regression: $\Phi(y, f) = \frac{1}{2}(y - f)^2$. Then we have

$$\nabla L(\boldsymbol{\beta} | \mathbf{D}) = -\frac{1}{n} \sum_{i=1}^{n} \tau_i (y_i - \mathbf{x}_i^\intercal \boldsymbol{\beta}) \mathbf{x}_i. \quad (5)$$

Because $\Phi''_f = 1$, Lemma 1 part (2) tell us that the QM condition holds with

$$\mathbf{H} = \mathbf{X}^\intercal \boldsymbol{\Gamma} \mathbf{X}/n \equiv \mathbf{H}^{\text{ls}}. \quad (6)$$

We now discuss several margin-based loss functions for binary classification. We code y by $\{+1, -1\}$. The logistic regression loss is defined as $\Phi(y, f) = \text{Logit}(yf) = \log(1+\exp(-yf))$. We have $\Phi'_f = -y\frac{1}{1+\exp(yf)}$ and $\Phi''_f = y^2\frac{\exp(yf)}{(1+\exp(yf))^2} = \frac{\exp(yf)}{(1+\exp(yf))^2}$. Then we write

$$\nabla L(\boldsymbol{\beta} \mid \mathbf{D}) = -\frac{1}{n}\sum_{i=1}^{n} \tau_i y_i \mathbf{x}_i \frac{1}{1+\exp(y_i \mathbf{x}_i^\intercal \boldsymbol{\beta})}. \tag{7}$$

Because $\Phi''_f \leq 1/4$, by Lemma 1 part (2) the QM condition holds for the logistic regression loss and

$$\mathbf{H} = \frac{1}{4}\mathbf{X}^\intercal \boldsymbol{\Gamma} \mathbf{X}/n \equiv \mathbf{H}^{\text{logit}}. \tag{8}$$

The squared hinge loss has the expression $\Phi(y, f) = \text{sqsvm}(yf) = [(1-yf)_+]^2$ where

$$(1-t)_+ = \begin{cases} 0, & t > 1 \\ 1-t, & t \leq 1. \end{cases}$$

By direct calculation we have

$$\Phi'_f = \begin{cases} 0, & yf > 1 \\ -2y(1-yf), & yf \leq 1, \end{cases}$$

$$\nabla L(\boldsymbol{\beta} \mid \mathbf{D}) = -\frac{1}{n}\sum_{i=1}^{n} 2\tau_i y_i \mathbf{x}_i (1 - y_i \mathbf{x}_i^\intercal \boldsymbol{\beta})_+. \tag{9}$$

We can also verify that $|\Phi'_f(y, f_1) - \Phi'_f(y, f_2)| \leq 2|f_1 - f_2|$. By Lemma 1 part (1) the QM condition holds for the squared hinge loss and

$$\mathbf{H} = 4\mathbf{X}^\intercal \boldsymbol{\Gamma} \mathbf{X}/n \equiv \mathbf{H}^{\text{sqsvm}}. \tag{10}$$

The Huberized hinge loss is defined as $\Phi(y, f) = \text{hsvm}(yf)$ where

$$\text{hsvm}(t) = \begin{cases} 0, & t > 1 \\ (1-t)^2/2\delta, & 1-\delta < t \leq 1 \\ 1-t-\delta/2, & t \leq 1-\delta. \end{cases}$$

Loss	$-\nabla L(\boldsymbol{\beta} \mid \mathbf{D})$	$\mathbf{H}$
Least squares	$\frac{1}{n}\sum_{i=1}^n \tau_i(y_i - \mathbf{x}_i^\intercal\boldsymbol{\beta})\mathbf{x}_i$	$\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}/n$
Logistic regression	$\frac{1}{n}\sum_{i=1}^n \tau_i y_i \mathbf{x}_i \frac{1}{1+\exp(y_i\mathbf{x}_i^\intercal\boldsymbol{\beta})}$	$\frac{1}{4}\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}/n$
Squared hinge loss	$\frac{1}{n}\sum_{i=1}^n 2\tau_i y_i \mathbf{x}_i(1 - y_i\mathbf{x}_i^\intercal\boldsymbol{\beta})_+$	$4\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}/n$
Huberized hinge loss	$\frac{1}{n}\sum_{i=1}^n \tau_i y_i \mathbf{x}_i \text{hsvm}'(y_i\mathbf{x}_i^\intercal\boldsymbol{\beta})$	$\frac{2}{\delta}\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}/n$

Table 1: The QM condition is verified for the least squares, logistic regression, squared hinge loss and Huberized hinge loss.

By direct calculation we have $\Phi'_f = y\text{hsvm}'(yf)$ where

$$\text{hsvm}'(t) = \begin{cases} 0, & t > 1 \\ (1-t)/\delta, & 1-\delta < t \le 1 \\ 1, & t \le 1-\delta, \end{cases}$$

$$\nabla L(\boldsymbol{\beta} \mid \mathbf{D}) = -\frac{1}{n}\sum_{i=1}^n \tau_i y_i \mathbf{x}_i \text{hsvm}'(y_i\mathbf{x}_i^\intercal\boldsymbol{\beta}). \tag{11}$$

We can also verify that $|\Phi'_f(y, f_1) - \Phi'_f(y, f_2)| \le \frac{1}{\delta}|f_1 - f_2|$. By Lemma 1 part (1) the QM condition holds for the Huberized hinge loss and

$$\mathbf{H} = \frac{2}{\delta}\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}/n \equiv \mathbf{H}^{\text{hsvm}}. \tag{12}$$

3 GMD Algorithm

3.1 Derivation

In this section we derive the *groupwise-majorization-descent* (GMD) algorithm for computing the solution of (3) when the loss function satisfies the QM condition. The objective function is

$$L(\boldsymbol{\beta} \mid \mathbf{D}) + \lambda \sum_{k=1}^{K} w_k \|\boldsymbol{\beta}^{(k)}\|_2. \tag{13}$$

Let $\widetilde{\boldsymbol{\beta}}$ denote the current solution of $\boldsymbol{\beta}$. Without loss of generality, let us derive the GMD update of $\widetilde{\boldsymbol{\beta}}^{(k)}$, the coefficients of group k. Define $\mathbf{H}^{(k)}$ as the sub-matrix of $\mathbf{H}$ corresponding to group k. For example, if group 2 is $\{2, 4\}$ then $\mathbf{H}_2$ is a 2×2 matrix with $\mathbf{H}^{(2)}_{11} = \mathbf{H}_{2,2}, \mathbf{H}^{(2)}_{12} = \mathbf{H}_{2,4}, \mathbf{H}^{(2)}_{21} = \mathbf{H}_{4,2}, \mathbf{H}^{(2)}_{22} = \mathbf{H}_{4,4}$.

Write $\boldsymbol{\beta}$ such that $\boldsymbol{\beta}^{(k')} = \widetilde{\boldsymbol{\beta}}^{(k')}$ for $k' \neq k$. Given $\boldsymbol{\beta}^{(k')} = \widetilde{\boldsymbol{\beta}}^{(k')}$ for $k' \neq k$, the optimal $\boldsymbol{\beta}^{(k)}$ is defined as

$$\operatorname*{argmin}_{\boldsymbol{\beta}^{(k)}} L(\boldsymbol{\beta} \mid \mathbf{D}) + \lambda w_k \|\boldsymbol{\beta}^{(k)}\|_2. \tag{14}$$

Unfortunately, there is no closed form solution to (14) for a general loss function with general design matrix. We overcome the computational obstacle by taking advantage of the QM condition. From (4) we have

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \leq L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) + (\boldsymbol{\beta} - \widetilde{\boldsymbol{\beta}})^{\intercal} \nabla L(\widetilde{\boldsymbol{\beta}}|\mathbf{D}) + \frac{1}{2}(\boldsymbol{\beta} - \widetilde{\boldsymbol{\beta}})^{\intercal}\mathbf{H}(\boldsymbol{\beta} - \widetilde{\boldsymbol{\beta}}).$$

Write $U(\widetilde{\boldsymbol{\beta}}) = -\nabla L(\widetilde{\boldsymbol{\beta}}|\mathbf{D})$. Using

$$\boldsymbol{\beta} - \widetilde{\boldsymbol{\beta}} = (\underbrace{0, \ldots, 0}_{k-1}, \boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)}, \underbrace{0, \ldots, 0}_{K-k}),$$

we can write

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \leq L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) - (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal} U^{(k)} + \frac{1}{2}(\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal}\mathbf{H}^{(k)}(\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)}). \tag{15}$$

Let η_k be the largest eigenvalue of $\mathbf{H}^{(k)}$. We set $\gamma_k = (1 + \varepsilon^*)\eta_k$, where $\varepsilon^* = 10^{-6}$. Then we can further relax the upper bound in (15) as

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \leq L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) - (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal} U^{(k)} + \frac{1}{2}\gamma_k(\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal}(\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)}). \tag{16}$$

It is important to note that the inequality strictly holds unless for $\boldsymbol{\beta}^{(k)} = \widetilde{\boldsymbol{\beta}}^{(k)}$. Instead of minimizing (14) we solve

$$\operatorname*{argmin}_{\boldsymbol{\beta}^{(k)}} L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) - (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal} U^{(k)} + \frac{1}{2}\gamma_k (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal}(\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)}) + \lambda w_k \|\boldsymbol{\beta}^{(k)}\|_2. \quad (17)$$

Denote by $\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})$ the solution to (17). It is straightforward to see that $\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})$ has a simple closed-from expression

$$\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) = \frac{1}{\gamma_k}\left(U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\right)\left(1 - \frac{\lambda w_k}{\|U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\|_2}\right)_+ . \quad (18)$$

Algorithm 1 summarizes the details of GMD.

Algorithm 1 The GMD algorithm for general group-lasso learning.

```
1. For $k = 1, \dots, K$, compute $\gamma_k$, the largest eigenvalue of $\mathbf{H}^{(k)}$.

2. Initialize $\widetilde{\boldsymbol{\beta}}$.

3. Repeat the following cyclic groupwise updates until convergence:

    — for $k = 1, \dots, K$, do step (3.1)–(3.3)

        3.1 Compute $U(\widetilde{\boldsymbol{\beta}}) = -\nabla L(\widetilde{\boldsymbol{\beta}}|\mathbf{D})$.
        3.2 Compute $\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) = \frac{1}{\gamma_k}\left(U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\right)\left(1 - \frac{\lambda w_k}{\|U^{(k)}+\gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\|_2}\right)_+$.
        3.3 Set $\widetilde{\boldsymbol{\beta}}^{(k)} = \widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})$.
```

We can prove the strict descent property of GMD by using the MM principle (Lange et al., 2000; Hunter and Lange, 2004; Wu and Lange, 2010). Define

$$Q(\boldsymbol{\beta} \mid \mathbf{D}) = L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) - (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal} U^{(k)} + \frac{1}{2}\gamma_k (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal}(\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)}) + \lambda w_k \|\boldsymbol{\beta}^{(k)}\|_2. \quad (19)$$

Obviously, $Q(\boldsymbol{\beta} \mid \mathbf{D}) = L(\boldsymbol{\beta} \mid \mathbf{D}) + \lambda w_k \|\boldsymbol{\beta}^{(k)}\|_2$ when $\boldsymbol{\beta}^{(k)} = \widetilde{\boldsymbol{\beta}}^{(k)}$ and (16) shows that $Q(\boldsymbol{\beta} \mid \mathbf{D}) > L(\boldsymbol{\beta} \mid \mathbf{D}) + \lambda w_k \|\boldsymbol{\beta}^{(k)}\|_2$ when $\boldsymbol{\beta}^{(k)} \neq \widetilde{\boldsymbol{\beta}}^{(k)}$. After updating $\widetilde{\boldsymbol{\beta}}^{(k)}$ using (18), we have

$$\begin{aligned}
L(\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) \mid \mathbf{D}) + \lambda w_k \|\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})\|_2 &\leq Q(\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) \mid \mathbf{D}) \\
&\leq Q(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) \\
&= L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) + \lambda w_k \|\widetilde{\boldsymbol{\beta}}^{(k)}\|_2.
\end{aligned}$$

Moreover, if $\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) \neq \widetilde{\boldsymbol{\beta}}^{(k)}$, then the first inequality becomes

$$L(\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) \mid \mathbf{D}) + \lambda w_k \|\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})\|_2 < Q(\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) \mid \mathbf{D}).$$

Therefore, the objective function is strictly decreased after updating all groups in a cycle, unless the solution does not change after each groupwise update. If this is the case, we can show that the solution must satisfy the KKT conditions, which means that the algorithm converges and finds the right answer. To see this, if $\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) = \widetilde{\boldsymbol{\beta}}^{(k)}$ for all k, then by the update formula (18) we have that for all k

$$\widetilde{\boldsymbol{\beta}}^{(k)} = \frac{1}{\gamma_k}\left(U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\right)\left(1 - \frac{\lambda w_k}{\|U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\|_2}\right) \qquad \text{if } \|U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\|_2 > \lambda w_k, \tag{20}$$

$$\widetilde{\boldsymbol{\beta}}^{(k)} = \mathbf{0} \qquad \text{if } \|U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\|_2 \leq \lambda w_k. \tag{21}$$

By straightforward algebra we obtain the KKT conditions:

$$\begin{aligned}
-U^{(k)} + \lambda w_k \cdot \frac{\widetilde{\boldsymbol{\beta}}^{(k)}}{\|\widetilde{\boldsymbol{\beta}}^{(k)}\|_2} = \mathbf{0} &\qquad \text{if } \widetilde{\boldsymbol{\beta}}^{(k)} \neq \mathbf{0}, \\
\left\|U^{(k)}\right\|_2 \leq \lambda w_k &\qquad \text{if } \widetilde{\boldsymbol{\beta}}^{(k)} = \mathbf{0},
\end{aligned}$$

where $k = 1, 2, \ldots, K$. Therefore, if the objective function stays unchanged after a cycle, the algorithm necessarily converges to the right answer.

3.2 Implementation

We have implemented Algorithm 1 for solving the group-lasso penalized least squares, logistic regression, Huberized SVM and squared SVM. These functions are contained in an R package `gglasso` publicly available from the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/web/packages/gglasso`. We always include the intercept term in the model. Without loss of generality we always center the design matrix beforehand.

We solve each group-lasso model for a sequence of λ values from large to small. The default number of points is 100. Let $\lambda^{[l]}$ denote these grid points. We use the warm-start trick to implement the solution path, that is, the computed solution at $\lambda = \lambda^{[l]}$ is used as the initial value for using Algorithm 1 to compute the solution at $\lambda = \lambda^{[l+1]}$. We define $\lambda^{[1]}$ as the smallest λ value such that all predictors have zero coefficients, except the intercept. In such a case let $\hat{\beta}_1$ be the optimal solution of the intercept. Then the solution at $\lambda^{[1]}$ is $\widehat{\boldsymbol{\beta}}^{[1]} = (\hat{\beta}_1, 0, \ldots, 0)$ as the null model estimates. By the Karush-Kuhn-Tucker conditions we can find that

$$\lambda^{[1]} = \max_{k=1,\ldots,K} \left\| \left[\nabla L(\widehat{\boldsymbol{\beta}}^{[1]} | \mathbf{D}) \right]^{(k)} \right\|_2 / w_k, \quad w_k \neq 0.$$

For least squares and logistic regression models, $\hat{\beta}_1$ has a simple expression:

$$\hat{\beta}_1(\mathrm{LS}) = \frac{\sum_{i=1}^n \tau_i y_i}{\sum_{i=1}^n \tau_i} \qquad \text{group-lasso penalized least squares} \tag{22}$$

$$\hat{\beta}_1(\mathrm{Logit}) = \log\left(\frac{\sum_{y_i=1} \tau_i}{\sum_{y_i=-1} \tau_i} \right) \qquad \text{group-lasso penalized logistic regression} \tag{23}$$

For the other two models, we use the following iterative procedure to solve for $\hat{\beta}_1$:

1. Initialize $\hat{\beta}_1 = \hat{\beta}_1(\mathrm{Logit})$ in large margin classifiers.

2. Compute $\hat{\beta}_1(\mathrm{new}) = \hat{\beta}_1 - \frac{1}{\gamma_1} \nabla L((\hat{\beta}_1, 0, \ldots, 0)|\mathbf{D})_1$ where $\gamma_1 = \frac{1}{n}\sum_{i=1}^n \tau_i$.

3. Let $\hat{\beta}_1 = \hat{\beta}_1(\text{new})$.

4. Repeat 2-3 until convergence.

For computing the solution at each λ we also utilize the strong rule introduced in Tibshirani et al. (2012). Suppose that we have computed $\widehat{\boldsymbol{\beta}}(\lambda^{[l]})$, the solution at $\lambda^{[l]}$. To compute the solution at $\lambda^{[l+1]}$, before using Algorithm 1 we first check if group k satisfies the following inequality:

$$\left\| [\nabla L(\widehat{\boldsymbol{\beta}}(\lambda^{[l]})|\mathbf{D})]^{(k)} \right\|_2 \geq w_k(2\lambda^{[l+1]} - \lambda^{[l]}). \tag{24}$$

Let $S = \{I_k : \text{group } k \text{ passes the check in (24)}\}$ and denote its complement as S^c. The strong rule claims that at $\lambda^{[l+1]}$ the groups in the set S are very likely to have nonzero coefficients and the groups in set S^c are very likely to have zero coefficients. If the strong rule guesses correctly, we only need to use Algorithm 1 to solve the group-lasso model with a reduced data set $\{\mathbf{y}, \mathbf{X}_S\}$ where $\mathbf{X}_S = [\cdots X_j \cdots]$, $j \in S$ corresponds to the design matrix with only the groups of variables in S. Suppose the solution is $\widehat{\boldsymbol{\beta}}_S$. We then need to check whether the strong rule indeed made correct guesses at $\lambda^{[l+1]}$ by checking whether $\widetilde{\boldsymbol{\beta}}(\lambda^{[l+1]}) = (\widehat{\boldsymbol{\beta}}_S, \mathbf{0})$ satisfies the KKT conditions. If for each group k where $I_k \in S^c$ the following KKT condition holds:

$$\left\| [\nabla L(\widetilde{\boldsymbol{\beta}}(\lambda^{[l+1]}) = (\widehat{\boldsymbol{\beta}}_S, \mathbf{0})|\mathbf{D})]^{(k)} \right\|_2 \leq \lambda^{[l+1]} w_k.$$

Then $\widetilde{\boldsymbol{\beta}}(\lambda^{[l+1]}) = (\widehat{\boldsymbol{\beta}}_S, \mathbf{0})$ is the desired solution at $\lambda = \lambda^{[l+1]}$. Otherwise, any group that violates the KKT conditions should be added to S. We update S by $S = S \bigcup V$ where

$$V = \left\{ I_k : I_k \in S^c \text{ and } \left\| [\nabla L(\widetilde{\boldsymbol{\beta}}(\lambda^{[l+1]}) = (\widehat{\boldsymbol{\beta}}_S, \mathbf{0})|\mathbf{D})]^{(k)} \right\|_2 > \lambda^{[l+1]} w_k \right\}.$$

Note that the strong rule will eventually make the correct guess since the set S can only grow larger after each update and hence the strong rule iteration will always stop after a finite number of updates. Algorithm 2 summarizes the details of the strong rule.

Note that not all the corresponding coefficients in S are nonzero. Therefore when we apply Algorithm 1 on the reduced data set $\{\mathbf{y}, \mathbf{X}_S\}$ to cyclically update $\widehat{\boldsymbol{\beta}}_S$, we only focus on a subset A of S which contains those groups whose current coefficients are nonzero $A = \{I_k : I_k \in S \text{ and } \widetilde{\boldsymbol{\beta}}^{(k)} \neq \mathbf{0}\}$. This subset is referred as the active-set. The similar idea has been adopted by previous work (Tibshirani et al., 2012; Meier et al., 2008; Vogt and Roth, 2012). In detail, we first create an active-set A by updating each group belonging to S once. Next Algorithm 1 will be applied on the active-set A until convergence. We then run a complete cycle again on S to see if any group is to be included into the active-set. If not, the algorithm is stopped. Otherwise, Algorithm 1 is repeated on the updated active-set.

Algorithm 2 The GMD algorithm with the strong rule at $\lambda^{[l+1]}$.

1. Initialize $\widetilde{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}(\lambda^{[l]})$.

2. Screen K groups using the strong rule, create an initial survival set S such that for group k where $I_k \in S$

$$\left\| [\nabla L(\widehat{\boldsymbol{\beta}}(\lambda^{[l]})|\mathbf{D})]^{(k)} \right\|_2 \geq w_k(2\lambda^{[l+1]} - \lambda^{[l]}).$$

3. Call Algorithm 1 on a reduced dataset $(y_i, \mathbf{x}_{iS})_{i=1}^n$ to solve $\widehat{\boldsymbol{\beta}}_S$.

4. Compute a set V as the part of S^c that failed KKT check:

$$V = \left\{ I_k : I_k \in S^c \text{ and } \left\| [\nabla L(\widetilde{\boldsymbol{\beta}}(\lambda^{[l+1]}) = (\widehat{\boldsymbol{\beta}}_S, \mathbf{0})|\mathbf{D})]^{(k)} \right\|_2 > \lambda^{[l+1]} w_k \right\}.$$

5. If $V = \varnothing$ then stop the loop and return $\widehat{\boldsymbol{\beta}}(\lambda^{[l+1]}) = (\widehat{\boldsymbol{\beta}}_S, \mathbf{0})$. Otherwise update $S = S \bigcup V$ and go to step 3.

In Algorithm 1 we use a simple updating formula to compute $\nabla L(\widetilde{\boldsymbol{\beta}}|\mathbf{D})$, because it only depends on $R = \mathbf{y} - \mathbf{X}\widetilde{\boldsymbol{\beta}}$ for regression and $R = \mathbf{y} \cdot \mathbf{X}\widetilde{\boldsymbol{\beta}}$ for classification. After updating $\widetilde{\boldsymbol{\beta}}^{(k)}$, for regression we can update R by $R - \mathbf{X}^{(k)}(\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) - \widetilde{\boldsymbol{\beta}}^{(k)})$, for classification update R by $R + \mathbf{y} \cdot \mathbf{X}^{(k)}(\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) - \widetilde{\boldsymbol{\beta}}^{(k)})$.

In order to make a fair comparison to `grplasso` and `SLEP`, we tested three different convergence criteria in `gglasso`:

1. $\max\limits_{j} \frac{\left|\tilde{\beta}_j(\text{current}) - \tilde{\beta}_j(\text{new})\right|}{1+|\tilde{\beta}_j(\text{current})|} < \epsilon$, for $j = 1, 2 \ldots, p$.
2. $\left\|\widetilde{\boldsymbol{\beta}}(\text{current}) - \widetilde{\boldsymbol{\beta}}(\text{new})\right\|_2 < \epsilon$.
3. $\max_k(\gamma_k) \cdot \max\limits_{j} \frac{\left|\tilde{\beta}_j(\text{current}) - \tilde{\beta}_j(\text{new})\right|}{1+|\tilde{\beta}_j(\text{current})|} < \epsilon$, for $j = 1, 2 \ldots, p$ and $k = 1, 2 \ldots, K$.

Convergence criterion 1 is used in `grplasso` and convergence criterion 2 is used in `SLEP`. For the group-lasso penalized least squares and logistic regression, we used both convergence criteria 1 and 2 in `gglasso`. For the group-lasso penalized Huberized SVM and squared SVM, we used convergence criterion 3 in `gglasso`. Compared to criterion 1, criterion 3 uses an extra factor $\max_k(\gamma_k)$ in order to take into account the observation that $\widetilde{\boldsymbol{\beta}}^{(k)}(\text{current}) - \widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})$ depends on $\frac{1}{\gamma_k}$. The default value for ϵ is 10^{-4}.

4 Numerical Examples

In this section, we use simulation and real data to demonstrate the efficiency of the GMD algorithm in terms of timing performance and solution accuracy. All numerical experiments were carried out on an Intel Xeon X5560 (Quad-core 2.8 GHz) processor. In this section, let `gglasso` (LS1) and `gglasso` (LS2) denote the group-lasso penalized least squares solutions computed by `gglasso` where the convergence criterion is criterion 1 and criterion 2, respectively. Likewise, we define `gglasso` (Logit1) and `gglasso` (Logit2) for the group-lasso penalized logistic regression.

4.1 Timing comparison

We design a simulation model by combining the *FHT* model introduced in Friedman et al. (2010) and the simulation model 3 in Yuan and Lin (2006). We generate original predictors $X_j,\ j = 1, 2 \ldots, q$ from a multivariate normal distribution with a compound symmetry correlation matrix such that the correlation between X_j and $X_{j'}$ is ρ for $j \neq j'$. Let

$$Y^* = \sum_{j=1}^{q} (\frac{2}{3}X_j - X_j^2 + \frac{1}{3}X_j^3)\beta_j,$$

where $\beta_j = (-1)^j \exp(-(2j-1)/20)$. When fitting a group-lasso model, we treat $\{X_j, X_j^2, X_j^3\}$ as a group, so the final predictor matrix has the number of variables $p = 3q$.

For regression data we generate a response $Y = Y^* + k \cdot e$ where the error term e is generated from $N(0, 1)$. k is chosen such that the signal-to-noise ratio is 3.0. For classification data we generate the binary response Y according to

$$\Pr(\mathrm{Y} = -1) = 1/(1 + \exp(-Y^*)), \quad \Pr(\mathrm{Y} = +1) = 1/(1 + \exp(Y^*)).$$

We considered the following combinations of (n, p):

Scenario 1. $(n, p) = (100, 3000)$ and $(n, p) = (300, 9000)$.

Scenario 2. $n = 200$ and $p = 600, 1200, 3000, 6000, 9000$, shown in Figure 2.

For each (n, p, ρ) combination we recorded the timing (in seconds) of computing the solution paths at 100 λ values of each group-lasso penalized model by `gglasso`, `SLEP` and `grplasso`. The results was averaged over 10 independent runs.

Table 2 shows results from Scenario 1. We see that gglasso has the best timing performance. In the group-lasso penalized least squares case, gglasso (LS2) is about 12 times faster than SLEP (LS). In the group-lasso penalized logistic regression case, gglasso (Logit2) is about 2-6 times faster than SLEP (Logit) and gglasso (Logit1) is about 5-10 times faster than grplasso (Logit).

Figure 2 shows results from Scenario 2 in which we examine the impact of dimension on the timing of gglasso. We fixed n at 200 and plotted the run time (in log scale) against p for three correlation levels 0.2, 0.5 and 0.8. We see that higher ρ increases the timing of gglasso in general. For each fixed correlation level, the timing increases linearly with the dimension.

4.2 Quality comparison

In this section we show that gglasso is also more accurate than grplasso and SLEP under the same convergence criterion. We test the accuracy of solutions by checking their KKT conditions. Theoretically, $\boldsymbol{\beta}$ is the solution of (3) if and only if the following KKT conditions hold:

$$\begin{aligned} [\nabla L(\boldsymbol{\beta}|\mathbf{D})]^{(k)} + \lambda w_k \cdot \frac{\boldsymbol{\beta}^{(k)}}{\|\boldsymbol{\beta}^{(k)}\|_2} &= \mathbf{0} \qquad \text{if } \boldsymbol{\beta}^{(k)} \neq \mathbf{0}, \\ \left\|[\nabla L(\boldsymbol{\beta}|\mathbf{D})]^{(k)}\right\|_2 &\leq \lambda w_k \qquad \text{if } \boldsymbol{\beta}^{(k)} = \mathbf{0}, \end{aligned}$$

where $k = 1, 2, \ldots, K$. The theoretical solution for the convex optimization problem (3) should be unique and always passes the KKT condition check. However, a numerical solution could only approach this analytical value within certain precision therefore may fail the KKT check. Numerically, we declare $\boldsymbol{\beta}^{(k)}$ passes the KKT condition check if

$$\begin{aligned} \left\| [\nabla L(\boldsymbol{\beta}|\mathbf{D})]^{(k)} + \lambda w_k \cdot \frac{\boldsymbol{\beta}^{(k)}}{\|\boldsymbol{\beta}^{(k)}\|_2} \right\|_2 &\le \varepsilon \qquad \text{if } \boldsymbol{\beta}^{(k)} \neq \mathbf{0}, \\ \left\| [\nabla L(\boldsymbol{\beta}|\mathbf{D})]^{(k)} \right\|_2 \le \lambda w_k &+ \varepsilon \qquad \text{if } \boldsymbol{\beta}^{(k)} = \mathbf{0}, \end{aligned}$$

for a small $\varepsilon > 0$. In this paper we set $\varepsilon = 10^{-4}$.

For the solutions of the FHT model scenario 1 computed in section 4.1, we also calculated the number of coefficients that violated the KKT condition check at each λ value. Then this number was averaged over the 100 values of λs. This process was then repeated 10 times on 10 independent datasets. As shown in Table 3, in the group-lasso penalized least squares case, `gglasso` (LS1) has zero violation count; `gglasso` (LS2) also has smaller violation counts compared with SLEP (LS). In the group-lasso penalized classification cases, `gglasso` (Logit1) has less KKT violation counts than `grplasso` (Logit) does when both use convergence criterion 1, and `gglasso` (Logit2) has less KKT violation counts than `SLEP` (Logit) when both use convergence criterion 2. Overall, it is clear that `gglasso` is numerically more accurate than `grplasso` and `SLEP`. gglasso (HSVM) and `gglasso` (SqSVM) both pass KKT checks without any violation.

4.3 Real data analysis

In this section we compare `gglasso`, `grplasso` and `SLEP` on several real data examples. Table 4 summarizes the datasets used in this section. We fit a sparse additive regression

Timing Comparison						
Data	$n = 100$ $p = 3000$			$n = 300$ $p = 9000$		
ρ	0.2	0.5	0.8	0.2	0.5	0.8
Regression						
SLEP (LS)	7.31	8.90	10.76	31.39	40.53	36.56
gglasso (LS1)	1.05	1.33	3.57	5.53	18.62	38.84
gglasso (LS2)	0.60	0.60	0.63	2.93	2.98	2.80
Classification						
grplasso (Logit)	31.78	38.19	58.45	111.70	158.44	239.67
gglasso (Logit1)	3.16	5.65	10.39	19.42	22.98	37.39
SLEP (Logit)	5.50	5.86	2.39	24.68	22.14	6.80
gglasso (Logit2)	0.95	0.95	0.76	6.11	5.35	3.68
gglasso (HSVM)	4.36	8.60	14.63	24.46	33.77	65.29
gglasso (SqSVM)	5.35	10.21	15.32	30.80	41.00	73.68

Table 2: The *FHT* model scenario 1. Reported numbers are timings (in seconds) of **gglasso**, **grplasso** and **SLEP** for computing solution paths at 100 λ values using the group-lasso penalized least squares, logistics regression, Huberized SVM and squared SVM models. Results are averaged over 10 independent runs.

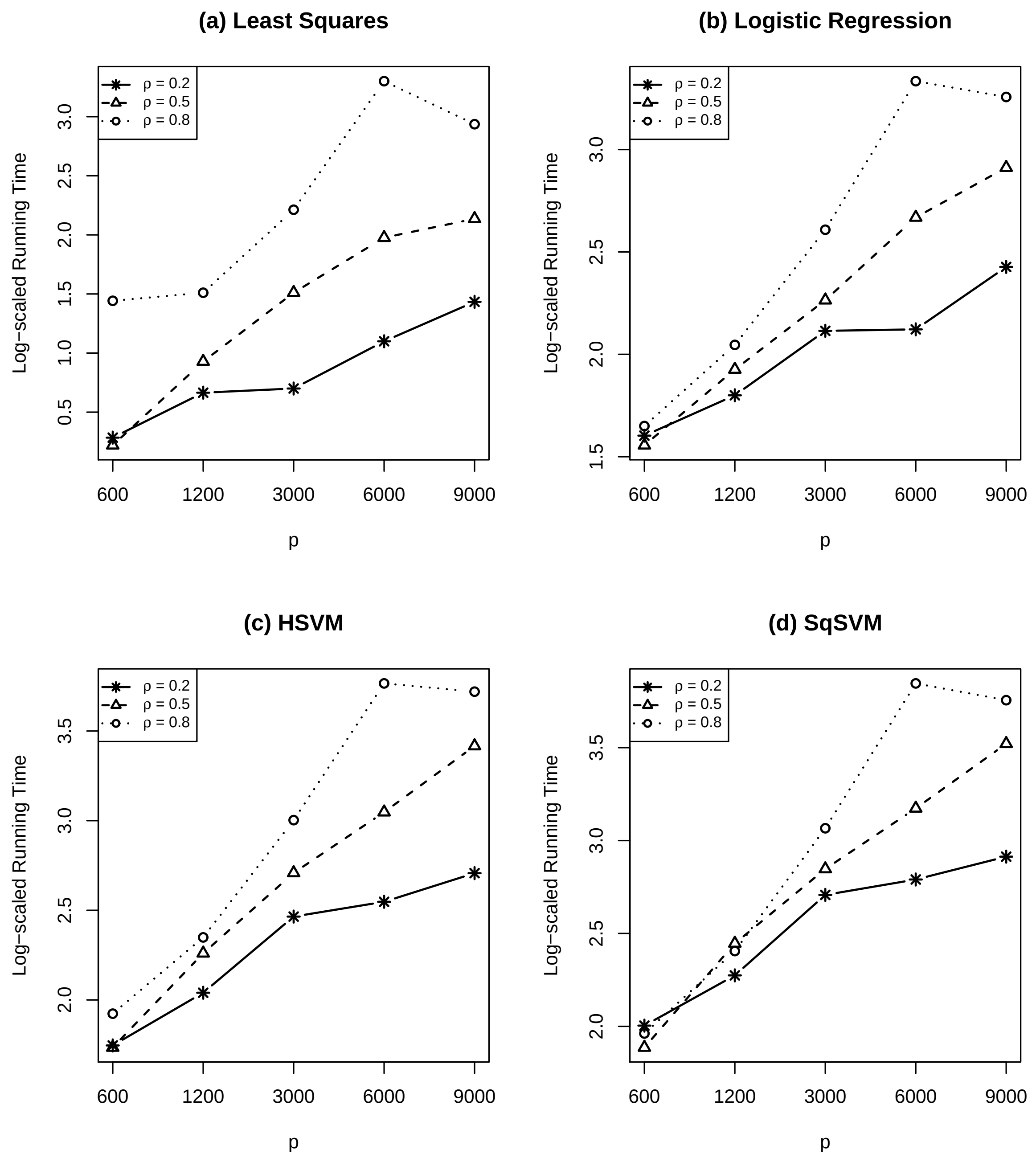


Figure 2: The *FHT* model scenario 2. The average running time of 10 independent runs (in the natural logarithm scale) of **gglasso** for computing solution paths of (a) least squares; (b) logistic regression; (c) Huberized SVM; (d) squared SVM. In all cases $n = 200$.

Quality Comparison: KKT Condition Check						
Data	$n = 100$ $p = 3000$			$n = 300$ $p = 9000$		
ρ	0.2	0.5	0.8	0.2	0.5	0.8
Regression						
SLEP (LS)	23	21	17	53	48	30
gglasso (LS1)	0	0	0	0	0	0
gglasso (LS2)	23	21	16	43	46	27
Classification						
grplasso (Logit)	0	5	17	0	28	37
gglasso (Logit1)	0	0	0	0	0	0
SLEP (Logit)	7	23	29	15	70	143
gglasso (Logit2)	9	24	24	4	50	48
gglasso (HSVM)	0	0	0	0	0	0
gglasso (SqSVM)	0	0	0	0	0	0

Table 3: The *FHT* model scenario 1. Reported numbers are the average number of coefficients among p coefficients that violated the KKT condition check (rounded down to the next smaller integer) using **gglasso**, **grplasso** and **SLEP**. Results are averaged over the λ sequence of 100 values and averaged over 10 independent runs.

Dataset	Type	n	q	p	Data Source
Autompg	R	392	7	31	(Quinlan, 1993)
Bardet	R	120	200	1000	(Scheetz et al., 2006)
Cardiomypathy	R	30	6319	31595	(Segal et al., 2003)
Spectroscopy	R	103	100	500	(Sæbø et al., 2008)
Breast	C	42	22283	111415	(Graham et al., 2010)
Colon	C	62	2000	10000	(Alon et al., 1999)
Prostate	C	102	6033	30165	(Singh et al., 2002)
Sonar	C	208	60	300	(Gorman and Sejnowski, 1988)

Table 4: Real Datasets. n is the number of instances. q is the number of original variables. p is the number of predictors after expansion. “R” means regression and “C” means classification.

model for the regression-type data and fit a sparse additive logistic regression model for the classification-type data. The group-lasso penalty is used to select important additive components. All data were standardized in advance such that each original variable has zero mean and unit sample variance. Some datasets contain only numeric variables but some datasets have both numeric and categorical variables. For any categorical variable with M levels of measurement, we recoded it by $M-1$ dummy variables and treated these dummy variables as a group. For each continuous variable, we used five B-Spline basis functions to represent its effect in the additive model. Those five basis functions are considered as a group. For example, in Colon data the original data have 2000 numeric variables. After basis function expansion there are 10000 predictors in 2000 groups.

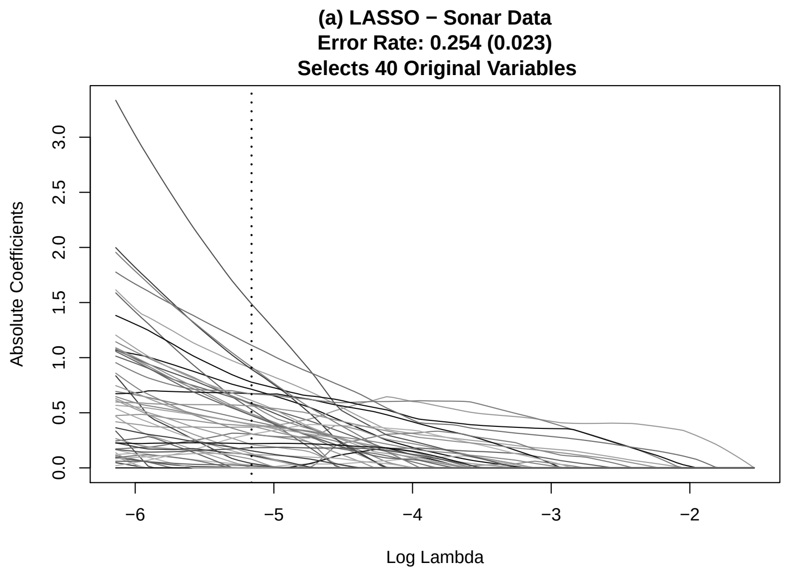
For each dataset, we report the average timings of 10 independent runs for computing the solution paths at 100 λ values. We also report the average number of the KKT check violations. The results are summarized in Table 5. It is clear that `gglasso` outperforms both `grplasso` and `SLEP`.

Before ending this section we would like to use a real data example to demonstrate why the group-lasso could be advantageous over the lasso. On sonar data we compared the lasso penalized logistic regression and the group-lasso penalized logistic regression. We randomly split the sonar data into training and test sets according to 4:1 ratio, and found the optimal λ for each method using five-fold cross-validation on the training data. Then we calculated the misclassification error rate on the test set. We used `glmnet` (Friedman et al., 2010) to compute the lasso penalized logistic regression. The process was repeated 100 times. In the group-lasso model, we define the group-wise L_2 coefficient norm $\theta_j(\lambda)$ for the jth variable by $\theta_j(\lambda) = \sqrt{\sum_{i=1}^{5} \hat{\beta}_{ji}^2(\lambda)}$. Then the jth variable enters the final model if and only if $\theta_j(\lambda) \neq 0$. Figure 3 shows the solution paths of the tuned lasso and group-lasso logistic model from one run, where in the group-lasso plot we plot $\theta_j(\lambda)$ against $\log \lambda$. To make a more direct comparison, we also plot the absolute value of each coefficient in the lasso plot. The fitted lasso logistic regression model selected 40 original variables while the group-lasso logistic regression model selected 26 original variables. When looking at the average misclassification error of 100 runs, we see that the group-lasso logistic regression model is significantly more accurate than the lasso logistic regression model. Note that the sample size is 208 in the Sonar data, thus the misclassification error calculation is meaningful.

Group-lasso Regression on Real Data				
Dataset	*Autompg*	*Bardet*	*Cardiomypathy*	*Spectroscopy*
	Sec. — KKT	Sec. — KKT	Sec. — KKT	Sec. — KKT
SLEP (LS)	3.14 — 0	9.96 — 0	78.23 — 0	9.37 — 0
gglasso (LS1)	1.79 — 1	8.49 — 1	0.38 — 2	0.50 — 0
gglasso (LS2)	1.29 — 0	1.04 — 0	2.53 — 0	2.26 — 0

Group-lasso Classification on Real Data				
Dataset	*Colon*	*Prostate*	*Sonar*	*Breast*
	Sec. — KKT	Sec. — KKT	Sec. — KKT	Sec. — KKT
grplasso (Logit)	60.42 — 0	111.75 — 0	24.55 — 0	439.76 — 0
gglasso (Logit1)	1.13 — 0	3.877 — 0	1.54 — 0	9.62 — 0
SLEP (Logit)	75.31 — 0	166.91 — 0	5.49 — 0	358.75 — 0
gglasso (Logit2)	2.23 — 0	4.36 — 0	2.88 — 0	10.24 — 0
gglasso (HSVM)	1.15 — 0	3.53 — 0	0.66 — 0	9.15 — 0
gglasso (SqSVM)	1.45 — 0	3.79 — 0	1.27 — 1	9.58 — 0

Table 5: Group-lasso penalized regression and classification on real datasets. Reported numbers are: (a) timings (in seconds), total time for 100 λ values; (b) the average number of coefficients among p coefficients that violated the KKT condition check. Results are averaged over 10 independent runs.



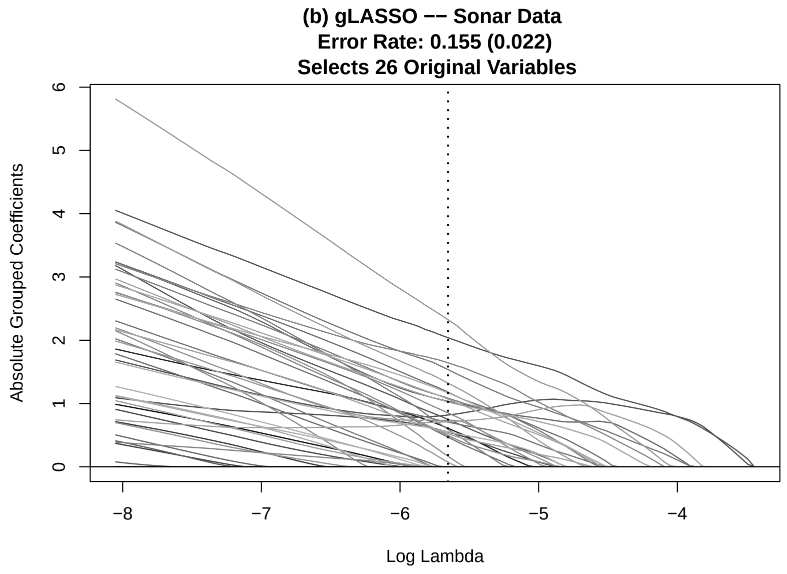


Figure 3: Compare the lasso penalized logistic regression and the group-lasso penalized logistic regression on Sonar data with $n = 208$ and $p = 60$. (a) solution paths of lasso penalized logistic regression model, in which each absolute coefficient $|\beta_j|$ is plotted against $\log \lambda$ for $j = 1, \ldots, p$. (b) solution paths of group-lasso penalized logistic regression model with $p^* = 300$ expanded variables grouped into $K = 60$ groups, in which each group norm $\|\boldsymbol{\beta}^{(k)}\|_2$ is plotted against $\log \lambda$ for $k = 1, \ldots, K$. The vertical dotted lines indicate the best models chosen by cross-validation.

5 Discussion

In this paper we have derived a unified groupwise-majorizatoin-descent algorithm for computing the solution paths of a class of group-lasso penalized models. We have demonstrated the efficiency of Algorithm 1 on four group-lasso models: the group-lasso penalized least squares, the group-lasso penalized logistic regression, the group-lasso penalized HSVM and the group-lasso penalized SqSVM. Algorithm 1 can be readily applied to other interesting group-lasso penalized models. All we need to do is to check the QM condition for the given loss function. For that, Lemma 1 is a handy tool. We also implemented the exact groupwise descent algorithm in which we solved the convex optimization problem defined in (14) for each group. We used the same computational tricks to implement the exact groupwise descent algorithm, including the strong rule, warm-start and the active set. We found that groupwise-majorization-descent is about 10 to 15 times faster than the exact groupwise descent algorithm. This comparison clearly shows the value of using majorization within the groupwise descent algorithm. For the sake of brevity, we opt not to show the timing comparison here.

`grplasso` is a popular R package for the group-lasso penalized logistic regression, but the underlying algorithm is limited to twice differentiable loss functions. `SLEP` implements Nesterov's method for the group-lasso penalized least squares and logistic regression. In principle, Nesterov's method can be used to solve other group-lasso penalized models. For the group-lasso penalized least squares and logistic regression cases, our package `gglasso` is faster than `SLEP` and `grplasso`. Although we do not claim that groupwise-majorizatoin-descent is superior than Nesterov's method, the numerical evidence clearly shows the practical usefulness of groupwise-majorizatoin-descent.

Finally, we should point out that Nesterov's method is a more general optimization algorithm than groupwise-descent or groupwise-majorizatoin-descent. Note that groupwise-descent or groupwise-majorizatoin-descent can only work for groupwise separable penalty functions in general. What we have shown in this paper is that a more general algorithm like Nesterov's method can be slower than a specific algorithm like GMD for a given set of problems. The same message was reported in a comparison done by Tibshirani in which the coordinate descent algorithm was shown to outperform Nesterov's method for the lasso regression. See https://statweb.stanford.edu/∼tibs/comparison.txt for more details.

Acknowledgements

The authors thank the editor, an associate editor and two referees for their helpful comments and suggestions. This work is supported in part by NSF Grant DMS-08-46068.

Appendix: proofs

Proof of Lemma 1. Part (1). For any $\boldsymbol{\beta}$ and $\boldsymbol{\beta}^*$, write $\boldsymbol{\beta} - \boldsymbol{\beta}^* = V$ and define $g(t) = L(\boldsymbol{\beta}^* + tV \mid \mathbf{D})$ so that

$$g(0) = L(\boldsymbol{\beta}^* \mid \mathbf{D}), \quad g(1) = L(\boldsymbol{\beta} \mid \mathbf{D}).$$

By the mean value theorem, $\exists\ a \in (0, 1)$ such that

$$g(1) = g(0) + g'(a) = g(0) + g'(0) + [g'(a) - g'(0)]. \tag{25}$$

Write $\Phi'_f = \frac{\partial \Phi(y,f)}{\partial f}$. Note that

$$g'(t) = \frac{1}{n} \sum_{i=1}^{n} \tau_i \Phi'_f(y_i, \mathbf{x}_i^{\intercal}(\boldsymbol{\beta}^* + tV))(\mathbf{x}_i^{\intercal} V). \tag{26}$$

Thus $g'(0) = (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \nabla L(\boldsymbol{\beta}^*|\mathbf{D})$. Moreover, from (26) we have

$$\begin{aligned}
|g'(a) - g'(0)| &= |\frac{1}{n}\sum_{i=1}^{n} \tau_i[\Phi'_f(y_i, \mathbf{x}_i^\intercal(\boldsymbol{\beta}^* + aV)) - \Phi'_f(y_i, \mathbf{x}_i^\intercal\boldsymbol{\beta}^*)](\mathbf{x}_i^\intercal V)| \\
&\le \frac{1}{n}\sum_{i=1}^{n} \tau_i|\Phi'_f(y_i, \mathbf{x}_i^\intercal(\boldsymbol{\beta}^* + aV)) - \Phi'_f(y_i, \mathbf{x}_i^\intercal\boldsymbol{\beta}^*)||\mathbf{x}_i^\intercal V| \\
&\le \frac{1}{n}\sum_{i=1}^{n} C\tau_i|\mathbf{x}_i^\intercal aV||\mathbf{x}_i^\intercal V| \qquad (27) \\
&\le \frac{1}{n}\sum_{i=1}^{n} C\tau_i\|\mathbf{x}_i^\intercal V\|_2^2 \\
&= \frac{C}{n}V^\intercal[\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}]V, \qquad (28)
\end{aligned}$$

where in (27) we have used the inequality $|\Phi'(y, f_1) - \Phi'(y, f_2)| \le C|f_1 - f_2|$. Plugging (28) into (25) we have

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \le L(\boldsymbol{\beta}^* \mid \mathbf{D}) + (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \nabla L(\boldsymbol{\beta}^*|\mathbf{D}) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \mathbf{H}(\boldsymbol{\beta} - \boldsymbol{\beta}^*),$$

with $\mathbf{H} = \frac{2C}{n}\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}$.

Part (2). Write $\Phi''_f = \frac{\partial\Phi^2(y,f)}{\partial f^2}$. By Taylor's expansion, $\exists\ b \in (0, 1)$ such that

$$g(1) = g(0) + g'(0) + g''(b). \qquad (29)$$

Note that

$$g''(b) = \frac{1}{n}\sum_{i=1}^{n} \tau_i\Phi''_f(y_i, \mathbf{x}_i^\intercal(\boldsymbol{\beta}^* + bV))(\mathbf{x}_i^\intercal V)^2 \le \frac{1}{n}\sum_{i=1}^{n} C_2\tau_i(\mathbf{x}_i^\intercal V)^2, \qquad (30)$$

where we have used the inequality $\Phi''_f \le C_2$. Plugging (30) into (29) we have

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \le L(\boldsymbol{\beta}^* \mid \mathbf{D}) + (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \nabla L(\boldsymbol{\beta}^*|\mathbf{D}) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\intercal \mathbf{H}(\boldsymbol{\beta} - \boldsymbol{\beta}^*),$$

with $\mathbf{H} = \frac{C_2}{n}\mathbf{X}^\intercal\boldsymbol{\Gamma}\mathbf{X}$.

$\square$

References

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. and Levine, A. (1999), 'Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays', *Proceedings of the National Academy of Sciences* **96**(12), 6745.

Daubechies, I., Defrise, M. and De Mol, C. (2004), 'An iterative thresholding algorithm for linear inverse problems with a sparsity constraint', *Communications on Pure and Applied Mathematics,* **57**, 1413–1457.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004), 'Least angle regression', *Annals of statistics* **32**(2), 407–451.

Friedman, J., Hastie, T. and Tibshirani, R. (2010), 'Regularized paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**, 1–22.

Fu, W. (1998), 'Penalized regressions: the bridge versus the lasso', *Journal of Computational and Graphical Statistics* **7**(3), 397–416.

Genkin, A., Lewis, D. and Madigan, D. (2007), 'Large-scale Bayesian logistic regression for text categorization', *Technometrics* **49**(3), 291–304.

Gorman, R. and Sejnowski, T. (1988), 'Analysis of hidden units in a layered network trained to classify sonar targets', *Neural networks* **1**(1), 75–89.

Graham, K., de Las Morenas, A., Tripathi, A., King, C., Kavanah, M., Mendez, J., Stone, M., Slama, J., Miller, M., Antoine, G. et al. (2010), 'Gene expression in histologically normal epithelium from breast cancer patients and from cancer-free prophylactic mastectomy patients shares a similar profile', *British journal of cancer* **102**(8), 1284–1293.

Hunter, D. and Lange, K. (2004), 'A tutorial on MM algorithms', *The American Statistician* **58**(1), 30–37.

Lange, K., Hunter, D. and Yang, I. (2000), 'Optimization transfer using sur- rogate objective functions (with discussion)', *Journal of Computational and Graphical Statistics* **9**, 1–20.

Liu, J., Ji, S. and Ye, J. (2009), *SLEP: Sparse Learning with Efficient Projections*, Arizona State University.

URL: *http://www.public.asu.edu/~jye02/Software/SLEP*

Meier, L., van de Geer, S. and Bühlmann, P. (2008), 'The group lasso for logistic regression', *Journal of the Royal Statistical Society, Series B* **70**, 53–71.

Nesterov, Y. (2004), 'Introductory lectures on convex optimization: A basic course', *Operations Research* .

Nesterov, Y. (2007), Gradient methods for minimizing composite objective function, Technical report, Technical Report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain (UCL).

Osborne, M., Presnell, B. and Turlach, B. (2000), 'A new approach to variable selection in least squares problems', *IMA Journal of Numerical Analysis* **20(3)**, 389–403.

Quinlan, J. (1993), Combining instance-based and model-based learning, *in* 'Proceedings of the Tenth International Conference on Machine Learning', pp. 236–243.

Sæbø, S., Almøy, T., Aarøe, J. and Aastveit, A. (2008), 'St-pls: a multi-directional nearest shrunken centroid type classifier via pls', *Journal of Chemometrics* **22**(1), 54–62.

Scheetz, T., Kim, K., Swiderski, R., Philp, A., Braun, T., Knudtson, K., Dorrance, A., DiBona, G., Huang, J., Casavant, T. et al. (2006), 'Regulation of gene expression in the mammalian eye and its relevance to eye disease', *Proceedings of the National Academy of Sciences* **103**(39), 14429–14434.

Segal, M., Dahlquist, K. and Conklin, B. (2003), 'Regression approaches for microarray data analysis', *Journal of Computational Biology* **10**(6), 961–980.

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., D'Amico, A., Richie, J. et al. (2002), 'Gene expression correlates of clinical prostate cancer behavior', *Cancer cell* **1**(2), 203–209.

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

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J. and Tibshirani, R. (2012), 'Strong rules for discarding predictors in lasso-type problems', *Journal of the Royal Statistical Society: Series B* **74**, 245–266.

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

Vogt, J. and Roth, V. (2012), A complete analysis of the $l_{1,p}$ group-lasso, *in* 'Proceedings of the 29th International Conference on Machine Learning (ICML-12)', ICML 2012, Omnipress, pp. 185–192.

Wang, H. and Leng, C. (2008), 'A note on adaptive group lasso', *Computational Statistics and Data Analysis* **52**, 5277–5286.

Wang, L., Zhu, J. and Zou, H. (2008), 'Hybrid huberized support vector machines for microarray classification and gene selection', *Bioinformatics* **24**, 412–419.

Wu, T. and Lange, K. (2008), 'Coordinate descent algorithms for lasso penalized regression', *The Annals of Applied Statistics* **2**, 224–244.

Wu, T. and Lange, K. (2010), 'The MM alternative to EM', *Statistical Science* **4**, 492–505.

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

Zhang, T. (2004), 'Statistical behavior and consistency of classification methods based on convex risk minimization', *Annals of Statistics* **32**, 56–85.

Zou, H. (2006), 'The adaptive lasso and its oracle properties', *Journal of the American Statistical Association* **101**, 1418–1429.