Efficient Methods for Overlapping Group Lasso

Lei Yuan  
Arizona State University  
Tempe, AZ, 85287  
Lei.Yuan@asu.edu

Jun Liu  
Arizona State University  
Tempe, AZ, 85287  
j.liu@asu.edu

Jieping Ye  
Arizona State University  
Tempe, AZ, 85287  
jieping.ye@asu.edu

Abstract

The group Lasso is an extension of the Lasso for feature selection on (predefined) non-overlapping groups of features. The non-overlapping group structure limits its applicability in practice. There have been several recent attempts to study a more general formulation, where groups of features are given, potentially with overlaps between the groups. The resulting optimization is, however, much more challenging to solve due to the group overlaps. In this paper, we consider the efficient optimization of the overlapping group Lasso penalized problem. We reveal several key properties of the proximal operator associated with the overlapping group Lasso, and compute the proximal operator by solving the smooth and convex dual problem, which allows the use of the gradient descent type of algorithms for the optimization. We have performed empirical evaluations using both synthetic and the breast cancer gene expression data set, which consists of 8,141 genes organized into (overlapping) gene sets. Experimental results show that the proposed algorithm is more efficient than existing state-of-the-art algorithms.

1 Introduction

Problems with high dimensionality have become common over the recent years. The high dimensionality poses significant challenges in building interpretable models with high prediction accuracy. Regularization has been commonly employed to obtain more stable and interpretable models. A well-known example is the penalization of the $\ell_1$ norm of the estimator, known as Lasso [25]. The $\ell_1$ norm regularization has achieved great success in many applications. However, in some applications [28], we are interested in finding important explanatory factors in predicting the response variable, where each explanatory factor is represented by a group of input features. In such cases, the selection of important features corresponds to the selection of groups of features. As an extension of Lasso, group Lasso [28] based on the combination of the $\ell_1$ norm and the $\ell_2$ norm has been proposed for group feature selection, and quite a few efficient algorithms [16, 17, 19] have been proposed for efficient optimization. However, the non-overlapping group structure in group Lasso limits its applicability in practice. For example, in microarray gene expression data analysis, genes may form overlapping groups as each gene may participate in multiple pathways [12].

Several recent work [3, 13, 15, 18, 29] studies the overlapping group Lasso, where groups of features are given, potentially with overlaps between the groups. The resulting optimization is, however, much more challenging to solve due to the group overlaps. When optimizing the overlapping group Lasso problem, one can reformulate it as a second order cone program and solve it by a generic toolbox, which, however, does not scale well. Jenatton et al. [13] proposed an alternating algorithm called SLasso for solving the equivalent reformulation. However, SLasso involves an expensive matrix inversion at each alternating iteration, and there is no known global convergence rate for such an alternating procedure. A reformulation [5] was also proposed such that the original problem can be solved by the Alternating Direction Method of Multipliers (ADMM), which involves solving a linear system at each iteration, and may not scale well for high dimensional problems. Argyriou et al. [1] adopted the proximal gradient method for solving the overlapping group lasso, and a fixed point method was developed to compute the proximal operator. Chen et al. [6] employed a
smoothing technique to solve the overlapping group Lasso problem. Mairal [18] proposed to solve the proximal operator associated with the overlapping group Lasso defined as the sum of the $\ell_\infty$ norms, which, however, is not applicable to the overlapping group Lasso defined as the sum of the $\ell_2$ norms considered in this paper.

In this paper, we develop an efficient algorithm for the overlapping group Lasso penalized problem via the accelerated gradient descent method. The accelerated gradient descent method has recently received increasing attention in machine learning due to the fast convergence rate even for non-smooth convex problems. One of the key operations is the computation of the proximal operator associated with the penalty. We reveal several key properties of the proximal operator associated with the overlapping group Lasso penalty, and proposed several possible reformulations that can be solved efficiently. The main contributions of this paper include: (1) we develop a low cost preprocessing procedure to identify (and then remove) zero groups in the proximal operator, which dramatically reduces the size of the problem to be solved; (2) we propose one dual formulation and two proximal splitting formulations for the proximal operator; (3) for the dual formulation, we further derive the duality gap which can be used to check the quality of the solution and determine the convergence of the algorithm. We have performed empirical evaluations using both synthetic data and the breast cancer gene expression data set, which consists of 8,141 genes organized into (overlapping) gene sets. Experimental results demonstrate the efficiency of the proposed algorithm in comparison with existing state-of-the-art algorithms.

**Notations:** $\| \cdot \|$ denotes the Euclidean norm, and $0$ denotes a vector of zeros. $\text{SGN}(\cdot)$ and $\text{sgn}(\cdot)$ are defined in a component wise fashion as: 1) if $t = 0$, then $\text{SGN}(t) = [-1, 1]$ and $\text{sgn}(t) = 0$; 2) if $t > 0$, then $\text{SGN}(t) = \{1\}$ and $\text{sgn}(t) = 1$; and 3) if $t < 0$, $\text{SGN}(t) = \{-1\}$ and $\text{sgn}(t) = -1$. $G_i \subseteq \{1, 2, \ldots, p\}$ denotes an index set, and $x_{G_i}$ denote a sub-vector of $x$ restricted to $G_i$.

### 2 The Overlapping Group Lasso

We consider the following overlapping group Lasso penalized problem:

$$
\min_{x \in \mathbb{R}^p} f(x) = l(x) + \phi_{\lambda_1, \lambda_2}^L(x)
$$

(1)

where $l(\cdot)$ is a smooth convex loss function, e.g., the least squares loss,

$$
\phi_{\lambda_1, \lambda_2}^L(x) = \lambda_1 \|x\|_1 + \lambda_2 \sum_{i=1}^{g} w_i \|x_{G_i}\|
$$

(2)

is the overlapping group Lasso penalty, $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are regularization parameters, $w_i > 0, i = 1, 2, \ldots, g, G_i \subseteq \{1, 2, \ldots, p\}$ contains the indices corresponding to the $i$-th group of features, and $\| \cdot \|$ denotes the Euclidean norm. Note that the first term in (2) can be absorbed into the second term, which however will introduce $p$ additional groups. The $g$ groups of features are pre-specified, and they may overlap. The penalty in (2) is a special case of the more general Composite Absolute Penalty (CAP) family [22]. When the groups are disjoint with $\lambda_1 = 0$ and $\lambda_2 > 0$, the model in (1) reduces to the group Lasso [28]. If $\lambda_1 > 0$ and $\lambda_2 = 0$, then the model in (1) reduces to the standard Lasso [25].

In this paper, we propose to make use of the accelerated gradient descent (AGD) [2, 21, 22] for solving (1), due to its fast convergence rate. The algorithm is called “FoGLasso”, which stands for Fast overlapping Group Lasso. One of the key steps in the proposed FoGLasso algorithm is the computation of the proximal operator associated with the penalty in (2), and we present an efficient algorithm for the computation in the next section.

In FoGLasso, we first construct a model for approximating $f(\cdot)$ at the point $x$ as:

$$
f_{L, x}(y) = [l(x) + \langle l'(x), y - x \rangle] + \phi_{\lambda_1, \lambda_2}^L(y) + \frac{L}{2} \|y - x\|^2,
$$

(3)

where $L > 0$. The model $f_{L, x}(y)$ consists of the first-order Taylor expansion of the smooth function $l(\cdot)$ at the point $x$, the non-smooth penalty $\phi_{\lambda_1, \lambda_2}^L(x)$, and a regularization term $\frac{L}{2} \|y - x\|^2$. Next, a sequence of approximate solutions $\{x_i\}$ is computed as follows: $x_{i+1} = \arg \min_y f_{L, s_i}(y)$, where the search point $s_i$ is an affine combination of $x_{i-1}$ and $x_i$ as $s_i = x_i + \beta_i (x_i - x_{i-1})$, for a properly chosen coefficient $\beta_i$. $L_i$ is determined by the line search according to the Armijo-Goldstein rule.
so that \( L_i \) should be appropriate for \( s_i \), i.e., \( f(x_{i+1}) \leq f_{L_i,s_i}(x_{i+1}) \). A key building block in FoGLasso is the minimization of (3), whose solution is known as the proximal operator \( \pi(\cdot) \). The computation of the proximal operator is the main technical contribution of this paper. The pseudo-code of FoGLasso is summarized in Algorithm [1] where the proximal operator \( \pi(\cdot) \) is defined in (4). In practice, we can terminate Algorithm [1] if the change of the function values corresponding to adjacent iterations is within a small value, say \( 10^{-5} \).

### Algorithm 1 The FoGLasso Algorithm

**Input:** \( L_0 > 0, x_0, \alpha \)

**Output:** \( x_{k+1} \)

1. Initialize \( x_i = x_0, \alpha_{-1} = 0, \alpha_0 = 1, \) and \( L = L_0 \).
2. for \( i = 1 \) to \( k \) do
   3. Set \( \beta_i = \frac{\alpha_i - 2 - 1}{\alpha_i} \), \( s_i = x_i + \beta_i(x_i - x_{i-1}) \)
   4. Find the smallest \( L = 2^i L_{i-1}, j = 0, 1, \ldots \) such that \( f(x_{i+1}) \leq f_{L_i,s_i}(x_{i+1}) \) holds, where \( x_{i+1} = \pi_{L_i/s_i}(s_i - \frac{1}{L_i} l'(s_i)) \)
   5. Set \( L_i = L \) and \( \alpha_{i+1} = \frac{1 + \sqrt{1 + 4\alpha_i^2}}{2} \)
3. end for

### 3 The Associated Proximal Operator and Its Efficient Computation

The proximal operator associated with the overlapping group Lasso penalty is defined as follows:

\[
\pi_{\lambda_2}(v) = \arg\min_{x \in \mathbb{R}^p} \left\{ g_{\lambda_2}(x) \equiv \frac{1}{2} \|x - v\|^2 + \phi_{\lambda_2}(x) \right\},
\]

(4)

which is a special case of (1) by setting \( l(x) = \frac{1}{2} \|x - v\|^2 \). It can be verified that the approximate solution \( x_{i+1} = \arg\min_y f_{L_i,s_i}(y) \) is given by \( x_{i+1} = \pi_{\lambda_2/L_i}(s_i - \frac{1}{L_i} l'(s_i)) \). Recently, it has been shown in [12] that the efficient computation of the proximal operator is key to many sparse learning algorithms. Next, we focus on the efficient computation of \( \pi_{\lambda_2}(v) \) in (4) for a given \( v \).

The rest of this section is organized as follows. In Section 3.1 we discuss some key properties of the proximal operator, based on which we propose a pre-processing technique that will significantly reduce the size of the problem. We then proposed to solve it via the dual formulation in Section 3.2 and the duality gap is also derived. Several alternative methods for solving the proximal operator via proximal splitting methods are discussed in Section 3.3.

#### 3.1 Key Properties of the Proximal Operator

We first reveal several basic properties of the proximal operator \( \pi_{\lambda_2}(v) \).

**Lemma 1.** Suppose that \( \lambda_1, \lambda_2 \geq 0, \) and \( w_i > 0, \) for \( i = 1, 2, \ldots, g \). Let \( x^* = \pi_{\lambda_2}(v) \). The following holds: 1) if \( v_i > 0, \) then \( 0 \leq x^*_i \leq v_i \); 2) if \( v_i < 0, \) then \( v_i \leq x^*_i \leq 0 \); 3) if \( v_i = 0, \) then \( x^*_i = 0 \); 4) \( \text{SGN}(v) \subseteq \text{SGN}(x^*) \); and 5) \( \pi_{\lambda_2}(v) = \text{sgn}(v) \odot \pi_{\lambda_2}(|v|) \).

**Proof.** When \( \lambda_1, \lambda_2 \geq 0, \) and \( w_i > 0, \) for \( i = 1, 2, \ldots, g, \) the objective function \( g_{\lambda_2}(\cdot) \) is strictly convex, thus \( x^* \) is the unique minimizer. We first show if \( v_i > 0, \) then \( 0 \leq x^*_i \leq v_i \). If \( x^*_i > v_i, \) then we can construct a \( \tilde{x} \) as follows: \( \tilde{x}_j = x^*_j, j \neq i \) and \( \tilde{x}_i = v_i \). Similarly, if \( x^*_i < 0, \) then we can construct a \( \tilde{x} \) as follows: \( \tilde{x}_j = x^*_j, j \neq i \) and \( \tilde{x}_i = 0 \). It is easy to verify that \( \tilde{x} \) achieves a lower objective function value than \( x^* \) in both cases. We can prove the second and the third properties using similar arguments. Finally, we can prove the fourth and the fifth properties using the definition of \( \text{SGN}(\cdot) \).

Next, we show that \( \pi_{\lambda_2}(\cdot) \) can be directly derived from \( \pi_{\lambda_2}(\cdot) \) by soft-thresholding. Thus, we only need to focus on the case when \( \lambda_1 = 0 \). This simplifies the optimization in (4). It is an extension of the result for Fused Lasso in [10].

**Theorem 1.** Let \( u = \text{sgn}(v) \odot \max(|v| - \lambda_1, 0) \), and

\[
\pi_{\lambda_2}^0(u) = \arg\min_{x \in \mathbb{R}^p} \left\{ h_{\lambda_2}(x) \equiv \frac{1}{2} \|x - u\|^2 + \lambda_2 \sum_{i=1}^g w_i \|x_{G_i}\| \right\},
\]

(5)

Then, the following holds: \( \pi_{\lambda_2}(v) = \pi_{\lambda_2}(u) \).

---

3
Proof. Denote the unique minimizer of \( h_{\lambda_2}(\cdot) \) as \( x^* \). The sufficient and necessary condition for the optimality of \( x^* \) is:

\[
0 \in \partial h_{\lambda_2}(x^*) = x^* - u + \partial \phi_{\lambda_2}(x^*),
\]

where \( \partial h_{\lambda_2}(x) \) and \( \partial \phi_{\lambda_2}(x) \) are the sub-differential sets of \( h_{\lambda_2}(\cdot) \) and \( \phi_{\lambda_2}(\cdot) \) at \( x \), respectively.

Next, we need to show \( 0 \in \partial g_{\lambda_2}(x^*) \). The sub-differential of \( g_{\lambda_2}(\cdot) \) at \( x^* \) is given by

\[
\partial g_{\lambda_2}(x^*) = x^* - v + \partial \phi_{\lambda_2}(x^*) = x^* - v + \lambda_1 \text{SGN}(x^*) + \partial \phi_{\lambda_2}(x^*).
\]

It follows from the definition of \( u \) that \( u \in v - \lambda_1 \text{SGN}(u) \). Using the fourth property in Lemma 1, we have \( \text{SGN}(u) \subseteq \text{SGN}(x^*) \). Thus,

\[
u \in v - \lambda_1 \text{SGN}(x^*).
\]

It follows from (6)-(8) that \( 0 \in \partial g_{\lambda_2}(x^*) \).

It follows from Theorem 1 that, we only need to focus on the optimization of (5) in the following discussion. The difficulty in the optimization of (5) lies in the large number of groups that may overlap. In practice, many groups will be zero, thus achieving a sparse solution (a sparse solution is desirable in many applications). However, the zero groups are not known in advance. The key question we aim to address is how we can identify as many zero groups as possible to reduce the complexity of the optimization. Next, we present a sufficient condition for a group to be zero.

**Lemma 2.** Denote the minimizer of \( h_{\lambda_2}(\cdot) \) in (5) by \( x^* \). If the \( i \)-th group satisfies \( \|u_{G_i}\| \leq \lambda_2 w_i \), then \( x^*_{G_i} = 0 \), i.e., the \( i \)-th group is zero.

Proof. We decompose \( h_{\lambda_2}(x) \) into two parts as follows:

\[
h_{\lambda_2}(x) = \left( \frac{1}{2} \|x_{G_i} - u_{G_i}\|^2 + \lambda_2 w_i \|x_{G_i}\| \right) + \left( \frac{1}{2} \|x_{\bar{G}_i} - u_{\bar{G}_i}\|^2 + \lambda_2 \sum_{j \neq i} w_j \|x_{G_j}\| \right),
\]

where \( \bar{G}_i = \{1, 2, \ldots, p\} - G_i \) is the complementary set of \( G_i \). We consider the minimization of \( h_{\lambda_2}(x) \) in terms of \( x_{G_i} \) when \( x_{\bar{G}_i} = x_{\bar{G}_i}^* \) is fixed. It can be verified that if \( \|u_{G_i}\| \leq \lambda_2 w_i \), then \( x^*_{G_i} = 0 \) minimizes both terms in (9) simultaneously. Thus we have \( x^*_{G_i} = 0 \). \( \Box \)

Lemma 2 may not identify many true zero groups due to the strong condition imposed. The lemma below weakens the condition in Lemma 2. Intuitively, for a group \( G_i \), we first identify all existing zero groups that overlap with \( G_i \), and then compute the overlapping index subset \( S_i \) of \( G_i \) as:

\[
S_i = \bigcup_{j \neq i, x^*_{G_j} = 0} (G_j \cap G_i).
\]

We can show that \( x^*_{G_i} = 0 \) if \( \|u_{G_i, -S_i}\| \leq \lambda_2 w_i \) is satisfied. Note that this condition is much weaker than the condition in Lemma 2 which requires that \( \|u_{G_i}\| \leq \lambda_2 w_i \).

**Lemma 3.** Denote the minimizer of \( h_{\lambda_2}(\cdot) \) by \( x^* \). Let \( S_i \), a subset of \( G_i \), be defined in (10). If \( \|u_{G_i, -S_i}\| \leq \lambda_2 w_i \) holds, then \( x^*_{G_i} = 0 \).

Proof. Suppose that we have identified a collection of zero groups. By removing these groups, the original problem (5) can then be reduced to:

\[
\min_{x_{\{f_1\}} \in \mathbb{R}^{|f_1|}} \left( \frac{1}{2} \|x_{\{I_1\}} - u_{\{I_1\}}\|^2 + \lambda_2 \sum_{i \in \{I_1\} \setminus \{G_i\}} w_i \|x_{\{G_i, -S_i\}}\| \right)
\]

where \( I_1 \) is the reduced index set, i.e., \( I_1 = \{1, 2, \ldots, p\} - \bigcup_{i \setminus x^*_{G_i} = 0} G_i \), and \( G_1 = \{i : x^*_{G_i} \neq 0\} \) is the index set of the remaining non-zero groups. Note that \( \forall i \in G_1, G_i \cap S_i \in I_1 \). By applying Lemma 2 again, we show that if \( \|u_{G_i, -S_i}\| \leq \lambda_2 w_i \) holds, then \( x^*_{G_i, -S_i} = 0 \). Thus, \( x^*_{G_i} = 0 \). \( \Box \)

Lemma 3 naturally leads to an iterative procedure for identifying the zero groups: For each group \( G_i \), if \( \|u_{G_i}\| \leq \lambda_2 w_i \), then we set \( u_{G_i} = 0 \); we cycle through all groups repeatedly until \( u \) does not change. Let \( p^* = |\{u_i : u_i \neq 0\}| \) be the number of nonzero elements in \( u \), \( g^* = |\{u_{G_i} : u_{G_i} \neq 0\}| \) be the number of the nonzero groups, and \( x^* \) denote the minimizer of \( h_{\lambda_2}(\cdot) \). It follows from Lemma 2 and Lemma 1 that, if \( u_i = 0 \), then \( x_i^* = 0 \). Therefore, by applying the above iterative procedure, we can find the minimizer of (5) by solving a reduced problem that has \( p^* \leq p \) variables and \( g^* \leq g \) groups. With some abuse of notation, we still use (5) to denote the resulting reduced problem. In addition, from Lemma 1 we only focus on \( u > 0 \) in the following discussion, and the analysis can be easily generalized to the general \( u \).
3.2 Reformulation as an Equivalent Smooth Convex Optimization Problem

It follows from the first two properties of Lemma 1 that, we can rewrite (5) as:

\[ \pi^0_{\lambda_2}(u) = \arg \min_{x \in \mathbb{R}^p} \max_{0 \leq x \leq u} h_{\lambda_2}(x), \]  
(11)

where \( \leq \) denotes the element-wise inequality, and

\[ h_{\lambda_2}(x) = \frac{1}{2} \|x - u\|^2 + \lambda_2 \sum_{i=1}^g w_i \|x_{G_i}\|, \]

and the minimizer of \( h_{\lambda_2}(\cdot) \) is constrained to be non-negative due to \( u > 0 \) (refer to the discussion at the end of Section 3).

Making use of the dual norm of the Euclidean norm \( \| \cdot \| \), we can rewrite \( h_{\lambda_2}(x) \) as:

\[ h_{\lambda_2}(x) = \max_{Y \in \Omega} \frac{1}{2} \|x - u\|^2 + \sum_{i=1}^g \langle x, Y^i \rangle, \]  
(12)

where \( \Omega \) is defined as follows:

\[ \Omega = \{ Y \in \mathbb{R}^{p \times g} : Y^i_{G_i} = 0, \|Y^i\| \leq \lambda_2 w_i, i = 1, 2, \ldots, g \}. \]

\( G_i \) is the complementary set of \( G_i \). \( Y \) is a sparse matrix satisfying \( Y^i_{ij} = 0 \) if the \( i \)-th feature does not belong to the \( j \)-th group, i.e., \( i \notin G_j \), and \( Y^i \) denotes the \( i \)-th column of \( Y \). As a result, we can reformulate (11) as the following min-max problem:

\[ \min_{x \in \mathbb{R}^p} \max_{0 \leq x \leq u} \left\{ \psi(x, Y) = \frac{1}{2} \|x - u\|^2 + \langle x, Ye \rangle \right\}, \]  
(13)

where \( e \in \mathbb{R}^g \) is a vector of ones. It is easy to verify that \( \psi(x, Y) \) is convex in \( x \) and concave in \( Y \), and the constraint sets are closed convex for both \( x \) and \( Y \). Thus, (13) has a saddle point, and the min-max can be exchanged.

It is easy to verify that for a given \( Y \), the optimal \( x \) minimizing \( \psi(x, Y) \) in (13) is given by:

\[ x = \max(u - Ye, 0). \]  
(14)

Plugging (14) into (13), we obtain the following minimization problem with regard to \( Y \):

\[ \min_{Y \in \mathbb{R}^{p \times g}} \{ \omega(Y) = -\psi(\max(u - Ye, 0), Y) \}. \]  
(15)

Our methodology for minimizing \( h_{\lambda_2}(\cdot) \) defined in (3) is to first solve (15), and then construct the solution to \( h_{\lambda_2}(\cdot) \) via (14). Using standard optimization techniques, we can show that the function \( \omega(\cdot) \) is continuously differentiable with Lipschitz continuous gradient. We include the detailed proof in the supplemental material for completeness. Therefore, we convert the non-smooth problem (11) to the smooth problem (15), making the smooth convex optimization tools applicable. In this paper, we employ the accelerated gradient descent to solve (15), due to its fast convergence property. Note that, the Euclidean projection onto the set \( \Omega \) can be computed in closed form. We would like to emphasize here that, the problem (15) may have a much smaller size than (4).

3.2.1 Computing the Duality Gap

We show how to estimate the duality gap of the min-max problem (13), which can be used to check the quality of the solution and determine the convergence of the algorithm.

For any given approximate solution \( \tilde{Y} \in \Omega \) for \( \omega(Y) \), we can construct the approximate solution \( \tilde{x} = \max(u - \tilde{Y} e, 0) \) for \( h_{\lambda_2}(x) \). The duality gap for the min-max problem (13) at the point \((\tilde{x}, \tilde{Y})\) can be computed as:

\[ \text{gap}(\tilde{Y}) = \max_{Y \in \Omega} \psi(\tilde{x}, Y) - \min_{0 \leq x \leq u} \psi(x, \tilde{Y}). \]  
(16)

The main result of this subsection is summarized in the following theorem:
Theorem 2. Let $\text{gap}(\hat{Y})$ be the duality gap defined in (16). Then, the following holds:

$$\text{gap}(\hat{Y}) = \lambda_2 \sum_{i=1}^{g} \left( w_i \| \hat{x}_{G_i} \| - \langle \hat{x}_{G_i}, \hat{Y}_{G_i} \rangle \right).$$

(17)

In addition, we have

$$\omega(\hat{Y}) - \omega(Y^*) \leq \text{gap}(\hat{Y}),$$

(18)

$$h(\bar{x}) - h(x^*) \leq \text{gap}(\hat{Y}).$$

(19)

Proof. Denote $(x^*, Y^*)$ as the optimal solution to the problem (13). From (12)-(15), we have

$$- \omega(\hat{Y}) = \psi(\hat{x}, \hat{Y}) = \min_{x \in \mathbb{R}^p, 0 \preceq x \preceq u} \psi(x, \hat{Y}) \leq \psi(x^*, \hat{Y}),$$

(20)

$$\psi(x^*, \hat{Y}) \leq \max_{Y \in \Omega} \psi(x^*, Y) = \psi(x^*, Y^*) = - \omega(Y^*),$$

(21)

$$h_{\lambda_2}(\bar{x}) = \psi(\bar{x}, Y^*) = \min_{x \in \mathbb{R}^p, 0 \preceq x \preceq u} \psi(x, Y^*) \leq \psi(\bar{x}, Y^*),$$

(22)

$$\psi(\bar{x}, Y^*) \leq \max_{Y \in \Omega} \psi(\bar{x}, Y) = h_{\lambda_2}(\bar{x}).$$

(23)

Incorporating (11), (20)-(23), we prove (17)-(19). □

In our experiments, we terminate the algorithm when the estimated duality gap is less than $10^{-10}$.

3.3 Proximal Splitting Methods

Recently, a family of proximal splitting methods [8] have been proposed for converting a challenging optimization problem into a series of sub-problems with a closed form solution. We consider two reformulations of the proximal operator (4), based on the Dykstra-like Proximal Splitting Method and the alternating direction method of multipliers (ADMM). The efficiency of these two methods for overlapping Group Lasso will be demonstrated in the next section.

In [5], Boyd et al. suggested that the original overlapping group lasso problem (1) can be reformulated and solved by ADMM directly. We include the implementation of ADMM for our comparative study. We provide the details of all three reformulations in the supplemental materials.

4 Experiments

In this section, extensive experiments are performed to demonstrate the efficiency of our proposed methods. We use both synthetic data sets and a real world data set and the evaluation is done in various problem size and precision settings. The proposed algorithms are mainly implemented in Matlab, with the proximal operator implemented in standard C for improved efficiency. Several state-of-the-art methods are also included for comparison purpose, including SLasso algorithm developed by Jenatton et al. [13], the ADMM reformulation [5], the Prox-Grad method by Chen et al. [6] and the Picard-Nesterov algorithm by Argyriou et al. [1].

4.1 Synthetic Data

In the first set of simulation we consider only the key component of our algorithm, the proximal operator. The group indices are predefined such that $G_1 = \{1, 2, \ldots, 10\}$, $G_2 = \{6, 7, \ldots, 20\}$, ..., with each group overlapping half of the previous group. 100 examples are generated for each set of fixed problem size $p$ and group size $g$, and the results are summarized in Figure[1] As we can observe from the figure, the dual formulation yields the best performance, followed closely by ADMM and then the Dykstra method. We can also observe that our method scales very well to high dimensional problems, since even with $p = 10^5$, the proximal operator can be computed in a few seconds. It is also not surprising that Dykstra method is much more sensitive to the number of groups, which equals to the number of projections in one Dykstra step.

To illustrate the effectiveness of our pre-processing technique, we repeat the previous experiment by removing the pre-processing step. The results are shown in the right plot of Figure[1]. As we can observe from the figure, the proposed pre-processing technique effectively reduces the computational
and we set $w_i = \sqrt{G_i}$, and $\lambda_1 = \lambda_2 = \gamma \times \lambda_i^{\text{max}}$, where $|G_i|$ denotes the size of the $i$-th group $G_i$, $\lambda_i^{\text{max}} = \|A^T b\|_\infty$ (the zero point is a solution to (1) if $\lambda_1 \geq \lambda_i^{\text{max}}$), and $\gamma$ is chosen from the set $\{5 \times 10^{-1}, 2 \times 10^{-1}, 1 \times 10^{-1}, 5 \times 10^{-2}, 2 \times 10^{-2}, 1 \times 10^{-2}, 5 \times 10^{-3}, 2 \times 10^{-3}, 1 \times 10^{-3}\}$.

Comparison with SLasso, Prox-Grad and ADMM We first compare our proposed FoGLasso with the SLasso algorithm [13], ADMM [5] and Prox-Grad [6]. The comparisons are based on the computational time, since all these methods have efficient Matlab implementations with key components written in C. For a given $\gamma$, we first run SLasso till a certain precision level is reached, and then run the others until they achieve an objective function value smaller than or equal to that of SLasso. Different precision levels of the solutions are evaluated such that a fair comparison can be made. We vary the number of genes involved, and report the total computational time (seconds) including all nine regularization parameters in Figure 2. We can observe that: 1) for all precision levels, our proposed FoGLasso is much more efficient than SLasso, ADMM and Prox-Grad; 2) the advantage of FoGLasso over other three methods in efficiency grows with the increasing number of genes (variables). For example, with the grouping by pathways, FoGLasso is about 25 and 70 times faster than SLasso for 1000 and 2000 genes, respectively; and 3) the efficiency on edges is inferior to that on pathways, due to the larger number of overlapping groups. Additional scalability study of our proposed method using larger problem size can be found in the supplemental materials.

Comparison with Picard-Nesterov Since the code acquired for Picard-Nesterov is implemented purely in Matlab, a computational time comparison might not be fair. Therefore, only the number of iterations required for convergence is reported, as both methods adopt the first order method. We use edges to generate the groups, and vary the problem size from 100 to 400, using the same set of regularization parameters. For each problem, we record both the number of outer iterations (the gradient steps) and the total number of inner iterations (the steps required for computing the
Figure 2: Comparison of SLasso \([13]\), ADMM \([5]\), Prox-Grad \([6]\) and our proposed FoGLasso algorithm in terms of computational time (in seconds and in the logarithmic scale) when different numbers of genes (variables) are involved. Different precision levels are used for comparison.

Table 1: Comparison of FoGLasso and Picard-Nesterov using different numbers \((p)\) of genes and various precision levels. For each particular method, the first row denotes the number of outer iterations required for convergence, while the second row represents the total number of inner iterations.

<table>
<thead>
<tr>
<th>Precision Level</th>
<th>(10^{-2})</th>
<th>(10^{-4})</th>
<th>(10^{-6})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FoGLasso</td>
<td>81</td>
<td>189</td>
<td>353</td>
</tr>
<tr>
<td></td>
<td>288</td>
<td>401</td>
<td>921</td>
</tr>
<tr>
<td>Picard-Nesterov</td>
<td>78</td>
<td>176</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>8271</td>
<td>6.8e4</td>
<td>2.2e5</td>
</tr>
</tbody>
</table>

proximal operators). The average number of iterations among all the regularization parameters are summarized in Table 1. As we can observe from the table, though Picard-Nesterov method often takes less outer iterations to converge, it takes a lot more inner iterations to compute the proximal operator. It is straightforward to verify that the inner iterations in Picard-Nesterov method and our proposed method have the same complexity of \(O(pg)\).

5 Conclusion

In this paper, we consider the efficient optimization of the overlapping group Lasso penalized problem based on the accelerated gradient descent method. We reveal several key properties of the proximal operator associated with the overlapping group Lasso, and compute the proximal operator via solving the smooth and convex dual problem. Numerical experiments on both synthetic and the breast cancer data set demonstrate the efficiency of the proposed algorithm. Although with an inexact proximal operator, the optimal convergence rate of the accelerated gradient descent might not be guaranteed \([23, 11]\), the algorithm performs quite well empirically. A theoretical analysis on the convergence property will be an interesting future direction. In the future, we also plan to apply the proposed algorithm to other real-world applications involving overlapping groups.

Acknowledgments

This work was supported by NSF IIS-0812551, IIS-0953662, MCB-1026710, CCF-1025177, NGA HM1582-08-1-0016, and NSFC 60905035, 61035003.
References


