Efficient Methods for Overlapping Group Lasso

Authored by:

Jieping Ye
Jun Liu
Lei Yuan

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 Paper Body

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 ?1 norm of the estimator, known as Lasso [25]. The ?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 ?1 norm and the ?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, 12, 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 1

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 ?? norms, which, however, is not applicable to the overlapping group Lasso defined as the sum of the ?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 nonsmooth 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 prepossessing 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: k ? k denotes the Euclidean norm, and 0 denotes a vector of zeros. SGN(?) and sgn(?) are defined in a component wise fashion as: 1) if t = 0, then SGN(t) = [?1, 1] and sgn(t) = 0; 2) if t ¿ 0, then SGN(t) = {1} and sgn(t) = 1; and 3) if t ¡ 0, SGN(t) = {?1} and sgn(t) = ?1. Gi ? {1, 2, . . . , p} denotes an index set, and xGi denote a sub-vector of x restricted to Gi .

2

The Overlapping Group Lasso

We consider the following overlapping group Lasso penalized problem: min f (x) = l(x) + ???12 (x)

(1)

x?Rp

where l(?) is a smooth convex loss function, e.g., the least squares loss, ???12 (x) = ?1 kxk1 + ?2

g X i=1

wi kxGi k

(2)

is the overlapping group Lasso penalty, ?1 ? 0 and ?2 ? 0 are regularization parameters, wi ¿ 0, i = 1, 2, . . . , g, Gi ? {1, 2, . . . , p} contains the indices corresponding to the i-th group of features, and k ? k 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 [29]. When the groups are disjoint with ?1 = 0 and ?2 ¿ 0, the model in (1) reduces to the group Lasso [28]. If ?1 ¿ 0 and ?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 (?) at the point x as: fL,x (y) = [l(x) + hl? (x), y ? xi] + ???21 (y) +

L ky ? xk2 , 2

(3)

where L ¿ 0. The model fL,x (y) consists of the first-order Taylor expansion of the smooth function l(?) at the point x, the non-smooth penalty ???21 (x), and a regularization term L2 ky ? xk2 . Next, a sequence of approximate solutions {xi } is computed as follows: xi+1 = arg miny fLi ,si (y), where the search point si is an affine combination of xi?1 and xi as si = xi +?i (xi ?xi?1), for a properly chosen coefficient ?i , Li is determined by the line search according to the Armijo-Goldstein rule 2

so that Li should be appropriate for si , i.e., f (xi+1) ? fLi ,si (xi+1). A key building block in FoGLasso is the minimization of (3), whose solution is known as the proximal operator [20]. The computation of the proximal operator is the main technical contribution of this paper. The pseudocode of FoGLasso is summarized in Algorithm 1, where the proximal operator ?(?) 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: L0 ¿ 0, x0 , k Output: xk+1 1: Initialize x1 = x0 , ??1 = 0, ?0 = 1, and L = L0 . 2: for i = 1 to k do i?2 ?1 3:

Set ?i = ?? , si = xi + ?i (xi ? xi?1) i?1 4: Find the smallest L = 2j Li?1 , j = 0, 1, . . . such that f (xi+1) ? fL,si (xi+1) holds, where ? /L xi+1 = ??21/L (si ? L1 l? (si)) ? 1+ 1+4?2i 5: Set Li = L and ?i+1 = 2 6: 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:

1 ???21 (v) = arg minp g??21 (x) ? kx ? vk2 + ???12 (x) , (4) x?R 2

which is a special case of (1) by setting l(x) = 12 kx ? vk2 . It can be verified that the approximate ? /L solution xi+1 = arg miny fLi ,si (y) is given by xi+1 = ??21/Lii (si ? L1i l? (si)). Recently, it has been shown in [14] that, the efficient computation of the proximal operator is key to many sparse learning algorithms. Next, we focus on the efficient computation of ???21 (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 ???21 (v). Lemma 1. Suppose that ?1 , ?2 ? 0, and wi ¿ 0, for i = 1, 2, . . . , g. Let x? = ???21 (v). The following holds: 1) if vi ¿ 0, then 0 ? x?i ? vi ; 2) if vi ¡ 0, then vi ? x?i ? 0; 3) if vi = 0, then x?i = 0; 4) SGN(v) ? SGN(x?); and 5) ???21 (v) = sgn(v) ? ???21 (—v—). Proof. When ?1 , ?2 ? 0, and wi ? 0, for i = 1, 2, . . . , g, the objective function g??21 (?) is strictly convex, thus x? is the unique minimizer. We first show if vi ¿ 0, then 0 ? x?i ? vi . If x?i ¿ vi , then we can construct a x ? as follows: x ?j = x?j , j 6= i and x ?i = vi . Similarly, if x?i ¡ 0, then ? we can construct a x ? as follows: x ?j = xj , j 6= i and x ?i = 0. It is easy to verify that 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 SGN(?) and the first three properties. Next, we show that ???21 (?) can be directly derived from ??0 2 (?) by soft-thresholding. Thus, we only need to focus on the case when ?1 = 0. This simplifies the optimization in (4). It is an extension of the result for Fused Lasso in [10]. Theorem 1. Let u = sgn(v) ? max(—v— ? ?1 , 0), and ( ) g X 1 0 2 ??2 (u) = arg minp h?2 (x) ? kx ? uk + ?2 (5) wi kxGi k . x?R 2 i=1 Then, the following holds: ???21 (v) = ??0 2 (u).

3

Proof. Denote the unique minimizer of h?2 (?) as x? . The sufficient and necessary condition for the optimality of x? is: 0 ? ?h?2 (x?) = x? ? u + ??0?2 (x?), (6) where ?h?2 (x) and ??0?2 (x) are the sub-differential sets of h?2 (?) and ?0?2 (?) at x, respectively. Next, we need to show 0 ? ?g??21 (x?). The sub-differential of g??21 (?) at x? is given by

?g??21 (x?) = x? ? v + ????12 (x?) = x? ? v + ?1 SGN(x?) + ??0?2 (x?).

(7)

It follows from the definition of u that u ? v ? ?1 SGN(u). Using the fourth property in Lemma 1, we have SGN(u) ? SGN(x?). Thus, u ? v ? ?1 SGN(x?). (8) It follows from (6)-(8) that 0 ? ?g??21 (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?2 (?) in (5) by x? . If the i-th group satisfies kuGi k ? ?2 wi , then x?Gi = 0, i.e., the i-th group is zero.

Proof. We decompose h?2 (x) into two parts as follows: ? ?

X 1 1 h?2 (x) = kxGi ? uGi k2 + ?2 wi kxGi k + ? kxGi ? uGi k2 + ?2 wj kxGj k? , 2 2

(9)

j6=i

where Gi = {1, 2, . . . , p} ? Gi is the complementary set of Gi . We consider the minimization of h?2 (x) in terms of xGi when xGi = x?G is fixed. It can be verified that if kuGi k ? ?2 wi , then i x?Gi = 0 minimizes both terms in (9) simultaneously. Thus we have x?Gi = 0.

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 Gi , we first identify all existing zero groups that overlap with Gi , and then compute the overlapping index subset Si of Gi as: [Si = (Gj ? Gi). (10) j6=i,x? G =0 j

We can show that x?Gi = 0 if kuGi ?Si k ? ?2 wi is satisfied. Note that this condition is much weaker than the condition in Lemma 2, which requires that kuGi k ? ?2 wi . Lemma 3. Denote the minimizer of h?2 (?) by x? . Let Si , a subset of Gi , be defined in (10). If kuGi ?Si k ? ?2 wi holds, then x?Gi = 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: X 1 wi kxGi ?Si k min kx(I1) ? u(I1)k2 + ?2 x(I1)?R—I1 — 2 i?G1 S where I1 is the reduced index set, i.e., I1 = {1, 2, . . . , p} ? i:x? =0 Gi , and G1 = {i : x?Gi 6= 0} Gi is the index set of the remaining non-zero groups. Note that ?i ? G1 , Gi ? Si ? I1 . By applying Lemma 2 again, we show that if kuGi ?Si k ? ?2 wi holds, then x?Gi ?Si = 0. Thus, x?Gi = 0.

Lemma 3 naturally leads to an iterative procedure for identifying the zero groups: For each group Gi , if kuGi k ? ?2 wi , then we set uGi = 0; we cycle through all groups repeatedly until u does not change. Let p? = —{ui : ui 6= 0}— be the number of nonzero elements in u, g ? = —{uGi : uGi 6= 0}— be the number of the nonzero groups, and x? denote the minimizer of h?2 (?). It follows from Lemma 3 and Lemma 1 that, if ui = 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? ? p variables and g ? ? 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. 4

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: ??0 2 (u) = arg minp h?2 (x),

(11)

x?R 0xu

where denotes the element-wise inequality, and

g X 1 2 h?2 (x) = kx ? uk + ?2 wi kxGi k, 2 i=1

and the minimizer of h?2 (?) is constrained to be non-negative due to u ¿ 0 (refer to the discussion at the end of Section 3.1). Making use of the dual norm of the Euclidean norm k ? k, we can rewrite h?2 (x) as: g X 1 h?2 (x) = max kx ? uk2 + hx, Y i i, Y ?? 2 i=1

(12)

where ? is defined as follows:

? = {Y ? Rp?g : YGi i = 0, kY i k ? ?2 wi , i = 1, 2, . . . , g}, Gi is the complementary set of Gi , Y is a sparse matrix satisfying Yij = 0 if the i-th feature does not belong to the j-th group, i.e., i 6? Gj , and Y i denotes the i-th column of Y . As a result, we can reformulate (11) as the following min-max problem:

1 2 min max ?(x, Y) = kx ? uk + hx, Y ei , (13) Y ?? x?Rp 2 0xu

where e ? Rg is a vector of ones. It is easy to verify that ?(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 ?(x, Y) in (13) is given by x = max(u ? Y e, 0).

(14)

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

Y ?Rp?g :Y ??

{?(Y) = ??(max(u ? Y e, 0), Y)} .

(15)

Our methodology for minimizing h?2 (?) defined in (5) is to first solve (15), and then construct the solution to h?2 (?) via (14). Using standard optimization techniques, we can show that the function ?(?) 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 ? 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 Y? ? ? for ?(Y), we can construct the approximate solution x, Y?) x ? = max(u ? Y? e, 0) for h?2 (x). The duality gap for the min-max problem (13) at the point (? can be computed as: (16) gap(Y?) = max ?(? x, Y) ? minp ?(x, Y?). Y ??

x?R 0xu

The main result of this subsection is summarized in the following theorem: 5

Theorem 2. Let gap(Y?) be the duality gap defined in (16). Then, the following holds: gap(Y?) = ?2

g X xGi , Y?Gi i i). (wi k? xGi k ? h?

(17)

i=1

In addition, we have

?(Y?) ? ?(Y ?) ? gap(Y?), h(? x) ? h(x?) ? gap(Y?).

(18) (19)

Proof. Denote (x? , Y ?) as the optimal solution to the problem (13). From (12)-(15), we have ? ?(Y?) = ?(? x, Y?) = minp ?(x, Y?) ? ?(x? , Y?),

(20)

?(x? , Y?) ? max ?(x? , Y) = ?(x? , Y ?) = ??(Y ?),

(21)

x?R 0xu

Y ?? ?

?

?

?

?

h?2 (x) = ?(x , Y) = minp ?(x, Y) ? ?(? x, Y),

(22)

x). ?(? x, Y ?) ? max ?(? x, Y) = h?2 (?

(23)

x?R 0xu

Y ??

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 G1 = {1, 2, . . . , 10}, G2 = {6, 7, . . . , 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 = 106 , 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 6

1
2
0
10
0
CPU Time
10
?2
10
10 Dual ADMM Dykstra
0
CPU Time
Dual ADMM Dykstra CPU Time
2
10
10
?1

10
?2
10
Dual ADMM Dual?no?pre ADMM?no?pre
?2
10
10 ?4
10
1e3
?3
1e4
1e5
1e6
10
10
?4
20
50 g
p
100
200
10
1e3
1e4
1e5
1e6
p

Figure 1: Time comparison for computing the proximal operators. The group number is fixed in the left figure and the problem size is fixed in the middle figure. In the right figure, the effectiveness of the pre-processing technique is illustrated. time. As is evident from Figure 1, the dual formulation proposed in Section 3.2 consistently outperforms other proximal splitting methods. In the following experiments, only the dual method will be used for computing the proximal operator, and our method will then be called as ?FoGLasso?. 4.2

Gene Expression Data

We have also conducted experiments to evaluate the efficiency of the proposed algorithm using the breast cancer gene expression data set [26], which consists of 8,141 genes in 295 breast cancer tumors (78 metastatic and 217 non-metastatic). For the sake of analyzing microarrays in terms of biologically meaningful gene sets, different approaches have been used to organize the genes into (overlapping) gene sets. In our experiments, we follow [12] and employ the following two approaches for generating the overlapping gene sets (groups): pathways [24] and edges [7]. For pathways, the canonical pathways from the Molecular Signatures Database (MSigDB) [24] are used. It contains 639 groups of genes, of which 637 groups involve the genes in our study. The statistics of the 637 gene groups are summarized as follows: the average number of genes in

each group is 23.7, the largest gene group has 213 genes, and 3,510 genes appear in these 637 groups with an average appearance frequency of about 4. For edges, the network built in [7] will be used, and we follow [12] to extract 42,594 edges from the network, leading to 42,594 overlapping gene sets of size 2. All 8,141 genes appear in the 42,594 groups with an average appearance frequency of about 10. The experimental settings are as follows: we solve (1) with the least squares loss l(x) = 21 kAx ? bk2 , p , where —Gi — denotes the size of the i-th group and we set wi = —Gi —, and ?1 = ?2 = ? ? ?max 1 T), and ? is chosen from the = kA bk (the zero point is a solution to (1) if ?1 ? ?max Gi , ?max ? 1 1 set {5 ? 10?1 , 2 ? 10?1 , 1 ? 10?1 , 5 ? 10?2 , 2 ? 10?2 , 1 ? 10?2 , 5 ? 10?3 , 2 ? 10?3 , 1 ? 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 ?, 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 7

Edges with Precision 1e?02
4
2
10
1
10
0
Pathways with Precision 1e?02
4
CPU Time
3

10
1
10
1
10
?1
10
100 200 300 400 500 1000 1500 2000 Number of involved genes Pathways with Precision 1e?04
100 200 300 400 500 1000 1500 2000 Number of involved genes 4
Pathways with Precision 1e?06
10 FoGLasso ADMM SLasso Prox?Grad
3
10
2
10
1
10
FoGLasso ADMM SLasso Prox?Grad
2
10
1
0
10
10
0
10
?1
10
2
10
0
10 FoGLasso ADMM SLasso Prox?Grad
Edges with Precision 1e?06 FoGLasso ADMM SLasso Prox?Grad
10
?1
CPU Time
2
1
10
10
100 200 300 400 500 1000 1500 2000 Number of involved genes
10
10
2
10

10
?1
3
3
10
0
10 10
4
10 FoGLasso ADMM SLasso Prox?Grad CPU Time
CPU Time
3
10 CPU Time
3
10
Edges with Precision 1e?04
10 FoGLasso ADMM SLasso Prox?Grad
CPU Time
4
10
0
?1
100 200 300 400 500 1000 1500 2000 Number of involved genes
10
100 200 300 400 500 1000 1500 2000 Number of involved genes
10
100 200 300 400 500 1000 1500 2000 Number of involved genes

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. Precision Level 10?2 10?4 10?6 p 100 200 400 100 200 400 100 200 400 81 189 353 192 371 1299 334 507 1796 FoGLasso 288 401 921 404 590 1912 547 727 2387 78 176 325 181 304 1028 318 504 1431 Picard-Nesterov 8271 6.8e4 2.2e5 2.6e4 1.0e5 7.8e5 5.1e4 1.3e5 1.1e6

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 straight forward 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. 8

2 References

[1] A. Argyriou, C.A. Micchelli, M. Pontil, L. Shen, and Y. Xu. Efficient first order methods for linear composite regularizers. Arxiv preprint arXiv:1104.1436, 2011. [2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2(1):183?202, 2009. [3] H. D. Bondell and B. J. Reich. Simultaneous regression shrinkage, variable selection and clustering of predictors with oscar. Biometrics, 64:115?123, 2008. [4] J. F. Bonnans and A. Shapiro. Optimization problems with perturbations: A guided tour. SIAM Review, 40(2):228?264, 1998. [5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. 2010. [6] X. Chen, Q. Lin, S. Kim, J.G. Carbonell, and E.P. Xing. An efficient proximal gradient method for general structured sparse learning. Arxiv preprint arXiv:1005.4717, 2010. [7] H. Y. Chuang, E. Lee, Y. T. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. Molecular Systems Biology, 3(140), 2007. [8] P.L. Combettes and J.C. Pesquet. Proximal splitting methods in signal processing. Arxiv preprint arXiv:0912.3522, 2009. [9] J. M. Danskin. The theory of max-min and its applications to weapons allocation problems. SpringerVerlag, New York, 1967. [10] J. Friedman, T. Hastie, H. H?ofling, and R. Tibshirani. Pathwise coordinate optimization. Annals of Applied Statistics, 1(2):302?332, 2007. [11] B. He and X. Yuan. An accelerated inexact proximal point algorithm for convex minimization. 2010. [12] L. Jacob, G. Obozinski, and J. Vert. Group lasso with overlap and graph lasso. In ICML, 2009. [13] R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, arXiv:0904.3523, 2009. [14] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In ICML, 2010. [15] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In ICML, 2010. [16] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate

descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In ICML, 2009. [17] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient ?2,1 -norm minimization. In UAI, 2009. [18] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In NIPS. 2010. [19] L. Meier, S. Geer, and P. B?uhlmann. The group lasso for logistic regression. Journal of the Royal Statistical Society: Series B, 70:53?71, 2008. [20] J.-J. Moreau. Proximit?e et dualit?e dans un espace hilbertien. Bull. Soc. Math. France, 93:273?299, 1965. [21] A. Nemirovski. Efficient methods in convex programming. Lecture Notes, 1994. [22] Y. Nesterov. Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, 2004. [23] R.T. Rockafellar. Monotone operators and the proximal point algorithm. SIAM Journal on Control and Optimization, 14:877, 1976. [24] A. Subramanian and et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences, 102(43):15545? 15550, 2005. [25] R. Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B, 58(1):267?288, 1996. [26] M. J. Van de Vijver and et al. A gene-expression signature as a predictor of survival in breast cancer. The New England Journal of Medicine, 347(25):1999?2009, 2002. [27] Y. Ying, C. Campbell, and M. Girolami. Analysis of svm with indefinite kernels. In NIPS. 2009. [28] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. Journal Of The Royal Statistical Society Series B, 68(1):49?67, 2006. [29] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. Annals of Statistics, 37(6A):3468?3497, 2009.

9