Feature Grouping and Selection Over an Undirected Graph

Sen Yang[1], Lei Yuan[1,2], Ying-Cheng Lai[3], Xiaotong Shen[4], Peter Wonka[1], Jieping Ye[1,2]
[1]Computer Science and Engineering, [3]Electrical Engineering,
[2]Center for Evolutionary Medicine and Informatics, The Biodesign Institute,
Arizona State University, Tempe, AZ 85287, USA
{senyang, lei.yuan, ying-cheng.lai, peter.wonka, jieping.ye}@asu.edu
[4]School of Statistics, University of Minnesota, Minneapolis, MN 55455, USA
xshen@stat.umn.edu

ABSTRACT

High-dimensional regression/classification continues to be an important and challenging problem, especially when features are highly correlated. Feature selection, combined with additional structure information on the features has been considered to be promising in promoting regression/classification performance. Graph-guided fused lasso (GFlasso) has recently been proposed to facilitate feature selection and graph structure exploitation, when features exhibit certain graph structures. However, the formulation in GFlasso relies on pairwise sample correlations to perform feature grouping, which could introduce additional estimation bias. In this paper, we propose three new feature grouping and selection methods to resolve this issue. The first method employs a convex function to penalize the pairwise l_∞ norm of connected regression/classification coefficients, achieving simultaneous feature grouping and selection. The second method improves the first one by utilizing a non-convex function to reduce the estimation bias. The third one is the extension of the second method using a truncated l_1 regularization to further reduce the estimation bias. The proposed methods combine feature grouping and feature selection to enhance estimation accuracy. We employ the alternating direction method of multipliers (ADMM) and difference of convex functions (DC) programming to solve the proposed formulations. Our experimental results on synthetic data and two real datasets demonstrate the effectiveness of the proposed methods.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms

KDD'12, August 12–16, 2012, Beijing, China.

Keywords

Feature grouping, feature selection, undirected graph, l_1 regularization, regression, classification

1. INTRODUCTION

High-dimensional regression/classification is challenging due to the *curse of dimensionality*. Lasso [18] and its various extensions, which can simultaneously perform feature selection and regression/classification, have received increasing attention in this situation. However, in the presence of highly correlated features lasso tends to only select one of those features resulting in suboptimal performance [27]. Several methods have been proposed to address this issue in the literature. Shen and Ye [15] introduce an adaptive model selection procedure that corrects the estimation bias through a data-driven penalty based on generalized degrees of freedom. The Elastic Net [27] uses an additional l_2 regularizer to encourage highly correlated features to stay together. However, these methods do not incorporate prior knowledge into the regression/classification process, which is critical in many applications. As an example, many biological studies have suggested that genes tend to work in groups according to their biological functions, and there are some regulatory relationships between genes [10]. This biological knowledge can be represented as a graph, where the nodes represent the genes, and the edges imply the regulatory relationships between genes. Therefore, we want to study how estimation accuracy can be improved using dependency information encoded as a graph.

Given feature grouping information, the group lasso [1, 7, 22, 11] yields a solution with grouped sparsity using l_1/l_2 penalty. The orignal group lasso does not consider the overlaps between groups. Zhao et al. [24] extend the group lasso to the case of overlapping groups. Jacob et al. [7] introduce a new penalty function leading to a grouped sparse solution with overlapping groups. Yuan et al. [21] propose an efficient method to solve the overlapping group lasso. Other extensions of group lasso with tree structured regularization include [11, 8]. Prior works have demonstrated the benefit of using feature grouping information for high-dimensional regression/classification. However, these methods need the feature groups to be pre-specified. In other words, they only utilize the grouping information to obtain solutions with grouped sparsity, but lack the capability of identifying groups.

There are also a number of existing methods for feature grouping. Fused lasso [19] introduces an l_1 regularization

method for estimating subgroups in a certain serial order, but pre-ordering features is required before using fused lasso. A study about parameter estimation of the fused lasso can be found in [12]; Shen et al. [13] propose a non-convex method to select all possible homogenous subgroups, but it fails to obtain sparse solutions. OSCAR [2] employs an l_1 regularizer and a pairwise l_∞ regularizer to perform feature selection and automatic feature grouping. Li and Li [10] suggest a grouping penalty using a Laplacian matrix to force the coefficients to be similar, which can be considered as a graph version of the Elastic Net. When the Laplacian matrix is an identity matrix, Laplacian lasso [10, 5] is identical to the Elastic Net. GFlasso employs an l_1 regularization over a graph, which penalizes the difference $|\beta_i - \text{sign}(r_{ij})\beta_j|$, to encourage the coefficients β_i, β_j for features i, j connected by an edge in the graph to be similar when $r_{ij} > 0$, but dissimilar when $r_{ij} < 0$, where r_{ij} is the sample correlation between two features [9]. Although these grouping penalties can improve the performance, they would introduce additional estimation bias due to strict convexity of the penalties or due to possible graph misspecification. For example, additional bias may occur when the signs of coefficients for two features connected by an edge in the graph are different in Laplacian lasso [10, 5], or when the sign of r_{ij} is inaccurate in GFlasso [9].

In this paper, we focus on simultaneous estimation of grouping and sparseness structures over a given undirected graph. Features tend to be grouped when they are connected by an edge in a graph. When features are connected by an edge in a graph, the absolute values of the model coefficients for these two features should be similar or identical. We propose one convex and two non-convex penalties to encourage both sparsity and equality of absolute values of coefficients for connected features. The convex penalty includes a pairwise l_∞ regularizer over a graph. The first non-convex penalty improves the convex penalty by penalizing the difference of absolute values of coefficients for connected features. The other one is the extension of the first non-convex penalty using a truncated l_1 regularization to further reduce the estimation bias. These penalties are designed to resolve the aforementioned issues of Laplacian lasso and GFlasso. The non-convex penalties shrink only small differences in absolute values so that estimation bias can be reduced; several recent works analyze their theoretical properties [26, 14]. Through ADMM and DC programming, we develop computational methods to solve the proposed formulations. The proposed methods can combine the benefit of feature selection and that of feature grouping to improve regression/classification performance. Due to the equality of absolute values of coefficients, the model complexity of the learned model can be reduced. We have performed experiments on synthetic data and two real datasets. The results demonstrate the effectiveness of the proposed methods.

The rest of the paper is organized as follows. We introduce the proposed convex method in Section 2, and the two proposed non-convex methods in Section 3. Experimental results are given in Section 4. We conclude the paper in Section 5.

2. A CONVEX FORMULATION

Consider a linear model in which response y_i depends on a vector of p features:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$ is a vector of coefficients, $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the data matrix, and $\boldsymbol{\varepsilon}$ is random noise. Given an undirected graph, we try to build a prediction model (regression or classification) incorporating the graph structure information to estimate the nonzero coefficients of $\boldsymbol{\beta}$ and to identify the feature groups when the number of features p is larger than the sample size n. Let (N, E) be the given undirected graph, where $N = \{1, 2, \ldots, p\}$ is a set of nodes, and E is the set of edges. Node i corresponds to feature $\mathbf{x}_i$. If nodes i and j are connected by an edge in E, then features $\mathbf{x}_i$ and $\mathbf{x}_j$ tend to be grouped. The formulation of graph OSCAR (GOSCAR) is given by

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{(i,j)\in E} \max\{|\beta_i|, |\beta_j|\} \tag{2}$$

where $\lambda_1,\ \lambda_2$ are regularization parameters. We use a pairwise l_∞ regularizer to encourage the coefficients to be equal [2], but we only put grouping constraints over the nodes connected over the given graph. The l_1 regularizer encourages sparseness. The pairwise l_∞ regularizer puts more penalty on the larger coefficients. Note that $\max\{|\beta_i|, |\beta_j|\}$ can be decomposed as

$$\max\{|\beta_i|, |\beta_j|\} = \frac{1}{2}(|\beta_i + \beta_j| + |\beta_i - \beta_j|).$$

$\frac{1}{2}(|\beta_i + \beta_j| + |\beta_i - \beta_j|)$ can be represented by

$$|\mathbf{u}^T\boldsymbol{\beta}| + |\mathbf{v}^T\boldsymbol{\beta}|,$$

where $\mathbf{u}, \mathbf{v}$ are sparse vectors, each with only two non-zero entries $u_i = u_j = \frac{1}{2},\ v_i = -v_j = \frac{1}{2}$. Thus Eq. (2) can be rewritten in a matrix form as

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\mathbf{T}\boldsymbol{\beta}\|_1, \tag{3}$$

where $\mathbf{T}$ is a sparse matrix constructed from the edge set E.

The proposed formulation is closely related to OSCAR [2]. The penalty of OSCAR is $\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{i<j} \max\{|\beta_i|, |\beta_j|\}$. The l_1 regularizer leads to a sparse solution, and the l_∞ regularizer encourages the coefficients to be equal. OSCAR can be efficiently solved by accelerated gradient methods, whose key projection can be solved by a simple iterative group merging algorithm [25]. However, OSCAR assumes each node is connected to all the other nodes, which is not sufficient for many applications. Note that OSCAR is a special case of GOSCAR when the graph is complete. GOSCAR, incorporating an arbitrary undirected graph, is much more challenging to solve.

2.1 Algorithm

We propose to solve GOSCAR using the alternating direction method of multipliers (ADMM) [3]. ADMM decomposes a large global problem into a series of smaller local subproblems and coordinates the local solutions to identify the globally optimal solution. ADMM attempts to combine the benefits of dual decomposition and augmented Lagrangian methods for constrained optimization [3]. The problem solved by ADMM takes the form of

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z})$$
$$s.t.\ \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}.$$

ADMM uses a variant of the augmented Lagrangian method and reformulates the problem as follows:

$$L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}) = f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\mu}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2,$$

with $\boldsymbol{\mu}$ being the augmented Lagrangian multiplier, and ρ being the non-negative dual update step length. ADMM solves this problem by iteratively minimizing $L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu})$ over $\mathbf{x}$, $\mathbf{z}$, and $\boldsymbol{\mu}$. The update rule for ADMM is given by

$$\begin{aligned}\mathbf{x}^{k+1} &:= \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^k, \boldsymbol{\mu}^k),\\ \mathbf{z}^{k+1} &:= \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \boldsymbol{\mu}^k),\\ \boldsymbol{\mu}^{k+1} &:= \boldsymbol{\mu}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}).\end{aligned}$$

Consider the unconstrained optimization problem in Eq. (3), which is equivalent to the following constrained optimization problem:

$$\begin{aligned}&\min_{\boldsymbol{\beta},\mathbf{q},\mathbf{p}} \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\mathbf{q}\|_1 + \lambda_2\|\mathbf{p}\|_1\\ &s.t. \quad \boldsymbol{\beta} - \mathbf{q} = \mathbf{0},\ \mathbf{T}\boldsymbol{\beta} - \mathbf{p} = \mathbf{0},\end{aligned} \tag{4}$$

where $\mathbf{q}, \mathbf{p}$ are slack variables. Eq. (4) can then be solved by ADMM. The augmented Lagrangian is

$$\begin{aligned}L_\rho(\boldsymbol{\beta}, \mathbf{q}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\upsilon}) &= \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\mathbf{q}\|_1 + \lambda_2\|\mathbf{p}\|_1\\ &+ \boldsymbol{\mu}^T(\boldsymbol{\beta} - \mathbf{q}) + \boldsymbol{\upsilon}^T(\mathbf{T}\boldsymbol{\beta} - \mathbf{p}) + \tfrac{\rho}{2}\|\boldsymbol{\beta} - \mathbf{q}\|^2 + \tfrac{\rho}{2}\|\mathbf{T}\boldsymbol{\beta} - \mathbf{p}\|^2,\end{aligned}$$

where $\boldsymbol{\mu}, \boldsymbol{\upsilon}$ are augmented Lagrangian multipliers.

Update $\boldsymbol{\beta}$: In the $(k+1)$-th iteration, $\boldsymbol{\beta}^{k+1}$ can be updated by minimizing L_ρ with $\mathbf{q}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\upsilon}$ fixed:

$$\begin{aligned}\boldsymbol{\beta}^{k+1} = \arg\min_{\boldsymbol{\beta}} &\tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + (\boldsymbol{\mu}^k + \mathbf{T}^T\boldsymbol{\upsilon}^k)^T\boldsymbol{\beta}\\ &+ \tfrac{\rho}{2}\|\boldsymbol{\beta} - \mathbf{q}^k\|^2 + \tfrac{\rho}{2}\|\mathbf{T}\boldsymbol{\beta} - \mathbf{p}^k\|^2.\end{aligned} \tag{5}$$

The above optimization problem is quadratic. The optimal solution is given by $\boldsymbol{\beta}^{k+1} = \mathbf{F}^{-1}\mathbf{b}^k$, where

$$\begin{aligned}\mathbf{F} &= \mathbf{X}^T\mathbf{X} + \rho(\mathbf{I} + \mathbf{T}^T\mathbf{T}),\\ \mathbf{b}^k &= \mathbf{X}^T\mathbf{y} - \boldsymbol{\mu}^k - \mathbf{T}^T\boldsymbol{\upsilon}^k + \rho\mathbf{T}^T\mathbf{p}^k + \rho\mathbf{q}^k.\end{aligned}$$

The computation of $\boldsymbol{\beta}^{k+1}$ involves solving a linear system, which is the most time-consuming part in the whole algorithm. To compute $\boldsymbol{\beta}^{k+1}$ efficiently, we compute the Cholesky factorization of $\mathbf{F}$ at the beginning of the algorithm:

$$\mathbf{F} = \mathbf{R}^T\mathbf{R}.$$

Note that $\mathbf{F}$ is a constant and positive definite matrix. Using the Cholesky factorization we only need to solve the following two linear systems at each iteration:

$$\mathbf{R}^T\hat{\boldsymbol{\beta}} = \mathbf{b}^k,\ \mathbf{R}\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}. \tag{6}$$

Since $\mathbf{R}$ is an upper triangular matrix, solving these two linear systems is very efficient.

Update $\mathbf{q}$: $\mathbf{q}^{k+1}$ can be obtained by solving

$$\mathbf{q}^{k+1} = \arg\min_{\mathbf{q}} \frac{\rho}{2}\|\mathbf{q} - \boldsymbol{\beta}^{k+1}\|^2 + \lambda_1\|\mathbf{q}\|_1 - (\boldsymbol{\mu}^k)^T\mathbf{q}$$

which is equivalent to the following problem:

$$\mathbf{q}^{k+1} = \arg\min_{\mathbf{q}} \frac{1}{2}\|\mathbf{q} - \boldsymbol{\beta}^{k+1} - \frac{1}{\rho}\boldsymbol{\mu}^k\|^2 + \frac{\lambda_1}{\rho}\|\mathbf{q}\|_1 \tag{7}$$

Eq. (7) has a closed-form solution, known as *soft-thresholding*:

$$\mathbf{q}^{k+1} = S_{\lambda_1/\rho}(\boldsymbol{\beta}^{k+1} + \frac{1}{\rho}\boldsymbol{\mu}^k), \tag{8}$$

where the *soft-thresholding operator* is defined as:

$$S_\lambda(x) = \text{sign}(x)\max(|x| - \lambda, 0).$$

Update $\mathbf{p}$: Similar to updating $\mathbf{q}$, $\mathbf{p}^{k+1}$ can also be obtained by *soft-thresholding*:

$$\mathbf{p}_i^{k+1} = S_{\lambda_2/\rho}(\mathbf{T}\boldsymbol{\beta}^{k+1} + \frac{1}{\rho}\boldsymbol{\upsilon}^k). \tag{9}$$

Update $\boldsymbol{\mu}, \boldsymbol{\upsilon}$:

$$\begin{aligned}\boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \rho(\boldsymbol{\beta}^{k+1} - \mathbf{q}^{k+1}),\\ \boldsymbol{\upsilon}^{k+1} &= \boldsymbol{\upsilon}^k + \rho(\mathbf{T}\boldsymbol{\beta}^{k+1} - \mathbf{p}^{k+1}).\end{aligned} \tag{10}$$

A summary of GOSCAR is shown in Algorithm 1.

Algorithm 1: The GOSCAR algorithm

Input: $\mathbf{X}, \mathbf{y}, E, \lambda_1, \lambda_2, \rho$
Output: $\boldsymbol{\beta}$
Initialization: $\mathbf{p}^0 \leftarrow \mathbf{0}, \mathbf{q}^0 \leftarrow \mathbf{0}, \boldsymbol{\mu}^0 \leftarrow \mathbf{0}, \boldsymbol{\upsilon}^0 \leftarrow \mathbf{0}$;
Compute the Cholesky factorization of $\mathbf{F}$;
do
  Compute $\boldsymbol{\beta}^{k+1}$ according to Eq. (6).
  Compute $\mathbf{q}^{k+1}$ according to Eq. (8).
  Compute $\mathbf{p}^{k+1}$ according to Eq. (9).
  Compute $\boldsymbol{\mu}^{k+1}, \boldsymbol{\upsilon}^{k+1}$ according to Eq. (10).
Until *Convergence*;
return $\boldsymbol{\beta}$;

In Algorithm 1, the Cholesky factorization only needs to be computed once, and each iteration involves solving one linear system and two soft-thresholding operations. The time complexity of the soft-thresholding operation in Eq. (8) is $O(p)$. The other one in Eq. (9) involves a matrix-vector multiplication. Due to the sparsity of $\mathbf{T}$, its time complexity is $O(n_e)$, where n_e is the number of edges. Solving the linear system involves computing $\mathbf{b}^k$ and solving Eq. (6), whose total time complexity is $O(p(p+n) + n_e)$. Thus the time complexity of each iteration is $O(p(p+n) + n_e)$.

3. TWO NON-CONVEX FORMULATIONS

The grouping penalty of GOSCAR overcomes the limitation of Laplacian lasso that the different signs of coefficients can introduce additional penalty. However, under the l_∞ regularizer, even if $|\beta_i|$ and $|\beta_j|$ are close to each other, the penalty on this pair may still be large due to the property of the max operator, resulting in the coefficient β_i or β_j being over penalized. The additional penalty would result in biased estimation, especially for large coefficient, as in the lasso case [18]. Another related grouping penalty is GFlasso, $|\beta_i - \text{sign}(r_{ij})\beta_j|$, where r_{ij} is the pairwise sample correlation. GFlasso relies on the pairwise sample correlation to decide whether β_i and β_j are enforced to be close or not. When the pairwise sample correlation wrongly estimates the sign between β_i and β_j, an additional penalty on β_i and β_j would occur, introducing estimation bias. This motivates our non-convex grouping penalty, $||\beta_i| - |\beta_j||$, that shrinks only small differences in absolutes values. As a result, estimation bias is reduced as compared to these convex grouping penalties. The proposed non-convex methods perform well even when the graph is wrongly specified, unlike GFlasso. Note that the proposed non-convex grouping penalty does not assume the sign of an edge is given; it only relies on the graph structure.

3.1 Non-Convex Formulation I: ncFGS

The proposed non-convex formulation (ncFGS) solves the following optimization problem:

$$\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{(i,j)\in E} ||\beta_i| - |\beta_j||, \tag{11}$$

where the grouping penalty $\sum_{(i,j)\in E} ||\beta_i| - |\beta_j||$ controls only magnitudes of differences of coefficients ignoring their signs over the graph. Through the l_1 regularizer and grouping penalty, simultaneous feature grouping and selection are performed, where only large coefficients as well as pairwise differences are shrunk.

A computational method for the non-convex optimization in Eq. (11) is through DC programming. We will first give a brief review of DC programming.

A particular DC program on $\mathbb{R}^p$ takes the form of

$$f(\boldsymbol{\beta}) = f_1(\boldsymbol{\beta}) - f_2(\boldsymbol{\beta})$$

with $f_1(\boldsymbol{\beta})$ and $f_2(\boldsymbol{\beta})$ being convex on $\mathbb{R}^p$. Algorithms to solve DC programming based on the duality and local optimality conditions have been introduced in [17]. Due to their local characteristic and the non-convexity of DC programming, these algorithms cannot guarantee the computed solution to be globally optimal. In general, these DC algorithms converge to a local solution, but some researchers observed that they converge quite often to a global one [16].

To apply DC programming to our problem we need to decompose the objective function into the difference of two convex functions. We propose to use:

$$\begin{aligned} f_1(\boldsymbol{\beta}) &= \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2 \textstyle\sum_{(i,j)\in E}(|\beta_i + \beta_j| \\ &\quad + |\beta_i - \beta_j|), \\ f_2(\boldsymbol{\beta}) &= \lambda_2 \textstyle\sum_{(i,j)\in E}(|\beta_i| + |\beta_j|). \end{aligned}$$

The above DC decomposition is based on the following identity: $||\beta_i| - |\beta_j|| = |\beta_i + \beta_j| + |\beta_i - \beta_j| - (|\beta_i| + |\beta_j|)$. Note that both $f_1(\boldsymbol{\beta})$ and $f_2(\boldsymbol{\beta})$ are convex functions.

Denote $f_2^k(\boldsymbol{\beta}) = f_2(\boldsymbol{\beta}^k) + \langle \boldsymbol{\beta} - \boldsymbol{\beta}^k, \partial f_2(\boldsymbol{\beta}^k)\rangle$ as the affine minorization of $f_2(\boldsymbol{\beta})$, where $\langle\cdot,\cdot\rangle$ is the inner product. Then DC programming solves Eq. (11) by iteratively solving a sub-problem as follows:

$$\min_{\beta} f_1(\boldsymbol{\beta}) - f_2^k(\boldsymbol{\beta}). \tag{12}$$

Since $\langle \boldsymbol{\beta}^k, \partial f_2(\boldsymbol{\beta}^k)\rangle$ is constant, Eq. (12) can be rewritten as

$$\min_{\boldsymbol{\beta}} f_1(\boldsymbol{\beta}) - \langle \boldsymbol{\beta}, \partial f_2(\boldsymbol{\beta}^k)\rangle. \tag{13}$$

Let $\mathbf{c}^k = \partial f_2(\boldsymbol{\beta}^k)$. Note that

$$c_i^k = \lambda_2 d_i \text{sign}(\beta_i^k)\mathbb{I}(\beta_i^k \neq 0), \tag{14}$$

where d_i is the degree of node i, and $\mathbb{I}(\cdot)$ is the indicator function. Hence, the formulation in Eq. (13) is

$$\begin{aligned} \min_{\boldsymbol{\beta}} &\tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 - (\mathbf{c}^k)^T\boldsymbol{\beta} \\ &+ \lambda_2 \textstyle\sum_{(i,j)\in E}(|\beta_i + \beta_j| + |\beta_i - \beta_j|), \end{aligned} \tag{15}$$

which is convex. Note that the only differences between the problems in Eq. (2) and Eq. (15) are the linear term $(\mathbf{c}^k)^T\boldsymbol{\beta}$ and the second regularization parameter. Similar to GOSCAR, we can solve Eq. (15) using ADMM, which is equivalent to the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\mathbf{q},\mathbf{p}} \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 - (\mathbf{c}^k)^T\boldsymbol{\beta} + \lambda_1\|\mathbf{q}\|_1 + 2\lambda_2\|\mathbf{p}\|_1 \\ &s.t \quad \boldsymbol{\beta} - \mathbf{q} = \mathbf{0},\ \mathbf{T}\boldsymbol{\beta} - \mathbf{p} = \mathbf{0}. \end{aligned} \tag{16}$$

There is an additional linear term $(\mathbf{c}^k)^T\boldsymbol{\beta}$ in updating $\boldsymbol{\beta}$ compared to Algorithm 1. Hence, we can use Algorithm 1 to solve Eq. (15) with a small change in updating $\boldsymbol{\beta}$:

$$\mathbf{F}\boldsymbol{\beta} - \mathbf{b}^s - \mathbf{c}^k = 0.$$

where s represents the iteration number in Algorithm 1.

The key steps of ncFGS are shown in Algorithm 2.

Algorithm 2: The ncFGS algorithm

Input: $\mathbf{X}, \mathbf{y}, E, \lambda_1, \lambda_2, \epsilon$
Output: $\boldsymbol{\beta}$
Initialization: $\boldsymbol{\beta}^0 \leftarrow \mathbf{0}$;
while $f(\boldsymbol{\beta}^k) - f(\boldsymbol{\beta}^{k+1}) > \epsilon$ **do**
  Compute $\mathbf{c}^k$ according to Eq. (14).
  Compute $\boldsymbol{\beta}^{k+1}$ using Algorithm 1 with $\mathbf{c}^k$ and $\lambda_1, 2\lambda_2$ as regularization parameters.
end
return $\boldsymbol{\beta}$;

3.2 Non-Convex Formulation II: ncTFGS

It is known that the bias of lasso is due to the looseness of convex relaxation of l_0 regularization. The truncated l_1 regularizer, a non-convex regularizer close to the l_0 regularizer, has been proposed to resolve the bias issue [23]. The truncated l_1 regularizer can recover the exact set of nonzero coefficients under a weaker condition, and has a smaller upper error bound than lasso [23]. Therefore, we propose a truncated grouping penalty to further reduce the estimation bias. The proposed formulation based on the truncated grouping penalty is

$$\min_{\boldsymbol{\beta}} f_T(\boldsymbol{\beta}) = \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 p_1(\boldsymbol{\beta}) + \lambda_2 p_2(\boldsymbol{\beta}) \tag{17}$$

where

$$\begin{aligned} p_1(\boldsymbol{\beta}) &= \textstyle\sum_i J_\tau(|\beta_i|), \\ p_2(\boldsymbol{\beta}) &= \textstyle\sum_{(i,j)\in E} J_\tau(||\beta_i| - |\beta_j||), \end{aligned}$$

and $J_\tau(x) = \min(\frac{x}{\tau}, 1)$ is the truncated l_1 regularizer, a surrogate of the l_0 function; τ is a non-negative tuning parameter. Figure 1 shows the difference between l_0 norm, l_1 norm and $J_\tau(|x|)$. When $\tau \to 0$, $J_\tau(|x|)$ is equivalent to the l_0 norm given by the number of nonzero entries of a vector. When $\tau \geq |x|$, $\tau J_\tau(|x|)$ is equivalent to the l_1 norm of x.

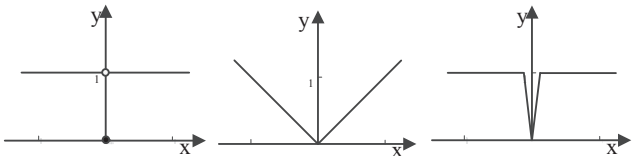


Figure 1: Example for l_0 norm (left), l_1 norm (middle), and $J_\tau(|x|)$ with $\tau = \frac{1}{8}$ (right).

Note that $J_\tau(||\beta_i| - |\beta_j||)$ can be decomposed as

$$\begin{aligned} J_\tau(||\beta_i| - |\beta_j||) &= \tfrac{1}{\tau}(|\beta_i + \beta_j| + |\beta_i - \beta_j|) \\ &\quad - \tfrac{1}{\tau}\max(2|\beta_i| - \tau, 2|\beta_j| - \tau, |\beta_i| + |\beta_j|), \end{aligned}$$

and a DC decomposition of $J_\tau(|\beta_i|)$ is

$$J_\tau(|\beta_i|) = \frac{1}{\tau}|\beta_i| - \frac{1}{\tau}\max(|\beta_i| - \tau, 0).$$

Hence, the DC decomposition of $f_T(\boldsymbol{\beta})$ can be written as

$$f_T(\boldsymbol{\beta}) = f_{T,1}(\boldsymbol{\beta}) - f_{T,2}(\boldsymbol{\beta}),$$

where

$$f_{T,1}(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \frac{\lambda_1}{\tau}\|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{\tau}\sum_{(i,j)\in E}(|\beta_i+\beta_j| + |\beta_i-\beta_j|),$$
$$f_{T,2}(\boldsymbol{\beta}) = \frac{\lambda_1}{\tau}\sum_i \max(|\beta_i|-\tau,0) + \frac{\lambda_2}{\tau}\sum_{(i,j)\in E}\max(2|\beta_i| - \tau, 2|\beta_j|-\tau, |\beta_i|+|\beta_j|).$$

Let $\mathbf{c}_T^k = \partial f_{T,2}(\boldsymbol{\beta}^k)$ be the subgradient of $f_{T,2}$ in the $(k+1)$-th iteration. We have

$$c_{T,i}^k = \text{sign}(\beta_i^k)\big(\frac{\lambda_1}{\tau}\mathbb{I}(|\beta_i^k|>\tau) + \frac{\lambda_2}{\tau}\sum_{j:(i,j)\in E} (2\mathbb{I}(|\beta_j^k|<|\beta_i^k|-\tau) + \mathbb{I}(||\beta_i^k|-|\beta_j^k||<\tau))\big). \quad (18)$$

Then the subproblem of ncTFGS is

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \frac{\lambda_1}{\tau}\|\boldsymbol{\beta}\|_1 - (\mathbf{c}_T^k)^T\boldsymbol{\beta} + \frac{\lambda_2}{\tau}\sum_{(i,j)\in E}(|\beta_i+\beta_j|+|\beta_i-\beta_j|), \quad (19)$$

which can be solved using Algorithm 1 as in ncFGS.

The key steps of ncTFGS are summarized in Algorithm 3.

Algorithm 3: The ncTFGS algorithm

Input: $\mathbf{X}, \mathbf{y}, E, \lambda_1, \lambda_2, \tau, \epsilon$
Output: $\boldsymbol{\beta}$
Initialization: $\boldsymbol{\beta}^0 \leftarrow \mathbf{0}$;
while $f(\boldsymbol{\beta}^k) - f(\boldsymbol{\beta}^{k+1}) > \epsilon$ **do**
  Compute $\mathbf{c}_T^k$ according to Eq. (18).
  Compute $\boldsymbol{\beta}^{k+1}$ using Algorithm 1 with $\mathbf{c}_T^k$ and $\frac{\lambda_1}{\tau}, \frac{2\lambda_2}{\tau}$ as regularization parameters.
end
return $\boldsymbol{\beta}$;

ncTFGS is an extension of ncFGS. When $\tau \geq |\beta_i|, \forall i$, ncTFGS with regularization parameters $\tau\lambda_1$ and $\tau\lambda_2$ is identical to ncFGS (see Figure 3). ncFGS and ncTFGS have the same time complexity. The subproblems of ncFGS and ncTFGS are solved by Algorithm 1. In our experiments, we observed ncFGS and ncTFGS usually converge in less than 10 iterations.

4. NUMERICAL RESULTS

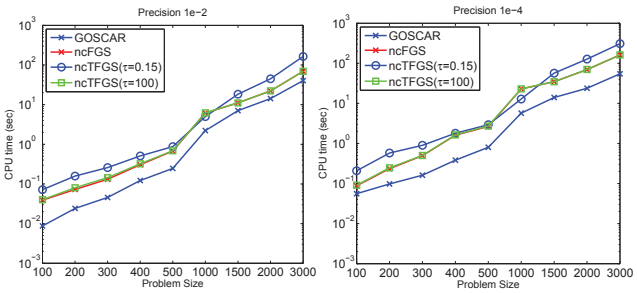
We examine the performance of the proposed methods and compare them against lasso, GFlasso, and OSCAR on synthetic datasets and two real datasets: FDG-PET images[1] and Breast Cancer[2]. The experiments are performed on a PC with dual-core Intel 3.0GHz CPU and 4GB memory. The code is written in MATLAB. The algorithms and their associated penalties are:

- Lasso: $\lambda_1\|\boldsymbol{\beta}\|_1$;
- OSCAR: $\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\sum_{i<j}\max\{|\beta_i|,|\beta_j|\}$;
- GFlasso: $\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\sum_{(i,j)\in E}|\beta_i - \text{sign}(r_{ij})\beta_j|$;
- GOSCAR: $\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\sum_{(i,j)\in E}\max\{|\beta_i|,|\beta_j|\}$;
- ncFGS: $\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\sum_{(i,j)\in E}||\beta_i|-|\beta_j||$;
- ncTFGS: $\lambda_1\sum_i J_\tau(|\beta_i|) + \lambda_2\sum_{(i,j)\in E}J_\tau(||\beta_i|-|\beta_j||)$;

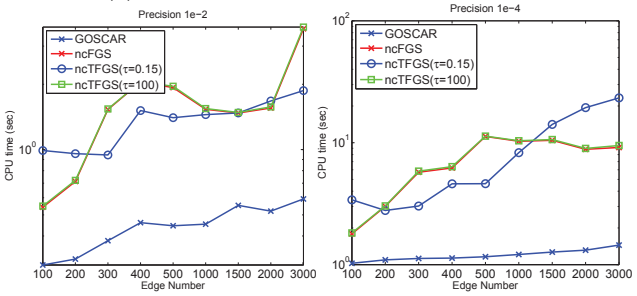
[1] http://adni.loni.ucla.edu/
[2] http://cbio.ensmp.fr/~jvert/publi/

4.1 Efficiency

To evaluate the efficiency of the proposed methods, we conduct experiments on a synthetic dataset with a sample size of 100 and dimensions varying from 100 to 3000. The regression model is $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_{p\times p})$, $\beta_i \sim \mathcal{N}(0,1)$, and $\varepsilon_i \sim \mathcal{N}(0, 0.01^2)$. The graph is randomly generated. The number of edges n_e varies from 100 to 3000. The regularization parameters are set as $\lambda_1 = \lambda_2 = 0.8\max\{|\beta_i|\}$ with n_e fixed. Since the graph size affects the penalty, λ_1 and λ_2 are scaled by $\frac{1}{n_e}$ to avoid trivial solutions with dimension p fixed. The average computational time based on 30 repetitions is reported in Figure 2. As can be seen in Figure 2, GOSCAR can achieve 1e-4 precision in less than 10s when the dimension and the number of edges are 1000. The computational time of ncTFGS is about 7 times higher than that of GOSCAR in this experiment. The computational time of ncFGS is the same as that of ncTFGS when $\tau = 100$, and very close to that of ncTFGS when $\tau = 0.15$. We can also observe that the proposed methods scale very well to the number of edges. The computational time of the proposed method increases less than 4 times when the number of edges increases from 100 to 3000. It is not surprising because the time complexity of each iteration in Algorithm 1 is linear with respect to n_e, and the sparsity of $\mathbf{T}$ makes the algorithm much more efficient. The increase of dimension is more costly than that of the number of edges, as the complexity of each iteration is quadratic with respect to p.



(a) The number of edges is fixed to 1000.



(b) The dimension is fixed to 500.

Figure 2: Comparison of GOSCAR, ncFGS, ncTFGS ($\tau = 0.15$), and ncTFGS ($\tau = 100$) in terms of computation time with different dimensions, precisions and the numbers of edges (in seconds and in logarithmic scale).

4.2 Simulations

We use five synthetic problems that have been commonly used in the sparse learning literature [2, 10] to compare the performance of different methods. The data is generated

from the regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The five problems are given by:

1. $n = 100$, $p = 40$, and $\sigma = 2, 5, 10$. The true parameter is given by

$$\boldsymbol{\beta} = (\underbrace{0, \ldots, 0}_{10}, \underbrace{2, \ldots, 2}_{10}, \underbrace{0, \ldots, 0}_{10}, \underbrace{2, \ldots, 2}_{10})^T.$$

   $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_{p\times p})$ with $s_{ii} = 1, \forall i$ and $s_{ij} = 0.5$ for $i \neq j$.

2. $n = 50$, $p = 40$, $\boldsymbol{\beta} = (\underbrace{3, \ldots, 3}_{15}, \underbrace{0, \ldots, 0}_{25})^T$, and $\sigma = 2, 5, 10$. The features are generated as

$$\begin{aligned} \mathbf{x}_i &= Z_1 + \varepsilon_i^x, \ Z_1 \sim \mathcal{N}(0,1), \quad i = 1, \ldots, 5 \\ \mathbf{x}_i &= Z_2 + \varepsilon_i^x, \ Z_2 \sim \mathcal{N}(0,1), \quad i = 6, \ldots, 10 \\ \mathbf{x}_i &= Z_3 + \varepsilon_i^x, \ Z_3 \sim \mathcal{N}(0,1), \quad i = 11, \ldots, 15 \\ \mathbf{x}_i &\sim \mathcal{N}(0,1) \quad i = 16, \ldots, 40 \end{aligned}$$

   with $\varepsilon_i^x \sim \mathcal{N}(0, 0.16)$, and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{40}]$.

3. Consider a regulatory gene network [10], where an entire network consists of n_{TF} subnetworks, each with one transcription factor (TF) and its 10 regulatory target genes. The data for each subnetwork can be generated as $\mathbf{X}_i^{TF} \sim \mathcal{N}(0, \mathbf{S}_{11\times 11})$ with $s_{ii} = 1, s_{1i} = s_{i1} = 0.7, \forall i, i \neq 1$ and $s_{ij} = 0$ for $i \neq j, j \neq 1, i \neq 1$. Then $\mathbf{X} = [\mathbf{X}_1^{TF}, \ldots, \mathbf{X}_{n_{TF}}^{TF}]$, $n = 100$, $p = 110$, and $\sigma = 5$. The true parameters are

$$\boldsymbol{\beta} = (\underbrace{\frac{5}{\sqrt{11}}, \ldots, \frac{5}{\sqrt{11}}}_{11}, \underbrace{\frac{-3}{\sqrt{11}}, \ldots, \frac{-3}{\sqrt{11}}}_{11}, \underbrace{0, \ldots, 0}_{p-22})^T.$$

4. Same as 3 except that

$$\boldsymbol{\beta} = (5, \underbrace{\frac{5}{\sqrt{10}}, \ldots, \frac{5}{\sqrt{10}}}_{10}, -3, \underbrace{\frac{-3}{\sqrt{10}}, \ldots, \frac{-3}{\sqrt{10}}}_{10}, \underbrace{0, \ldots, 0}_{p-22})^T$$

5. Same as 3 except that

$$\boldsymbol{\beta} = (5, \underbrace{\frac{5}{\sqrt{10}}, \ldots, \frac{5}{\sqrt{10}}}_{10}, -5, \underbrace{\frac{-5}{\sqrt{10}}, \ldots, \frac{-5}{\sqrt{10}}}_{10}, 3, \underbrace{\frac{3}{\sqrt{10}}, \ldots, \frac{3}{\sqrt{10}}}_{10}, -3, \underbrace{\frac{-3}{\sqrt{10}}, \ldots, \frac{-3}{\sqrt{10}}}_{10}, \underbrace{0, \ldots, 0}_{p-44})^T$$

We assume that the features in the same group are connected in a graph, and those in different groups are not connected. We use MSE to measure the performance of estimation of $\boldsymbol{\beta}$, which is defined as

$$MSE(\boldsymbol{\beta}) = (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^T \mathbf{X}^T \mathbf{X} (\boldsymbol{\beta} - \boldsymbol{\beta}^*).$$

For feature grouping and selection, we introduce two separate metrics to measure the accuracy of feature grouping and selection. Denote $I_i, i = 0, 1, 2, ..., K$ as the index of different groups, where I_0 is the index of zero coefficients. Then the metric for feature selection is defined as

$$s_0 = \frac{\sum_{i\in I_0} \mathbb{I}(\beta_i = 0) + \sum_{i\notin I_0} \mathbb{I}(\beta_i \neq 0)}{p},$$

and the metric for feature grouping is defined as

$$s = \frac{\sum_{i=1}^K s_i + s_0}{K+1},$$

where

$$s_i = \frac{\sum_{i\neq j, i,j\in I_i} \mathbb{I}(|\beta_i| = |\beta_j|) + \sum_{i\neq j, i\in I_i, j\notin I_i} \mathbb{I}(|\beta_i| \neq |\beta_j|)}{|I_i|(p-1)}.$$

s_i measures the grouping accuracy of group i under the assumption that the absolute values of entries in the same group should be the same, but different from those in different groups. s_0 measures the accuracy of feature selection. It is clear that $0 \leq s_0, s_i, s \leq 1$.

For each dataset, we generate n samples for training, as well as n samples for testing. To make the synthetic datasets more challenging, we first randomly select $\lfloor n/2 \rfloor$ coefficients, and change their signs, as well as those of the corresponding features. Denote $\tilde{\boldsymbol{\beta}}$ and $\tilde{\mathbf{X}}$ as the coefficients and features after changing signs. Then $\tilde{\beta}_i = -\beta_i$, $\tilde{\mathbf{x}}_i = -\mathbf{x}_i$, if the i-th coefficient is selected; otherwise, $\tilde{\beta}_i = \beta_i$, $\tilde{\mathbf{x}}_i = \mathbf{x}_i$. So that $\tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} = \mathbf{X}\boldsymbol{\beta}$. We apply different approaches on $\tilde{\mathbf{X}}$. The covariance matrix of $\mathbf{X}$ is used in GFlasso to simulate the graph misspecification. The results of $\boldsymbol{\beta}$ converted from $\tilde{\boldsymbol{\beta}}$ are reported.

Figure 3 shows that ncFGS obtains the same results as ncTFGS on dataset 1 with $\sigma = 2$ when τ is larger than $|\beta_i|$. The regularization parameters are $\tau\lambda_1$ and $\tau\lambda_2$ for ncTFGS, and λ_1 and λ_2 for ncFGS. Figure 4 shows the average nonzero coefficients obtained on dataset 1 with $\sigma = 2$. As can be seen in Figure 4, GOSCAR, ncFGS, and ncTFGS are able to utilize the graph information, and achieve good parameter estimation. Although GFlasso can use the graph information, it performs worse than GOSCAR, ncFGS, and ncTFGS due to the graph misspecification.

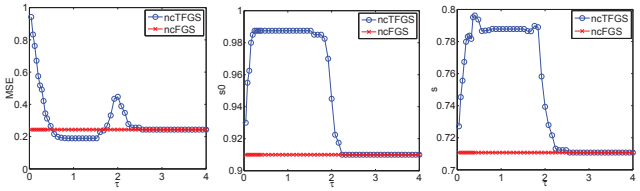


Figure 3: MSEs (left), s_0 (middle), and s (right) of ncFGS and ncTFGS on dataset 1 for fixed λ_1 and λ_2. The regularization parameters for ncTFGS are $\tau\lambda_1$ and $\tau\lambda_2$. τ ranges from 0.04 to 4.

The performance in terms of MSEs averaged over 30 simulations is shown in Table 1. As indicated in Table 1, among existing methods (Lasso, GFlasso, OSCAR), GFlasso is the best, except in the two cases where OSCAR is better. GOSCAR is better than the best existing method in all cases except for two, and ncFGS and ncTFGS outperform all the other methods.

Table 2 shows the results in terms of accuracy of feature grouping and selection. Since Lasso does not perform feature grouping, we only report the results of the other five methods: OSCAR, GFlasso, GOSCAR, ncFGS, and ncTFGS. Table 2 shows that ncFGS and ncTFGS achieve higher accuracy than other methods.

Table 3 shows the comparison of feature selection alone ($\lambda_2 = 0$), feature grouping alone ($\lambda_1 = 0$), and simultaneous feature grouping and selection using ncTFGS. From

Table 1: Comparison of performance in terms of MSEs and estimated standard deviations (in parentheses) for different methods based on 30 simulations on different synthetic datasets.

Datasets	Lasso	OSCAR	GFlasso	GOSCAR	ncFGS	ncTFGS
Data1 ($\sigma = 2$)	1.807(0.331)	1.441(0.318)	1.080(0.276)	0.315(0.157)	0.123(0.075)	**0.116(0.075)**
Data1 ($\sigma = 5$)	5.294(0.983)	5.328(1.080)	3.480(1.072)	1.262(0.764)	0.356(0.395)	**0.356(0.395)**
Data1 ($\sigma = 10$)	12.628(3.931)	13.880(4.031)	13.411(4.540)	6.061(4.022)	1.963(1.600)	**1.958(1.593)**
Data2 ($\sigma = 2$)	1.308(0.435)	1.084(0.439)	0.623(0.250)	0.291(0.208)	0.226(0.175)	**0.223(0.135)**
Data2 ($\sigma = 5$)	4.907(1.496)	4.868(1.625)	2.538(0.656)	0.781(0.598)	0.721(0.532)	**0.705(0.535)**
Data2 ($\sigma = 10$)	18.175(6.611)	18.353(6.611)	6.930(2.858)	4.601(2.623)	4.232(2.561)	**4.196(2.577)**
Data3 ($\sigma = 5$)	5.163(1.708)	4.503(1.677)	4.236(1.476)	3.336(1.725)	0.349(0.282)	**0.348(0.283)**
Data4 ($\sigma = 5$)	7.664(2.502)	7.167(2.492)	7.516(2.441)	7.527(2.434)	5.097(0.780)	**4.943(0.764)**
Data5 ($\sigma = 5$)	9.893(1.965)	7.907(2.194)	9.622(2.025)	9.810(2.068)	7.684(1.1191)	**7.601(1.038)**

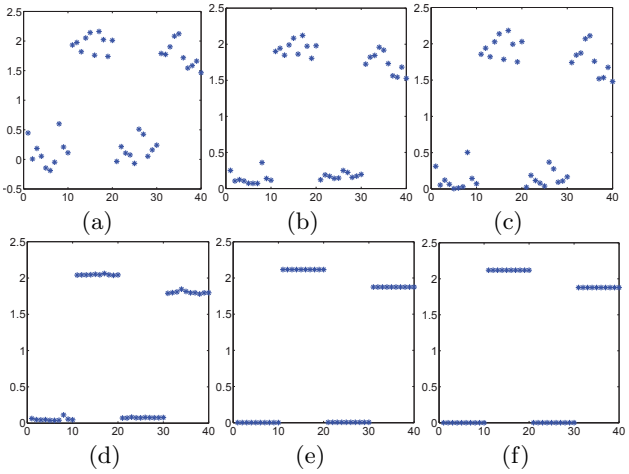


Figure 4: The average nonzero coefficients obtained on dataset 1 with $\sigma = 2$: (a) Lasso; (b) GFlasso; (c) OSCAR; (d) GOSCAR; (e); ncFGS; (f) ncTGS

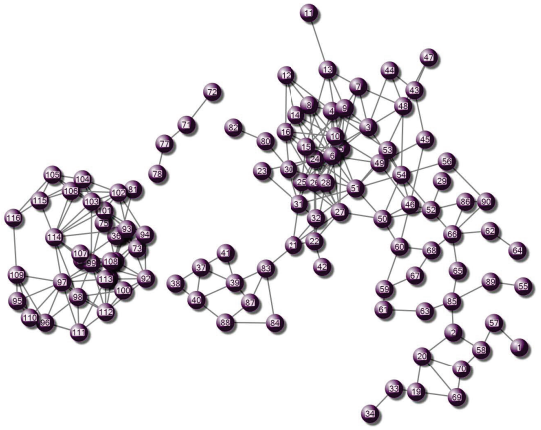


Figure 5: Subgraphs of the graph built by SICE on FDG-PET dataset, which consists of 265 edges.

Table 3, we can observe that simultaneous feature grouping and selection outperforms either feature grouping or feature selection, demonstrating the benefit of joint feature grouping and selection in the proposed non-convex method.

4.3 Real Data

We conduct experiments on two real datasets: FDG-PET and Breast Cancer. The metrics to measure the performance of different algorithms include accuracy (acc.), sensitivity (sen.), specificity (spe.), degrees of freedom (dof.), and the number of nonzero coefficients (nonzero coeff.). The dof. of lasso is the number of nonzero coefficients [18]. For the algorithms capable of feature grouping, we use the same definition of dof. in [2], which is the number of estimated groups.

4.3.1 FDG-PET

In this experiment, we use FDG-PET 3D images from 74 Alzheimer's disease (AD), 172 mild cognitive impairment (MCI), and 81 normal control (NC) subjects downloaded from the Alzheimer's disease neuroimaging initiative (ADNI) database. The different regions of whole brain volume can be represented by 116 anatomical volumes of interest (AVOI), defined by Automated Anatomical Labeling (AAL) [20]. Then we extracted data from each of the 116 AVOIs, and derived average of each AVOI for each subject.

In our study, we compare different methods in distinguishing AD and NC subjects, which is a two-class classification problem over a dataset with 155 samples and 116 features. The dataset is randomly split into two subset, one training set consisting of 104 samples, and one testing set consisting of the remaining 51 samples. The tuning of the parameter is achieved by 5-fold cross validation. Sparse inverse covariance estimation (SICE) has been recognized as an effective tool for identifying the structure of the inverse covariance matrix. We use SICE developed in [6] to model the connectivity of brain regions. Figure 5 shows sample subgraphs built by SICE consisting of 115 nodes and 265 edges.

The results based on 20 replications are shown in Table 4. From Table 4, we can see that ncTFGS achieves more accurate classification while obtaining smaller degrees of freedom. ncFGS and GOSCAR achieve similar classification, while ncFGS selects more features than GOSCAR.

Figure 6 shows the comparison of accuracy with either λ_1 or λ_2 fixed. The λ_1 and λ_2 values range from 1e-4 to 100. As we can see, the performance of ncTFGS is slightly better than that of the other competitors. Since the regularization parameters of subproblems in ncTFGS are $\frac{\lambda_1}{\tau}$ and $\frac{2\lambda_2}{\tau}$, the solution of ncTFGS is more sparse than those of other competitors when λ_1 and λ_2 are large and τ is small ($\tau = 0.15$ in this case).

4.3.2 Breast Cancer

We conduct experiments on the breast cancer dataset, which consists of gene expression data for 8141 genes in 295

Table 2: Accuracy of feature grouping and selection based on 30 simulations for five feature grouping methods: the first row for each dataset corresponds to the accuracy of feature selection; the second row corresponds to the accuracy of feature grouping. The numbers in parentheses are the standard deviations.

Datasets	OSCAR	GFlasso	GOSCAR	ncFGS	ncTFGS
Data1 ($\sigma = 2$)	0.675(0.098)	0.553(0.064)	0.513(0.036)	0.983(0.063)	**1.000(0.000)**
	0.708(0.021)	0.709(0.017)	0.702(0.009)	0.994(0.022)	**1.000(0.000)**
Data1 ($\sigma = 5$)	0.565(0.084)	0.502(0.009)	0.585(0.085)	**1.000(0.000)**	**1.000(0.000)**
	0.691(0.011)	0.709(0.016)	0.708(0.017)	**1.000(0.000)**	**1.000(0.000)**
Data1 ($\sigma = 10$)	0.532(0.069)	0.568(0.088)	0.577(0.061)	0.983(0.063)	**1.000(0.000)**
	0.675(0.031)	0.725(0.022)	0.708(0.020)	0.994(0.022)	**0.999(0.001)**
Data2 ($\sigma = 2$)	0.739(0.108)	0.544(0.272)	**1.000(0.000)**	0.958(0.159)	0.958(0.159)
	0.625(0.052)	0.823(0.029)	0.837(0.014)	0.831(0.052)	**0.846(0.041)**
Data2 ($\sigma = 5$)	0.763(0.114)	0.717(0.275)	**0.999(0.005)**	0.979(0.114)	0.975(0.115)
	0.650(0.066)	0.741(0.062)	0.833(0.011)	**0.845(0.030)**	0.842(0.037)
Data2 ($\sigma = 10$)	0.726(0.101)	0.818(0.149)	0.993(0.024)	**1.000(0.000)**	**1.000(0.000)**
	0.597(0.058)	0.680(0.049)	0.829(0.025)	0.851(0.015)	**0.856(0.014)**
Data3 ($\sigma = 5$)	0.886(0.135)	0.736(0.103)	0.382(0.084)	0.992(0.026)	**0.996(0.014)**
	0.841(0.056)	0.739(0.041)	0.689(0.013)	**0.995(0.017)**	0.978(0.028)
Data4 ($\sigma = 5$)	0.875(0.033)	0.881(0.026)	0.882(0.037)	0.796(0.245)	**0.950(0.012)**
	0.834(0.030)	0.805(0.035)	0.805(0.036)	**0.895(0.114)**	0.890(0.074)
Data5 ($\sigma = 5$)	0.760(0.203)	0.802(0.153)	0.861(0.051)	0.881(0.174)	**0.894(0.132)**
	0.858(0.031)	0.821(0.037)	0.805(0.037)	**0.920(0.056)**	0.919(0.057)

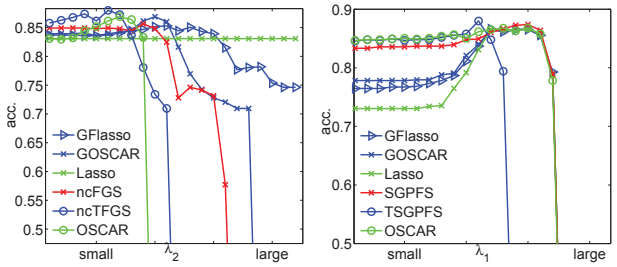
Table 4: Comparison of classification accuracy, sensitivity, specificity, degrees of freedom, and the number of nonzero coefficients averaged over 20 replications for different methods on FDG-PET dataset.

Metrics	Lasso	OSCAR	GFlasso	GOSCAR	ncFGS	ncTFGS
acc.	0.886(0.026)	0.891(0.026)	0.901(0.029)	0.909(0.026)	0.909(0.031)	**0.920(0.024)**
sen.	0.870(0.041)	0.876(0.038)	0.904(0.038)	0.909(0.041)	0.915(0.043)	**0.933(0.047)**
pec.	0.913(0.0446)	**0.917(0.050)**	0.902(0.046)	0.915(0.046)	0.908(0.047)	0.915(0.052)
dof.	22.150	29.150	24.350	21.300	31.250	**18.250**
nonzero coeff.	22.150	38.000	41.900	24.250	37.350	**19.350**

Table 3: Comparison of feature selection alone (FS), feature grouping alone (FG), and simultaneous feature grouping and feature selection (Both). The average results based on 30 replications of three datasets with $\sigma = 5$: Data3 (top), Data4 (middle), and Data5 (bottom) are reported. The numbers in parentheses are the standard deviations.

Meth.	MSE	s_0	s
FG	2.774(0.967)	0.252(0.156)	0.696(0.006)
FS	6.005(1.410)	0.945(0.012)	0.773(0.037)
Both	**0.348(0.283)**	**0.996(0.014)**	**0.978(0.028)**
FG	9.4930(1.810)	0.613(0.115)	0.770(0.038)
FS	6.437(1.803)	0.947(0.016)	0.782(0.046)
Both	**4.944(0.764)**	**0.951(0.166)**	**0.890(0.074)**
FG	10.830(2.161)	0.434(0.043)	0.847(0.014)
FS	10.276(1.438)	0.891(0.018)	0.768(0.026)
Both	**7.601(1.038)**	**0.894(0.132)**	**0.919(0.057)**

breast cancer tumors (78 metastatic and 217 non-metastatic). The network described in [4] is used as the input graph in this experiment. Figure 7 shows a subgraph consisting of 80 nodes of the used graph. We restrict our analysis to the 566 genes most correlated to the output, but also connected in the graph. 2/3 data is randomly chosen as training data, and the remaining 1/3 data is used as testing data. The tuning parameter is estimated by 5-fold cross validation. Table 5 shows the results averaged over 30 replications. As indicated in Table 5, GOSCAR, ncFGS and ncTFGS outperform the other three methods, and ncTFGS achieves the best performance.



(a) λ_1 fixed (b) λ_2 fixed

Figure 6: Comparison of accuracies for various methods with λ_1 fixed (left) and λ_2 fixed (right) on FDT-PET dataset.

5. CONCLUSION

In this paper, we consider simultaneous feature grouping and selection over a given undirected graph. We propose one convex and two non-convex penalties to encourage both sparsity and equality of absolute values of coefficients for features connected in the graph. We employ ADMM and DC programming to solve the proposed formulations. Numerical experiments on synthetic and real data demonstrate the effectiveness of the proposed methods. Our results also demonstrate the benefit of simultaneous feature grouping and feature selection through the proposed non-convex

Table 5: Comparison of classification accuracy, sensitivity, specificity, degrees of freedom, and the number of nonzero coefficients averaged over 30 replications for various methods on Breast Cancer dataset.

Metrics	Lasso	OSCAR	GFlasso	GOSCAR	ncFGS	ncTFGS
acc.	0.739(0.054)	0.755(0.055)	0.771(0.050)	0.783(0.042)	0.779(0.041)	**0.790(0.036)**
sen.	0.707(0.056)	0.720(0.060)	0.749(0.060)	0.755(0.050)	0.755(0.055)	**0.762(0.044)**
pec.	0.794(0.071)	0.810(0.068)	0.805(0.056)	0.827(0.061)	0.819(0.058)	**0.834(0.060)**
dof.	239.267	165.633	108.633	70.267	57.233	**45.600**
nonzero coeff.	239.267	243.867	144.867	140.667	**79.833**	116.567

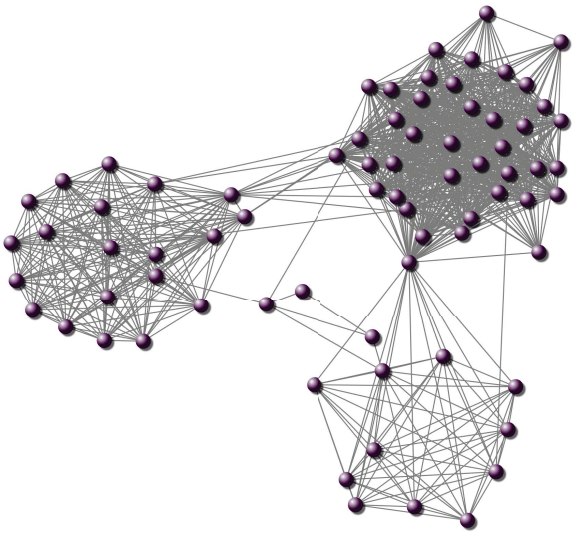


Figure 7: A subgraph of the network in Breast Cancer dataset [4]. The subgraph consists of 80 nodes.

methods. In this paper, we focus on undirected graphs. A possible future direction is to extend the formulations to directed graphs. In addition, we plan to study the generalization performance of the proposed formulations.

Acknowledgments

This work was supported in part by NSF (IIS-0953662, MCB-1026710, CCF-1025177, DMS-0906616) and NIH (R01LM010 730, 2R01GM081535-01, R01HL105397).

6. REFERENCES

[1] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

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

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers.* 2011.

[4] H. Chuang, E. Lee, Y. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. *Molecular systems biology*, 3(1), 2007.

[5] H. Fei, B. Quanz, and J. Huan. Regularization and feature selection for networked features. In *CIKM*, pages 1893–1896, 2010.

[6] S. Huang, J. Li, L. Sun, J. Liu, T. Wu, K. Chen, A. Fleisher, E. Reiman, and J. Ye. Learning brain connectivity of Alzheimer's disease from neuroimaging data. *NIPS*, 22:808–816, 2009.

[7] L. Jacob, G. Obozinski, and J. Vert. Group lasso with overlap and graph lasso. In *ICML*, pages 433–440, 2009.

[8] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, 2010.

[9] S. Kim and E. Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS genetics*, 5(8):e1000587, 2009.

[10] C. Li and H. Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, 2008.

[11] J. Liu and J. Ye. Moreau-Yosida regularization for grouped tree structure learning. *NIPS*, 2010.

[12] A. Rinaldo. Properties and refinements of the fused lasso. *The Annals of Statistics*, 37(5B):2922–2952, 2009.

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

[14] X. Shen, H. Huang, and W. Pan. Simultaneous supervised clustering and feature selection over a graph. *Submitted*.

[15] X. Shen and J. Ye. Adaptive model selection. *Journal of the American Statistical Association*, 97(457):210–221, 2002.

[16] P. Tao and L. An. Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Math. Vietnam*, 22(1):289–355, 1997.

[17] P. Tao and S. El Bernoussi. Duality in DC (difference of convex functions) optimization. subgradient methods. *Trends in Mathematical Optimization*, 84:277–293, 1988.

[18] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, pages 267–288, 1996.

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

[20] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *Neuroimage*, 15(1):273–289, 2002.

[21] L. Yuan, J. Liu, and J. Ye. Efficient methods for overlapping group lasso. *NIPS*, 2011.

[22] 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.

[23] T. Zhang. Multi-stage convex relaxation for feature selection. *stat*, 1050:3, 2011.

[24] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497, 2009.

[25] L. Zhong and J. Kwok. Efficient sparse modeling with automatic feature grouping. *ICML*, 2011.

[26] Y. Zhu, X. Shen, and W. Pan. Simultaneous grouping pursuit and feature selection in regression over an undirected graph. *Submitted*.

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