

Feature Grouping and Selection Over an Undirected Graph

Sen Yang, Lei Yuan, Ying-Cheng Lai, Xiaotong Shen, Peter Wonka, and Jieping Ye

1 Introduction

High-dimensional regression/classification is challenging due to the *curse of dimensionality*. Lasso [18] and its various extensions [10], 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 [25]. 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 [25] 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 [9]. 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, 6, 11, 21] yields a solution with grouped sparsity using l_1/l_2 penalty. The original group lasso does

S. Yang • L. Yuan • Y.-C. Lai • P. Wonka • J. Ye (✉)

Arizona State University, Tempe, AZ 85287, USA

e-mail: senyang@asu.edu; lei.yuan@asu.edu; ying-cheng.lai@asu.edu; peter.wonka@asu.edu
jieping.ye@asu.edu

X. Shen

University of Minnesota, Minneapolis, MN 55455, USA

e-mail: xshen@stat.umn.edu

not consider the overlaps between groups. Zhao et al. [22] extend the group lasso to the case of overlapping groups. Jacob et al. [6] introduce a new penalty function leading to a grouped sparse solution with overlapping groups. Yuan et al. [20] propose an efficient method to solve the overlapping group lasso. Other extensions of group lasso with tree structured regularization include [7, 11]. 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 [9] 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 [5, 9] 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 [8]. 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 [5, 9], or when the sign of r_{ij} is inaccurate in GFlasso [8].

In this chapter, 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 a convex and non-convex penalty 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 non-convex penalty improves the convex penalty by penalizing the difference of absolute values of coefficients for connected features. These penalties are designed to resolve the aforementioned issues of Laplacian lasso and GFlasso. Several recent works analyze their theoretical properties [14, 24]. 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 a real dataset. The results demonstrate the effectiveness of the proposed methods.

The rest of the chapter is organized as follows. We introduce the proposed convex method in Sect. 2 and the proposed non-convex method in Sect. 3. Experimental results are given in Sect. 4. We conclude the chapter in Sect. 5.

2 A Convex Formulation

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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad (1)$$

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 $N = \{1, 2, \dots, p\}$ 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

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

where λ_1, λ_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 nonzero entries $u_i = u_j = \frac{1}{2}$, $v_i = -v_j = \frac{1}{2}$. Thus (2) can be rewritten in a matrix form as

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

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 $\lambda_1 \|\beta\|_1 + \lambda_2 \sum_{i < j} \max\{|\beta_i|, |\beta_j|\}$. 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 [23]. 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

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \\ \text{s.t. } \mathbf{Ax} + \mathbf{Bz} = \mathbf{c}. \end{aligned}$$

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

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

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

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

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

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

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

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

where μ, ν are augmented Lagrangian multipliers.

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

$$\beta^{k+1} = \arg \min_{\beta} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + (\mu^k + \mathbf{T}^T \nu^k)^T \beta + \frac{\rho}{2} \|\beta - \mathbf{q}^k\|^2 + \frac{\rho}{2} \|\mathbf{T}\beta - \mathbf{p}^k\|^2. \tag{5}$$

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

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

The computation of β^{k+1} involves solving a linear system, which is the most time-consuming part in the whole algorithm. To compute β^{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:

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

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

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

which is equivalent to the following problem:

$$\mathbf{q}^{k+1} = \arg \min_{\mathbf{q}} \frac{1}{2} \|\mathbf{q} - \beta^{k+1} - \frac{1}{\rho} \mu^k\|^2 + \frac{\lambda_1}{\rho} \|\mathbf{q}\|_1. \quad (7)$$

Equation (7) has a closed-form solution, known as *soft-thresholding*

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

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:

$$\mathbf{p}_i^{k+1} = S_{\lambda_2/\rho} \left(\mathbf{T} \beta^{k+1} + \frac{1}{\rho} \mathbf{v}^k \right). \quad (9)$$

Update μ, \mathbf{v} :

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

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: β

Initialization: $\mathbf{p}^0 \leftarrow \mathbf{0}, \mathbf{q}^0 \leftarrow \mathbf{0}, \mu^0 \leftarrow \mathbf{0}, \mathbf{v}^0 \leftarrow \mathbf{0}$;

Compute the Cholesky factorization of \mathbf{F} ;

do

 Compute β^{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 $\mu^{k+1}, \mathbf{v}^{k+1}$ according to Eq. (10).

Until *Convergence*;

return β ;

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 (8) is $O(p)$. The other one in (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 (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 A Non-convex Formulation

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, $||\beta_i| - |\beta_j||$, which shrinks only small differences in absolute values. As a result, estimation bias is reduced as compared to these convex grouping penalties. The proposed non-convex method performs well even when the graph is wrongly specified, unlike GFlasso. Note that the proposed non-convex grouping penalty does not assume that the sign of an edge is given; it only relies on the graph structure.

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

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

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 (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

$$f(\beta) = f_1(\beta) - f_2(\beta)$$

with $f_1(\beta)$ and $f_2(\beta)$ 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:

$$f_1(\beta) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \sum_{(i,j) \in E} (|\beta_i + \beta_j| + |\beta_i - \beta_j|),$$

$$f_2(\beta) = \lambda_2 \sum_{(i,j) \in E} (|\beta_i| + |\beta_j|).$$

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 $f_1(\beta)$ and $f_2(\beta)$ are convex functions.

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

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

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

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

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

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

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

$$\min_{\beta} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda_1 \|\beta\|_1 - (\mathbf{c}^k)^T \beta + \lambda_2 \sum_{(i,j) \in E} (|\beta_i + \beta_j| + |\beta_i - \beta_j|), \quad (15)$$

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

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

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

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

where s represents the iteration number in Algorithm 1.

Table 1 The algorithms and their associated penalties

Algorithm	Penalty
Lasso	$\lambda_1 \ \beta\ _1$
OSCAR	$\lambda_1 \ \beta\ _1 + \lambda_2 \sum_{i < j} \max\{ \beta_i , \beta_j \}$
GFlasso	$\lambda_1 \ \beta\ _1 + \lambda_2 \sum_{(i,j) \in E} \beta_i - \text{sign}(r_{ij})\beta_j $
GOSCAR	$\lambda_1 \ \beta\ _1 + \lambda_2 \sum_{(i,j) \in E} \max\{ \beta_i , \beta_j \}$
ncFGS	$\lambda_1 \ \beta\ _1 + \lambda_2 \sum_{(i,j) \in E} \beta_i - \beta_j $

The key steps of ncFGS are shown in Algorithm 2.

Algorithm 2: The ncFGS algorithm

Input: $\mathbf{X}, \mathbf{y}, E, \lambda_1, \lambda_2, \varepsilon$

Output: β

Initialization: $\beta^0 \leftarrow \mathbf{0}$;

while $f(\beta^k) - f(\beta^{k+1}) > \varepsilon$ **do**

 Compute \mathbf{c}^k according to Eq. (14).

 Compute β^{k+1} using Algorithm 1 with \mathbf{c}^k and $\lambda_1, 2\lambda_2$ as regularization parameters.

end

return β ;

4 Numerical Results

We compare GOSCAR and ncFGS against lasso, GFlasso, and OSCAR on synthetic datasets and a real dataset: Breast Cancer.¹ 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 summarized in Table 1:

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 3,000. The regression model is $\mathbf{y} = \mathbf{X}\beta + \varepsilon$, 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 3,000. 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 Fig. 1. As can be seen in Fig. 1, GOSCAR can achieve $1e-4$ precision in less than 10s when the dimension and the number of edges are 1,000. The computational time of ncFGS

¹<http://cbio.ensmp.fr/~jvert/publi/>

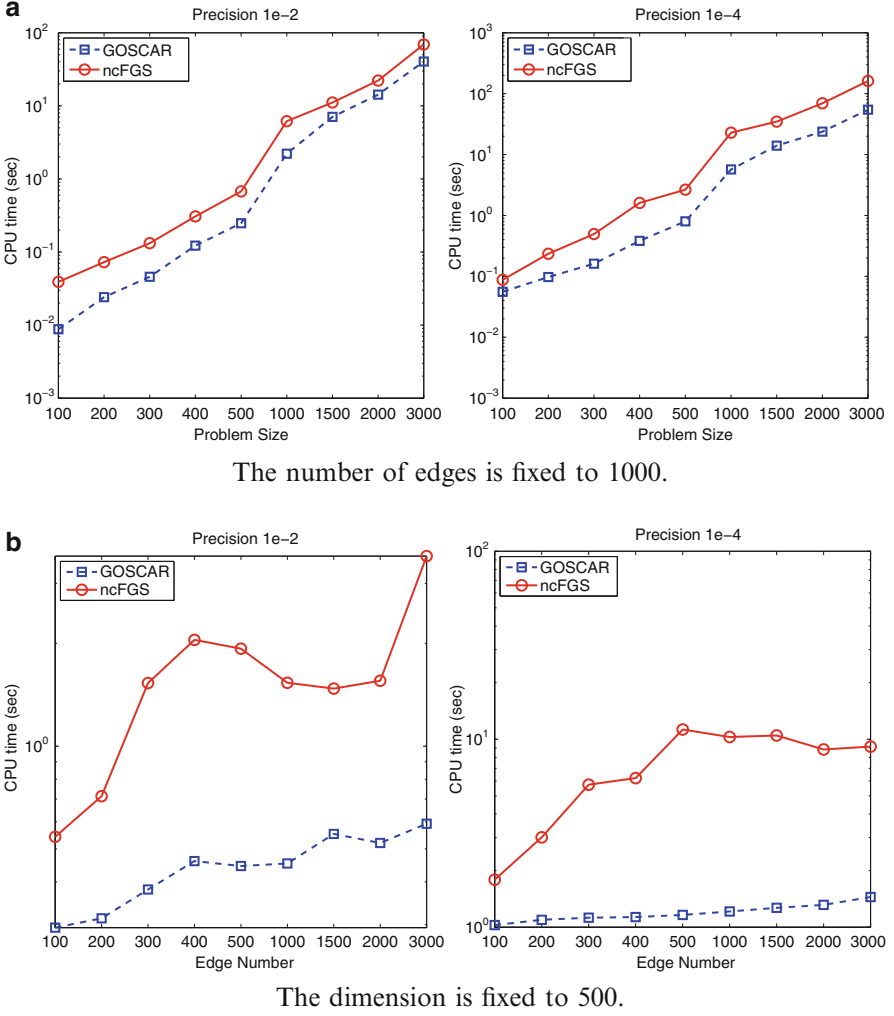


Fig. 1 Comparison of GOSCAR and ncFGS in terms of computation time with different dimensions, precisions, and the numbers of edges (in seconds and in logarithmic scale)

is about seven times higher than that of GOSCAR in this experiment. 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 3,000. It is not surprising because the 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 is quadratic with respect to p .

4.2 Simulations

We use five synthetic problems that have been commonly used in the sparse learning literature [2, 9] to compare the performance of different methods. The data is generated from the regression model $\mathbf{y} = \mathbf{X}\beta + \varepsilon$, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The five problems are given by:

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

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

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

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

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

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

3. Consider a regulatory gene network [9], 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 $\mathbf{X}_i^{\text{TF}} \sim \mathcal{N}(0, \mathbf{S}_{11 \times 11})$ 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 $\mathbf{X} = [\mathbf{X}_1^{\text{TF}}, \dots, \mathbf{X}_{n_{\text{TF}}}^{\text{TF}}]$, $n = 100$, $p = 110$, and $\sigma = 5$. The true parameters are

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

4. Same as Problem 3 except that

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

5. Same as Problem 3 except that

$$\beta = \left(5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{10}, -5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{10}, \right. \\ \left. 3, \underbrace{\frac{3}{\sqrt{10}}, \dots, \frac{3}{\sqrt{10}}}_{10}, -3, \underbrace{\frac{-3}{\sqrt{10}}, \dots, \frac{-3}{\sqrt{10}}}_{10}, \underbrace{0, \dots, 0}_{p-44} \right)^T$$

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 β , which is defined as

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

For feature grouping and selection, we introduce two separate metrics to measure the accuracy of feature grouping and selection. Denote $I_i, i = 0, 1, 2, \dots, K$ 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 $\tilde{\beta}$ 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 $\tilde{\mathbf{X}}\tilde{\beta} = \mathbf{X}\beta$. 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 β converted from $\tilde{\beta}$ are reported.

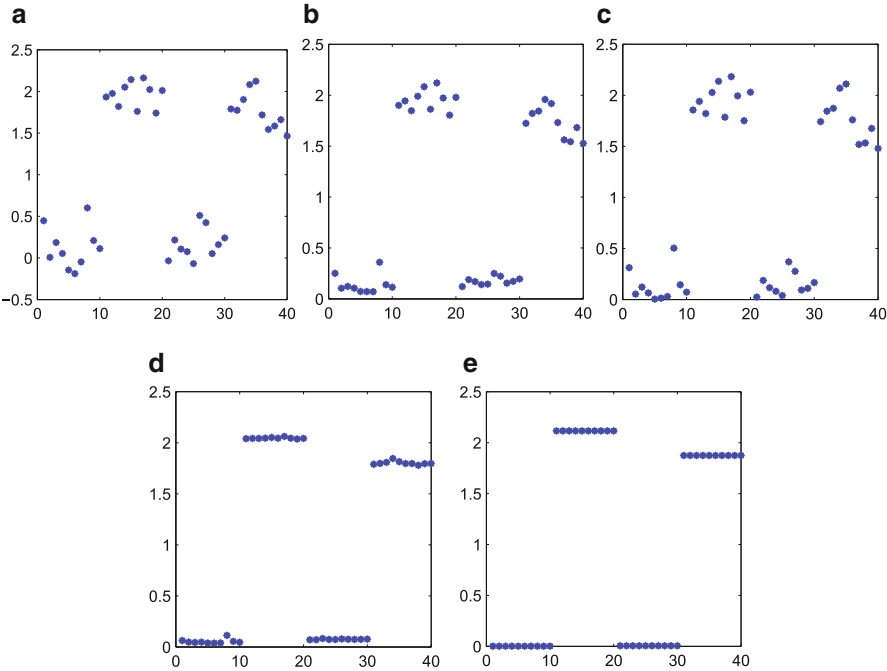


Fig. 2 The average nonzero coefficients obtained on dataset 1 with $\sigma = 2$: (a) Lasso; (b) GFlasso; (c) OSCAR; (d) GOSCAR; (e) ncFGS

Table 2 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
Data1 ($\sigma = 2$)	1.807 (0.331)	1.441 (0.318)	1.080 (0.276)	0.315 (0.157)	0.123 (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)
Data1 ($\sigma = 10$)	12.628 (3.931)	13.880 (4.031)	13.411 (4.540)	6.061 (4.022)	1.963 (1.600)
Data2 ($\sigma = 2$)	1.308 (0.435)	1.084 (0.439)	0.623 (0.250)	0.291 (0.208)	0.226 (0.175)
Data2 ($\sigma = 5$)	4.907 (1.496)	4.868 (1.625)	2.538 (0.656)	0.781 (0.598)	0.721 (0.532)
Data2 ($\sigma = 10$)	18.175 (6.611)	18.353 (6.611)	6.930 (2.858)	4.601 (2.623)	4.232 (2.561)
Data3 ($\sigma = 5$)	5.163 (1.708)	4.503 (1.677)	4.236 (1.476)	3.336 (1.725)	0.349 (0.282)
Data4 ($\sigma = 5$)	7.664 (2.502)	7.167 (2.492)	7.516 (2.441)	7.527 (2.434)	5.097 (0.780)
Data5 ($\sigma = 5$)	9.893 (1.965)	7.907 (2.194)	9.622 (2.025)	9.810 (2.068)	7.684 (1.1191)

Figure 2 shows the average nonzero coefficients obtained on dataset 1 with $\sigma = 2$. As can be seen in Fig. 2, GOSCAR and ncFGS are able to utilize the graph information and achieve good parameter estimation. Although GFlasso can use the graph information, it performs worse than GOSCAR and ncFGS due to the graph misspecification.

The performance in terms of MSEs averaged over 30 simulations is shown in Table 2. As indicated in Table 2, among existing methods (Lasso, GFlasso,

Table 3 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
Data1 ($\sigma = 2$)	0.675 (0.098)	0.553 (0.064)	0.513 (0.036)	0.983 (0.063)
	0.708 (0.021)	0.709 (0.017)	0.702 (0.009)	0.994 (0.022)
Data1 ($\sigma = 5$)	0.565 (0.084)	0.502 (0.009)	0.585 (0.085)	1.000 (0.000)
	0.691 (0.011)	0.709 (0.016)	0.708 (0.017)	1.000 (0.000)
Data1 ($\sigma = 10$)	0.532 (0.069)	0.568 (0.088)	0.577 (0.061)	0.983 (0.063)
	0.675 (0.031)	0.725 (0.022)	0.708 (0.020)	0.994 (0.022)
Data2 ($\sigma = 2$)	0.739 (0.108)	0.544 (0.272)	1.000 (0.000)	0.958 (0.159)
	0.625 (0.052)	0.823 (0.029)	0.837 (0.014)	0.831 (0.052)
Data2 ($\sigma = 5$)	0.763 (0.114)	0.717 (0.275)	0.999 (0.005)	0.979 (0.114)
	0.650 (0.066)	0.741 (0.062)	0.833 (0.011)	0.845 (0.030)
Data2 ($\sigma = 10$)	0.726 (0.101)	0.818 (0.149)	0.993 (0.024)	1.000 (0.000)
	0.597 (0.058)	0.680 (0.049)	0.829 (0.025)	0.851 (0.015)
Data3 ($\sigma = 5$)	0.886 (0.135)	0.736 (0.103)	0.382 (0.084)	0.992 (0.026)
	0.841 (0.056)	0.739 (0.041)	0.689 (0.013)	0.995 (0.017)
Data4 ($\sigma = 5$)	0.875 (0.033)	0.881 (0.026)	0.882 (0.037)	0.796 (0.245)
	0.834 (0.030)	0.805 (0.035)	0.805 (0.036)	0.895 (0.114)
Data5 ($\sigma = 5$)	0.760 (0.203)	0.802 (0.153)	0.861 (0.051)	0.881 (0.174)
	0.858 (0.031)	0.821 (0.037)	0.805 (0.037)	0.920 (0.056)

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 outperforms all the other methods.

Table 3 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 four methods: OSCAR, GFlasso, GOSCAR, and ncFGS. Table 3 shows that ncFGS achieves higher accuracy than other methods in most cases.

4.3 Real Data

We conduct experiments on the breast cancer dataset. 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.

The breast cancer dataset consists of gene expression data for 8,141 genes in 295 breast cancer tumors (78 metastatic and 217 non-metastatic). The network described in [4] is used as the input graph in this experiment. Figure 3 shows a subgraph

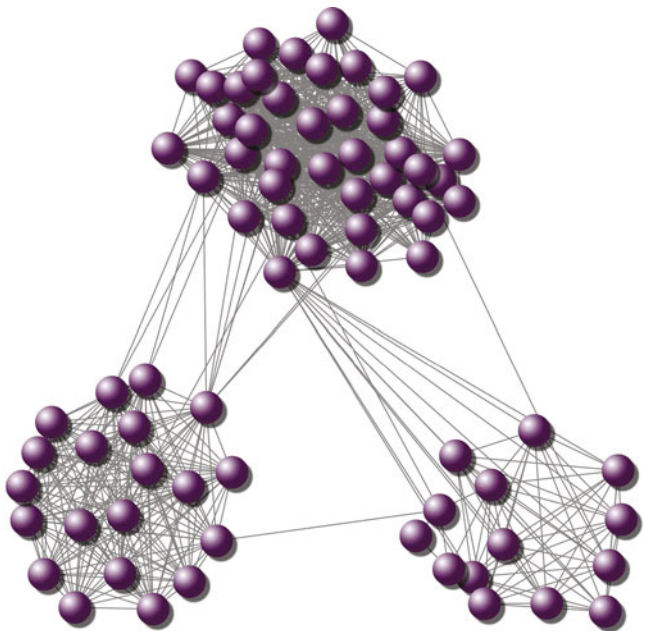


Fig. 3 A subgraph of the network in breast cancer dataset [4]. The subgraph consists of 80 nodes

Table 4 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
acc.	0.739 (0.054)	0.755 (0.055)	0.771 (0.050)	0.783 (0.042)	0.779 (0.041)
sen.	0.707 (0.056)	0.720 (0.060)	0.749 (0.060)	0.755 (0.050)	0.755 (0.055)
pec.	0.794 (0.071)	0.810 (0.068)	0.805 (0.056)	0.827 (0.061)	0.819 (0.058)
dof.	239.267	165.633	108.633	70.267	57.233
nonzero coeff.	239.267	243.867	144.867	140.667	79.833

The numbers in parentheses are the standard deviations

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. About 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 fivefold cross validation. Table 4 shows the results averaged over 30 replications. As indicated in Table 4, GOSCAR and ncFGS outperform the other three methods.

5 Summary

In this chapter, we consider simultaneous feature grouping and selection over a given undirected graph. We propose a convex and non-convex penalty 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 method. In this chapter, we focus on undirected graphs. A possible future direction is to extend the formulations to directed graphs.

Acknowledgements This work was supported in part by NSF (IIS-0953662, MCB-1026710, CCF-1025177, DMS-0906616) and NIH (R01LM010730, 2R01GM081535-01, R01HL105397).

References

1. Bach F, Lanckriet G, Jordan M (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML ACM New York, NY, USA. DOI 10.1145/1015330.1015424
2. Bondell H, Reich B (2008) Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics* 64(1):115–123
3. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers *Foundations and Trends® in Machine Learning*, Now Publishers Inc 3(1):1–122
4. Chuang H, Lee E, Liu Y, Lee D, Ideker T (2007) Network-based classification of breast cancer metastasis. *Mol Syst Biol* 3(1)
5. Fei H, Quanz B, Huan J (2010) Regularization and feature selection for networked features. In: CIKM, ACM New York, NY, USA pp 1893–1896. DOI 10.1145/1871437.1871756
6. Jacob L, Obozinski G, Vert J (2009) Group lasso with overlap and graph lasso. In: ICML, ACM New York, NY, USA pp 433–440. DOI 10.1145/1553374.1553431
7. Jenatton R, Mairal J, Obozinski G, Bach F (2010) Proximal methods for sparse hierarchical dictionary learning. In: ICML ACM New York, NY, USA
8. Kim S, Xing E (2009) Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS Genet* 5(8):e1000587
9. Li C, Li H (2008) Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics* 24(9):1175–1182
10. Liu J, Ji S, Ye J (2009) SLEP: Sparse learning with efficient projections. Arizona State University, <http://www.public.asu.edu/~jye02/Software/SLEP/>
11. Liu J, Ye J (2010) Moreau-Yosida regularization for grouped tree structure learning. In: NIPS
12. Rinaldo A (2009) Properties and refinements of the fused lasso. *Ann Stat* 37(5B):2922–2952
13. Shen X, Huang H (2009) Grouping pursuit through a regularization solution surface. *J Am Stat Assoc* 105(490):727–739
14. Shen X, Huang H, Pan W (2012) Simultaneous supervised clustering and feature selection over a graph. *Biometrika*, to appear
15. Shen X, Ye J (2002) Adaptive model selection. *J Am Stat Assoc* 97(457):210–221
16. Tao P, An L (1997) Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Math Vietnam* 22(1):289–355

17. Tao P, El Bernoussi S (1988) Duality in DC (difference of convex functions) optimization. Subgradient methods. *Trends Math Optimiz* 84:277–293
18. Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J Roy Stat Soc Ser B*, 58(1): 267–288
19. Tibshirani R, Saunders M, Rosset S, Zhu J, Knight K (2005) Sparsity and smoothness via the fused lasso. *J Roy Stat Soc Ser B* 67(1):91–108
20. Yuan L, Liu J, Ye J (2011) Efficient methods for overlapping group lasso. In: *NIPS*
21. Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. *J Roy Stat Soc Ser B* 68(1):49–67
22. Zhao P, Rocha G, Yu B (2009) The composite absolute penalties family for grouped and hierarchical variable selection. *Ann Stat* 37(6A):3468–3497
23. Zhong L, Kwok J (2011) Efficient sparse modeling with automatic feature grouping. In: *ICML*
24. Zhu Y, Shen X, Pan W (2012) Simultaneous grouping pursuit and feature selection in regression over an undirected graph. Manuscript
25. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *J Roy Stat Soc Ser B* 67(2):301–320



<http://www.springer.com/978-1-4614-4456-5>

Graph Embedding for Pattern Analysis

Fu, Y.; Ma, Y. (Eds.)

2013, VIII, 260 p., Hardcover

ISBN: 978-1-4614-4456-5