Exact Hybrid Covariance Thresholding for Joint Graphical Lasso

Qingming Tang[1], Chao Yang[1], Jian Peng[2], and Jinbo Xu[1(✉)]

[1] Toyota Technological Institute at Chicago, Chicago, USA
{qmtang,harryyang}@ttic.edu
[2] University of Illinois at Urbana-Champaign, Champaign, USA
jianpeng@illinois.edu

Abstract. This paper studies precision matrix estimation for multiple related Gaussian graphical models from a dataset consisting of different classes, based upon the formulation of this problem as group graphical lasso. In particular, this paper proposes a novel hybrid covariance thresholding algorithm that can effectively identify zero entries in the precision matrices and split a large joint graphical lasso problem into many small subproblems. Our hybrid covariance thresholding method is superior to existing uniform thresholding methods in that our method can split the precision matrix of each individual class using different partition schemes and thus, split group graphical lasso into much smaller subproblems, each of which can be solved very fast. This paper also establishes necessary and sufficient conditions for our hybrid covariance thresholding algorithm. Experimental results on both synthetic and real data validate the superior performance of our thresholding method over the others.

1 Introduction

Graphs have been widely used to describe the relationship between variables (or features). Estimating an undirected graphical model from a dataset has been extensively studied. When the dataset has a Gaussian distribution, the problem is equivalent to estimating a precision matrix from the empirical (or sample) covariance matrix. In many real-world applications, the precision matrix is sparse. This problem can be formulated as graphical lasso [1,22] and many algorithms [4,9,16,18,19] have been proposed to solve it. To take advantage of the sparsity of the precision matrix, some covariance thresholding (also called screening) methods are developed to detect zero entries in the matrix and then split the matrix into smaller submatrices, which can significantly speed up the process of estimating the entire precision matrix [12,19].

Recently, there are a few studies on how to jointly estimate multiple related graphical models from a dataset with a few distinct class labels [3,6–8,11,13,14,20,23–25]. The underlying reason for joint estimation is that the graphs of these classes are similar to some degree, so it can increase statistical power and estimation accuracy by aggregating data of different classes. This joint graph

A. Appice et al. (Eds.): ECML PKDD 2015, Part II, LNAI 9285, pp. 593–607, 2015.
DOI: 10.1007/978-3-319-23525-7_36

estimation problem can be formulated as joint graphical lasso that makes use of similarity of the underlying graphs. In addition to group graphical lasso, Guo et al. used a non-convex hierarchical penalty to promote similar patterns among multiple graphical models [6] ; [3] introduced popular group and fused graphical lasso; and [20,25] proposed efficient algorithms to solve fused graphical lasso. To model gene networks, [14] proposed a node-based penalty to promote hub structure in a graph.

Existing algorithms for solving joint graphical lasso do not scale well with respect to the number of classes, denoted as K, and the number of variables, denoted as p. Similar to covariance thresholding methods for graphical lasso, a couple of thresholding methods [20,25] are developed to split a large joint graphical lasso problem into subproblems [3]. Nevertheless, these algorithms all use uniform thresholding to decompose the precision matrices of distinct classes in exactly the same way. As such, it may not split the precision matrices into small enough submatrices especially when there are a large number of classes and/or the precision matrices have different sparsity patterns. Therefore, the speedup effect of covariance thresholding may not be very significant.

In contrast to the above-mentioned uniform covariance thresholding, this paper presents a novel hybrid (or non-uniform) thresholding approach that can divide the precision matrix for each individual class into smaller submatrices without requiring that the resultant partition schemes be exactly the same across all the classes. Using this method, we can split a large joint graphical lasso problem into much smaller subproblems. Then we employ the popular ADMM (Alternating Direction Method of Multipliers [2,5]) method to solve joint graphical lasso based upon this hybrid partition scheme. Experiments show that our method can solve group graphical lasso much more efficiently than uniform thresholding.

This hybrid thresholding approach is derived based upon group graphical lasso. The idea can also be generalized to other joint graphical lasso such as fused graphical lasso. Due to space limit, the proofs of some of the theorems in the paper are presented in supplementary material.

2 Notation and Definition

In this paper, we use a script letter, like $\mathcal{H}$, to denote a set or a set partition. When $\mathcal{H}$ is a set, we use $\mathcal{H}_i$ to denote the i^{th} element. Similarly we use a bold letter, like $\boldsymbol{H}$ to denote a graph, a vector or a matrix. When $\boldsymbol{H}$ is a matrix we use $\boldsymbol{H}_{i,j}$ to denote its $(i,j)^{th}$ entry. We use $\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(N)}\}$ and $\{\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)} \ldots, \boldsymbol{H}^{(N)}\}$ to denote N objects of same category.

Let $\{\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots, \boldsymbol{X}^{(K)}\}$ denote a sample dataset of K classes and the data in $\boldsymbol{X}^{(k)}$ $(1 \leq k \leq K)$ are independently and identically drawn from a p-dimension normal distribution $N(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$. Let $\boldsymbol{S}^{(k)}$ and $\hat{\boldsymbol{\Theta}}^{(k)}$ denote the empirical covariance and (optimal) precision matrices of class k, respectively. By "optimal" we mean the precision matrices are obtained by exactly solving

joint graphical lasso. Let a binary matrix $\boldsymbol{E}^{(k)}$ denote the sparsity pattern of $\hat{\boldsymbol{\Theta}}^{(k)}$, i.e., for any $i, j (1 \leq i, j \leq p)$, $\boldsymbol{E}^{(k)}_{i,j} = 1$ if and only if $\hat{\boldsymbol{\Theta}} T^{(k)}_{i,j} \neq 0$.

Set Partition. A set $\mathcal{H}$ is a partition of a set $\mathcal{C}$ when the following conditions are satisfied: 1) any element in $\mathcal{H}$ is a subset of $\mathcal{C}$; 2) the union of all the elements in $\mathcal{H}$ is equal to $\mathcal{C}$; and 3) any two elements in $\mathcal{H}$ are disjoint. Given two partitions $\mathcal{H}$ and $\mathcal{F}$ of a set $\mathcal{C}$, we say that $\mathcal{H}$ is **finer** than $\mathcal{F}$ (or $\mathcal{H}$ is a **refinement** of $\mathcal{F}$), denoted as $\mathcal{H} \preceq \mathcal{F}$, if every element in $\mathcal{H}$ is a subset of some element in $\mathcal{F}$. If $\mathcal{H} \preceq \mathcal{F}$ and $\mathcal{H} \neq \mathcal{F}$, we say that $\mathcal{H}$ is strictly finer than $\mathcal{F}$ (or $\mathcal{H}$ is a strict refinement of $\mathcal{F}$), denoted as $\mathcal{H} \prec \mathcal{F}$.

Let $\boldsymbol{\Theta}$ denote a matrix describing the pairwise relationship of elements in a set $\mathcal{C}$, where $\boldsymbol{\Theta}_{i,j}$ corresponds to two elements $\mathcal{C}_i$ and $\mathcal{C}_j$. Given a partition $\mathcal{H}$ of $\mathcal{C}$, we define $\boldsymbol{\Theta}_{\mathcal{H}_k}$ as a $|\mathcal{H}_k| \times |\mathcal{H}_k|$ submatrix of $\boldsymbol{\Theta}$ where $\mathcal{H}_k$ is an element of $\mathcal{H}$ and $(\boldsymbol{\Theta}_{\mathcal{H}_k})_{i,j} \cong \boldsymbol{\Theta}_{(\mathcal{H}_k)_i (\mathcal{H}_k)_j}$ for any suitable (i, j).

Graph-based Partition. Let $\mathcal{V} = \{1, 2, \ldots, p\}$ denote the variable (or feature) set of the dataset. Let graph $\boldsymbol{G}^{(k)} = (\mathcal{V}, \boldsymbol{E}^{(k)})$ denote the k^{th} estimated concentration graph $1 \leq k \leq K$. This graph defines a partition $\boxplus^{(k)}$ of $\mathcal{V}$, where an element in $\boxplus^{(k)}$ corresponds to a connected component in $\boldsymbol{G}^{(k)}$. The matrix $\hat{\boldsymbol{\Theta}}^{(k)}$ can be divided into disjoint submatrices based upon $\boxplus^{(k)}$. Let $\boldsymbol{E}$ denote the mix of $\boldsymbol{E}^{(1)}, \boldsymbol{E}^{(2)}, \ldots, \boldsymbol{E}^{(K)}$, i.e., one entry $\boldsymbol{E}_{i,j}$ is equal to 1 if there exists at least one k $(1 \leq k \leq K)$ such that $\boldsymbol{E}^{(k)}_{i,j}$ is equal to 1. We can construct a partition $\boxplus$ of $\mathcal{V}$ from graph $\boldsymbol{G} = \{\mathcal{V}, \boldsymbol{E}\}$, where an element in $\boxplus$ corresponds to a connected component in $\boldsymbol{G}$. Obviously, $\boxplus^{(k)} \preccurlyeq \boxplus$ holds since $\boldsymbol{E}^{(k)}$ is a subset of $\boldsymbol{E}$. This implies that for any k, the matrix $\hat{\boldsymbol{\Theta}}^{(k)}$ can be divided into disjoint submatrices based upon $\boxplus$.

Feasible Partition. A partition $\mathcal{H}$ of $\mathcal{V}$ is feasible for class k or graph $\boldsymbol{G}^{(k)}$ if $\boxplus^{(k)} \preccurlyeq \mathcal{H}$. This implies that 1) $\mathcal{H}$ can be obtained by merging some elements in $\boxplus^{(k)}$; 2) each element in $\mathcal{H}$ corresponds to a union of some connected components in graph $\boldsymbol{G}^{(k)}$; and 3) we can divide the precision matrix $\hat{\boldsymbol{\Theta}}^{(k)}$ into independent submatrices according to $\mathcal{H}$ and then separately estimate the submatrices without losing accuracy. $\mathcal{H}$ is uniformly feasible if for all k $(1 \leq k \leq K)$, $\boxplus^{(k)} \preccurlyeq \mathcal{H}$ holds.

Let $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(K)}$ denote K partitions of the variable set V. If for each k $(1 \leq k \leq K)$, $\boxplus^{(k)} \preccurlyeq \mathcal{H}^{(k)}$ holds, we say $\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(K)}\}$ is a feasible partition of $\mathcal{V}$ for the K classes or graphs. When at least two of the K partitions are not same, we say $\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(K)}\}$ is a non-uniform partition. Otherwise, $\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(K)}\}$ is a class-independent or uniform partition and abbreviated as $\mathcal{H}$. That is, $\mathcal{H}$ is uniformly feasible if for all k $(1 \leq k \leq K)$, $\boxplus^{(k)} \preccurlyeq \mathcal{H}$ holds. Obviously, $\{\boxplus^{(1)}, \boxplus^{(2)}, \ldots, \boxplus^{(K)}\}$ is finer than any non-uniform feasible partition of the K classes. Based upon the above definitions, we have the following theorem, which is proved in supplementary material.

Theorem 1. *For any uniformly feasible partition $\mathcal{H}$ of the variable set $\mathcal{V}$, we have $\boxplus \preccurlyeq \mathcal{H}$. That is, $\mathcal{H}$ is feasible for graph $\boldsymbol{G}$ and $\boxplus$ is the finest uniform feasible partition.*

Proof. First, for any element $\mathcal{H}_j$ in $\mathcal{H}$, $\boldsymbol{G}$ does not contain edges between $\mathcal{H}_j$ and $\mathcal{H} - \mathcal{H}_j$. Otherwise, since $\boldsymbol{G}$ is the mixing (or union) of all $\boldsymbol{G}^{(k)}$, there exists at least one graph $\boldsymbol{G}^{(k)}$ such that it contains at least one edge between $\mathcal{H}_j$ and $\mathcal{H} - \mathcal{H}_j$. Since $\mathcal{H}_j$ is the union of some elements in $\boxplus^{(k)}$, this implies that there exist two different elements in $\boxplus^{(k)}$ such that $\boldsymbol{G}^{(k)}$ contains edges between them, which contradicts with the fact that $\boldsymbol{G}^{(k)}$ does not contain edges between any two elements in $\boxplus^{(k)}$. That is, $\mathcal{H}$ is feasible for graph $\boldsymbol{G}$.

Second, if $\boxplus \preccurlyeq \mathcal{H}$ does not hold, then there is one element $\boxplus_i$ in $\boxplus$ and one element $\mathcal{H}_j$ in $\mathcal{H}$ such that $\boxplus_i \cap \mathcal{H}_j \neq \emptyset$ and $\boxplus_i - \mathcal{H}_j \neq \emptyset$. Based on the above paragraph, $\forall x \in \boxplus_i \cap \mathcal{H}_j$ and $\forall y \in \boxplus_i - \mathcal{H}_j = \boxplus_i \cap (\mathcal{H}_i - \mathcal{H}_j)$, we have $\boldsymbol{E}_{x,y} = \boldsymbol{E}_{y,x} = 0$. That is, $\boxplus_i$ can be split into at least two disjoint subsets such that $\boldsymbol{G}$ does not contain any edges between them. This contradicts with the fact that $\boxplus_i$ corresponds to a connected component in graph $\boldsymbol{G}$.

3 Joint Graphical Lasso

To learn the underlying graph structure of multiple classes simultaneously, some penalty functions are used to promote similar structural patterns among different classes, including [3,6,7,13,14,16,20,21,25]. A typical joint graphical lasso is formulated as the following optimization problem:

$$\min \sum_{k=1}^{K} L(\boldsymbol{\Theta}^{(k)}) + P(\boldsymbol{\Theta}) \tag{1}$$

Where $\boldsymbol{\Theta}^{(k)} \succ 0$ is the precision matrix ($k = 1, \ldots, K$) and $\boldsymbol{\Theta}$ represents the set of $\boldsymbol{\Theta}^{(k)}$. The negative log-likelihood $L(\boldsymbol{\Theta}^{(k)})$ and the regularization $P(\boldsymbol{\Theta})$ are defined as follows.

$$L(\boldsymbol{\Theta}^{(k)}) = -\log\det(\boldsymbol{\Theta}^{(k)}) + \mathrm{tr}(\mathcal{S}^{(k)}\boldsymbol{\Theta}^{(k)}) \tag{2}$$

$$P(\boldsymbol{\Theta}) = \lambda_1 \sum_{k=1}^{K} \|\boldsymbol{\Theta}^{(k)}\|_1 + \lambda_2 J(\boldsymbol{\Theta}) \tag{3}$$

Here $\lambda_1 > 0$ and $\lambda_2 > 0$ and $J(\boldsymbol{\Theta})$ is some penalty function used to encourage similarity (of the structural patterns) among the K classes. In this paper, we focus on group graphical lasso. That is,

$$J(\boldsymbol{\Theta}) = 2 \sum_{1 \leq i < j \leq p} \sqrt{\sum_{k=1}^{K} (\boldsymbol{\Theta}_{i,j}^{(k)})^2} \tag{4}$$

4 Uniform Thresholding

Covariance thresholding methods, which identify zero entries in a precision matrix before directly solving the optimization problem like Eq.(1), are widely

used to accelerate solving graphical lasso. In particular, a screening method divides the variable set into some disjoint groups such that when two variables (or features) are not in the same group, their corresponding entry in the precision matrix is guaranteed to be 0. Using this method, the precision matrix can be split into some submatrices, each corresponding to one distinct group. To achieve the best computational efficiency, we shall divide the variable set into as small groups as possible subject to the constraint that two related variables shall be in the same group. Meanwhile, [3] described a screening method for group graphical lasso. This method uses a single thresholding criterion (i.e., uniform thresholding) for all the K classes, i.e., employs a uniformly feasible partition of the variable set across all the K classes. Existing methods such as those described in [3,20,25] for fused graphical lasso and that in [15] for node-based learning all employ uniform thresholding.

Uniform thresholding may not be able to divide the variable set into the finest feasible partition for each individual class when the K underlying concentration graphs are not exactly the same. For example, Figure 1(a) and (c) show two concentration graphs of two different classes. These two graphs differ in variables 1 and 6 and each graph can be split into two connected components. However, the mixing graph in (b) has only one connected component, so it cannot be split further. According to **Theorem 1**, no uniform feasible partition can divide the variable set into two disjoint groups without losing accuracy. It is expected that when the number of classes and variables increases, uniform thresholding may perform even worse.

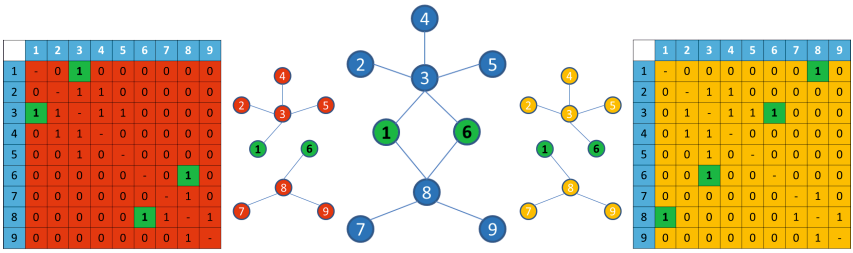


Fig. 1. Illustration of uniform thresholding impacted by minor structure difference between two classes. (a) and (c): the edge matrix and concentration graph for each of the two classes. (b): the concentration graph resulting from the mixing of two graphs in (a) and (c).

5 Non-uniform Thresholding

Non-uniform thresholding generates a non-uniform feasible partition by thresholding the K empirical covariance matrices separately. In a non-uniform partition, two variables of the same group in one class may belong to different

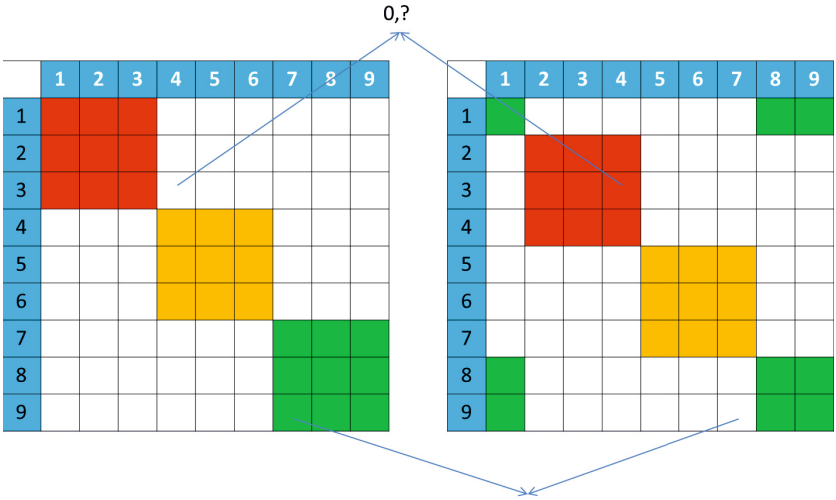


Fig. 2. Illustration of a non-uniform partition. White color indicates zero entries detected by covariance thresholding. Entries with the same color other than white belong to the same group.

groups in another class. Figure 2 shows an example of non-uniform partition. In this example, all the matrix elements in white color are set to 0 by non-uniform thresholding. Except the white color, each of the other colors indicates one group. The 7^{th} and 9^{th} variables belong to the same group in the left matrix, but not in the right matrix. Similarly, the 3^{rd} and 4^{th} variables belong to the same group in the right matrix, but not in the left matrix.

We now present necessary and sufficient conditions for identifying a non-uniform feasible partition for group graphical lasso, with penalty defined in Eq (3) and (4).

Given a non-uniform partition $\{\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(K)}\}$ for the K classes, let $F^{(k)}(i) = t$ denote the group which the variable i belongs to in the k^{th} class, i.e., $F^{(k)}(i) \Leftrightarrow i \in \mathrm{P}_t^{(k)}$. We define pairwise relationship matrices $\mathbf{I}^{(k)}$ $(1 \leq k \leq K)$ as follows:

$$\begin{cases} \mathbf{I}_{i,j}^{(k)} = \mathbf{I}_{j,i}^{(k)} = 0; \text{ if } F^{(k)}(i) \neq F^{(k)}(j) \\ \mathbf{I}_{i,j}^{(k)} = \mathbf{I}_{j,i}^{(k)} = 1; \text{ otherwise} \end{cases} \tag{5}$$

Also, we define $\boldsymbol{Z}^{(k)} (1 \leq k \leq K)$ as follows:

$$\boldsymbol{Z}_{i,j}^{(k)} = \boldsymbol{Z}_{j,i}^{(k)} = \lambda_1 + \lambda_2 \times \tau((\sum_{t \neq k} |\hat{\boldsymbol{\Theta}}_{i,j}^{(t)}|) = 0) \tag{6}$$

Here $\tau(b)$ is the indicator function.

The following two theorems state the necessary and sufficient conditions of a non-uniform feasible partition. See supplementary material for their proofs.

Algorithm 1 Hybrid Covariance Screening Algorithm

for $k = 1$ *to* K **do**
  Initialize $\mathbf{I}_{i,j}^{(k)} = \mathbf{I}_{j,i}^{(k)} = 1, \forall 1 \leq i < j \leq p$
  Set $\mathbf{I}_{i,j}^{(k)} = 0$, if $|\boldsymbol{S}_{i,j}^{(k)}| \leq \lambda_1$ and $i \neq j$
  Set $\mathbf{I}_{i,j}^{(k)} = 0$, if $\sum_{k=1}^{K}(|\boldsymbol{S}_{i,j}^{(k)}| - \lambda_1)_+^2 \leq \lambda_2^2$ and $i \neq j$
end for
for $k = 1$ *to* K **do**
  Construct a graph $\boldsymbol{G}^{(k)}$ for $\mathcal{V}$ from $\boldsymbol{I}^{(k)}$
  Find connected components of $G^{(k)}$
  for $\forall(i,j)$ *in the same component of* $\boldsymbol{G}^{(k)}$ **do**
    Set $\boldsymbol{I}_{i,j}^{(k)} = \boldsymbol{I}_{j,i}^{(k)} = 1$
  end for
end for
repeat
  Search for triple (x,i,j) satisfying the following condition:
  $\boldsymbol{I}_{i,j}^{(x)} = 0$, $|\boldsymbol{S}_{i,j}^{(x)}| > \lambda_1$ and $\exists s$, s.t. $\boldsymbol{I}_{i,j}^{(s)} = 1$
  if $\exists(x,i,j)$ satisfies the condition above **then**
    merge the two components of $\boldsymbol{G}^{(x)}$ that containing variable i and j into new component;
    for $\forall(m,n)$ in this new component **do**
      Set $\boldsymbol{I}_{m,n}^{(x)} = \boldsymbol{I}_{n,m}^{(x)} = 1$;
    end for
  end if
until No such kind of triple.
return the connected components of each graph which define the non-uniform feasible solution

Theorem 2. *If* $\{\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(K)}\}$ *is a non-uniform feasible partition of the variable set* $\mathcal{V}$*, then for any pair* (i,j) $(1 \leq i \neq j \leq p)$ *the following conditions must be satisfied:*

$$\begin{cases} \sum_{k=1}^{K}(|\boldsymbol{S}_{i,j}^{(k)}| - \lambda_1)_+^2 \leq \lambda_2^2; \ \textit{if } \forall k \in 1,2,\ldots,K, \boldsymbol{I}_{i,j}^{(k)} = 0 \\ |\boldsymbol{S}_{i,j}^{(k)}| \leq \boldsymbol{Z}_{i,j}^{(k)}; \ \textit{if } \boldsymbol{I}_{i,j}^{(k)} = 0 \textit{ and } \exists t \neq k, \boldsymbol{I}_{i,j}^{(t)} = 1 \end{cases} \tag{7}$$

Here, each $\boldsymbol{S}^{(k)}$ *is a covariance matrix of the* k^{th} *class and* $x_+ = \max(0,x)$.

Theorem 3. *If for any pair* $(i,j)(1 \leq i \neq j \leq p)$ *the following conditions hold, then* $\{\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(K)}\}$ *is a non-uniform feasible partition of the variable set* $\mathcal{V}$.

$$\begin{cases} \sum_{k=1}^{K}(|\boldsymbol{S}_{i,j}^{(k)}| - \lambda_1)_+^2 \leq \lambda_2^2; \ \textit{if } \forall k \in 1,2,\ldots,K, \boldsymbol{I}_{i,j}^{(k)} = 0 \\ |\boldsymbol{S}_{i,j}^{(k)}| \leq \lambda_1; \ \textit{if } \boldsymbol{I}_{i,j}^{(k)} = 0 \textit{ and } \exists t \neq k, \boldsymbol{I}_{i,j}^{(t)} = 1 \end{cases} \tag{8}$$

Algorithm 1 is a covariance thresholding algorithm that can identify a non-uniform feasible partition satisfying condition (8). We call **Algorithm 1** hybrid screening algorithm as it utilizes both class-specific thresholding (e.g. $|\mathbf{S}_{i,j}^{(k)}| \leq \lambda_1$)

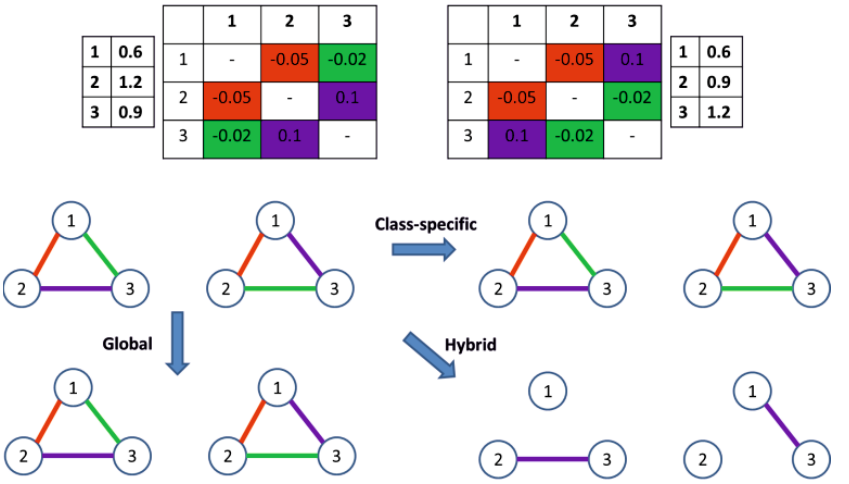


Fig. 3. Comparison of three thresholding strategies. The dataset contains 2 slightly different classes and 3 variables. The two sample covariance matrices are shown on the top of the figure. The parameters used are $\lambda_1 = 0.04$ and $\lambda_2 = 0.02$.

and global thresholding (e.g. $\sum_{k=1}^{K}(|\mathbf{S}_{i,j}^{(k)}| - \lambda_1)_+^2 \leq \lambda_2^2$) to identify a non-uniform partition. This hybrid screening algorithm can terminate rapidly on a typical Linux machine, tested on the synthetic data described in section 7 with $K = 10$ and $p = 10000$.

We can generate a uniform feasible partition using only the global thresholding and generate a non-uniform feasible partition by using only the class-specific thresholding, but such a partition is not as good as using the hybrid thresholding algorithm. Let $\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(K)}\}$, $\{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \ldots, \mathcal{L}^{(K)}\}$ and $\mathcal{G}$ denote the partitions generated by hybrid, class-specific and global thresholding algorithms, respectively. It is obvious that $\mathcal{H}^{(k)} \preccurlyeq \mathcal{L}^{(k)}$ and $\mathcal{H}^{(k)} \preccurlyeq \mathcal{G}$ for $k = 1, 2, \ldots, K$ since condition (8) is a combination of both global thresholding and class-specific thresholding.

Figure 3 shows a toy example comparing the three screening methods using a dataset of two classes and three variables. In this example, the class-specific or the global thresholding alone cannot divide the variable set into disjoint groups, but their combination can do so.

We have the following theorem regarding our hybrid thresholding algorithm, which will be proved in Supplemental File.

Theorem 4. *The hybrid screening algorithm yields the finest non-uniform feasible partition satisfying condition (8).*

6 Hybrid ADMM (HADMM)

In this section, we describe how to apply ADMM (Alternating Direction Method of Multipliers [2,5]) to solve joint graphical lasso based upon a non-uniform

feasible partition of the variable set. According to [3], solving Eq.(1) by ADMM is equivalent to minimizing the following scaled augmented Lagrangian form:

$$\sum_{k=1}^{K} L(\boldsymbol{\Theta}^{(k)}) + \frac{\rho}{2} \sum_{k=1}^{K} \|\boldsymbol{\Theta}^{(k)} - \boldsymbol{Y}^{(k)} + \boldsymbol{U}^{(k)}\|_F^2 + P(\boldsymbol{Y}) \tag{9}$$

where $\boldsymbol{Y} = \{\boldsymbol{Y}^{(1)}, \boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(K)}\}$ and $\boldsymbol{U} = \{\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(K)}\}$ are dual variables. We use the ADMM algorithm to solve Eq.(9) iteratively, which updates the three variables $\boldsymbol{\Theta}$, $\boldsymbol{Y}$ and $\boldsymbol{U}$ alternatively. The most computational-insensitive step is to update $\boldsymbol{\Theta}$ given $\boldsymbol{Y}$ and $\boldsymbol{U}$, which requires eigen-decomposition of K matrices. We can do this based upon a non-uniform feasible partition $\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(K)}\}$. For each k, updating $\boldsymbol{\Theta}^{(k)}$ given $\boldsymbol{Y}^{(k)}$ and $\boldsymbol{U}^{(k)}$ for Eq (9) is equivalent to solving in total $|\mathcal{H}^{(k)}|$ independent sub-problems. For each $\mathcal{H}_j^{(k)} \in \mathcal{H}^{(k)}$, its independent sub-problem solves the following equation:

$$(\boldsymbol{\Theta}^{(k)}_{H_j^{(k)}})^{-1} = \mathcal{S}^{(k)}_{\mathcal{H}_j^{(k)}} + \rho \times (\boldsymbol{\Theta}^{(k)}_{\mathcal{H}_j^{(k)}} - \boldsymbol{Y}^{(k)}_{\mathcal{H}_j^{(k)}} + \boldsymbol{U}^{(k)}_{\mathcal{H}_j^{(k)}}) \tag{10}$$

Solving Eq.(10) requires eigen-decomposition of small submatrices, which shall be much faster than the eigen-decomposition of the original large matrices. Based upon our non-uniform partition, updating $\boldsymbol{Y}$ given $\boldsymbol{\Theta}$ and $\boldsymbol{U}$ and updating $\boldsymbol{U}$ given $\boldsymbol{Y}$ and $\boldsymbol{\Theta}$ are also faster than the corresponding components of the plain ADMM algorithm described in [3], since our non-uniform thresholding algorithm can detect many more zero entries before ADMM is applied.

7 Experimental Results

We tested our method, denoted as HADMM (i.e., hybrid covariance thresholding algorithm + ADMM), on both synthetic and real data and compared HADMM with two control methods: 1) GADMM: global covariance thresholding algorithm + ADMM; and 2) LADMM: class-specific covariance thresholding algorithm +ADMM. We implemented these methods with C++ and R, and tested them on a Linux machine with Intel Xeon E5-2670 2.6GHz.

To generate a dataset with K classes from Gaussian distribution, we first randomly generate K precision matrices and then use them to sample $5 \times p$ data points for each class. To make sure that the randomly-generated precision matrices are positive definite, we set all the diagonal entries to 5.0, and an off-diagonal entry to either 0 or $\pm r \times 5.0$. We generate three types of datasets as follows.

- **Type A**: 97% of the entries in a precision matrix are 0.
- **Type B**: the K precision matrices have same diagonal block structure.
- **Type C**: the K precision matrices have slightly different diagonal block structures.

For **Type A**, r is set to be less than 0.0061. For **Type B** and **Type C**, r is smaller than 0.0067. For each type we generate 18 datasets by setting $K = 2, 3, \ldots, 10$, and $p = 1000,\ 10000$, respectively.

Table 1. Objective function values of HADMM and ADMM on the six classes type C data (first 4 iterations, $p = 1000$, $\lambda_1 = 0.0082$, $\lambda_2 = 0.0015$)

Iteration	**1**	**2**	**3**	**4**
ADMM	1713.66	-283.743	-1191.94	-1722.53
HADMM	1734.42	-265.073	-1183.73	-1719.78

7.1 Correctness of HADMM by Experimental Validation

We first show that HADMM can converge to the same solution obtained by the plain ADMM (i.e., ADMM without any covariance thresholding) through experiments.

To evaluate the correctness of our method HADMM, we compare the objective function value generated by HADMM to that by ADMM with respect to the number of iterations. We run the two methods for 500 iterations over the three types of data with $p = 1000$. As shown in Table 1, in the first 4 iterations, HADMM and ADMM yield slightly different objective function values. However, along with more iterations passed, both HADMM and ADMM converge to the same objective function value, as shown in Figure 4 and Supplementary Figures S3-5. This experimental result confirms that our hybrid covariance thresholding algorithm is correct. We tested several pairs of hyper-parameters (λ_1 and λ_2) in our experiment. Please refer to the supplementary material for model selection. Note that although in terms of the number of iterations HADMM and ADMM

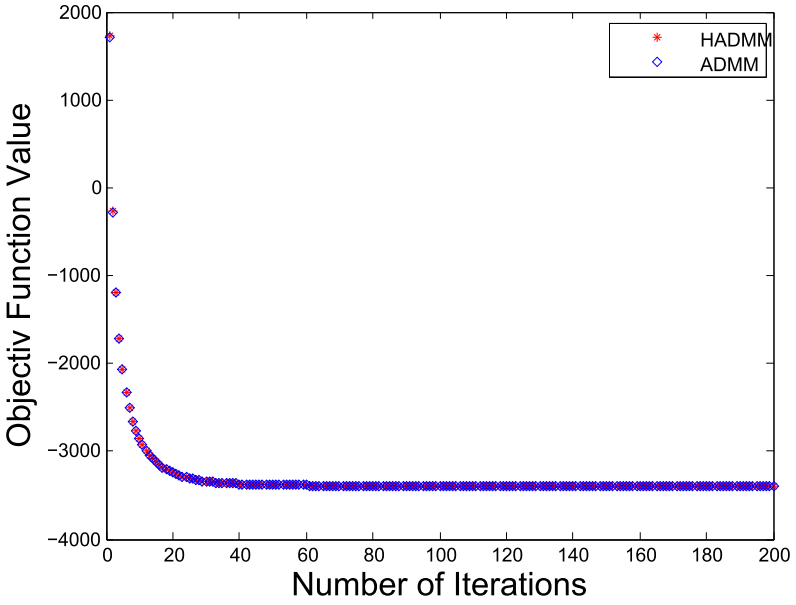
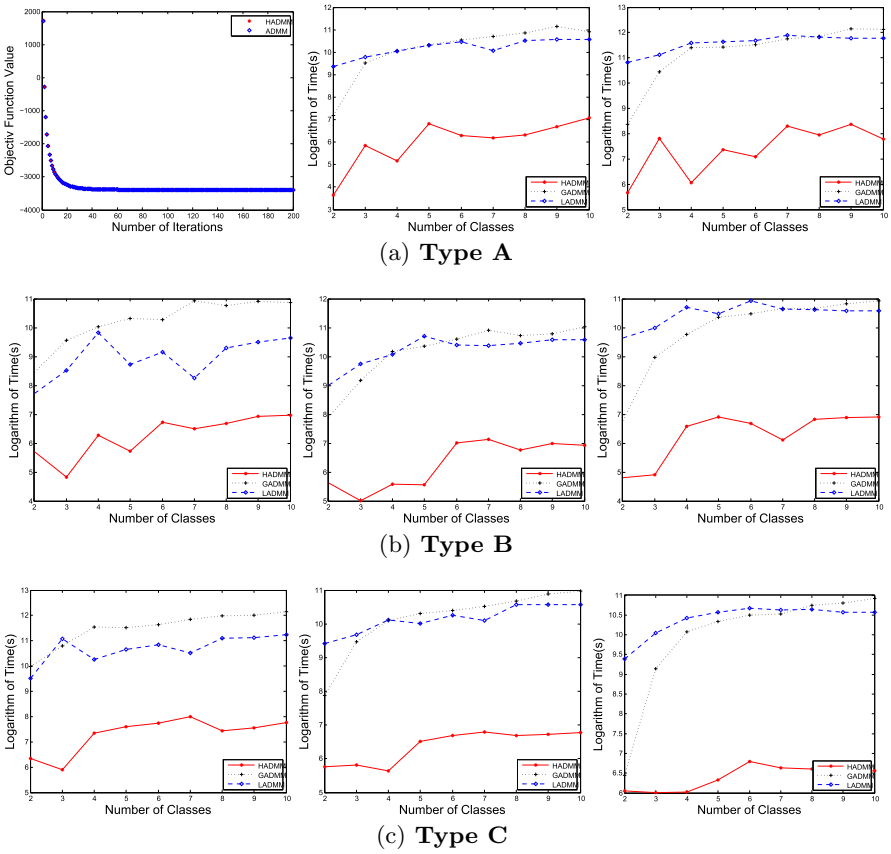


Fig. 4. The objective function value with respect to the number of iterations on a six classes type C data with $p = 1000$, $\lambda_1 = 0.0082$ and $\lambda_2 = 0.0015$.



(a) **Type A**

(b) **Type B**

(c) **Type C**

Fig. 5. Logarithm of the running time (in seconds) of HADMM, LADMM and GADMM for $p = 1000$ on **Type A, Type B** and **Type C** data.

converge similarly, HADMM runs much faster than ADMM at each iteration, so HADMM converges in a much shorter time.

7.2 Performance on Synthetic Data

In previous section we have shown that our HADMM converges to the same solution as ADMM. Here we test the running times of HADMM, LADMM and GADMM needed to reach the following stop criteria for $p = 1000$: $\sum_{i=1}^{k} ||\boldsymbol{\Theta}^{(k)} - \boldsymbol{Y}^{(k)}|| < 10^{-6}$ and $\sum_{i=1}^{k} ||\boldsymbol{Y}^{(k+1)} - \boldsymbol{Y}^{(k)}|| < 10^{-6}$. For $p = 10000$, considering the large amount of running time needed for LADMM and GADMM, we run only 50 iterations for all the three methods and then compare the average running time for a single iteration.

We tested the running time of the three methods using different parameters λ_1 and λ_2 over the three types of data. See supplementary material for model selection. We show the result for $p = 1000$ in Figure 5 and that for $p = 10000$ in Figure S15-23 in supplementary material, respectively.

In Figure 5, each row shows the experimental results on one type of data (**Type A**, **Type B** and **Type C** from top to bottom). Each column has the experimental results for the same hyper-parameters ($\lambda_1 = 0.009$ and $\lambda_2 = 0.0005$, $\lambda_1 = 0.0086$ and $\lambda_2 = 0.001$, and $\lambda_1 = 0.0082$ and $\lambda_2 = 0.0015$ from left to right). As shown in Figure 5, HADMM is much more efficient than LADMM and GADMM. GADMM performs comparably to or better than LADMM when λ_2 is large. The running time of LADMM increases as λ_1 decreases. Also, the running time of all the three methods increases along with the number of classes. However, GADMM is more sensitive to the number of classes than our HADMM. Moreover, as our hybrid covariance thresholding algorithm yields finer non-uniform feasible partitions, the precision matrices are more likely to be split into many more smaller submatrices. This means it is potentially easier to parallelize HADMM to obtain even more speedup.

We also compare the three screening algorithms in terms of the estimated computational complexity for matrix eigen-decomposition, a time-consuming subroutine used by the ADMM algorithms. Given a partition $\mathcal{H}$ of the variable set of $\mathcal{V}$, the computational complexity can be estimated by $\sum_{\mathcal{H}_i \in \mathcal{H}} |\mathcal{H}_i|^3$. As shown in Supplementary Figures S6-14, when $p = 1000$, our non-uniform thresholding algorithm generates partitions with much smaller computational complexity, usually $\frac{1}{10} \sim \frac{1}{1000}$ of the other two methods. Note that in these figures the Y-axis is the logarithm of the estimated computational complexity. When $p = 10000$, the advantage of our non-uniform thresholding algorithm over the other two are even larger, as shown in Figure S24-32 in Supplemental File.

7.3 Performance on Real Gene Expression Data

We test our proposed method on real gene expression data. We use a lung cancer data (accession number GDS2771 [17]) downloaded from Gene Expression Omnibus and a mouse immune dataset described in [10]. The immune dataset consists of 214 observations. The lung cancer data is collected from 97 patients with lung cancer and 90 controls without lung cancer, so this lung cancer dataset consists of two different classes: patient and control. We treat the 214 observations from the immune dataset, the 97 lung cancer observations and the 90 controls as three classes of a compound dataset for our joint inference task. These three classes share 10726 common genes, so this dataset has 10726 features and 3 classes. As the absolute value of entries of covariance matrix of first class (corresponds to immune observations) are relatively larger, so we divide each entry of this covariance matrix by 2 to make the three covariance matrices with similar magnitude before performing joint analysis using unique λ_1 and λ_2.

The running time (first 10 iterations) of HADMM, LADMM and GADMM for this compound dataset under different settings are shown in Table 2 and the resultant gene networks with different sparsity are shown in Fig 6 and Supplemental File.

As shown in Table 2, HADMM (ADMM + our hybrid screening algorithm) is always more efficient than the other two methods in different settings. Typically, when λ_1 is small and λ_2 is large (**Setting 1**), our method is much faster than

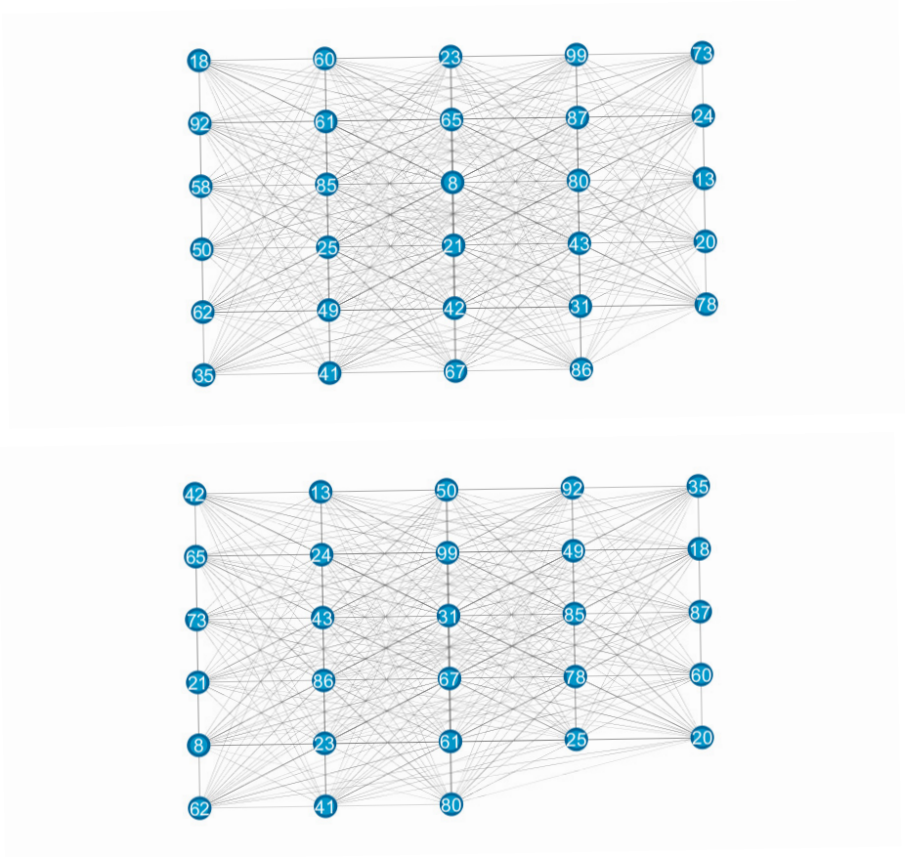


Fig. 6. Network of the first 100 genes of class one and class three for **Setting 1**.

Table 2. Running time (hours) of HADMM, LADMM and GADMM on real data. (**Setting 1**: $\lambda_1 = 0.1$ and $\lambda_2 = 0.5$; **Setting 2**: $\lambda_1 = 0.2$ and $\lambda_2 = 0.2$; **Setting 3**: $\lambda_1 = 0.3$ and $\lambda_2 = 0.1$; **Setting 4**: $\lambda_1 = 0.4$ and $\lambda_2 = 0.05$, and **Setting 5**: $\lambda_1 = 0.5$ and $\lambda_2 = 0.01$)

Method	**Setting 1**	**Setting 2**	**Setting 3**	**Setting 4**	**Setting 5**
HADMM	3.46	8.23	3.9	1.71	1.11
LADMM	> 20	> 20	13.6	3.72	1.98
GADMM	4.2	> 20	> 20	11.04	6.93

LADMM. In contrast, when λ_2 is small and λ_1 is large enough (**Setting 4** and **Setting 5**), our method is much faster than GADMM. What's more, when both λ_1 and λ_2 are with moderate values (**Setting 2** and **Setting 3**), HADMM is still much faster than both GADMM and LADMM.

As shown in Fig 6, the two resultant networks are with very similar topology structure. This is reasonable because we use large λ_2 in **Setting 1**. Actually,

the networks of all the three classes under **Setting 1** share very similar topology structure. What's more, the number of edges in the network does decrease significantly as λ_1 goes to 0.5, as shown in Supplementary material.

8 Conclusion and Discussion

This paper has presented a non-uniform or hybrid covariance thresholding algorithm to speed up solving group graphical lasso. We have established necessary and sufficient conditions for this thresholding algorithm. Theoretical analysis and experimental tests demonstrate the effectiveness of our algorithm. Although this paper focuses only on group graphical lasso, the proposed ideas and techniques may also be extended to fused graphical lasso.

In the paper, we simply show how to combine our covariance thresholding algorithm with ADMM to solve group graphical lasso. In fact, our thresholding algorithm can be combined with other methods developed for (joint) graphical lasso such as the QUIC algorithm [9], the proximal gradient method [16], and even the quadratic method developed for fused graphical lasso [20].

The thresholding algorithm presented in this paper is static in the sense that it is applied as a pre-processing step before ADMM is applied to solve group graphical lasso. We can extend this "static" thresholding algorithm to a "dynamic" version. For example, we can identify zero entries in the precision matrix of a specific class based upon intermediate estimation of the precision matrices of the other classes. By doing so, we shall be able to obtain finer feasible partitions and further improve the computational efficiency.

References

1. Banerjee, O., El Ghaoui, L., d'Aspremont, A.: Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. The Journal of Machine Learning Research **9**, 485–516 (2008)
2. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning **3**(1), 1–122 (2011)
3. Danaher, P., Wang, P., Witten, D.M.: The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **76**(2), 373–397 (2014)
4. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**(3), 432–441 (2008)
5. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Computers & Mathematics with Applications **2**(1), 17–40 (1976)
6. Guo, J., Levina, E., Michailidis, G., Zhu, J.: Joint estimation of multiple graphical models. Biometrika, asq060 (2011)
7. Hara, S., Washio, T.: Common substructure learning of multiple graphical gaussian models. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 1–16. Springer, Heidelberg (2011)

8. Honorio, J., Samaras, D.: Multi-task learning of gaussian graphical models. In: Proceedings of the 27th International Conference on Machine Learning, ICML 2010, pp. 447–454 (2010)
9. Hsieh, C.J., Dhillon, I.S., Ravikumar, P.K., Sustik, M.A.: Sparse inverse covariance matrix estimation using quadratic approximation. In: Advances in Neural Information Processing Systems, pp. 2330–2338 (2011)
10. Jojic, V., Shay, T., Sylvia, K., Zuk, O., Sun, X., Kang, J., Regev, A., Koller, D., Consortium, I.G.P., et al.: Identification of transcriptional regulators in the mouse immune system. Nature Immunology **14**(6), 633–643 (2013)
11. Liu, J., Yuan, L., Ye, J.: An efficient algorithm for a class of fused lasso problems. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 323–332. ACM (2010)
12. Mazumder, R., Hastie, T.: Exact covariance thresholding into connected components for large-scale graphical lasso. The Journal of Machine Learning Research **13**(1), 781–794 (2012)
13. Mohan, K., Chung, M., Han, S., Witten, D., Lee, S.I., Fazel, M.: Structured learning of gaussian graphical models. In: Advances in Neural Information Processing Systems, pp. 620–628 (2012)
14. Mohan, K., London, P., Fazel, M., Witten, D., Lee, S.I.: Node-based learning of multiple gaussian graphical models. The Journal of Machine Learning Research **15**(1), 445–488 (2014)
15. Oztoprak, F., Nocedal, J., Rennie, S., Olsen, P.A.: Newton-like methods for sparse inverse covariance estimation. In: Advances in Neural Information Processing Systems, pp. 755–763 (2012)
16. Rolfs, B., Rajaratnam, B., Guillot, D., Wong, I., Maleki, A.: Iterative thresholding algorithm for sparse inverse covariance estimation. In: Advances in Neural Information Processing Systems, pp. 1574–1582 (2012)
17. Spira, A., Beane, J.E., Shah, V., Steiling, K., Liu, G., Schembri, F., Gilman, S., Dumas, Y.M., Calner, P., Sebastiani, P., et al.: Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. Nature Medicine **13**(3), 361–366 (2007)
18. Tseng, P., Yun, S.: Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. Journal of Optimization Theory and Applications **140**(3), 513–535 (2009)
19. Witten, D.M., Friedman, J.H., Simon, N.: New insights and faster computations for the graphical lasso. Journal of Computational and Graphical Statistics **20**(4), 892–900 (2011)
20. Yang, S., Lu, Z., Shen, X., Wonka, P., Ye, J.: Fused multiple graphical lasso. arXiv preprint arXiv:1209.2139 (2012)
21. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **68**(1), 49–67 (2006)
22. Yuan, M., Lin, Y.: Model selection and estimation in the gaussian graphical model. Biometrika **94**(1), 19–35 (2007)
23. Yuan, X.: Alternating direction method for covariance selection models. Journal of Scientific Computing **51**(2), 261–273 (2012)
24. Zhou, S., Lafferty, J., Wasserman, L.: Time varying undirected graphs. Machine Learning **80**(2–3), 295–319 (2010)
25. Zhu, Y., Shen, X., Pan, W.: Structural pursuit over multiple undirected graphs. Journal of the American Statistical Association **109**(508), 1683–1696 (2014)