Efficient Algorithm for Extremely Large Multi-task Regression with Massive Structured Sparsity

Seunghak Lee*

Eric P. Xing**

School of Computer Science

Carnegie Mellon University, Pittsburgh, PA, U.S.A.

*email: seunghak@cs.cmu.edu

**email: epxing@cs.cmu.edu

August 16, 2012

Abstract

We develop a highly scalable optimization method called "hierarchical group-thresholding" for solving a multi-task regression model with complex structured sparsity constraints on both input and output spaces. Despite the recent emergence of several efficient optimization algorithms for tackling complex sparsity-inducing regularizers, true scalability in practical high-dimensional problems where a huge amount (e.g., millions) of sparsity patterns need to be enforced remains an open challenge, because all existing algorithms must deal with ALL such patterns exhaustively in every iteration, which is computationally prohibitive. Our proposed algorithm addresses the scalability problem by screening out multiple groups of coefficients simultaneously and systematically. We employ a hierarchical tree representation of group constraints to accelerate the process of removing irrelevant constraints by taking advantage of the inclusion relationships between group sparsities, thereby avoiding dealing with all constraints in every optimization step, and necessitating optimization operation only on a small number of outstanding coefficients. In our experiments, we demonstrate the efficiency of our method on simulation datasets, and in an application of detecting genetic variants associated with gene expression traits.

1. INTRODUCTION

In this paper, we propose a very efficient optimization technique for multi-task regression with structured sparsity. We are interested in the optimization problem with the following general form:

$$\min_{\mathbf{B}} \frac{1}{2}\|\mathbf{Y} - \mathbf{B}\mathbf{X}\|_F^2 + \lambda_1 |\mathbf{B}| + \lambda_2 \Omega_{in}(\mathbf{B}) + \lambda_3 \Omega_{out}(\mathbf{B}) \qquad (1)$$

where $\mathbf{X} \in \mathbb{R}^{J \times N}$ is the input data for J inputs and N samples, $\mathbf{Y} \in \mathbb{R}^{K \times N}$ is the K output data (equivalently K tasks), and $\mathbf{B} \in \mathbb{R}^{K \times J}$ is the regression coefficient matrix. Here Ω_{in} is an ℓ_1/ℓ_2 norm for inducing group sparsity among correlated inputs (grouping effects in the same rows of $\mathbf{B}$) and Ω_{out} is an ℓ_1/ℓ_2 norm for inducing group sparsity among correlated outputs (grouping effects in the same columns of $\mathbf{B}$). In this setting, it is possible that there exists overlap between/within input and output groups (i.e., a row group and a column group may intersect and hence overlap). Note that this formulation subsumes popular special cases such as single task lasso, group lasso, etc.. However, throughout this paper, we use the formulation in (1), as it explicitly presents a highly general regression problem, and one can still use our algorithm for a single task regression problem by setting $\lambda_3 = 0$ and $K = 1$.

Unfortunately, problem (1) is non-trivial to optimize as it poses two major challenges for large scale problems. First, we need to be able to handle a large number of group sparsities efficiently. For example, in eQTL mapping problems in bioinformatics, there exist a very large number of groups since the number of input and output groups are proportional to K (e.g., 2×10^4) and J (e.g., 5×10^5), respectively. Second, we need to deal with overlap of groups within and between Ω_{in} and Ω_{out}. Note that a simple coordinate descent algorithm is not applicable when Ω_{in} or Ω_{out} is non-separable.

The second challenge has been addressed by many optimization techniques including [9, 8, 11, 5, 20, 14, 17, 1, 10, 12, 3]. For example, Jacob et al. [8] proposed to select the union of overlapping groups as the support of sparse vectors. In their optimization procedure, input variables are duplicated to convert Ω_{in} with overlap into the norm with disjoint groups, and an optimization technique for group lasso [13] is applied. Jenatton et al. developed Structured-Lasso (SLasso) algorithm for sparsity-inducing norms with overlapping groups [9]. A smoothing proximal gradient method (SPG) [5] is developed to efficiently deal with overlapping group lasso penalty and graph-guided fusion penalty. Also, an efficient algorithm based on alternating direction methods [17] was proposed for overlapping group lasso with both ℓ_1/ℓ_2 norm and ℓ_1/ℓ_∞ norm. Recently, fast overlapping group lasso (FoGLasso) [20] was proposed for fast optimization of overlapping group lasso problem based on accelerated gradient descent method and a proximal operator.

However, the first challenge is a scalability problem when there exist a very large number of (overlapping) groups, and it has been relatively less studied in previous works. For example, the time complexity of smoothing proximal gradient method (SPG) [5] is $O(\sum_{\mathbf{g}_m \in \mathcal{G}} |\mathbf{g}_m|)$, where $\mathcal{G}$ is a set of groups, and a primal-dual algorithm for overlapping group lasso [14] has time complexity of $O(|\hat{\mathcal{G}}^3|)$, where $\hat{\mathcal{G}}$ is the set of active groups (groups having non-zero elements). At each iteration of SLasso algorithm [9], there is an expensive matrix inversion operation, and the inner loop of Picard-Nesterov method [1] and FoGLasso [20] have the time complexity of $O(J|\mathcal{G}|)$. As the number of groups in large-scale problems can be very large (e.g. 10^6), the scalability of existing algorithms could be severely affected by a large number of groups. Thus, there is an urgent need to develop an algorithm highly scalable to the number of groups, and in this paper, we present a highly efficient algorithm given a very large number of (overlapping) groups. Figure 2 illustrates the efficiency of our method in comparison to other competitors including FoGLasso, SPG, and SLasso.

We present a simple and efficient algorithm called hierarchical group-thresholding method

(HiGT) to address the scalability problem for overlapping group lasso. We use the following optimization strategy. First, we screen a large number of zero groups simultaneously by testing the zero condition of multiple groups. We further improved the speed of this step by employing a tree data structure where nodes represent the zero patterns encoded by Ω_{in} and Ω_{out} at different granularity, and edges indicate the inclusion relations among them. Using the tree data structure, we can avoid checking a large number of zero groups. Second, given a small number of nonzero groups of coefficients from the previous step, we solve our problem using an efficient method for overlapping group lasso. We used FoGLasso for the second step. It is also noteworthy that the accuracy of our screening step is not affected by the number of overlapping groups as it relies on exact optimality conditions of zero groups. Unlike our method, a large number of overlapping groups can degrade the accuracy of some approximation approaches (see Figure 2(b)).

In our experiments, we first evaluate the efficiency of the first step (screening step). Then, we demonstrate the performance of our method in terms of the speed and the accuracy for the recovery of structured sparsity via simulation study, in comparison to three state-of-the-art methods. As an example of biological analysis, we report a novel and significant SNP pair identified by our method, and present discussions.

Remark The problem (1) is originally motivated by expression quantitative trait loci (eQTLs) mapping in computational biology. Here eQTLs refer to the genomic locations or single nucleotide polymorphisms (SNPs) associated with gene expressions. In eQTL mapping problems, it is believed that many inputs (i.e., SNPs) impose small or medium effects on outputs (i.e., expression traits), and we usually have $J >> N$ ($J \sim 10^6, N \sim 10^3$) which exacerbate the noise to signal ratio. Thus, it is desirable to explore the groups of inputs to increase effective signal strength (individual inputs have too small effects to be detected) for more accurate causal SNP identification. It is also desirable to perform multi-task learning by jointly considering multiple (possibly correlated) responses to decrease the sample size required for successful support recovery [15] (the number of samples is too small to detect small signals). Thus, to take advantage of both input groups Ω_{in} and output groups Ω_{out} simultaneously, we are interested in solving problem (1).

Notations Given a matrix $\mathbf{B} \in \mathbb{R}^{K\times J}$, we denote the k-th row by $\boldsymbol{\beta}_k$, the j-th column by $\boldsymbol{\beta}^j$, and the (k,j) element by β_k^j. Given the set of groups $\mathcal{G} = \{\mathbf{g}_{m1}, \ldots, \mathbf{g}_{m|\mathcal{G}|}\}$ defined as a subset of the power set of $\{1,\ldots,J\}$, $\boldsymbol{\beta}_k^{\mathbf{g}_m}$ represents the row vector with elements $\{\beta_k^j : j \in \mathbf{g}_m,\ \mathbf{g}_m \in \mathcal{G}\}$. Similarly, for the set of groups $\mathcal{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{|\mathcal{H}|}\}$ over K rows of matrix $\mathbf{B}$, we denote by $\boldsymbol{\beta}_{\mathbf{h}_o}^j$ the column vector with elements $\{\beta_k^j : k \in \mathbf{h}_o,\ \mathbf{h}_o \in \mathcal{H}\}$. We also define the submatrix of $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$ as a $|\mathbf{h}_o| \times |\mathbf{g}_m|$ matrix with elements $\{\beta_k^j : k \in \mathbf{h}_o,\ j \in \mathbf{g}_m,\ \mathbf{h}_o \in \mathcal{H},\ \mathbf{g}_m \in \mathcal{G}\}$.

2. MULTI-TASK REGRESSION WITH STRUCTURED SPARSITY

We use a linear model parametrized by unknown regression coefficients $\mathbf{B} \in \mathbb{R}^{K\times J}$: $\mathbf{Y} = \mathbf{B}\mathbf{X} + \mathbf{E}$, where $\mathbf{E} \in \mathbb{R}^{K\times N}$ is i.i.d. Gaussian noise with zero mean and the identity covariance matrix. Throughout the paper, we assume that x_j^is and y_k^is are standardized, and consider a model without an intercept.

Suppose that we are given a set of input groups $\mathcal{G}$ and a set of output groups $\mathcal{H}$. We consider a multi-task regression model with structured sparsity:

$$\min_{\mathbf{B}} \frac{1}{2}\|\mathbf{Y} - \mathbf{B}\mathbf{X}\|_F^2 + \lambda_1 \|\mathrm{diag}\,(\boldsymbol{w}^T\mathbf{B})\|_1 + \lambda_2 \sum_{k=1}^{K} \sum_{\mathbf{g}_m \in \mathcal{G}} \rho_t \|\boldsymbol{\beta}_k^{\mathbf{g}_m}\|_2 + \lambda_3 \sum_{j=1}^{J} \sum_{\mathbf{h}_o \in \mathcal{H}} \nu_o \|\boldsymbol{\beta}_{\mathbf{h}_o}^j\|_2, \quad (2)$$

where $\mathbf{g}_m \in \mathcal{G}$ is the mth group of inputs, $\mathbf{h}_o \in \mathcal{H}$ is the oth group of outputs, $\|\boldsymbol{\beta}_k^{\mathbf{g}_m}\|_2 = \sqrt{\sum_{j\in\mathbf{g}_m} (\beta_k^j)^2}$, and $\|\boldsymbol{\beta}_{\mathbf{h}_o}^k\|_2 = \sqrt{\sum_{k\in\mathbf{h}_o} (\beta_k^j)^2}$. Here individual or groups of coefficients are differently penalized with weights $\boldsymbol{w} \in \mathbb{R}^{K\times J}$, $\boldsymbol{\rho} \in \mathbb{R}^{|\mathcal{G}|}$ and $\boldsymbol{\nu} \in \mathbb{R}^{|\mathcal{H}|}$. There may exist overlap between groups in $\mathcal{G}$ and groups in $\mathcal{H}$, and within groups in $\mathcal{G}$ or $\mathcal{H}$. Note that $\mathbf{B}$ will have zero patterns which are the union of groups in $\mathcal{G}$ and $\mathcal{H}$ and individual coefficients. The supports of $\mathbf{B}$ (nonzero β_k^j's) will be the complement of zero patterns. As the contribution of this paper is to propose an efficient optimization method, for simplicity, we assume that all weights are set to 1.

Example We illustrate an example of the penalty used for problem (2). Suppose we have two inputs and outputs, $\{\mathbf{x}_1, \mathbf{x}_2\}$, $\{\mathbf{y}_1, \mathbf{y}_2\}$, and $\mathbf{B}$ which includes $\{\beta_1^1, \beta_1^2, \beta_2^1, \beta_2^2\}$. For the input and output groups, we have $\mathcal{G} = \{\mathbf{g}_1\}$, $\mathbf{g}_1 = \{1, 2\}$, $\mathcal{H} = \{\mathbf{h}_1\}$ and $\mathbf{h}_1 = \{1, 2\}$. Under this setting, the penalty for problem (2) is given by

$$\Omega(\mathbf{B}) = \lambda_1 \sum_{k=1}^{2} \sum_{j=1}^{2} |\beta_k^j| + \lambda_2 \sum_{k=1}^{2} \sqrt{\sum_{j=1}^{2} (\beta_k^j)^2} + \lambda_3 \sum_{j=1}^{2} \sqrt{\sum_{k=1}^{2} (\beta_k^j)^2}. \tag{3}$$

3. HIERARCHICAL GROUP-THRESHOLDING

In this section, we propose an efficient method to optimize problem (2) referred to as Hierarchical Group-Thresholding (HiGT). Our algorithm consists of two steps. First, We identify zero groups by checking optimality conditions (called thresholding) as we walk through a predefined hierarchical tree. After walking though the nodes in the tree, some groups of coefficients might not achieve zero. Second, we optimize problem (2) with only these groups of non-zero β_k^j's using an efficient optimization technique available for overlapping group lasso.

Let us characterize the zero patterns induced by ℓ_1/ℓ_2 norms in problem (2). We first consider a block of $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$ which consists of one input group ($\mathbf{g}_m \in \mathcal{G}$) and one output group ($\mathbf{h}_o \in \mathcal{H}$). Since each group can be zero simultaneously ($\boldsymbol{\beta}_k^{\mathbf{g}_m} = \mathbf{0}$, $\boldsymbol{\beta}_{\mathbf{h}_o}^j = \mathbf{0}$), there exist zero patterns for $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$ when $\boldsymbol{\beta}_k^{\mathbf{g}_m} = \mathbf{0}$, $\forall k \in \mathbf{h}_o$ or $\boldsymbol{\beta}_{\mathbf{h}_o}^j = \mathbf{0}$, $\forall j \in \mathbf{g}_m$. Furthermore, the union of multiple $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$'s can generate zero patterns for $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} = \mathbf{0}$ which consists of multiple input groups and multiple output groups, $\{\mathbf{h}_o\} \in \mathbf{H}$ and $\{\mathbf{g}_m\} \in \mathbf{G}$. One might be able to check these zero patterns by checking optimality conditions for each $\boldsymbol{\beta}_k^{\mathbf{g}_m} = \mathbf{0}$ and $\boldsymbol{\beta}_{\mathbf{h}_o}^j = \mathbf{0}$. However, this approach may be inefficient as it needs to examine a large number of groups. Instead, to efficiently check the zero patterns, we will test multiple groups simultaneously (i.e., all groups in $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$ or $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$). Also, we will construct a hierarchical tree, and exploit the inclusion relations between the zero patterns so that we can identify zero groups efficiently by traversing the tree while avoiding unnecessary optimality checks.

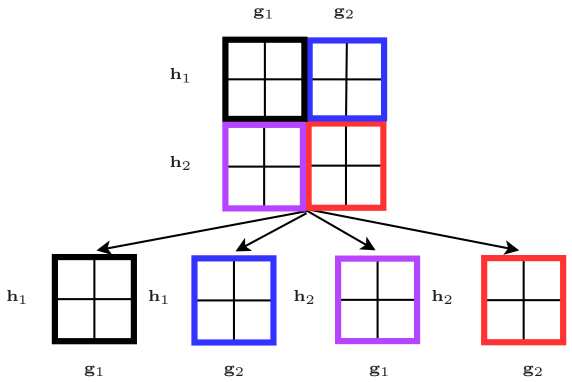


Figure 1: An example of a tree that contains $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$, where $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2\}$, and $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2\}$. The root node contains zero pattern for $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} = \mathbf{0}$, and the leaf nodes represent the zero patterns for $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$.

In Figure 1, we show an example of the tree for $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$ when $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2\}$, $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2\}$, and $|\mathbf{g}_1| = |\mathbf{g}_2| = |\mathbf{h}_1| = |\mathbf{h}_2| = 2$. We denote the set of zero patterns of $\mathbf{B}$ (i.e., $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$'s or $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$'s) by $\mathcal{Z} = \{Z_1, \ldots, Z_{|\mathcal{Z}|}\}$. For example, Z_1 can be a zero pattern for $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} = \mathbf{0}$ (the root node in Figure 1). Let us denote $\mathbf{B}(Z_t)$ by the coefficients of $\mathbf{B}$ corresponding to Z_t's zero pattern. Then we define a tree as follows. A node is represented by $Z \in \mathcal{Z}$, and there exists a directed edge from $Z_1 \in \mathcal{Z}$ to $Z_2 \in \mathcal{Z}$ if and only if $Z_1 \supset Z_2$ and $\nexists Z \in \mathcal{Z} : Z_1 \supset Z \supset Z_2$. Note that each layer encodes different granularities of sparsity pattern. When we have multiple $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$'s, we can generate a subtree for each $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$ separately, and then connect all the subtrees to the dummy root node for $\mathbf{B} = \mathbf{0}$.

We can observe that our procedure has the following properties. First, by testing zero conditions for each node, we can identify multiple zero groups simultaneously. Second, walking through the tree, if $\mathbf{B}(Z_t) = \mathbf{0}$, we know that all the descendants of Z_t are also zero due to the inclusion relations of the tree. Hence, we can skip to check the optimality conditions that the descendants of Z_t are zero.

Considering these properties, we develop our optimization method for the following reasons. First, if $\mathbf{B}$ is sparse, our method is very efficient since we can skip optimality checks for many zero patterns in $\mathcal{Z}$. Mostly we will check only nodes located at the high levels of the tree. Second, our method is simple to implement. All we need is to check whether each node in the tree attains zero. After identifying zero groups, we solve problem (2) with a small number of non-zero groups of coefficients using an available optimization technique.

Specifically, our hierarchical group-thresholding method has the following procedure:

1. Construct a tree that contains the groups of zero patterns of $\mathbf{B}$ (i.e., $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$ and $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$). In our experiments, we used two input and two output groups for each $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$, i.e., $|\mathbf{H}| = |\mathbf{G}| = 2$.

2. Use depth-first-search (DFS) to traverse the tree, and check optimality conditions to see if the zero patterns at each node Z achieve zero. If Z satisfies the optimality condition to be zero, skip the descendants of Z, and visit the next node according to the DFS order.

3. With the groups of β_k^js which did not achieve zero in the previous step, we solve problem (2) using an available optimization algorithm for overlapping group lasso. We used FoGLasso [20] for this step.

In the next section, we show two main ingredients of our optimization method that include 1) the construction of a hierarchical tree, and 2) the optimality condition of each $Z \in \mathcal{Z}$ in the tree.

3.1 Construction of Hierarchical Tree

Here we consider each $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$ separately. We first generate a tree for each $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$, and then combine them to make a single tree. In each block of $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$, we examine the zero patterns of $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$, which are included in $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$, $\{\mathbf{g}_m\} \in \mathbf{G}$, $\{\mathbf{h}_o\} \in \mathbf{H}$. These zero patterns of $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$ are shown in the leaf nodes in Figure 1. Note that even though we present a two-level tree throughout this paper, one can design a tree with multiple levels. Then we need to determine the edges of the tree by investigating the relations of the nodes. Given their relations between $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$ and $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$ (i.e., $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} \supset \mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m}$), we create a directed edge $Z_1 \rightarrow Z_2$. Finally, we make a dummy root node and generate an edge from the dummy node to the roots of all subtrees for $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} = \mathbf{0}$.

3.2 Screening Rules for Multiple Groups

We present rules for checking zero conditions of each node in the tree. We start with optimality condition for problem (2) by computing a subgradient of its objective function with respect to β_k^j and set it to zero:

$$\left(\mathbf{y}_k - \boldsymbol{\beta}_k\mathbf{X}\right)\left(\mathbf{x}_j\right)^T = \lambda_1 s_k^j + \lambda_2 c_k^j + \lambda_3 d_k^j, \tag{4}$$

where s_k^j, c_k^j and d_k^j are a subgradient of penalties in problem (2) with respect to β_k^j.

We first show a rule for identifying $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$ which includes $|\mathbf{h}_o|$ output groups and $|\mathbf{g}_m|$ input groups of coefficients. We assume that our algorithm starts with $\mathbf{B} = \mathbf{0}$, and set $\boldsymbol{\beta}_k\mathbf{X} = \mathbf{0}$, $\forall k$. Under the assumption, we can test $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$ separately using Eq. (4).

Proposition 1 $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$ *if* $\sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}\left|\mathbf{y}_k(\mathbf{x}_j)^T - \lambda_1 s_k^j\right| \leq \left|\lambda_2\sqrt{|\mathbf{h}_o|} - \lambda_3\sqrt{|\mathbf{g}_m|}\right|$ *where*

$$s_k^j = \begin{cases} \frac{\mathbf{y}_k(\mathbf{x}_j)^T}{\lambda_1} & \text{if } \left|\mathbf{y}_k(\mathbf{x}_j)^T\right| \leq \lambda_1 \\ sign\left(\mathbf{y}_k(\mathbf{x}_j)^T\right) & \text{if } \left|\mathbf{y}_k(\mathbf{x}_j)^T\right| > \lambda_1. \end{cases}$$

Proof From the optimality condition in (4), $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$ if

$$\begin{aligned} \sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}\left\{\mathbf{y}_k(\mathbf{x}_j)^T - \lambda_1 s_k^j\right\}^2 &= \sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}\left\{\lambda_2(c_k^j) + \lambda_3(d_k^j)\right\}^2 \\ &\leq \lambda_2^2|\mathbf{h}_o| + \lambda_3^2|\mathbf{g}_m| + 2\lambda_2\lambda_3\sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}(c_k^j)(d_k^j) \\ &\leq \left(\lambda_2\sqrt{|\mathbf{h}_o|} - \lambda_3\sqrt{|\mathbf{g}_m|}\right)^2. \end{aligned}$$

Here we used the fact that $\sum_{j\in\mathbf{g}_m}(c_k^j)^2 \leq 1$, $\sum_{k\in\mathbf{h}_o}(d_k^j)^2 \leq 1$ and $\left|\sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}(c_k^j)(d_k^j)\right|^2 \leq \sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}(c_k^j)^2\sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_o}(d_k^j)^2 \leq |\mathbf{g}_m||\mathbf{h}_o|$ by Cauchy-Schwarz inequality. The above inequality holds since $-\sqrt{|\mathbf{g}_m||\mathbf{h}_o|} \leq \sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}(c_k^j)(d_k^j)$. Also, $s_k^j \in [-1, 1]$ is determined to minimize the left-hand side of the inequality, which is equivalent to applying soft-thresholding to $\mathbf{y}_k(\mathbf{x}_j)^T$. □

Note that Proposition 1 becomes the condition to identify a zero group for overlapping group lasso when $\lambda_3 = 0$ and $K = 1$ (Lemma 2 in [20]). Based on Proposition 1, we further propose a rule for identifying $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} = \mathbf{0}, \{\mathbf{g_m}\} \in \mathbf{G}, \{\mathbf{h_o}\} \in \mathbf{H}$ as follows:

$$\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} = \mathbf{0} \text{ if } \sum_{k\in\mathbf{h}_o,\mathbf{h}_o\in\mathbf{H}} \sum_{j\in\mathbf{g}_m,\mathbf{g}_m\in\mathbf{G}} \left|\mathbf{y}_k(\mathbf{x}_j)^T - \lambda_1 s_k^j\right| \le \sum_{\mathbf{h}_o\in\mathbf{H}} \sum_{\mathbf{g}_m\in\mathbf{G}} \left|\lambda_2\sqrt{|\mathbf{h}_o|} - \lambda_3\sqrt{|\mathbf{g}_m|}\right|. \tag{5}$$

This rule does not guarantee that optimality conditions hold for $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$ for all $\mathbf{h}_o \in \mathbf{H}$ and $\mathbf{g}_m \in \mathbf{G}$. However, in all of our experiments, we observed no violations when $|\mathbf{G}| = |\mathbf{H}| = 2$, and it was very efficient to identify a large number of zero groups simultaneously. Here we give some motivation for this rule. Let us denote $\sum_{k\in\mathbf{h}_o}\sum_{j\in\mathbf{g}_m}\left|\mathbf{y}_k(\mathbf{x}_j)^T - \lambda_1 s_k^j\right|$ by L_{om}, and $\left|\lambda_2\sqrt{|\mathbf{h}_o|} - \lambda_3\sqrt{|\mathbf{g}_m|}\right|$ by R_{om}. If $L_{om} \le R_{om}$ for all (o, m), this rule is satisfied, and it correctly discards groups in $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$. Now we claim that if $L_{om} > R_{om}$ for some (o, m) (there exist some nonzero blocks, i.e., $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} \neq \mathbf{0}$), $\Pr(\sum_{o,m} L_{om} \le \sum_{o,m} R_{om})$ is small, and we are unlikely to discard nonzero blocks. Suppose $L_{om} \sim \mathcal{N}(\gamma, \sigma)$ if $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} = \mathbf{0}$, and $L_{om} \sim \mathcal{N}(\tau, \sigma)$ if $\mathbf{B}_{\mathbf{h}_o}^{\mathbf{g}_m} \neq \mathbf{0}$, where σ is a constant, and $0 < \gamma < R_{om} \le (1 + S/Q)R_{om} << \tau$. Here S and Q are the number of zero and nonzero blocks in $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}}$, respectively, and thus $S + Q = |\mathbf{H}||\mathbf{G}|$. Then, by Hoeffding's inequality, $\Pr(\sum_{o,m} L_{om} \le \sum_{o,m} R_{om}) \le \exp\left\{-2\left(\mathbb{E}(\sum_{o,m} L_{om}) - \sum_{o,m} R_{om}\right)^2 / C\right\}$, where C is a constant. We can see that if $\mathbf{B}_{\mathbf{H}}^{\mathbf{G}} \neq \mathbf{0}$, $\Pr(\sum_{o,m} L_{om} \le \sum_{o,m} R_{om})$ is likely to be small since $\mathbb{E}(\sum_{o,m} L_{om}) = \tau Q + \gamma S >> (1 + S/Q)\sup\{R_{om}\}Q + \gamma S > \sum_{o,m} R_{om}$. Therefore, the rule in (5) would work well when S/Q is small since the assumption for τ can be weak. However, it should be noted that if $|\mathbf{G}| + |\mathbf{H}|$ is large, S/Q can be very large ($S >> Q$), and the assumption for τ becomes too strong. As a result, this rule may be violated if we test very large blocks.

From computational perspective, the rule in (5) significantly decreases the number of iterations for identifying zero groups as we can test a block of coefficients consisting of multiple input groups and multiple output groups. Note that each test can be performed very efficiently by summation of elements in a pre-computed matrix, and the speed for each test can potentially be further improved by GPU [2].

4. EXPERIMENTS

In this section, we show the efficiency and accuracy of our proposed method using simulated datasets, and present its usefulness for eQTL mapping, an important application in bioinformatics. We also present comparison between our optimization method and three other competitors including Fast overlapping Group Lasso (FoGLasso) [20], Smoothing Proximal Gradient method (SPG) [5], and Structured Lasso algorithm (SLasso) [9]. Note that FoGLasso is a state-of-the-art method for overlapping group lasso, and Yuan et al. showed that FoGLasso is significantly faster than other alternative methods [20].

We designed our experiments as follows. In section 4.1, we first present the efficiency of screening step in our method for a wide range of tuning parameters. Then, we present the speed and accuracy of our method under various settings in comparison to other methods. Finally, we confirm the usefulness of our method by showing an interesting interaction effect between a pair of genetic variants in yeast that we identified using our method.

4.1 Evaluation of Efficiency of Our Method Via Simulation Study

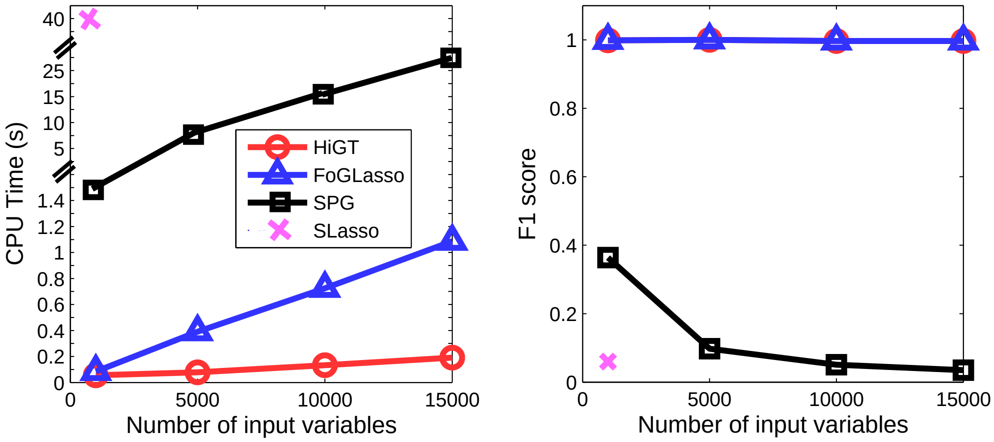
To systematically evaluate the efficiency of our method, we generated simulated datasets as follows. For generating $\mathbf{X} \in \mathbb{R}^{J \times N}$, we first selected J input covariates from a uniform distribution over $[0, 1]$ for N samples. Then we defined input and output groups as follows. For input groups, we selected the size of input groups from a uniform distribution over $[5, 10]$, denoted by $\mathcal{U}(5, 10)$, and the size of overlapping inputs between two consecutive groups was selected from $\mathcal{U}(1, 4)$. For output groups, the size of output groups was selected from $\mathcal{U}(3, 5)$, and the size of overlap with the previous output group was drawn from $\mathcal{U}(1, 2)$. We then simulated $\mathbf{B} \in \mathbb{R}^{K \times J}$, i.e, the ground-truth coefficients, which includes 52 nonzero coefficients ($\beta_k^j = 3$). Given $\mathbf{X}$ and $\mathbf{B}$, we generated K outputs by $\mathbf{Y} = \mathbf{B}\mathbf{X} + \mathbf{E}$, $\mathbf{E} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We generated 10 different datasets for each simulation setting with N, K and J, and report the average CPU time and average accuracy using F1 score, which is harmonic mean of precision and recall rates. Given an estimated $\mathbf{B}$, precision is defined by the ratio of the number of correctly found nonzero coefficients to the total number of estimated nonzero coefficients, and recall is denoted by the number of correctly found nonzero coefficients divided by the total number of true nonzero coefficients. Throughout all the experiments, we employed a two-level tree (excluding the dummy root node), where the nodes at the first level contain a block of coefficients consisting of two input groups and two output groups, and the leaf nodes include a block of cofficients with one input group and one output group.

Evaluation of Efficiency of Screening Step Via Simulation Study We first evaluate the efficiency of screening step (the first step in Algorithm 1) for a range of tuning parameters $\{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ using simulation datasets with $N = 1000$, $J = 5000$ and $K = 5$, and Table 1 shows the results. For simplicity, we set $\lambda_1 = \lambda_2 = \lambda_3$ denoted by λ. From the table, we can observe that screening time drops significantly as λ changes from 0.05 to 0.02, which indicates that many coefficients were discarded in the first level of our hierarchical tree. Indeed, the number of selected groups was decreased from 17071 to 116 without missing true nonzero coefficients. It should be noted that the updating time (the second step in Algorithm 1) was also substantially reduced when λ is changed from 0.02 to 0.05 due to the small number of groups selected by screening step. Thus, our algorithm became very efficient from $\lambda = 0.05$ since both screening and updating step were very fast. For large tuning parameters (e.g. $\lambda \geq 0.2$), we started to miss true coefficients, and when $\lambda = 0.5$, all coefficients were set to zero due to heavy penalization. We can observe that $\lambda = 0.05$ or 0.1 are appropriate for our simulation datasets, and in the following experiments, we will use these two tuning parameters.

Evaluation of Speed and Induced Structured Sparsity Via Simulation Study We compared the speed and the accuracy of our HiGT method with the three alternatives of FoGLasso, SPG and SLasso. We first show the results under a single task regression setting where $\lambda_3 = 0$, and $K = 1$ (this setting was used in previous papers for FoGLasso, SPG and SLasso). Figure 2(a,b) show CPU time and F1 score of the four methods with different number of input variables from 1000 to 20000, fixing $N = 1000$, $K = 1, \lambda_1 = \lambda_2 = 0.05$, and

Table 1: Efficiency of our screening step for a range of tuning parameters. For comparison, CPU time for updating step (the second step in Algorithm 1) is also presented. The fourth column denotes the number of groups selected by our screening step (total number of groups: 19152), and the last column represents the number of true nonzero coefficient discarded by our screening step.

$\lambda_1 = \lambda_2 = \lambda_3$	Screening Time (s)	Updating Time (s)	# Selected Groups	# Missing $\beta_k^j \neq 0$
0.001	0.465	13.246	19152	0
0.002	0.482	13.497	19152	0
0.005	0.470	12.515	19151	0
0.01	0.481	9.343	19124	0
0.02	0.476	6.018	17071	0
0.05	0.246	0.022	116	0
0.1	0.255	0.010	51	0
0.2	0.249	0.003	16	20
0.5	0.239	0	0	52



(a)

(b)

Figure 2: (a) CPU time and (b) F1 score comparison of our proposed HiGT method, FoGLasso, SPG, and SLasso with different number of input variables under a single task regression setting. We used simulation datasets with $N = 1000$, $K = 1$, $\lambda_1 = \lambda_2 = 0.05$, and $\lambda_3 = 0$.

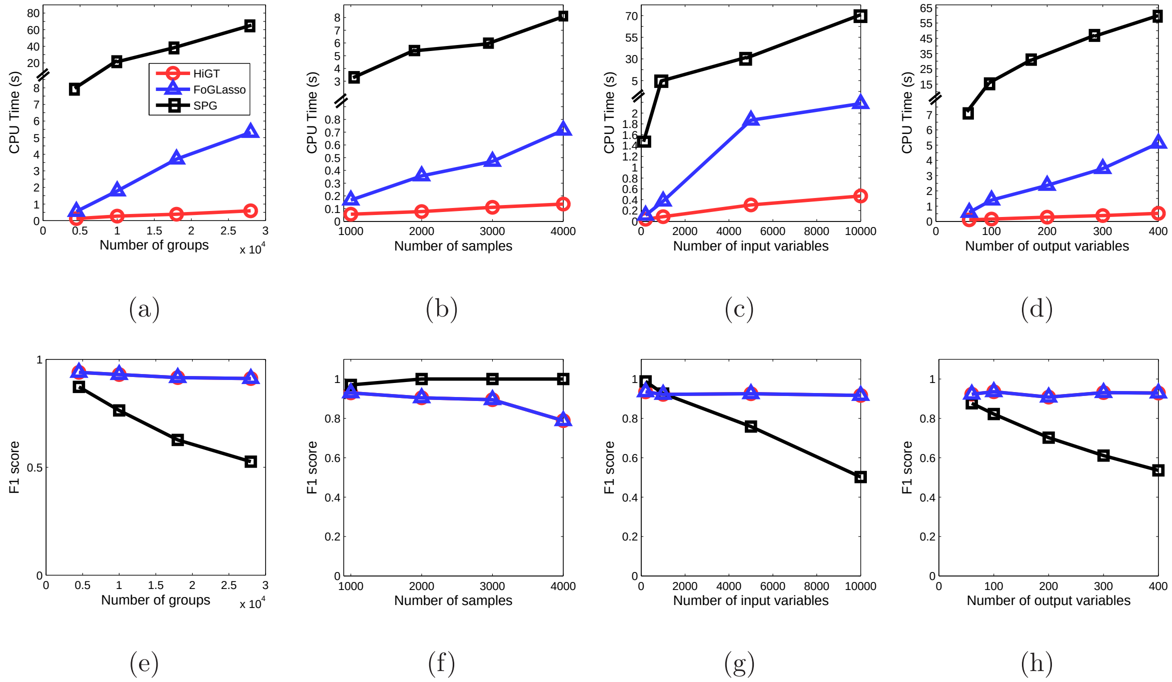
$\lambda_3 = 0$. We observed that our method was much more scalable than other methods, and perfectly recovered true nonzero coefficients. FoGLasso achieved the same accuracy but it was not as fast as HiGT due to the lack of hierarchical group screening step. In the following comparison analysis, we included only FoGLasso and SPG which showed good performance.

Figure 3 shows efficiency and F1 score of three methods including HiGT, FoGLasso, and SPG under various simulation settings. For all experiments, we set $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$. From this figure, we can observe the following:

- For all settings with different number of groups, samples, input and output variables, our algorithm was much more efficient than the other methods.

- Our HiGT algorithm and FoGLasso showed the same F1 score (close to 1) for all simulation settings.

- Screening step in our algorithm never made a mistake for all experiments.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3: CPU time and F1 score comparison of our proposed HiGT method, FoGLasso, and SPG with different (a,e) number of groups ($N = 1000$), (b,f) samples ($J = 500, K = 5$), (c,g) input variables ($N = 1000, K = 5$), and (d,h) output variables ($N = 1000, J = 150$).

- In general, the accuracy of HiGT and FoGLasso did not decrease as the problem size increased.

- For all methods, CPU time increased linearly with the number of groups but the slopes were significantly different. Our HiGT method has a very small slope due to the efficient screening step.

4.2 Detecting eQTLs Having Interaction Effects in Yeast Genome

We also solved problem (2) using our HiGT method with yeast data [4] which contains 1260 unique SNPs and observed gene-expression levels of 5637 genes. To show the usefulness of our method, we briefly report the most significant eQTLs having interaction effects that we identified (chr1:154328-chr5:350744). According to our estimation, it turns out this pair of genetic variants affected 455 genes enriched with the GO category of ribosome biogenesis with corrected p-value $< 10^{-35}$. This SNP pair was very closely located on gene NUP60 and gene RAD51, respectively, and we found that there exists a significant genetic interaction between the two genes [6]. As both SNPs are closely located to NUP60 and RAD51 (within 500bp), we can assume that the two SNPs affected the two genes (NUP60 and RAD51), and their genetic interaction in turn acted on a large number of genes related to ribosome biogenesis. It implies that this pair of SNPs can be a truly meaningful biological finding. We consider that our detection of this SNP pair is novel as the exact locations of the SNP pair were not reported in both Storey et al. [18] and a statistical test for pairwise interactions [16].

5. DISCUSSIONS

In this paper, we presented an efficient algorithm for a large-scale overlapping group lasso problem in highly general settings. Our method relies on a screening step which can efficiently discard a large number of irrelevant groups simultaneously. Our simulation confirmed that our model is significantly faster than other competitors while maintaining high accuracy. In our analysis of yeast eQTL datasets, we reported a pair of genetic variants that potentially interact with each other and influence on ribosome biogenesis.

One of promising research directions of this work would be to consider parallelization of our method. Note that we can naturally parallelize the screening step as it considers a set of groups separately. However, the second step of our algorithm needs to be performed sequentially after the screening step is completed. A efficiently parallelized algorithm would not only further speed up the algorithm but also allow us to deal with very large problems which cannot fit into memory. We are also interested in theoretical analysis of our screening step in terms of sure screening property for ultra high dimensional problems [7] or the properties of strong rules for discarding covariates [19]. Finally, we plan to apply our efficient algorithm to very large-scale eQTL mapping problems in bioinformatics for understanding the biological mechanisms of complex human diseases.

REFERENCES

[1] A. Argyriou, C.A. Micchelli, M. Pontil, L. Shen, and Y. Xu. Efficient first order methods for linear composite regularizers. *Arxiv preprint arXiv:1104.1436*, 2011.

[2] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 917–924. ACM, 2003.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–124, 2011.

[4] R.B. Brem and L. Kruglyak. The landscape of genetic complexity across 5,700 gene expression traits in yeast. *PNAS*, 102(5):1572, 2005.

[5] X. Chen, Q. Lin, S. Kim, and E.P. Xing. An efficient proximal-gradient method for single and multi-task regression with structured sparsity. *Annals of Applied Statistics*, 2010.

[6] M. Costanzo et al. The genetic landscape of a cell. *Science*, 327(5964):425, 2010.

[7] J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

[8] L. Jacob, G. Obozinski, and J.P. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM, 2009.

[9] R. Jenatton, J.Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12:2777–2824, 2011.

[10] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334, 2011.

[11] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. *Advances in Neural Information Processing Systems*, 2010.

[12] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Convex and network flow optimization for structured sparsity. *Journal of Machine Learning Research*, 12:2681–2720, 2011.

[13] L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[14] S. Mosci, S. Villa, A. Verri, and L. Rosasco. A primal-dual algorithm for group sparse regularization with overlapping groups. In *Neural Information Processing Systems*, 2010.

[15] S.N. Negahban and M.J. Wainwright. Simultaneous support recovery in high dimensions: Benefits and perils of block ℓ_1/ℓ_∞-regularization. *Information Theory, IEEE Transactions on*, 57(6):3841–3863, 2011.

[16] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M.A.R. Ferreira, D. Bender, J. Maller, P. Sklar, P.I.W. de Bakker, M.J. Daly, et al. PLINK: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81(3):559–575, 2007.

[17] Z. Qin and D. Goldfarb. Structured sparsity via alternating directions methods. *ArXiv e-prints*, 2011.

[18] J.D. Storey, J.M. Akey, L. Kruglyak, et al. Multiple locus linkage analysis of genomewide expression in yeast. *PLoS Biology*, 3(8):1380, 2005.

[19] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R.J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2011.

[20] L. Yuan, J. Liu, and J. Ye. Efficient methods for overlapping group lasso. *Advances in Neural Information Processing Systems*, 2011.

Algorithm 1 Hierarchical Group Thresholding (HiGT) algorithm

```
$\mathcal{G} \leftarrow$ groups of inputs; $\mathcal{H} \leftarrow$ groups of outputs
$T(\mathcal{Z}, \mathcal{E}) \leftarrow$ a hierarchical tree with groups of zero patterns (see Section 3.1)
$\{Z_{(1)}, Z_{(2)}, \ldots, Z_{(|\mathcal{Z}|)}\} \leftarrow$ DFS order of $\mathcal{Z}$ in $T(\mathcal{Z}, \mathcal{E})$

(1. Screening Step)
$V \leftarrow \emptyset$
$t \leftarrow 1$
while $t \leq |\mathcal{Z}|$ do
    if $Z_{(t)}$ corresponds to $\mathbf{B}^{\mathbf{G}}_{\mathbf{H}} = \mathbf{0}$ then
        p $\leftarrow$ Rule in (5)
    else if $Z_{(t)}$ corresponds to $\mathbf{B}^{\mathbf{g}_m}_{\mathbf{h}_o} = \mathbf{0}$ then
        p $\leftarrow$ Rule in Proposition 1
    else
        $t \leftarrow t + 1$; continue; (Skip dummy root node)
    end if
    if p holds (condition for $\mathbf{B}(Z_{(t)}) = \mathbf{0}$) then
        $t \leftarrow$ DFS order of $t'$ such that $Z_{(t')}$ is not a descendant of $Z_{(t)}$, $t' > t$ and $\nexists t'' : t' > t'' > t$
        (Skip the descendants of $Z_{(t)}$)
    else if p = Rule in Proposition 1 then
        $V \leftarrow V \cup$ groups in $Z_{(t)}$ (Keep the groups in $Z_{(t)}$)
        $t \leftarrow t + 1$
    else
        $t \leftarrow t + 1$
    end if
end while

(2. Updating Step)
With the coefficients in $V$ and their corresponding groupings in $\mathcal{G}$ and $\mathcal{H}$, we optimize problem (2) using
an efficient optimization technique for overlapping group lasso (We used FoGLasso [20] for this step).
```