An Efficient Proximal-Gradient Method for Single and Multi-task Regression with Structured Sparsity

Xi Chen[1] **Qihang Lin**[2] **Seyoung Kim**[1]
Javier Peña[2] **Jaime G. Carbonell**[1]
Eric P. Xing[1*]

[1] School of Computer Science
[2] Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We consider the optimization problem of learning regression models with a mixed-norm penalty that is defined over overlapping groups to achieve structured sparsity. It has been previously shown that such penalty can encode prior knowledge on the input or output structure to learn an structured-sparsity pattern in the regression parameters. However, because of the non-separability of the parameters of the overlapping groups, developing an efficient optimization method has remained a challenge. An existing method casts this problem as a second-order cone programming (SOCP) and solves it by interior-point methods. However, this approach is computationally expensive even for problems of moderate size. In this paper, we propose an efficient proximal-gradient method that achieves a faster convergence rate and is much more efficient and scalable than solving the SOCP formulation. Our method exploits the structure of the non-smooth structured-sparsity-inducing norm, introduces its smooth approximation, and solves this approximation function instead of optimizing the original objective function directly. We demonstrate the efficiency and scalability of our method on simulated datasets and show that our method can be successfully applied to a very large-scale dataset in genetic association analysis.

Keywords: structured sparsity, overlapping group structure, proximal-gradient method, multi-task sparse learning

1 Introduction

The problem of high-dimensional sparse feature learning arises in many areas in science and engineering. In a typical setting, the input lies in a high-dimensional space, and we are interested in

*To whom correspondence should be addressed: epxing@cs.cmu.edu

selecting a small number of input features that influence the output. Although a popular approach has been to optimize the loss function with an ℓ_1-norm penalization (e.g., lasso [13]) to shrink the parameters to zero for the irrelevant input features , this does not take advantage of any prior knowledge on the structure of the inputs to learn *structured-sparsity* pattern in the estimated parameters. In the simple case of a non-overlapping group structure over the inputs, group lasso with a ℓ_1/ℓ_2-norm penalty has been used to learn structured-sparsity pattern in which multiple inputs in a cluster are jointly relevant to the output [14]. Recently, in order to handle a more general structure, the original group lasso with non-overlapping groups has been extended to overlapping groups [15, 6]. For example, when inputs are organized as a tree with a multi-level clustering structure, a general *structured-sparsity-inducing norm* penalty can allow the sparsity pattern in the parameters to reflect overlapping groups of the complex input structure. Similar ideas have been used in multi-task learning in which tasks (or outputs) are organized *a priori* into a particular structure and the closely related tasks according to this structure share a similar sparsity pattern for relevant features [7].

A practical challenge in using any structured-sparsity-inducing norms with overlapping groups is to develop an efficient optimization method. In the simple case of group lasso with non-overlapping groups, the optimization is relatively straightforward in that the block coordinate-descent algorithm [4, 9] can be applied that computes the subgradient and then update parameters for each group iteratively according to its closed-form update equation. In contrast, when the groups overlap, this block coordinate-descent method cannot be applied because the subgradient with respect to each group has a complex form and a closed-form update equation cannot be obtained. Instead, the most widely adopted method is to formulate the problem as a second-order cone programming (SOCP) and solve it by the interior-point method (IPM). This approach is computationally very expensive [8]. To improve the scalability, very recently, an unpublished manuscript [6] proposed an active-set algorithm which solves a sequence of subproblems with a smaller set of "active" variables. However, this method can only solve the regression problems regularized by the *square* of the structured-sparsity-inducing norm. In addition, this method formulates each subproblem either as an SOCP, which can be computationally expensive for a large active set, or as a jointly convex (but still non-convex as a whole) problem with auxiliary variables, which is then solved by an alternating gradient descent. This latter approach lacks the guarantee in optimization convergence and may lead to numerical problems.

In this paper, we propose an efficient proximal-gradient method for estimating regression parameters with the overlapping group structure encoded in the structured-sparsity-inducing norm for both single and multi-task learning settings. Our approach is called "proximal" gradient method in that instead of optimizing the original problem directly, we introduce a smooth approximation of the structured-sparsity-inducing norm and then apply the accelerated gradient method [11]. Our method can achieve $O(\frac{1}{\epsilon})$ convergence rate for a desired accuracy ϵ. In other words, it can find a solution $\boldsymbol{\beta}^t$ for minimizing f function after $t = O(\frac{1}{\epsilon})$ iterations such that $f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*) \leq \epsilon$, where $\boldsymbol{\beta}^*$ is the optimal solution. There are several advantages in using our proximal-gradient method: (a) Our method is a first-order method, i.e. it only uses the gradient information. Thus, it is significantly more efficient and scalable than IPM for SOCP. (b) Our method is a general approach that can be used to solve any optimization problems with a smooth convex loss and a

structured-sparsity-inducing norm with overlapping groups, including both single-task and multi-task regressions. (c) Theoretically, our optimization method has a faster convergence rate of $O(\frac{1}{\epsilon})$ than the subgradient method with a convergence rate of $O(\frac{1}{\epsilon^2})$. In fact, for the optimization problems with overlapping groups that we consider in this paper, there are no implementations of the subgradient method available in the literature. (d) Our method is easy to implement with only a few lines of MATLAB code.

The rest of this paper is organized as follows. In Section 2, we present our proximal-gradient method for structured sparse feature learning for univariate-regression along with the complexity results. In Section 3, we present the generalization of our algorithm to the multi-task learning setting. In Section 4, we present numerical results on both simulated and genetic association datasets, followed by conclusions in Section 5.

2 Proximal-Gradient Method for Univariate-Response Regression with Structured Sparsity

In this section, we introduce our proximal-gradient method for the sparse univariate-response (also called single-task) regression problems with overlapping group structure over the inputs encoded in the structured-sparsity-inducing norm. Then, in the next section, we will show how this method can be applied to a multi-task setting.

First, we define our sparse regression problem and structured-sparsity-inducing norm. Given the input data $\mathbf{X} \in \mathbb{R}^{N \times J}$ for J input features and the output data $\mathbf{y} \in \mathbb{R}^{N \times 1}$ for N samples, we assume a linear-regression model,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{\beta}$ is the vector of regression coefficients and $\boldsymbol{\epsilon}$ is the vector of length N for noise distributed as $N(0, \sigma^2 I_{N \times N})$. We study the following optimization problem that minimizes the squared-error loss with a penalty function:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^J} f(\boldsymbol{\beta}) \equiv \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \Omega(\boldsymbol{\beta}). \tag{1}$$

While various forms of the penalty function $\Omega(\boldsymbol{\beta})$ such as ℓ_1-norm, ℓ_2-norm, and ℓ_1/ℓ_2-norm have been considered in literature, in this paper, we are interested in taking advantage of the available structural information in the inputs to achieve structured sparsity in $\boldsymbol{\beta}$, and focus on the structured-sparsity-inducing norm $\Omega(\boldsymbol{\beta})$ to encode the overlapping group structures. We assume that the set of groups of inputs $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\}$ is defined as a subset of the power set of $\{1, \ldots, J\}$, and is available as prior knowledge. Note that the members (groups) of $\mathcal{G}$ are allowed to overlap. Then, we define the structured-sparsity-inducing norm as:

$$\Omega(\boldsymbol{\beta}) = \lambda \sum_{g \in \mathcal{G}} w_g \|\boldsymbol{\beta}_g\|_2, \tag{2}$$

where $\boldsymbol{\beta}_g \in \mathbb{R}^{|g|}$ is the subvector of $\boldsymbol{\beta}$ corresponding to the inputs in group g, w_g is the predefined weight for group g, λ is the regularization parameter that controls the sparsity level, and $\|\cdot\|_2$

denotes the vector ℓ_2-norm. A simple strategy for setting w_g is $w_g = \sqrt{|g|}$ as in [14] so that the amount of penalization is adjusted by the size of each group.

We notice that the penalty in (2) has a general form that includes ridge regression, lasso [13], group lasso [14], elastic net [16] and hierarchical mixed-norms [2] as special cases. Although we focus on the squared-error loss in this paper, our oprimization can handle any smooth convex loss functions such as logistic loss. In addition, our method can be easily generalized to solve optimization problems with other variants of (2) (e.g., [6]).

The main difficulty in optimizing (1) arises from the non-separable $\{\boldsymbol{\beta}_g\}_{g\in\mathcal{G}}$ in the non-smooth penalty term $\Omega(\boldsymbol{\beta})$. While the block coordinate descent method can be used for the problem with non-overlapping groups in $\mathcal{G}$, it cannot be applied to the case of overlapping groups because the overlap among $\{\boldsymbol{\beta}_g\}_{g\in\mathcal{G}}$ makes the computation of the subgradient with respect to $\boldsymbol{\beta}_g$ difficult. As we show in this section, the key in our approach is to decouple the overlapped $\{\boldsymbol{\beta}_g\}_{g\in\mathcal{G}}$ into a simple linear transformation of $\boldsymbol{\beta}$ by introducing auxiliary variables. Then, we introduce a smooth approximation to $\Omega(\boldsymbol{\beta})$ such that its gradient with respect to $\boldsymbol{\beta}$ can be easily calculated.

2.1 Reformulation of the Structured-Sparsity-Inducing Norm

Since the dual norm of ℓ_2-norm is also an ℓ_2-norm, we can write $\|\boldsymbol{\beta}_g\|_2$ as $\|\boldsymbol{\beta}_g\|_2 = \max_{\|\boldsymbol{\alpha}_g\|_2\leq 1} \boldsymbol{\alpha}_g^T\boldsymbol{\beta}_g$, where $\boldsymbol{\alpha}_g \in \mathbb{R}^{|g|}$ is the vector of auxiliary variables associated with $\boldsymbol{\beta}_g$. Let $\boldsymbol{\alpha} = \left[\boldsymbol{\alpha}_{g_1}^T, \ldots, \boldsymbol{\alpha}_{g_{|\mathcal{G}|}}^T\right]^T$. Then, $\boldsymbol{\alpha}$ is a vector of length $\sum_{g\in G}|g|$ with domain $\mathcal{Q} \equiv \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha}_g\|_2 \leq 1,\ \forall g \in \mathcal{G}\}$, where $\mathcal{Q}$ is the Cartesian product of unit balls in Euclidean space and thus, a closed and convex set. We can rewrite the structured-sparsity-inducing norm in (2) as:

$$\Omega(\boldsymbol{\beta}) = \lambda \sum_{g\in\mathcal{G}} w_g \max_{\|\boldsymbol{\alpha}_g\|_2\leq 1} \boldsymbol{\alpha}_g^T\boldsymbol{\beta}_g = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} \sum_{g\in\mathcal{G}} \lambda w_g \boldsymbol{\alpha}_g^T\boldsymbol{\beta}_g = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} \boldsymbol{\alpha}^T C\boldsymbol{\beta}, \tag{3}$$

where $C \in \mathbb{R}^{\sum_{g\in\mathcal{G}}|g|\times J}$ is a matrix defined as follows. The rows of C are indexed by all pairs of $(i,g) \in \{(i,g)|i \in g, i \in \{1,\ldots,J\}\}$, the columns are indexed by $j \in \{1,\ldots,J\}$, and each element of C is given as:

$$C_{(i,g),j} = \begin{cases} \lambda w_g & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Then, we have $C\boldsymbol{\beta} = \left[\lambda w_{g_1}\boldsymbol{\beta}_{g_1}^T, \ldots, \lambda w_{g_{|\mathcal{G}|}}\boldsymbol{\beta}_{g_{|\mathcal{G}|}}^T\right]^T$.

Example. We give a concrete example of C. Assume $\boldsymbol{\beta} \in \mathbb{R}^3$, i.e. $J = 3$ with groups $\mathcal{G} = \{g_1 = \{1,2\}, g_2 = \{2,3\}\}$. Then, the matrix C is defined as follows:

$$\begin{array}{c} \\ i=1\in g_1 \\ i=2\in g_1 \\ i=2\in g_2 \\ i=3\in g_2 \end{array} \begin{array}{c} \begin{array}{ccc} j=1 & j=2 & j=3 \end{array} \\ \begin{pmatrix} \lambda w_{g_1} & 0 & 0 \\ 0 & \lambda w_{g_1} & 0 \\ 0 & \lambda w_{g_2} & 0 \\ 0 & 0 & \lambda w_{g_2} \end{pmatrix} \end{array}$$

Note that C is a highly sparse matrix with only a single non-zero element in each row and $\sum_{g\in\mathcal{G}}|g|$ non-zero elements in the entire matrix, and can be stored with only a small amount of memory during the optimization procedure.

According to (3), $\Omega(\boldsymbol{\beta})$ can be viewed as the inner product of the auxiliary variable $\boldsymbol{\alpha}$ and the linear mapping of $\boldsymbol{\beta}$ given as $\Gamma(\boldsymbol{\beta}) \equiv C\boldsymbol{\beta}$, where the linear operator Γ is a mapping from $\mathbb{R}^J$ to $\mathbb{R}^{\sum_{g\in\mathcal{G}}|g|}$. This linear operator allows us to decouple the overlapping $\{\boldsymbol{\beta}_g\}_{g\in\mathcal{G}}$ in (2) while letting $\boldsymbol{\beta}$ appear in its original form in (3). From $\langle C\boldsymbol{\beta}, \boldsymbol{\alpha}\rangle = \langle \boldsymbol{\beta}, C^T\boldsymbol{\alpha}\rangle$, the adjoint operator of Γ is $\Gamma^*(\boldsymbol{\alpha}) = C^T\boldsymbol{\alpha}$ that maps $\mathbb{R}^{\sum_{g\in\mathcal{G}}|g|}$ back into $\mathbb{R}^J$. Essentially, the adjoint operator Γ^* is the linear operator induced by Γ in the space of auxiliary variables. The use of Γ and its adjoint Γ^* will simplify our notation and provide a uniform treatment of the single-task and multi-task learning as shown in the later sections.

2.2 Proximal-Gradient Method

The formulation in (3) is still a non-smooth function of $\boldsymbol{\beta}$, and this makes the optimization challenging. To tackle this problem, we introduce an auxiliary strongly-convex function to construct a smooth approximation of (3). More precisely, we define:

$$f_\mu(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} \boldsymbol{\alpha}^T C\boldsymbol{\beta} - \mu d(\boldsymbol{\alpha}), \tag{5}$$

where μ is a positive smoothness parameter and $d(\boldsymbol{\alpha})$ is an arbitrary smooth strongly-convex function defined on $\mathcal{Q}$. The original penalty term can be viewed as $f_\mu(\boldsymbol{\beta})$ with $\mu = 0$ (i.e., $f_0(\boldsymbol{\beta}) = \Omega(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} \boldsymbol{\alpha}^T C\boldsymbol{\beta}$). Since our algorithm will utilize the optimal solution $\boldsymbol{\alpha}^*$ to (5), we choose $d(\boldsymbol{\alpha}) \equiv \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2$ so that we can obtain a closed-form equation for $\boldsymbol{\alpha}^*$.

It is easy to see that $f_\mu(\boldsymbol{\beta})$ is a lower bound of $f_0(\boldsymbol{\beta})$. In order to bound the gap between $f_\mu(\boldsymbol{\beta})$ and $f_0(\boldsymbol{\beta})$, let $D = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} d(\boldsymbol{\alpha})$. It is easy to verify that

$$D = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} \frac{1}{2}\sum_{g\in\mathcal{G}} \|\boldsymbol{\alpha}_g\|_2^2 = |\mathcal{G}|/2. \tag{6}$$

Then, we have $f_0(\boldsymbol{\beta}) - f_\mu(\boldsymbol{\beta}) \leq \mu D = \mu|\mathcal{G}|/2$. From Theorem 1 as presented below, we know that $f_\mu(\boldsymbol{\beta})$ is a smooth function for any $\mu > 0$. Therefore, $f_\mu(\boldsymbol{\beta})$ can be viewed as a smooth approximation of $f_0(\boldsymbol{\beta})$ with the maximum gap of $\mu|\mathcal{G}|/2$, and the μ controls the gap between $f_\mu(\boldsymbol{\beta})$ and $f_0(\boldsymbol{\beta})$. According to the convergence result in the next section, we need to set $\mu = \frac{\epsilon}{2D}$ to achieve the best convergence rate given the desired accuracy ϵ.

Now we present the key theorem. It is also stated in [11] but without a proof of smoothness property and a derivation of the gradient. In this paper, we provide a simple proof based on Fenchel Conjugate and properties of subdifferential. Intuitively, the strong convexity of $d(\boldsymbol{\alpha})$ leads to the smoothness of $f_\mu(\boldsymbol{\beta})$. A vivid geometric illustration and the proof of this theorem are presented in Appendix.

Theorem 1. *For any $\mu > 0$, $f_\mu(\boldsymbol{\beta})$ is a convex and continuously-differentiable function in $\boldsymbol{\beta}$, and the gradient of $f_\mu(\boldsymbol{\beta})$ takes the following form:*

$$\nabla f_\mu(\boldsymbol{\beta}) = \Gamma^*(\boldsymbol{\alpha}^*) = C^T\boldsymbol{\alpha}^*, \tag{7}$$

where Γ^* is the adjoint operator of Γ as defined in Section 2.1, and $\boldsymbol{\alpha}^*$ is the optimal solution to (5)*. Furthermore, the gradient $\nabla f_\mu(\boldsymbol{\beta})$ is Lipschitz continuous with the Lipschitz constant $L_\mu = \frac{1}{\mu}\|\Gamma\|^2$, where $\|\Gamma\|$ is the norm of the linear operator Γ defined as:*

$$\|\Gamma\| \equiv \max_{\|\mathbf{v}\|_2 \leq 1} \|\Gamma(\mathbf{v})\|_2 = \max_{\|\mathbf{v}\|_2 \leq 1} \|C\mathbf{v}\|_2.$$

To compute the $\nabla f_\mu(\boldsymbol{\beta})$ and L_μ, we need to know $\boldsymbol{\alpha}^*$ and $\|\Gamma\|$. We present the closed-form equations for $\boldsymbol{\alpha}^*$ and $\|\Gamma\|$ in the following two lemmas. The proof of Lemma 2 can be found in Appendix.

Lemma 1. *Let $\boldsymbol{\alpha}^*$, which is composed of $\boldsymbol{\alpha}_g^*$'s, be the optimal solution to* (5)*. For any $g \in \mathcal{G}$,*

$$\boldsymbol{\alpha}_g^* = S(\frac{\lambda w_g \boldsymbol{\beta}_g}{\mu}),$$

where S is the shrinkage operator defined for any vector $\mathbf{u}$ as follows:

$$S(\mathbf{u}) = \begin{cases} \frac{\mathbf{u}}{\|\mathbf{u}\|_2} & \|\mathbf{u}\|_2 > 1, \\ \mathbf{u} & \|\mathbf{u}\|_2 \leq 1. \end{cases}$$

Proof. Taking the derivative of (5) with respect to $\boldsymbol{\alpha}$ and setting it to zeros, we obtain $\boldsymbol{\alpha}_g = \frac{\lambda w_g \boldsymbol{\beta}_g}{\mu}$. We project the solution onto the $\mathcal{Q}$ to obtain the optimal solution. □

Lemma 2.

$$\|\Gamma\| = \lambda \max_{j \in \{1,\ldots,J\}} \sqrt{\sum\nolimits_{g \in \mathcal{G} \text{ s.t. } j \in g} (w_g)^2}. \tag{8}$$

Given the results in Theorem 1, now we present our proximal-gradient method. We substitute the penalty term $\Omega(\boldsymbol{\beta})$ in (1) with its smooth approximation $f_\mu(\boldsymbol{\beta})$ and obtain the smooth optimization problem:

$$\min_{\boldsymbol{\beta}} \widetilde{f}(\boldsymbol{\beta}) \equiv \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + f_\mu(\boldsymbol{\beta}).$$

According to Theorem 1, the gradient of $\widetilde{f}(\boldsymbol{\beta})$ is given as:

$$\nabla \widetilde{f}(\boldsymbol{\beta}) = \mathbf{X}^T(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}) + C^T\boldsymbol{\alpha}^*. \tag{9}$$

Moreover, $\nabla \widetilde{f}(\boldsymbol{\beta})$ is Lipschitz continuous with the Lipschitz constant:

$$L = \lambda_{\max}(\mathbf{X}^T\mathbf{X}) + L_\mu = \lambda_{\max}(\mathbf{X}^T\mathbf{X}) + \frac{\|\Gamma\|^2}{\mu}, \tag{10}$$

where $\lambda_{\max}(\mathbf{X}^T\mathbf{X})$ is the largest eigenvalue of $(\mathbf{X}^T\mathbf{X})$.

We call our method a "proximal" method because instead of optimizing the original function $f(\boldsymbol{\beta})$ in (1), we optimize $\widetilde{f}(\boldsymbol{\beta})$, which is a smooth approximation of $f(\boldsymbol{\beta})$. Since $\widetilde{f}(\boldsymbol{\beta})$ is a *smooth*

Algorithm 1 Proximal-Gradient Method for Structured Variable Selection

Input: $\mathbf{X}$, $\mathbf{y}$, Group Structure $\mathcal{G}$, λ, $\{w_g\}_{g\in\mathcal{G}}$, desired accuracy ϵ.
Initialization: Construct C as in (4); compute L as in (10); set $\mu = \frac{\epsilon}{2D} = \frac{\epsilon}{|\mathcal{G}|}$; set $\mathbf{w}^0 = \mathbf{0} \in \mathbb{R}^J$.
Iterate For $t = 0, 1, 2, \ldots$, until convergence of $\boldsymbol{\beta}^t$:

1. Compute $\nabla\widetilde{f}(\mathbf{w}^t)$ according to (9).
2. Perform the gradient descent step: $\boldsymbol{\beta}^t = \mathbf{w}^t - \frac{1}{L}\nabla\widetilde{f}(\mathbf{w}^t)$.
3. Set $\mathbf{z}^t = -\frac{1}{L}\sum_{i=0}^{t}\frac{i+1}{2}\nabla\widetilde{f}(\mathbf{w}^i)$.
4. Set $\mathbf{w}^{t+1} = \frac{t+1}{t+3}\boldsymbol{\beta}^t + \frac{2}{t+3}\mathbf{z}^t$.

Output: $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}^t$.

function, we can adopt the framework of the accelerated gradient-descent method, so called Nesterov's method [11], to minimize $\widetilde{f}(\boldsymbol{\beta})$ as shown in Algorithm 1.

In contrast to the standard gradient-descent algorithm, Algorithm 1 involves updating three sequences, $\{\mathbf{w}^t\}$, $\{\boldsymbol{\beta}^t\}$, and $\{\mathbf{z}^t\}$, where $\boldsymbol{\beta}^t$ is obtained from the gradient-descent update based on $\mathbf{w}^t$ with the stepsize $\frac{1}{L}$, $\mathbf{z}^t$ is the weighted combination of all the previous gradient information, and $\mathbf{w}^{t+1}$ is the convex combination of $\boldsymbol{\beta}^t$ and $\mathbf{z}^t$. Intuitively, the reason why this method is superior to the standard gradient descent is that it utilizes all the gradient information from the first step to the current step for each update, while the standard gradient-descent update is based only on the gradient information at the current step.

2.3 Convergence Rate and Time Complexity

Although we optimize the approximation function $\widetilde{f}$, it can be proven that the $\widehat{\boldsymbol{\beta}}$ obtained from Algorithm 1 is sufficiently close to the optimal solution $\boldsymbol{\beta}^*$ to the original objective function in (1). We present the convergence rate of Algorithm 1 in the next theorem.

Theorem 2. *Let $\boldsymbol{\beta}^*$ be the optimal solution to* (1) *and $\boldsymbol{\beta}^t$ be the approximate solution at the t-th iteration in Algorithm 1. If we require $f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*) \leq \epsilon$ and set $\mu = \frac{\epsilon}{2D}$, then, the number of iterations t is upper-bounded by*

$$\sqrt{\frac{4\|\boldsymbol{\beta}^*\|_2^2}{\epsilon}\left(\lambda_{\max}(\mathbf{X}^T\mathbf{X}) + \frac{2D\|\Gamma\|^2}{\epsilon}\right)}, \tag{11}$$

where D and $\|\Gamma\|$ are given in (6) *and Lemma 2 respectively.*

The key idea behind the proof is to decompose $f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*)$ into three parts: (i) $f(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}^t)$, (ii) $\widetilde{f}(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}^*)$, and (iii) $\widetilde{f}(\boldsymbol{\beta}^*) - f(\boldsymbol{\beta}^*)$. (i) and (iii) can be bounded by the gap of the approximation μD. Since $\widetilde{f}$ is a smooth function, we can bound (ii) by the accuracy bound when applying

the accelerated gradient method to minimize smooth functions [11]. We obtain (11) by balancing these three terms. The details of the proof are presented in Appendix. According to Theorem 2, Algorithm 1 converges in $O\left(\frac{1}{\epsilon}\right)$ iterations, which is substantially faster than the subgradient method with the convergence rate of $O(\frac{1}{\epsilon^2})$. On the other hand, the best convergence result for optimizing the purely smooth function using the first-order method is $O(\frac{1}{\sqrt{\epsilon}})$ [10]. The gap between $O\left(\frac{1}{\epsilon}\right)$ and $O(\frac{1}{\sqrt{\epsilon}})$ is due to the approximation of the non-smooth penalty term. It remains an open question whether we can further boost our algorithm to achieve $O(\frac{1}{\sqrt{\epsilon}})$ convergence rate. In addition, note that setting $\mu = \frac{\epsilon}{h}$ for any $h > 1$ instead of $\mu = \frac{\epsilon}{2D}$ affects the results in (11) only by a constant factor and still achieves $O\left(\frac{1}{\epsilon}\right)$ convergence rate.

As for the time complexity, assuming that we pre-compute and store $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$ with the time complexity of $O(J^2N)$, the main computational cost in each iteration comes from calculating the gradient $\nabla\widetilde{f}(\mathbf{w}_t)$ with the time complexity of $O(J^2 + \sum_{g\in\mathcal{G}}|g|)$. Therefore, the total complexity to achieve ϵ accuracy is $O\left(J^2N + (J^2 + \sum_{g\in\mathcal{G}}|g|)/\epsilon\right)$. In comparison, according to [8], solving SOCP with IPM has the complexity of $O(J + |\mathcal{G}|)^2(N + 2|\mathcal{G}|)$ per iteration.

Remarks (Time complexity) The per-iteration time complexity of our proximal-gradient method does not depend on the sample size N, which can be very large in large-scale data analysis, whereas that of SOCP with IPM grows linearly in N. Moreover, each IPM iteration of SOCP requires significantly more memory to store the Newton linear system.

3 Proximal-Gradient Method for Multi-task Regression with Structured Sparsity

Our proximal-gradient method as presented in the previous section is a general approach for solving any types of regression problems with overlapping group structures encoded in the structured-sparsity-inducing norm. In this section, we show that our method can be applied in a straightforward manner to multi-task learning setting, where the structural information is available for outputs instead of inputs and the overlapping group structure is defined over multiple related response variables.

Let $\mathbf{X} \in \mathbb{R}^{N\times J}$ denote the matrix of input data for J inputs and $\mathbf{Y} \in \mathbb{R}^{N\times K}$ denote the matrix of output data for K outputs collected over N samples. We assume that the k-th column of $\mathbf{Y}$ for the k-th task is generated from a linear model:

$$\mathbf{y}_k = \mathbf{X}\boldsymbol{\beta}_k + \boldsymbol{\epsilon}_k, \quad \forall k = 1, \dots K,$$

where $\boldsymbol{\beta}_k = [\beta_{1k}, \dots, \beta_{Jk}]^T$ is the regression-coefficient vector for the k-th task and $\boldsymbol{\epsilon}_k$ is Gaussian noise. Let $\mathbf{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K] \in \mathbb{R}^{J\times K}$ be the matrix of regression coefficients for all of the K tasks.

Then, the multi-task regression problem can be naturally formulated as the following optimization problem:

$$\min_{\mathbf{B}\in\mathbb{R}^{J\times K}} f(\mathbf{B}) \equiv \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\mathbf{B}\|_F^2 + \Omega(\mathbf{B}), \tag{12}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm and $\Omega(\mathbf{B})$ is a structured-sparsity-inducing norm defined as follows. Assume that we have available a group structure over response variables, denoted by $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\}$, which is a subset of the power set of $\{1, \ldots, K\}$. For any variable $j \in \{1, \ldots, J\}$ and group $g \in \mathcal{G}$, let $\boldsymbol{\beta}_{jg}$ be the vector of regression coefficients $\{\beta_{jk}, k \in g\}$. Then, we define the structured-sparsity-inducing penalty as:

$$\Omega(\mathbf{B}) \equiv \lambda \sum_{j=1}^{J} \sum_{g \in \mathcal{G}} w_g \|\boldsymbol{\beta}_{jg}\|_2. \tag{13}$$

This penalty has a general form that includes many previously studied penalty functions as a special case. For example, the ℓ_1/ℓ_2-norm that has been adopted in multi-task regression is a special case of $\Omega(\mathbf{B})$, where $\mathcal{G}$ consists of a single group of the entire set of tasks $\{1, \ldots, K\}$. Tree-guided group-lasso penalty [7] is another special case of $\Omega(\mathbf{B})$.

Following a technique similar to that in Section 2.1, for each input j and group g, we introduce a vector of auxiliary variables $\boldsymbol{\alpha}_{jg}$ so that $\|\boldsymbol{\beta}_{jg}\|_2 = \max_{\|\boldsymbol{\alpha}_{jg}\|_2 \leq 1} \boldsymbol{\alpha}_{jg}^T \boldsymbol{\beta}_{jg}$. Let

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\alpha}_{1g_1} & \ldots & \boldsymbol{\alpha}_{Jg_1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}_{1g_{|G|}} & \ldots & \boldsymbol{\alpha}_{Jg_{|G|}} \end{pmatrix}.$$

Note that $\mathbf{A}$ is a $(\sum_{g \in \mathcal{G}} |g|) \times J$ matrix with domain $\mathcal{Q} \equiv \{\mathbf{A} \mid \|\boldsymbol{\alpha}_{jg}\|_2 \leq 1, \forall\, j \in \{1, \ldots, J\}, g \in \mathcal{G}\}$. Now, the penalty function in (13) can be written as

$$\Omega(\mathbf{B}) = \lambda \sum_{j=1}^{J} \sum_{g \in \mathcal{G}} w_g \max_{\|\boldsymbol{\alpha}_{jg}\|_2 \leq 1} \boldsymbol{\alpha}_{jg}^T \boldsymbol{\beta}_{jg} = \max_{\mathbf{A} \in \mathcal{Q}} \langle C\mathbf{B}^T, \mathbf{A} \rangle, \tag{14}$$

where $\langle \mathbf{U}, \mathbf{V} \rangle \equiv \mathrm{Tr}(\mathbf{U}^T \mathbf{V})$ denotes a matrix inner product. The matrix $C \in \mathbb{R}^{\sum_{g \in \mathcal{G}} |g| \times K}$ is defined similarly to (4), where rows are indexed by (i, g) such that $i \in \{1, \ldots, K\}, i \in g$, columns are indexed by $k \in \{1, \ldots, K\}$, and the value of each element is set to $C_{(i,g),k} = \lambda w_g$ if $i = k$ and 0 otherwise. The linear operator Γ now becomes $\Gamma(\mathbf{B}) = C\mathbf{B}^T$ with adjoint $\Gamma^*(\mathbf{A}) = \mathbf{A}^T C$.

We introduce the uniform smooth approximation of (14):

$$f_\mu(\mathbf{B}) = \max_{\mathbf{A} \in \mathcal{Q}} \langle C\mathbf{B}^T, \mathbf{A} \rangle - \mu d(\mathbf{A}), \tag{15}$$

where $d(\mathbf{A}) \equiv \frac{1}{2}\|\mathbf{A}\|_F^2$ with the maximum value $D \equiv \max_{\mathbf{A} \in \mathcal{Q}} d(\mathbf{A}) = \frac{J|\mathcal{G}|}{2}$.

Following a proof strategy similar to that in Theorem 1, we can show that $f_\mu(\mathbf{B})$ is convex and smooth with gradient $\nabla f_\mu(\mathbf{B}) = \Gamma^*(\mathbf{A}^*) = (\mathbf{A}^*)^T C$, where $\mathbf{A}^*$ is the optimal solution to (15), composed of $\boldsymbol{\alpha}_{jg}^* = S(\frac{\lambda w_g \boldsymbol{\beta}_{jg}}{\mu})$. In addition, $\nabla f_\mu(\mathbf{B})$ is Lipschitz continuous with the Lipschitz constant $L_\mu = \|\Gamma\|^2/\mu$, where $\|\Gamma\| \equiv \max_{\|\mathbf{V}\|_F \leq 1} \|\Gamma(\mathbf{V})\|_F = \max_{\|\mathbf{V}\|_F \leq 1} \|C\mathbf{V}^T\|_F$. Similar to Lemma 2, we can show that $\|\Gamma\| = \lambda \max_{k \in \{1,\ldots,K\}} \sqrt{\sum_{g \in \mathcal{G}\ s.t.\ k \in g} (w_g)^2}$.

By substituting $\Omega(\mathbf{B})$ in (12) with $f_\mu(\mathbf{B})$, we can adopt Algorithm 1 to solve (12) with convergence rate of $O(\frac{1}{\epsilon})$ iterations. The time complexity per iteration of our method is $O(J^2K +$

$J\sum_{g\in\mathcal{G}}|g|)$, whereas an IPM for SOCP costs $O\left(J^2(K+|\mathcal{G}|)^2(KN+J(|\mathcal{G}|+\sum_{g\in\mathcal{G}}|g|))\right)$ per iteration.

Remarks. (Time Complexity) If we pre-compute and store $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ with the time complexity of $O(J^2N+JKN)$, the per-iteration time complexity of our proximal-gradient method is independent of the number of samples N and linear in K. In contrast, the time complexity of IPM for SOCP is linear in N and cubic in K. Thus, our method has a significantly lower time complexity than solving the SOCP formulation.

4 Experiments

In this section, we evaluate our proximal-gradient method (Prox-Grad) on both synthetic and real datasets, and compare the performance of our method with that of IPM for SOCP formulation using the standard MATLAB package SeDuMi [12].* All of the experiments are performed on a PC with Intel Core 2 Quad Q6600 CPU 2.4GHz CPU and 4GB RAM. The software is written in MATLAB, and we terminate our optimization procedure when the relative change in the objective is below 10^{-6}. We select the tuning parameter λ by three-fold cross-validation, and report the computation time as the CPU time for running the optimization procedure on the entire dataset with the selected λ. We find that setting the desired accuracy $\epsilon = 0.1$ generally gave us a good performance on the recovery of the sparsity pattern, and use this value in all of our experiments to set $\mu = \frac{\epsilon}{2D}$ according to Theorem 2. We assume that for each group g, $w_g = \sqrt{|g|}$ as in [14] for simulation studies.

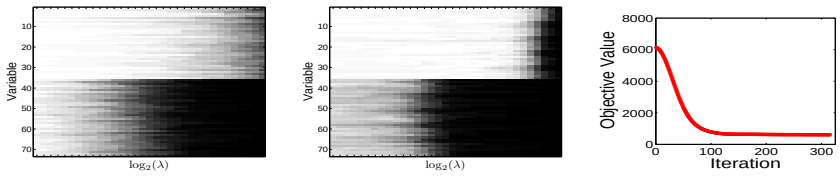
4.1 Synthetic Data

4.1.1 Univariate-Response Sparse Regression

We simulate data for a single-task regression, assuming that the inputs have an overlapping group structure as described below. Assuming that the inputs are ordered, we define a sequence of groups of 10 adjacent inputs with an overlap of three variables between two successive groups so that $\mathcal{G} = \{\{1,\ldots,10\},\{8,\ldots,17\},\ldots,\{J-9,\ldots,J\}\}$ with $J = 7|\mathcal{G}|+3$. We set the support of $\boldsymbol{\beta}$ to the first half of the input variables We sample each element of $\mathbf{X}$ and the non-zero elements of $\boldsymbol{\beta}$ from *i.i.d.* Gaussian distribution, and generate the output data from $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}\sim\mathcal{N}(0, I_{N\times N})$.

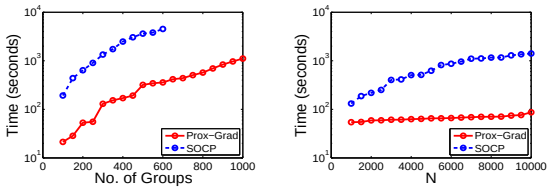
In Figure 1, we compare the performance of lasso and the structured sparse regression method with overlapping groups $\mathcal{G}$ on variable-selection problem. We generate 100 datasets with $N = 200$ and $|\mathcal{G}| = 10$ ($J = 73$) using a fixed $\boldsymbol{\beta}$ and a randomly generated input matrix $\mathbf{X}$ in each repetition, and plot the frequency of selection of each variable (y-axis) as a function of the regularization parameter λ (x-axis). The black pixels represent selections in all of the 100 datasets and the white pixels represent selections in none of the 100 datasets. It can be clearly seen that the structured

*We do not compare with the active-set method in [6] since it optimizes a different objective and requires a heuristic search.



(a) (b) (c)

Figure 1: Results on synthetic data for single-task regression. Frequency of selection of each variable via (a) lasso and (b) the structured sparse regression. (c) Objective values of the Prox-Grad method over iterations.



(a) (b)

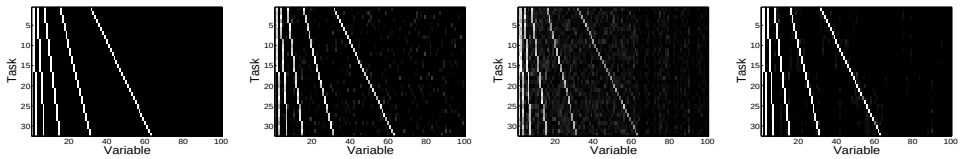
Figure 2: Comparisons of scalability for Prox-Grad and SOCP. (a) Fix $N = 5000$ and vary $|\mathcal{G}|$ from 100 to 1000 with a step size of 50. (b) Fix $|\mathcal{G}| = 200$ and vary N from 1000 to 10000 with a step size of 500. The y-axis denotes the computation time in seconds in logarithmic scale.

variable selection outperforms lasso. The values of the objective function over iterations in a typical run of Prox-Grad is plotted in Figure 1 (c).

To demonstrate the efficiency and scalability of Prox-Grad as compared to SOCP, we present the computation time for datasets with varying N and $|\mathcal{G}|$ in Figure 2. The computation time is measured in seconds and plotted in logarithmic scale. We omit the computation time for the SOCP formulation when it exceeds 2 hours. Clearly, Prox-Grad is more efficient and scalable by orders of magnitude than IPM with SOCP formulation. Moreover, we notice that the increase of N almost does not affect the computation time of Prox-Grad since N affects only the computation time of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$ that can be pre-computed before Prox-Grad iterations start. In contrast, the increase of N leads to an approximately linear increase of the computation time for SOCP. This observation is consistent with our complexity analysis in Section 2.3.

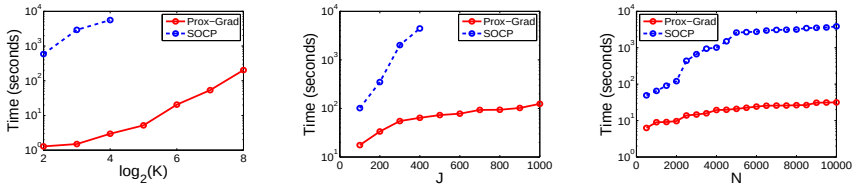
4.1.2 Multi-task Sparse Regression

In this section, we consider the multi-task regression problem, where the structure is defined over the outputs and related outputs share a similar sparsity pattern in their regression coefficients. We assume that K tasks are organized as a perfect binary tree of depth l with leaf nodes corresponding to tasks and internal nodes representing clusters of the tasks for the subtree. Furthermore, we assume that the clusters of the tasks near the bottom of the tree are more closely related as in hierarchical clustering tree and more likely to share the sparsity pattern in their regression coefficients. Given this output structure, we set the sparsity pattern in the true regression-coefficient matrix as shown in Figure 3(a), where the white pixels represent non-zero elements. Then, we sample the



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

Figure 3: Comparison of different regression methods on the recovery of tree sparsity pattern in multi-task regression. (a) The matrix of true regression coefficients $\mathbf{B}$. Estimated regression coefficients are shown for (b) lasso, (c) ℓ_1/ℓ_2-regularized regression, (d) tree-structured sparse regression. The rows and columns represent tasks and inputs, respectively. The white pixels correspond to non-zero regression coefficients.



(a) (b) (c)

Figure 4: Comparisons of scalability of Prox-Grad and SOCP. (a) Fix $N = 1000$ and $J = 600$, and vary $\log_2(K)$ from 2 to 8 with a step size of 1. (b) Fix $N = 1000$ and $K = 32$, and vary J from 100 to 1000 with a step size of 100. (c) Fix $J = 100$ and $K = 32$, and vary N from 500 to 5000 with a step size of 500. The y-axis shows the computation time in seconds in logarithmic scale.

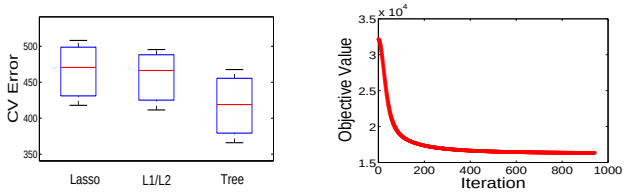
elements of $\mathbf{X}$ from *i.i.d* standard Gaussian and generate the output data using $\mathbf{Y} = \mathbf{XB} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is the standard Gaussian noise.

First, we compare the performance of lasso, the ℓ_1/ℓ_2-regularized multi-task regression [1], and the tree-structured multi-task regression [7] in terms of recovery of true sparsity pattern. We use a dataset simulated with $N = 100, J = 100$, and $K = 32$. For the tree-structured multi-task regression, each node in the tree over the outputs defines a group $g \in \mathcal{G}$ of the tasks in the subtree. See [7] for more details. The recovered regression-coefficient matrix is plotted in Figures 3 (b)–(d). It is visually clear that the tree-structured multi-task regression recovers the true underlying sparsity pattern significantly better than other methods.

We compare the scalability of the proposed Prox-Grad algorithm with that of IPM for SOCP. We simulate datasets with varying K, J and N, and present the computational time on these datasets for our Prox-Grad method and IPM with SOCP formulation in Figure 4. We omit the computation time for SOCP when it exceeds 2 hours. As can be seen in Figure 4, our method is significantly faster than the SOCP formulation and can scale up to a very high-dimensional dataset with many tasks.

4.2 Analysis of Yeast Dataset

We analyze the yeast data with 1,260 genotypes (inputs) and expression levels (outputs) of 3,684 genes collected for 114 yeast strains [3]. We apply the tree-guided group lasso with groups overlapping according to the hierarchical clustering tree over the genes as described in [7].



(a) (b)

Figure 5: Results on yeast dataset. (a) Cross-validated prediction errors. (b) The change in values of the objective function over iterations in Prox-Grad.

We present the boxplot of cross-validated prediction errors in Figure 5 (a). Clearly, the tree-based method has a superior performance to lasso and the ℓ_1/ℓ_2-regularized multi-task regression. The decrease of the values of the objective function over iterations in a typical run of Prox-Grad for tree-guided group lasso is plotted in Figure 5 (b). While it takes approximately an hour for the Prox-Grad method to reach convergence in this dataset, to the best of our knowledge, there are no known optimization algorithms that can solve multi-task regression problems with structured sparsity of this scale. Although the same dataset has been analyzed using tree-guided group lasso in [7], their optimization method based on a variational formulation could handle only a small-scale dataset because it involves an inversion of $J \times J$ matrix in each iteration. Thus, their analysis is focused only on a single chromosome with 21 genotypes instead of the entire set of 1260 genotypes.

5 Conclusions

In this paper, we considered an optimization problem for learning a structured-sparsity pattern in regression parameters, where the overlapping group structure in inputs or outputs is encoded the structured-sparsity-inducing norm. Under single-task and multi-task regression settings, we developed a proximal-gradient method that is extremely efficient and can scale up to very high-dimensional datasets. Using synthetic and real datasets, we demonstrated the efficiency and scalability of our method.

Appendix

A.1 Geometric Illustration of the Smoothness of $f_\mu(\beta)$

In order to provide a geometric illustration of the smoothness of $f_\mu(\beta)$ as stated in Theorem 1, we consider the function $f_0(\beta)$ and its smooth approximation $f_\mu(\beta)$ in one-dimensional space (i.e., $\beta \in \mathbb{R}$). For the sake of simplicity, we assume that μ and C are set to 1.

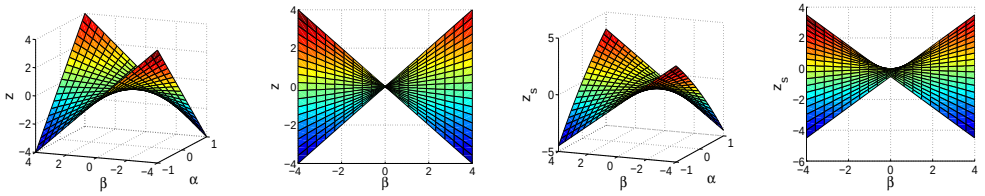
First, we show geometrically that $f_0(\beta) = \max_{\alpha \in [-1,1]} z(\alpha, \beta)$ with $z(\alpha, \beta) \equiv \alpha\beta$ is a non-smooth function. The 3-D plot for $z(\alpha, \beta)$ with α restricted to $[-1, 1]$ is shown in Figure 6(a). We project the surface in Figure 6(a) onto the $\beta - z$ space as shown in Figure 6(b). For each β, the

value of $f_0(\beta)$ is the highest point along the z-axis since we maximize over α in $[-1, 1]$. We can see that $f_0(\beta)$ is composed of two segments with a sharp point at $\beta = 0$. In fact $f_0(\beta) = |\beta|$.

Now, we introduce the auxiliary function $d(\alpha) = \frac{1}{2}\alpha^2$ and let $z_s(\alpha, \beta) \equiv \alpha\beta - \frac{1}{2}\alpha^2$ and $f_\mu(\beta) = \max_{\alpha\in[-1,1]} z_s(\alpha, \beta)$. We show geometrically that $f_\mu(\beta)$ is a smooth approximation of $f_0(\beta)$. The 3-D plot for $z_s(\alpha, \beta)$ with α restricted to $[-1, 1]$ is shown in Figure 6(c). Similarly, we project the surface in Figure 6(c) onto the $\beta - z_s$ space as shown in Figure 6(d). For fixed β, the value of $f_\mu(\beta)$ is the highest point along the z-axis. In Figure 6(d), we can see that $f_\mu(\beta)$ is composed of three parts: (i) a line with slope -1 when $\beta < 1$, (ii) a line with slope 1 when $\beta > 1$, and (iii) a quadratic function when $-1 \leq \beta \leq 1$. By introducing a quadratic auxiliary function, we remove the sharp point at $\beta = 0$ and $f_\mu(\beta)$ becomes a smooth function. In fact, f_μ takes the following form:

$$f_\mu(\beta) = \begin{cases} \frac{\beta^2}{2} & -1 < \beta < 1 \\ \beta - \frac{1}{2} & \beta \geq 1 \\ -\beta - \frac{1}{2} & \beta \leq -1. \end{cases}$$

We note that f_μ is smooth at $\beta = \pm 1$.



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

Figure 6: A geometric illustration of the smoothness of $f_\mu(\beta)$. (a) The 3-D plot of $z(\alpha, \beta)$, (b) the projection of (a) onto the β-z space, (c) the 3-D plot of $z_s(\alpha, \beta)$, and (d) the projection of (c) onto the β-z space.

A.2 Proof of Theorem 1

The $f_\mu(\boldsymbol{\beta})$ is a convex function since it is the maximum of a set of functions that are linear in $\boldsymbol{\beta}$. For the smoothness property, let the function d^* be the Fenchel conjugate of the distance function d defined as:

$$d^*(\boldsymbol{\gamma}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} \langle \boldsymbol{\alpha}, \boldsymbol{\gamma} \rangle - d(\boldsymbol{\alpha}). \tag{16}$$

We want to prove d^* is differentiable everywhere by showing that the subdifferential ∂d^* of d^* is a singleton set for any $\boldsymbol{\gamma}$.

From the definition in (16), for any $\boldsymbol{\gamma}$ and any $\boldsymbol{\alpha} \in \mathcal{Q}$, we have:

$$d^*(\boldsymbol{\gamma}) + d(\boldsymbol{\alpha}) \geq \langle \boldsymbol{\alpha}, \boldsymbol{\gamma} \rangle, \tag{17}$$

where the inequality holds as an equality if and only if $\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}'\in\mathcal{Q}} \langle \boldsymbol{\alpha}', \boldsymbol{\gamma} \rangle - d(\boldsymbol{\alpha}')$.

Since d is convex and closed, we have $d^{**} \equiv d$ (Chapter E in [5]). Thus, (17) can be written as:

$$d^*(\boldsymbol{\gamma}) + d^{**}(\boldsymbol{\alpha}) \geq \langle \boldsymbol{\alpha}, \boldsymbol{\gamma} \rangle, \tag{18}$$

where the inequality holds as an equality if and only if $\boldsymbol{\gamma} = \arg\max_{\boldsymbol{\gamma}' \in \mathbb{R}^J} \langle \boldsymbol{\alpha}, \boldsymbol{\gamma}' \rangle - d^*(\boldsymbol{\gamma}')$.

Since (17) and (18) are equivalent, we know that $\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}' \in \mathcal{Q}} \boldsymbol{\alpha}'^T \boldsymbol{\gamma} - d(\boldsymbol{\alpha}')$ if and only if $\boldsymbol{\gamma} = \arg\max_{\boldsymbol{\gamma}' \in \mathbb{R}^J} \langle \boldsymbol{\alpha}, \boldsymbol{\gamma}' \rangle - d^*(\boldsymbol{\gamma}')$. The latter equality implies that for any $\boldsymbol{\gamma}'$:

$$d^*(\boldsymbol{\gamma}') \geq d^*(\boldsymbol{\gamma}) + \langle \boldsymbol{\alpha}, \boldsymbol{\gamma}' - \boldsymbol{\gamma} \rangle,$$

which further means that $\boldsymbol{\alpha}$ is a subgradient of d^* at $\boldsymbol{\gamma}$ by the definition of subgradient.

Summarizing the above arguments, we conclude that $\boldsymbol{\alpha}$ is a subgradient of d^* at $\boldsymbol{\gamma}$ if and only if

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}' \in \mathcal{Q}} \langle \boldsymbol{\alpha}', \boldsymbol{\gamma} \rangle - d(\boldsymbol{\alpha}'). \tag{19}$$

Since d is a strongly-convex function, this maximization problem in (19) has a unique optimal solution. Thus, the subdifferential ∂d^* of d^* at any point $\boldsymbol{\gamma}$ is a singleton set that contains only $\boldsymbol{\alpha}$. Therefore, d^* is differentiable everywhere (Chapter D in [5]) and $\boldsymbol{\alpha}$ is its gradient:

$$\nabla d^*(\boldsymbol{\gamma}) = \boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}' \in \mathcal{Q}} \langle \boldsymbol{\alpha}', \boldsymbol{\gamma} \rangle - d(\boldsymbol{\alpha}'). \tag{20}$$

Now we return to our original problem of $f_\mu(\boldsymbol{\beta})$ and rewrite it as:

$$f_\mu(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha} \in \mathcal{Q}} \langle \boldsymbol{\alpha}, \Gamma(\boldsymbol{\beta}) \rangle - \mu d(\boldsymbol{\alpha}) = \mu \max_{\boldsymbol{\alpha} \in \mathcal{Q}} [\langle \boldsymbol{\alpha}, \frac{\Gamma(\boldsymbol{\beta})}{\mu} \rangle - d(\boldsymbol{\alpha})] = \mu d^*(\frac{\Gamma(\boldsymbol{\beta})}{\mu}).$$

Using (20) and the chain rule, we know that $f_\mu(\boldsymbol{\beta})$ is continuously differentiable and its gradient takes the following form:

$$\begin{aligned} \nabla f_\mu(\boldsymbol{\beta}) &= \mu \Gamma^*(\nabla d^*(\frac{\Gamma(\boldsymbol{\beta})}{\mu})) = \mu \Gamma^*(\arg\max_{\boldsymbol{\alpha}' \in \mathcal{Q}} [\langle \boldsymbol{\alpha}', \frac{\Gamma(\boldsymbol{\beta})}{\mu} \rangle - d(\boldsymbol{\alpha}')]) \\ &= \Gamma^*(\arg\max_{\boldsymbol{\alpha}' \in \mathcal{Q}} [\langle \boldsymbol{\alpha}', \Gamma(\boldsymbol{\beta}) \rangle - \mu d(\boldsymbol{\alpha}')]) = \Gamma^*(\boldsymbol{\alpha}^*). \end{aligned}$$

For the proof of Lipschitz constant of $f_\mu(\boldsymbol{\beta})$, readers can refer to [11].

A.3 Proof of Lemma 2

Since we have

$$\|\Gamma(\mathbf{v})\|_2 = \|C\mathbf{v}\|_2 = \lambda \sqrt{\sum_{g \in \mathcal{G}} \sum_{j \in g} (w_g)^2 v_j^2} = \lambda \sqrt{\sum_{j=1}^{J} \left(\sum_{g \in \mathcal{G} \ s.t. \ j \in g} (w_g)^2 \right) v_j^2},$$

the maximum value of $\|\Gamma(\mathbf{v})\|_2$, given $\|\mathbf{v}\|_2 \leq 1$, can be achieved by setting $v_{\hat{j}}$ for j corresponding to the largest summation $\sum_{g \in \mathcal{G} \ s.t. \ j \in g} (w_g)^2$ to one, and setting other v_j's to zeros. Hence, we have $\|\Gamma(\mathbf{v})\|_2 = \lambda \max_{j \in \{1,\ldots,J\}} \sqrt{\sum_{g \in \mathcal{G} \text{ s.t. } j \in g} (w_g)^2}$.

For the norm of the linear operator Γ in the multi-task setting, following the same proof strategy, we can show that

$$\|\Gamma\| = \lambda \max_{k\in\{1,\ldots,K\}} \sqrt{\sum\nolimits_{g\in\mathcal{G}\ s.t.\ k\in g}(w_g)^2}.$$

A.4 Proof of Theorem 2

Based on Theorem 2 in [11], we have the following lemma:

Lemma 3. *Assume that function $\widetilde{f}(\boldsymbol{\beta})$ is an arbitrary convex smooth function and its gradient $\nabla\widetilde{f}(\boldsymbol{\beta})$ is Lipschitz continuous with the Lipschitz constant L. We apply Algorithm 1 to minimize $\widetilde{f}(\boldsymbol{\beta})$ and let $\boldsymbol{\beta}^t$ be the approximate solution at the t-th iteration. For any $\boldsymbol{\beta}$, we have the following bound:*

$$\widetilde{f}(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}) \leq \frac{2L\|\boldsymbol{\beta}\|_2^2}{t^2}. \tag{21}$$

Recall that the smooth approximation of the function $f(\boldsymbol{\beta})$, $\widetilde{f}(\boldsymbol{\beta})$, is defined as:

$$\widetilde{f}(\boldsymbol{\beta}) \equiv \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + f_\mu(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \max_{\boldsymbol{\alpha}\in\mathcal{Q}}(\boldsymbol{\alpha}^T C\boldsymbol{\beta} - \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2).$$

Since Algorithm 1 optimizes the smooth function $\widetilde{f}(\boldsymbol{\beta})$, according to Lemma 3, we have

$$\widetilde{f}(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}^*) \leq \frac{2L\|\boldsymbol{\beta}^*\|_2^2}{t^2}, \tag{22}$$

where $L = \lambda_{\max}(\mathbf{X}^T\mathbf{X}) + \frac{\|\Gamma\|^2}{\mu}$ is the Lipschitz constant for $\nabla\widetilde{f}(\boldsymbol{\beta})$.

In order to use the bound in (22), we decompose $f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*)$ into three terms:

$$f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*) = \left(f(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}^t)\right) + \left(\widetilde{f}(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}^*)\right) + \left(\widetilde{f}(\boldsymbol{\beta}^*) - f(\boldsymbol{\beta}^*)\right). \tag{23}$$

According to the definition of $\widetilde{f}$, we know that for any $\boldsymbol{\beta}$

$$\widetilde{f}(\boldsymbol{\beta}) \leq f(\boldsymbol{\beta}) \leq \widetilde{f}(\boldsymbol{\beta}) + \mu D,$$

where $D \equiv \max_{\boldsymbol{\alpha}\in\mathcal{Q}} d(\boldsymbol{\alpha})$. Therefore, the first term in (23), $f(\boldsymbol{\beta}^t) - \widetilde{f}(\boldsymbol{\beta}^t)$, is upper-bounded by μD, and the last term in (23) is less than or equal to 0 (i.e., $\widetilde{f}(\boldsymbol{\beta}^*) - f(\boldsymbol{\beta}^*) \leq 0$). Combining (22) with these two simple bounds, we have:

$$f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*) \leq \mu D + \frac{2L\|\boldsymbol{\beta}^*\|_2^2}{t^2} \leq \mu D + \frac{2\|\boldsymbol{\beta}^*\|_2^2}{t^2}\left(\lambda_{\max}(\mathbf{X}^T\mathbf{X}) + \frac{\|\Gamma\|^2}{\mu}\right). \tag{24}$$

By setting $\mu = \frac{\epsilon}{2D}$ and plugging this into the right-hand side of (24), we obtain

$$f(\boldsymbol{\beta}^t) - f(\boldsymbol{\beta}^*) \leq \frac{\epsilon}{2} + \frac{2\|\boldsymbol{\beta}^*\|_2^2}{t^2}\left(\lambda_{\max}\left(\mathbf{X}^T\mathbf{X}\right) + \frac{2D\|\Gamma\|^2}{\epsilon}\right). \tag{25}$$

If we require the right-hand side of (25) to be equal to ϵ and solve it for t, we obtain the bound of t in (11).

Note that we can set $\mu = \frac{\epsilon}{h}$ for any $h > 1$ to achieve $O\left(\frac{1}{\epsilon}\right)$ convergence rate, which is different from (11) only by a constant factor.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73:243–272, 2006.

[2] Francis Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[3] Jun Zhu et al. Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature Genetics*, 40:854–861, 2008.

[4] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.

[5] Jean-Baptiste Hiriart-Urruty and Claude Lemarechal. *Fundamentals of Convex Analysis*. Springer, 2001.

[6] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. Technical report, INRIA, 2009.

[7] Seyoung Kim and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[8] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Herve Lebret. Applications of second-order cone programming. *Linear Algebra and Its Applications*, 284:193–228, 1998.

[9] Lukas Meier, Sara van de Geer, and Peter Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B*, 70:53–71, 2008.

[10] Yurii Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Pub, 2003.

[11] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[12] Jos F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(12):625:653, 1999.

[13] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–288, 1996.

[14] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68:49–67, 2006.

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

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