Stat Biosci (2012) 4:3–26
DOI 10.1007/s12561-011-9048-z

An Efficient Optimization Algorithm for Structured Sparse CCA, with Applications to eQTL Mapping

Xi Chen · Han Liu

Received: 3 July 2011 / Accepted: 2 December 2011 / Published online: 21 December 2011

Abstract In this paper we develop an efficient optimization algorithm for solving canonical correlation analysis (CCA) with complex *structured-sparsity-inducing penalties*, including overlapping-group-lasso penalty and network-based fusion penalty. We apply the proposed algorithm to an important genome-wide association study problem, eQTL mapping. We show that, with the efficient optimization algorithm, one can easily incorporate rich structural information among genes into the sparse CCA framework, which improves the interpretability of the results obtained. Our optimization algorithm is based on a general excessive gap optimization framework and can scale up to millions of variables. We demonstrate the effectiveness of our algorithm on both simulated and real eQTL datasets.

Keywords Sparse CCA · Structured sparsity · Group structure · Network structure · Genome-wide association study · eQTL mapping · Optimization algorithm

1 Introduction

In recent years, significant progress has been made on developing different variants of canonical correlation analysis (CCA) models and apply them to genome-wide association study (GWAS or GWA study) problems. Examples include identifying genes that are correlated with regions of DNA copy number change [26, 28]; identifying genes that are correlated with single nucleotide polymorphisms (SNPs) [18, 22];

X. Chen (✉)
Machine Learning Department, Carnegie Mellon University, Pittsburg, USA
e-mail: xichen@cs.cmu.edu

H. Liu
Biostatistics Department, Computer Science Department, Johns Hopkins University, Baltimore, USA
e-mail: hanliu@cs.jhu.edu

and identifying sets of genes on two different microarray platforms that have correlated expressions [6]. More specifically, let us take *expression quantitative trait loci* (*eQTLs*) *mapping* as an example. The problem of eQTL mapping searches for associations between a large number of SNPs and gene expression levels collected over a number of individuals. We denote SNP genotype data as $\mathbf{X}$ of dimensions $n \times d$ and expression levels as $\mathbf{Y}$ of $n \times p$. CCA finds two canonical vectors $\mathbf{u}$ and $\mathbf{v}$ to maximize the correlation between $\mathbf{Xu}$ and $\mathbf{Yv}$. Since the number of SNPs and expressions levels are far greater than that of subjects, the standard CCA models cannot be directly applied. To handle this challenge, a popular approach is to impose an ℓ_1-norm penalty on $\mathbf{u}$ and $\mathbf{v}$ to shrink the coefficients of the irrelevant variables toward zero [22, 28, 29] and the corresponding model is referred to *sparse CCA*. However, the simple ℓ_1-norm penalty is limited in that it neglects the rich structural information among variables. When dealing with high-dimensional data, prior structural knowledge is crucial for improving the estimation performance and model interpretability. For example, a biological pathway is a group of genes that participate in a particular biological process to perform certain functionality in a cell. To find controlling factors related to a disease, it is more meaningful to study the genes by considering their pathway information. Similarly, we could also exploit the network structure among genes (e.g., gene regulatory network) to obtain enhanced estimation performance.

Recently, various extensions of ℓ_1-norm penalty have been proposed to take advantage of the prior knowledge of the structure among variables. Examples include mixed-norm group-lasso penalty [30], mixed-norm tree-structured penalty [15, 31] and network-structured fusion penalty [14]. However, these structured-sparsity-inducing penalties have not been incorporated into the CCA framework for analysis of GWAS data. The main challenge arises from the computational side. More specifically, it is known that sparse CCA model can be naturally formulated into a biconvex problem and solved by an alternating optimization strategy: fix $\mathbf{u}$ and optimize with respect to $\mathbf{v}$; then fix $\mathbf{v}$ and optimize with respect to $\mathbf{u}$; and iterate over these two steps. Since sparse CCA needs many iterations of these two steps and one may have to run sparse CCA with multiple starting parameters to avoid local minima, it is crucial to solve each subproblem (i.e. optimization with respect to $\mathbf{u}$ or $\mathbf{v}$) efficiently. When the simple ℓ_1-norm is imposed, the subproblem can be solved in closed form according to [29]. For other simple structures, e.g. non-overlapping group-lasso penalty or a chain-structured fusion penalty, one can apply coordinate descent scheme to solve the corresponding optimization problem. However, for the more general structured-sparsity-inducing penalties, there still lacks an efficient and scalable optimization algorithm, which prevents the wide application of sparse CCA models in GWAS.

In this paper, we propose an efficient optimization algorithm, which solves the sparse CCA with a wide class of *structured sparsity-inducing penalties*. Our method is based on a general excessive gap optimization framework [19]. We consider two widely used structured-sparsity-inducing penalties in this paper:

1. *Overlapping-group-lasso penalty* [11]. Compare to the standard group lasso [30], such a penalty allows arbitrary overlaps among groups which reflects the fact that a gene can belong to multiple pathways. We refer to the corresponding CCA model as the *group-structured sparse CCA*.

2. *Network-based fusion penalty* [14]. By leveraging any prior knowledge of the network structure, the network-based fusion penalty enforces the coefficients on two connected nodes to be similar. In addition, it can conduct automatic group pursuit to group the genes into different clusters based on the prior network structure. We refer to this model as the *network-structured sparse CCA*.

We show that it is possible to decouple the non-separable overlapping-group-lasso and network-based fusion penalties via the dual norm and reformulate them into a maximization form [7] where the excessive gap framework can be applied. Since it is a first-order method only using the gradient information, the per-iteration time complexity is very low (e.g. linear in the sum of group sizes or the number of edges) and the method can scale up to millions of variables. Moreover, unlike in many first-order methods where only the primal solutions are computed, it is a primal-dual approach which diminishes the primal-dual gap over iterations. For each subproblem in the alternating optimization procedure (optimization with respect to $\mathbf{u}$ or $\mathbf{v}$), the algorithm provably converges to an ϵ accurate solution (i.e. the duality gap is less than ϵ) in $O(\mathcal{L}/\sqrt{\epsilon})$ iterations, where $\mathcal{L}$ is an input dependent constant. According to [20], it has already achieved the optimal rate of convergence for solving smooth convex problem only using the first-order information.

The rest of this paper is organized as follows. Section 2 introduces background for sparse CCA in [28, 29]; Sect. 3 presents the group-structured sparse CCA and proposes the corresponding optimization algorithm; Sect. 4 proposes the network-structured sparse CCA; in Sect. 5, we first demonstrate the efficiency and scalability of the proposed optimization algorithm, then apply the proposed structured sparse CCA models to both simulated and real datasets. We conclude our paper in the final section.

2 Background: Sparse CCA

Given two datasets $\mathbf{X}$ and $\mathbf{Y}$ of dimensions $n \times d$ and $n \times p$ on the same set of n observations, we assume that each column of $\mathbf{X}$ and $\mathbf{Y}$ is normalized to have mean zero and standard deviation one. The sparse CCA proposed in [28, 29] takes the following form:

$$\begin{aligned} &\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^T \mathbf{X}^T \mathbf{Y} \mathbf{v} \\ &\text{s.t.} \quad \|\mathbf{u}\|_2 \le 1, \qquad \|\mathbf{v}\|_2 \le 1, \\ &\qquad P_1(\mathbf{u}) \le c_1, \qquad P_2(\mathbf{v}) \le c_2, \end{aligned} \tag{1}$$

where P_1 and P_2 are convex and non-smooth sparsity-inducing penalties that yield sparse $\mathbf{u}$ and $\mathbf{v}$. Witten et al. [29] studied two specific forms of the penalty P (either P_1 or P_2): (1) ℓ_1-norm penalty $P(\mathbf{w}) = \|\mathbf{w}\|_1$, which will result in a sparse $\mathbf{w}$ vector. (2) Chain-structured fusion penalty $P(\mathbf{w}) = \|\mathbf{w}\|_1 + \gamma \sum_j |w_j - w_{j-1}|$, which assumes that variables have a natural ordering and will result in $\mathbf{w}$ sparse and smooth along the ordering.

In this work, we extend the sparse CCA to more general forms of P that incorporate the group or network structural information among variables. In eQTL mapping,

the structural knowledge among genes on $\mathbf{Y}$ side is often of more interest. To ease the illustration of our algorithm, we always assume that $P_1(\mathbf{u}) = \|\mathbf{u}\|_1$ and mainly focus on $P_2(\mathbf{v})$, which incorporates the structural information. As has been discussed in the introduction, (1) is biconvex in $\mathbf{u}$ and $\mathbf{v}$ individually. The optimization problem can then be solved by an alternating approach. In our setting, the optimization with respect to $\mathbf{u}$ with $P_1(\mathbf{u}) = \|\mathbf{u}\|_1$ is relatively simple and the closed-form solution has been obtained in [29]. However, due to the complicated structure of $P_2(\mathbf{v})$, the optimization with respect to $\mathbf{v}$ cannot be easily solved, which is the exact challenge addressed in this paper.

3 Group-Structured Sparse CCA

3.1 Model

In this section, we study the problem in which the group structure information among variables in $\mathbf{Y}$ is pre-given from the domain knowledge, e.g. pathways in the genetic data; and our goal is to identify a small subset of groups under the sparse CCA framework. More formally, let us assume that the set of groups of variables in $\mathbf{Y}$: $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\}$ is defined as a subset of the power set of $\{1, \ldots, p\}$, and is available as prior knowledge. Note that the members (groups) of $\mathcal{G}$ are allowed to overlap. Inspired by the *group-lasso penalty* [30] and the *elastic-net penalty* [33], we define the penalty $P_2(\mathbf{v})$ as follows:

$$P_2(\mathbf{v}) = \sum_{g \in \mathcal{G}} w_g \|\mathbf{v}_g\|_2 + \frac{c}{2}\mathbf{v}^T\mathbf{v}, \tag{2}$$

where $\mathbf{v}_g \in \mathbb{R}^{|g|}$ is the subvector of $\mathbf{v}$ in group g, w_g is the predefined weight for group g; c is the tuning parameter and $\|\cdot\|_2$ is the vector ℓ_2-norm. The ℓ_1/ℓ_2 mixed-norm penalty in $P_2(\mathbf{v})$ plays the role of group selection. Since some gene expression levels are highly correlated, the ridge penalty $\frac{c}{2}\mathbf{v}^T\mathbf{v}$ addresses the problem of the collinearity, enforcing strongly correlated variables to be in or out of the model together. In addition, according to [17, 33], the ridge penalty is crucial to ensure the stable variable selection when $p \gg n$, which is a typical setting of eQTL mapping. It is noteworthy that, to perform variable selection within the group, we can also include the penalty for individual variable as singleton group in $P_2(\mathbf{v})$ by adding the term $w_j|v_j|$. We also note that the widely used *tree-structured-sparsity-inducing penalty* [15, 31] is a special case of the overlapping-group-lasso penalty where each tree node corresponds to a group.

Rather than solving the constraint form of $P_2(\mathbf{v})$, we solve the regularized problem using the Lagrangian form:

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{v}} -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} + \frac{\tau}{2}\mathbf{v}^T\mathbf{v} + \theta \sum_{g \in \mathcal{G}} w_g \|\mathbf{v}_g\|_2 \\ &\text{s.t.} \quad \|\mathbf{u}\|_2 \le 1, \qquad \|\mathbf{v}\|_2 \le 1, \qquad \|\mathbf{u}\|_1 \le c_1, \end{aligned} \tag{3}$$

where there exists a one-to-one correspondence between (θ, τ) and (c, c_2) (c_2 is the upper bound of $P_2(\mathbf{v})$). We refer to this model (3) as the *group-structured sparse CCA*.

3.2 Optimization Algorithm

The main difficulty in solving (3) arises from optimizing with respect to $\mathbf{v}$. Let the domain of $\mathbf{v}$ be denoted as $Q_1 = \{\mathbf{v} \,|\, \|\mathbf{v}\|_2 \leq 1\}$, $\boldsymbol{\beta} = \frac{1}{\tau}\mathbf{Y}^T\mathbf{X}\mathbf{u}$ and $\gamma = \frac{\theta}{\tau}$, the optimization of (3) with respect to $\mathbf{v}$ can be written as

$$\min_{\mathbf{v}\in Q_1} f(\mathbf{v}) \equiv l(\mathbf{v}) + P(\mathbf{v}), \tag{4}$$

where $l(\mathbf{v}) = \frac{1}{2}\|\mathbf{v} - \boldsymbol{\beta}\|_2^2$ is the Euclidean distance loss function and $P(\mathbf{v})$ is the overlapping-group-lasso penalty: $P(\mathbf{v}) = \gamma \sum_{g\in\mathcal{G}} w_g \|\mathbf{v}_g\|_2$. The optimization problem in (4) is so-called *proximal mapping* associated with the function $P(\mathbf{v})$.[1]

3.2.1 Related Optimization Methods

When $\mathbf{v}$ is unconstrained and the groups are non-overlapped, the closed-form optimal solution can be easily obtained by computing the subgradient with respect to each $\mathbf{v}_g$ as shown in [8]. In contrast, when the groups are overlapped, the subgradient with respect to each group becomes very complicated and hence there is no closed-form solution. A number of first-order methods [7, 8, 11, 12, 16] have recently been developed for solving variants of overlapping-group-lasso problem. The methods in [12, 16] can only be applied to the tree-structured groups or ℓ_1/ℓ_∞-regularized group structure for the unconstrained $\mathbf{v}$. Generally speaking, for the complicated structured non-smooth penalty, there are two common first-order approaches for optimizing it: (1) compute the subgradient of the penalty and then apply the projected subgradient descent. However, it has a very slow convergence rate of $O(\frac{1}{\epsilon^2})$. (2) Smooth the penalty [7] and then apply any first-order method to solve the "smoothed" problem. However, this approach does not fully utilize the special structure of the loss function in the CCA setting, that is, the design matrix is the identity matrix and the loss is essentially a signal approximator. Therefore, it can only achieve a sub-optimal rate of $O(\frac{1}{\epsilon})$. For other possible methods, interior-point method for the second-order cone formulation and iterated reweighted least squares [1] suffer from the high computational cost of solving a linear system. Alternating direction augmented Lagrangian method [27] have no known results on the convergence rate.

In this section, by specializing a general *excessive gap* framework [19], we present an efficient and provably optimal optimization algorithm for solving the proximal mapping with ℓ_1/ℓ_2 regularized overlapping-group-lasso penalty in (4).

[1] To be more precise, (4) is the proximal mapping under the constraint on Q_1 or the proximal mapping associated with the function $P(\mathbf{v}) + I_{Q_1}(\mathbf{v})$, where $I_{Q_1}(\mathbf{v})$ is the indicator function of Q_1.

3.2.2 *Reformulation of the Penalty*

To specialize the excessive gap framework, we first reformulate the group-structured-sparsity-inducing-penalty using the technique in [7]. For the purpose of completeness, we present the detailed derivation here. Using the dual norm, $\|\mathbf{v}_g\|_2$ can be written as $\max_{\|\boldsymbol{\alpha}_g\|_2 \leq 1} \boldsymbol{\alpha}_g^T \mathbf{v}_g$, where $\boldsymbol{\alpha}_g \in \mathbb{R}^{|g|}$. Then we can rewrite $P(\mathbf{v})$ as

$$P(\mathbf{v}) = \gamma \sum_{g \in \mathcal{G}} w_g \|\mathbf{v}_g\|_2 = \gamma \sum_{g \in \mathcal{G}} w_g \max_{\|\boldsymbol{\alpha}_g\|_2 \leq 1} \boldsymbol{\alpha}_g^T \mathbf{v}_g = \max_{\boldsymbol{\alpha} \in Q_2} \boldsymbol{\alpha}^T C \mathbf{v}. \tag{5}$$

Here $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_{g_1}^T, \ldots, \boldsymbol{\alpha}_{g_{|\mathcal{G}|}}^T]^T$ is the concatenation of the vectors $\{\boldsymbol{\alpha}_g\}_{g \in \mathcal{G}}$; and we denote the domain of $\boldsymbol{\alpha}$ as

$$Q_2 \equiv \big\{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha}_g\|_2 \leq 1,\ \forall g \in \mathcal{G}\big\}.$$

The matrix $C \in \mathbb{R}^{(\sum_{g \in \mathcal{G}} |g|) \times p}$ is defined as

$$C_{(i,g),j} = \begin{cases} \gamma w_g & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where the rows of C are indexed by all pairs of $(i, g) \in \{(i, g) | i \in g, i \in \{1, \ldots, p\}\}$, the columns are indexed by $j \in \{1, \ldots, p\}$.

Example Assume $\mathbf{v} \in \mathbb{R}^3$ with groups $\mathcal{G} = \{g_1 = \{1, 2\}, g_2 = \{2, 3\}\}$. Then, the matrix C is

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

To provide a deeper insight into this reformation, we show that (5) can be viewed as *Fenchel Conjugate* [9] of the indicator function.

Definition 1 The Fenchel conjugate of a function $f(\mathbf{x})$ is the function f^* defined by

$$f^*(\mathbf{y}) = \sup_{\mathbf{x} \in \mathrm{dom}(f)} \big(\mathbf{x}^T \mathbf{y} - f(\mathbf{x})\big). \tag{7}$$

Let $\delta_{Q_2}(\mathbf{x})$ be the indicator function:

$$\delta_{Q_2}(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in Q_2, \\ +\infty, & \mathbf{x} \notin Q_2; \end{cases}$$

the penalty function $P(\mathbf{v})$ is the Fenchel conjugate of δ_{Q_2} at $C\mathbf{v}$:

$$P(\mathbf{v}) = \max_{\boldsymbol{\alpha} \in Q_2} \boldsymbol{\alpha}^T C \mathbf{v} = \delta_{Q_2}^*(C\mathbf{v}). \tag{8}$$

3.2.3 Smoothing the Penalty

The difficulty for the optimization mainly arises from the non-smooth penalty $P(\mathbf{v})$. To tackle this problem, we introduce an auxiliary quadratic function to construct a smooth approximation of $P(\mathbf{v})$ using the smoothing technique in [21]. Our smooth approximation function is given as follows:

$$P_\mu(\mathbf{v}) = \max_{\boldsymbol{\alpha} \in Q_2} \big(\boldsymbol{\alpha}^T C\mathbf{v} - \mu d(\boldsymbol{\alpha})\big), \tag{9}$$

where μ is the positive smoothness parameter and $d(\boldsymbol{\alpha})$ is defined as $\frac{1}{2}\|\boldsymbol{\alpha}\|_2^2$. It is obvious that $P_\mu(\mathbf{v}) \le P(\mathbf{v})$. Let $D = \max_{\boldsymbol{\alpha} \in Q_2} d(\boldsymbol{\alpha}) = |\mathcal{G}|/2$, where $|\mathcal{G}|$ is the number of groups. It is easy to verify that the maximum gap between $P_\mu(\mathbf{v})$ and $P(\mathbf{v})$ is μD:

$$P(\mathbf{v}) - \mu D \le P_\mu(\mathbf{v}) \le P(\mathbf{v}). \tag{10}$$

The next theorem from [21] shows that for any $\mu > 0$, $P_\mu(\mathbf{v})$ is a smooth function with a simple form of gradient.

Theorem 1 ([21]) *For any $\mu > 0$, $P_\mu(\mathbf{v})$ is a smooth and convex function in $\mathbf{v}$, and the gradient of $P_\mu(\mathbf{v})$ takes the following form*:

$$\nabla P_\mu(\mathbf{v}) = C^T \boldsymbol{\alpha}_\mu(\mathbf{v}), \tag{11}$$

where $\boldsymbol{\alpha}_\mu(\mathbf{v})$ is the optimal solution to (9):

$$\boldsymbol{\alpha}_\mu(\mathbf{v}) = \arg\max_{\boldsymbol{\alpha} \in Q_2} \boldsymbol{\alpha}^T C\mathbf{v} - \mu d(\boldsymbol{\alpha}). \tag{12}$$

To compute the gradient of $P_\mu(\mathbf{v})$ in (11), we need to know $\boldsymbol{\alpha}_\mu(\mathbf{v})$. The closed-form equation for $\boldsymbol{\alpha}_\mu(\mathbf{v})$ can be derived with some simple algebra and is presented in the next proposition.

Proposition 1 *The $\boldsymbol{\alpha}_\mu(\mathbf{v}) \in \mathbb{R}^{\sum_{g \in \mathcal{G}} |g|}$ in* (12) *is the concatenation of subvectors $\{[\boldsymbol{\alpha}_\mu(\mathbf{v})]_g\}$ for all groups $g \in \mathcal{G}$. For any group g,*

$$\big[\boldsymbol{\alpha}_\mu(\mathbf{v})\big]_g = S_2\left(\frac{\gamma w_g \mathbf{v}_g}{\mu}\right), \tag{13}$$

where S_2 is the projection operator (to the ℓ_2-ball) defined as follows:

$$S_2(\mathbf{x}) = \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|_2}, & \|\mathbf{x}\|_2 > 1, \\ \mathbf{x}, & \|\mathbf{x}\|_2 \le 1. \end{cases} \tag{14}$$

We substitute $P(\mathbf{v})$ in the original objective function $f(\mathbf{v})$ with $P_\mu(\mathbf{v})$ and construct the smooth approximation of $f(\mathbf{v})$: $f_\mu(\mathbf{v}) \equiv l(\mathbf{v}) + P_\mu(\mathbf{v})$. According to (10), the relationship of $f_\mu(\mathbf{v})$ and $f(\mathbf{v})$ can be characterized by the following inequality:

$$f(\mathbf{v}) - \mu D \le f_\mu(\mathbf{v}) \le f(\mathbf{v}). \tag{15}$$

3.2.4 Fenchel Dual of $f(\mathbf{v})$

The fundamental idea of the excessive gap method is to diminish the duality gap between the objective $f(\mathbf{v})$ and its *Fenchel dual* over iterations. In this section, we derive the Fenchel dual of $f(\mathbf{v})$ and study its property. According to Theorem 3.3.5 in [4], the Fenchel dual problem of $f(\mathbf{v})$, $\phi(\boldsymbol{\alpha})$, takes the following form:

$$\phi(\boldsymbol{\alpha}) = -l^*\big(-C^T\boldsymbol{\alpha}\big) - \delta_{Q_2}(\boldsymbol{\alpha}), \tag{16}$$

where l^* is the Fenchel Conjugate of l and

$$-l^*\big(-C^T\boldsymbol{\alpha}\big) = -\max_{\mathbf{v}\in Q_1} -\mathbf{v}^T C^T\boldsymbol{\alpha} - l(\mathbf{v}) = \min_{\mathbf{v}\in Q_1} \mathbf{v}^T C^T\boldsymbol{\alpha} + \frac{1}{2}\|\mathbf{v}-\boldsymbol{\beta}\|_2^2.$$

With the similar proof technique as in Theorem 1, we present the gradient of $\phi(\boldsymbol{\alpha})$ in the following theorem:

Theorem 2 *The gradient of $\phi(\boldsymbol{\alpha})$ takes the following form*:

$$\nabla\phi(\boldsymbol{\alpha}) = C\mathbf{v}(\boldsymbol{\alpha}), \tag{17}$$

where

$$\mathbf{v}(\boldsymbol{\alpha}) = \arg\min_{\mathbf{v}\in Q_1} \mathbf{v}^T C^T\boldsymbol{\alpha} + \frac{1}{2}\|\mathbf{v}-\boldsymbol{\beta}\|_2^2. \tag{18}$$

Moreover, $\nabla\phi(\boldsymbol{\alpha})$ is Lipschitz continuous with the Lipschitz constant $L(\phi) = \frac{1}{\sigma}\|C\|^2$, where $\sigma = 1$ is the strongly convex parameter for function $l(\mathbf{v})$ and $\|C\|$ is the matrix spectral norm of C: $\|C\| \equiv \max_{\|\mathbf{x}\|_2=1} \|C\mathbf{x}\|_2$.

According to the next proposition [7], the closed-form equations for $\mathbf{v}(\boldsymbol{\alpha})$ and $\|C\|$ can be written as follows:

Proposition 2 $\mathbf{v}(\boldsymbol{\alpha})$ *takes the following form*:

$$\mathbf{v}(\boldsymbol{\alpha}) = S_2\big(\boldsymbol{\beta} - C^T\boldsymbol{\alpha}\big), \tag{19}$$

where S_2 is the projection operator (shrinking to the ℓ_2-ball) defined in (14). $\|C\|$ takes the following form:

$$\|C\| = \gamma \max_{j\in\{1,\ldots,p\}} \sqrt{\sum_{g\in\mathcal{G}\ s.t.\ j\in g} (w_g)^2}. \tag{20}$$

According to Proposition 2, the value of the Lipschitz constant for $\nabla\phi(\boldsymbol{\alpha})$ is

$$L(\phi) = \|C\|^2 = \gamma^2 \max_{j\in\{1,\ldots,p\}} \sum_{g\in\mathcal{G}\ \text{s.t.}\ j\in g} (w_g)^2. \tag{21}$$

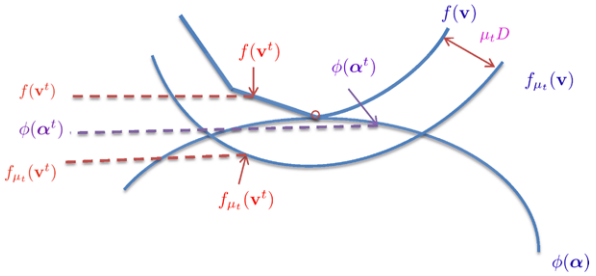

Fig. 1 Illustration of the excessive gap method

3.2.5 Excessive Gap Method

According to the Fenchel duality theorem [4], we know that under certain mild conditions which hold for our problem: $\min_{\mathbf{v}\in Q_1} f(\mathbf{v}) = \max_{\boldsymbol{\alpha}\in Q_2} \phi(\boldsymbol{\alpha})$, and for any $\mathbf{v} \in Q_1$ and $\boldsymbol{\alpha} \in Q_2$:

$$\phi(\boldsymbol{\alpha}) \le f(\mathbf{v}). \tag{22}$$

The key idea of the *excessive gap method* [19] is to simultaneously maintain two sequences $\{\mathbf{v}^t\}$, $\{\boldsymbol{\alpha}^t\}$ and a diminishing smoothness parameter sequence $\{\mu_t\}$ such that

$$f_{\mu_t}(\mathbf{v}^t) \le \phi(\boldsymbol{\alpha}^t); \qquad \mu_{t+1} \le \mu_t; \quad \text{and} \quad \lim_{t\to\infty} \mu_t = 0. \tag{23}$$

The geometric illustration of this idea is presented in Fig. 1. Combining (15), (22) and (23), we have

$$f_{\mu_t}(\mathbf{v}^t) \le \phi(\boldsymbol{\alpha}^t) \le f(\mathbf{v}^t) \le f_{\mu_t}(\mathbf{v}^t) + \mu_t D. \tag{24}$$

From (24), when $\mu_t \to 0$, we have $f(\mathbf{v}^t) \approx \phi(\boldsymbol{\alpha}^t)$, which are hence the optimal primal and dual solutions.

Moreover, in the excessive gap method, a gradient mapping operator $\psi : Q_2 \to Q_2$ is defined as follows: for any $\mathbf{z} \in Q_2$,

$$\psi(\mathbf{z}) = \arg\max_{\boldsymbol{\alpha}\in Q_2} \left\{ \langle \nabla\phi(\mathbf{z}), \boldsymbol{\alpha} - \mathbf{z} \rangle - \frac{1}{2} L(\phi) \|\boldsymbol{\alpha} - \mathbf{z}\|_2^2 \right\}. \tag{25}$$

As in Proposition 1, the gradient mapping operator ψ for our problem can also be computed in closed form as presented in the next proposition.

Proposition 3 *For any $\mathbf{z} \in Q_2$, $\psi(\mathbf{z})$ in (25) is the concatenation of subvectors $[\psi(\mathbf{z})]_g$ for all groups $g \in \mathcal{G}$. For any group g,*

$$[\psi(\mathbf{z})]_g = S_2\left(\mathbf{z}_g + \frac{[\nabla\phi(\mathbf{z})]_g}{L(\phi)}\right),$$

where $[\nabla\phi(\mathbf{z})]_g = \gamma w_g [\mathbf{v}(\mathbf{z})]_g$ is the gth subvector of $\nabla\phi(\mathbf{z})$ and the projection operator S_2 is defined in (14).

Algorithm 1 Excessive Gap Algorithm for Proximal Mapping Associated with Overlapping-group-lasso Penalty

Input: $\boldsymbol{\beta}$, γ, $\mathcal{G}$ and $\{w_g\}_{g\in\mathcal{G}}$
Initialization: (1) Construct C; (2) Compute $L(\phi)$ as in (21) and set $\mu_0 = 2L(\phi)$; (3) Set $\mathbf{v}^0 = \mathbf{v}(\mathbf{0}) = S_2(\boldsymbol{\beta})$; (4) Set $\boldsymbol{\alpha}^0 = \psi(\mathbf{0})$
Iterate For $t = 0, 1, 2, \ldots$, until convergence of $\mathbf{v}^t$:

1. Set $\tau_t = \frac{2}{t+3}$
2. Compute $\boldsymbol{\alpha}_{\mu_t}(\mathbf{v}^t)$ as in Proposition 1
3. Set $\mathbf{z}^t = (1-\tau_t)\boldsymbol{\alpha}^t + \tau_t \boldsymbol{\alpha}_{\mu_t}(\mathbf{v}^t)$
4. Update $\mu_{t+1} = (1-\tau_t)\mu_t$
5. Compute $\mathbf{v}(\mathbf{z}^t) = S_2(\boldsymbol{\beta} - C^T\mathbf{z}^t)$ as in (19)
6. Update $\mathbf{v}^{t+1} = (1-\tau_t)\mathbf{v}^t + \tau_t \mathbf{v}(\mathbf{z}^t)$
7. Update $\boldsymbol{\alpha}^{t+1} = \psi(\mathbf{z}^t)$ as in Proposition 3

Output: $\mathbf{v}^{t+1}$

Propositions 1, 2 and 3 have shown that, for our problem, all the essential ingredients of the excessive gap framework can be computed in closed form. We present the excessive gap method to solve proximal mapping associated with overlapping-group-lasso penalty in Algorithm 1.

3.2.6 Convergence Rate and Time Complexity

The Lemma 7.4 and Theorem 7.5 in [19] guarantee that both the starting points, $\mathbf{v}^0$ and $\boldsymbol{\alpha}^0$, and the sequences, $\{\mathbf{v}^t\}$ and $\{\boldsymbol{\alpha}^t\}$ in Algorithm 1 satisfy the key condition $f_{\mu_t}(\mathbf{v}^t) \le \phi(\boldsymbol{\alpha}^t)$ in (23). Using (24), the convergence rate can be established via the duality gap:

$$f\left(\mathbf{v}^t\right) - \phi\left(\boldsymbol{\alpha}^t\right) \le f_{\mu_t}\left(\mathbf{v}^t\right) + \mu_t D - \phi\left(\boldsymbol{\alpha}^t\right) \le \mu_t D. \tag{26}$$

From (26), we can see that the duality gap which characterizes the convergence rate is reduced at the same rate at which μ_t approaches to 0. According to Step 4 in Algorithm 1, the closed-form equation of μ_t can be written as

$$\begin{aligned}\mu_t &= (1-\tau_{t-1})\mu_{t-1} = \frac{t}{t+2}\mu_{t-1} = \frac{t}{t+2}\cdot\frac{t-1}{t+1}\cdots\frac{2}{4}\cdot\frac{1}{3}\cdot\mu_0 \\ &= \frac{2}{(t+1)(t+2)}\mu_0 = \frac{4L(\phi)}{(t+1)(t+2)} = \frac{4\|C\|^2}{(t+1)(t+2)}.\end{aligned} \tag{27}$$

Combining (26) and (27), we immediately obtain the convergence rate of Algorithm 1 [19].

Theorem 3 (Rate of convergence for duality gap) *The duality gap between the primal solution* $\{\mathbf{v}^t\}$ *and dual solution* $\{\boldsymbol{\alpha}^t\}$ *generated from Algorithm 1 satisfies*

$$f\left(\mathbf{v}^t\right) - \phi\left(\boldsymbol{\alpha}^t\right) \leq \mu_t D = \frac{4\|C\|^2 D}{(t+1)(t+2)}, \tag{28}$$

where $D = \max_{\boldsymbol{\alpha} \in Q_2} d(\boldsymbol{\alpha})$. *In other words, if we require that the duality gap is less than* ϵ, *Algorithm 1 needs at most* $\lceil 2\|C\|\sqrt{\frac{D}{\epsilon}} - 1 \rceil$ *iterations*.

According to [20], the convergence rate in Theorem 3 has already achieved the optimal rate for solving any convex smooth problem using only the first-order information.

As for time complexity, it is easy to verify that the per-iteration complexity time 1 is *linear* in $p + \sum_{g \in \mathcal{G}} |g|$, which is very cheap and hence can easily scale up to high-dimensional data.

4 Network-Structured Sparse CCA

In this section, we propose to adopt a network-based fusion penalty to incorporate the prior knowledge of the network structure into the sparse CCA framework. In addition, such a penalty can be naturally used for the group pursuit purpose which automatically groups the relevant variables into clusters.

The fused lasso model [24] assumes a linear ordering on the variables and uses the penalty $\sum_{j=1}^{p-1} |v_{j+1} - v_j|$ to enforce the learned parameters to be piece-wise constant. The network-based fusion penalty [14] extends the chain-structured fusion penalty to a general graph $G = (V, E)$, where $V = \{1, \ldots, p\}$ corresponds to p variables and E is the edge set. The graph G can be obtained from any prior knowledge. For the ease of illustration, we assume that the graph G is constructed from the correlation network with the edge set:

$$E = \left\{e = (i, j) : |r_{ij}| > \delta\right\}, \tag{29}$$

where r_{ij} is the correlation between ith and jth variable and $0 < \delta < 1$ is a predefined thresholding constant. We note that any prior knowledge can be used to construct G such as gene regulatory network. Using the correlation network, the network-based fusion penalty is defined as

$$\sum_{(i,j) \in E} w_{ij} \left| v_i - \text{sign}(r_{ij}) v_j \right| + c\|\mathbf{v}\|_1, \tag{30}$$

where as in fused lasso [24], ℓ_1-norm penalty is used to enforce the sparsity; c is the tuning parameter to balance the ℓ_1-norm penalty and the fusion penalty; w_{ij} is the predefined weight for each edge $e = (i, j)$. The term $\text{sign}(r_{ij})$ enforces the coefficients for positively correlated variables to be similar; while makes the sum of the coefficients for negatively correlated variables close to zero. If one has a prior belief of the correlation network structure, a natural way for assigning w_{ij} is to set

$w_{ij} = |r_{ij}|^q$, where r_{ij} is the correlation between the ith and jth variable; q models the strength of the prior: a larger q results in a stronger belief of the correlation network. For the purpose of simplicity, we set $w_{ij} = |r_{ij}|$ with $q = 1$ throughout this paper.

The network-based fusion penalty cannot only leverage the prior knowledge of the network structure to enhance the estimation but also be used for the group pursuit purpose. It can automatically group the relevant variables into clusters. In particular, assuming that G is a complete graph and $r_{ij} = 1$ and for all (i, j), the network-based fusion penalty directly conducts pairwise comparison between variables: when the estimated $\widehat{v}_i = \widehat{v}_j$ (i.e., $|\widehat{v}_i - \widehat{v}_j| = 0$), the ith and jth variables are grouped together. We identify all subgroups among p variables by conducting such a pairwise comparison and applying transitivity rule, i.e., $\widehat{v}_i = \widehat{v}_j$ and $\widehat{v}_j = \widehat{v}_k$ implies that the ith, jth, and kth variables are in the same group. Note that another non-convex group pursuit penalty has recently been proposed in [23]. However, it is computationally very expensive due to the non-convexity of the penalty and could be easily trapped by local minima.

As in group-structured sparse CCA, we also add an elastic-net penalty to address the problem of the collinearity and improve the stability. Our *network-based sparse CCA* is defined as

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{v}} -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} + \frac{\tau}{2}\mathbf{v}^T\mathbf{v} + \theta_1\|\mathbf{v}\|_1 + \theta_2 \sum_{(i,j)\in E} w_{ij}\left|v_i - \operatorname{sign}(r_{ij})v_j\right| \\ &\text{s.t.} \quad \|\mathbf{u}\|_2 \le 1, \qquad \|\mathbf{v}\|_2 \le 1, \qquad \|\mathbf{u}\|_1 \le c_1. \end{aligned} \tag{31}$$

4.1 Optimization

The optimization of (31) with respect to $\mathbf{v}$ can be written as

$$\min_{\mathbf{v}\in Q_1} f(\mathbf{v}) = l(\mathbf{v}) + P(\mathbf{v}), \tag{32}$$

where $\boldsymbol{\beta} = \frac{\mathbf{Y}^T\mathbf{X}\mathbf{u}}{\tau}$; $Q_1 = \{\mathbf{v} \,|\, \|\mathbf{v}\|_2 \le 1\}$ is the domain of $\mathbf{v}$; $l(\mathbf{v}) = \frac{1}{2}\|\mathbf{v} - \boldsymbol{\beta}\|_2^2$ is the Euclidean distance loss function and $P(\mathbf{v})$ is the network-based fusion penalty with $\lambda = \frac{\theta_1}{\tau}$ and $\gamma = \frac{\theta_2}{\tau}$:

$$P(\mathbf{v}) = \gamma \sum_{(i,j)\in E} w_{ij}\left|v_i - \operatorname{sign}(r_{ij})v_j\right| + \lambda\|\mathbf{v}\|_1. \tag{33}$$

The optimization problem in (32) is essentially a generalized fused lasso signal approximator problem where the fusion penalty is defined on an arbitrary graph. Here, the penalty $P(\mathbf{v})$ can also be reformulated into maximization problem over the auxiliary variables. Then the excessive gap algorithm described in Algorithm 1 can be applied in a straightforward manner.

As shown in [7], the first part of $P(\mathbf{v})$ in (33) can be written as $\sum_{e=(i,j)\in E} w_{ij}|v_i - \operatorname{sign}(r_{ij})v_j| \equiv \|H\mathbf{v}\|_1$, where $H \in \mathbb{R}^{|E|\times p}$ is the edge-vertex incident matrix with the

rows indexed by the edge set E and columns by p variables of $\mathbf{v}$:

$$H_{e=(i,j),k} = \begin{cases} w_{ij}, & \text{if } k = i, \\ -\text{sign}(r_{ij})w_{ij}, & \text{if } k = j, \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

Therefore, the overall penalty function $P(\mathbf{v})$ can be written as $\|C\mathbf{v}\|_1$, where

$$C = \begin{pmatrix} \gamma H \\ \lambda I \end{pmatrix} \in \mathbb{R}^{(|E|+p)\times p}, \tag{35}$$

and I is $p \times p$ identity matrix. By the fact that the ℓ_∞-norm and the ℓ_1-norm are dual of each other:

$$P(\mathbf{v}) \equiv \|C\mathbf{v}\|_1 = \max_{\boldsymbol{\alpha}\in Q_2} \boldsymbol{\alpha}^T C\mathbf{v}, \tag{36}$$

where $\boldsymbol{\alpha}$ is a vector of auxiliary variables with its domain $Q_2 = \{\boldsymbol{\alpha} \in \mathbb{R}^{|E|+p} \mid \|\boldsymbol{\alpha}\|_\infty \leq 1\}$.

After reformulating $P(\mathbf{v})$ into $\max_{\boldsymbol{\alpha}\in Q_2} \boldsymbol{\alpha}^T C\mathbf{v}$, we can directly apply the excessive gap method as described Sect. 3.2. The main difference is that $\boldsymbol{\alpha}_\mu(\mathbf{v}) = \arg\max_{\boldsymbol{\alpha}\in Q_2} \boldsymbol{\alpha}^T C\mathbf{v} - \mu d(\boldsymbol{\alpha})$ is now

$$\boldsymbol{\alpha}_\mu(\mathbf{v}) = S_\infty\left(\frac{C\mathbf{v}}{\mu}\right),$$

where S_∞ is the projection operator (shrinking to the ℓ_∞-ball) defined as follows:

$$S_\infty(x) = \begin{cases} x, & \text{if } -1 \leq x \leq 1, \\ 1, & \text{if } x > 1, \\ -1, & \text{if } x < -1. \end{cases} \tag{37}$$

And the gradient mapping $\psi(\mathbf{z})$ becomes

$$\psi(\mathbf{z}) = \arg\max_{\boldsymbol{\alpha}\in Q_2} \left\{ \langle \nabla\phi(\mathbf{z}), \boldsymbol{\alpha} - \mathbf{z} \rangle - \frac{1}{2} L(\phi)\|\boldsymbol{\alpha} - \mathbf{z}\|_2^2 \right\} = S_\infty\left(\mathbf{z} + \frac{\nabla\phi(\mathbf{z})}{L(\phi)}\right).$$

The excessive gap method will converge in at most $\lceil 2\|C\|\sqrt{\frac{D}{\epsilon}} - 1 \rceil$ iterations according to Theorem 3; and the per-iteration time complexity for network-based fusion penalty is *linear* in $p + |E|$.

5 Experiment

In this section, we present the numerical results on both simulated and real datasets to illustrate the performance of the proposed algorithm.

Table 1 Comparison between ExGap with Grad and IPM for SOCP

$\|\mathcal{G}\| = 20$, $p = 20{,}100$					$\|\mathcal{G}\| = 40$, $p = 40{,}100$				
$\gamma = 0.2$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 0.4$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	4.059E-3	4.4063E+3	4.900E-12	2	ExGap	1.960E-2	8.8682E+3	5.259E-11	3
Grad	3.135E+1	4.4063E+3	–	69	Grad	1.753E-1	8.8682E+3	–	19
SOCP	4.866E+1	4.4063E+3	8.723E-8	17	SOCP	9.642E+2	8.8682E+3	7.048E-9	20
$\gamma = 2$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 4$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	2.591E-2	4.4123E+3	1.197E-9	9	ExGap	1.626E-1	8.8851E+3	8.384E-7	18
Grad	3.496E+1	4.4123E+3	–	773	Grad	9.462E+1	8.8851E+3	–	1036
SOCP	4.440E+1	4.4123E+3	8.267E-5	13	SOCP	2.952E+3	8.8851E+3	3.844E-8	18
$\|\mathcal{G}\| = 100$, $p = 100{,}100$					$\|\mathcal{G}\| = 500$, $p = 500{,}100$				
$\gamma = 1$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 5$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	2.172E-1	2.2296E+4	6.646E-8	9	ExGap	4.381E+1	1.1211E+5	4.816E-8	51
Grad	5.219E+1	2.2296E+4	–	199	Grad	1.021E+2	1.1211E+5	–	723
$\gamma = 10$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 50$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	1.288E+0	2.2362E+4	7.891E-7	48	ExGap	1.855E+2	1.1250E+5	9.757E-7	2144
Grad	9.463E+1	2.2363E+4	–	1293	Grad	2.831E+3	1.1286E+5	–	*20000*
$\|\mathcal{G}\| = 1000$, $p = 1{,}000{,}100$					$\|\mathcal{G}\| = 5000$, $p = 5{,}000{,}100$				
$\gamma = 10$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 50$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	2.232E+2	2.2456E+5	9.376E-7	102	ExGap	1.579E+3	1.1245E+6	9.615E-7	1752
Grad	2.432E+2	2.2456E+5	–	867	Grad	2.977E+5	1.1261E+6	–	*20000*
$\gamma = 100$	CPU	Primal Obj	Rel_Gap	Iter	$\gamma = 500$	CPU	Primal Obj	Rel_Gap	Iter
ExGap	8.108E+3	2.2500E+5	8.677E-7	3872	ExGap	7.432E+3	1.1250E+6	9.851E-7	9080
Grad	5.718E+3	2.2668E+5	–	*20000*	Grad	2.981E+5	1.1498E+6	–	*20000*

5.1 Computational Efficiency of Excessive Gap Method

In this section, we evaluate the scalability and efficiency of the excessive gap method (ExGap) for solving the proximal mapping associated with the overlapping-group-lasso penalty:

$$\arg\min_{\mathbf{v}:\|\mathbf{v}\|_2 \le 1} f(\mathbf{v}) = \frac{1}{2}\|\mathbf{v} - \boldsymbol{\beta}\|_2^2 + \gamma \sum_{g \in \mathcal{G}} w_g \|\mathbf{v}_g\|_2,$$

where $\boldsymbol{\beta}$ is given and all w_g are assumed to be 1 for simplicity. We compare ExGap with two widely used optimization methods: (1) formulating the problem into a second-order cone programming (SOCP) and solving by interior-point method (IPM) using the state-of-the-art MATLAB package SPDT3 [25] and (2) projected subgradient-descent method (Grad) The stepsize of the Grad is set to $\frac{\eta}{\sqrt{t}}$ as suggested in [8], where the constant η is carefully tuned to be $\frac{0.1}{\sqrt{p}}$. All of the experiments are performed on a PC with Intel Core 2 Quad Q6600 2.4 GHz CPU and 4 GB RAM. The software is written in MATLAB. We terminate the optimization proce-

dure of ExGap and SOCP when the relative duality gap (Rel_Gap) is less than 10^{-6}: $\text{Rel\_Gap} = \frac{|f(\mathbf{v}^t)-\phi(\boldsymbol{\alpha}^t)|}{1+|f(\mathbf{v}^t)|+|\phi(\boldsymbol{\alpha}^t)|} \leq 10^{-6}$. For Grad, since there is no dual solutions, we use objective of ExGap as the "optimal" objective for Grad and stop Grad when its objective is less than 1.00001 times the objective of ExGap. We set the maximum iteration for all methods to be 20,000.

More specifically, we generate the data using the similar approach as in [10], with an overlapping group structure imposed on $\boldsymbol{\beta}$ as described below. Assuming that inputs are ordered and each group is of size 1000, we define a sequence of groups of 1000 adjacent inputs with an overlap of 100 variables between two successive groups, i.e. $\mathcal{G} = \{\{1, \ldots, 1000\}, \{901, \ldots, 1900\}, \ldots, \{p-999, \ldots, p\}\}$ with $p = 1000|\mathcal{G}| + 100$. We set the support of $\boldsymbol{\beta}$ to be the first half of the variables and set the values of $\boldsymbol{\beta}$ in the support to be 1 and otherwise 0.

We vary the number of the groups $|\mathcal{G}|$ and report the CPU time in *seconds* (CPU), primal objective value (Primal Obj), relative duality gap (Rel_Gap) and the number of iterations (Iter) in Table 1. For each setting of $|\mathcal{G}|$, we use two levels of regularization: (1) $\gamma = \frac{|\mathcal{G}|}{100}$ and (2) $\gamma = \frac{|\mathcal{G}|}{10}$. Note that when $|\mathcal{G}| \geq 50$ ($p \geq 50, 100$), we are unable to collect results for SOCP, because they lead to out-of-memory errors due to the large storage requirement for solving the Newton linear system. In addition, for large $|\mathcal{G}|$, Grad cannot converge in 20,000 iterations. From Table 1, we can see that ExGap achieves the same objective value as SOCP with small relative duality gap. For all different scales of the problem, ExGap is much more efficient than SOCP and Grad. It can easily scale up to high-dimensional data with millions of variables. Another interesting observation is that, for smaller γ, which leads to smaller $\|C\|$ and $L(\phi)$, the convergence of ExGap is much faster. This observation is consistent with convergence result in Theorem 3. It suggests that ExGap is more efficient when the non-smooth part plays less important role in the optimization problem.

5.2 Simulations

In this and next subsections, we use simulated data and a real eQTL dataset to investigate the performance of the overlapping group-structured and network-structured sparse CCA. All the regularization parameters are chosen from $\{0.01, 0.02, \ldots, 0.09, 0.1, 0.2, \ldots, 0.9, 1, 2, 10\}$ and set using the permutation-based method in [28]. Instead of tuning all the parameters on a multi-dimensional grid which is computationally heavy, we first train the ℓ_1-regularized sparse CCA (i.e. $P_1(\mathbf{u}) = \|\mathbf{u}\|_1$, $P_2(\mathbf{v}) = \|\mathbf{v}\|_1$) and the tuned regularization parameter c_1 in (1) is used for all structured models. For the overlapping-group-lasso penalty in (4), all the group weights $\{w_g\}$ are set to 1. In addition, we observe that the learned sparsity pattern is quite insensitive to the parameter τ (the regularization parameter for the quadratic penalty); and therefore we set it to 1 for simplicity. For all algorithms, we use 10 random initializations of $\mathbf{u}$ and select the results that lead to the largest correlation.

5.2.1 Group-Structured Sparse CCA

In this section, we conduct the simulation where the overlapping group structure in $\mathbf{v}$ is given as *a priori*. We generate the data $\mathbf{X}$ and $\mathbf{Y}$ with $n = 50$, $d = 100$ and $p = 82$

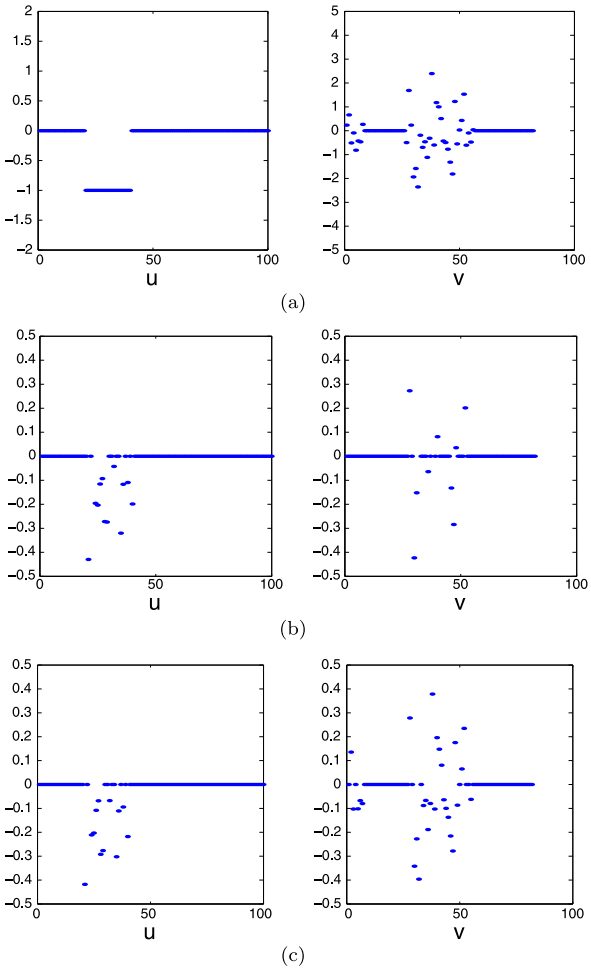

Fig. 2 (**a**) True **u** and **v**; (**b**) Estimated **u** and **v** using the ℓ_1-regularized sparse CCA; (**c**) Estimated **u** and **v** using the group-structured sparse CCA

as follows. Let **u** be a vector of length d with 20 0s, 20 -1s, and 60 0s. We construct **v** with $p = 82$ variables using the same approach as in [10]: assuming that **v** is covered by 10 groups; each group has 10 variables with 2 variables overlapped between every two successive groups, i.e. $\mathcal{G} = \{\{1, \ldots, 10\}, \{9, \ldots, 18\}, \ldots, \{73, \ldots, 82\}\}$. For the indices of the 2nd, 3rd, 8th, 9th and 10th groups, we set the corresponding entries of **v** to be zeros and the other entries are sampled from i.i.d. $N(0, 1)$. In addition, we randomly generate a latent vector **z** of length n and normalize it to unit length.

We generate the data matrix **X** with each $X_{ij} \sim N(z_i u_j, 1)$ and **Y** with each $Y_{ij} \sim N(z_i v_j, 1)$. The true and estimated vectors for **u** and **v** are presented in Fig. 2. For the group-structured sparse CCA, we add the regularization $\sum_{g \in \mathcal{G}} \|\mathbf{v}_g\|_2$ on **v** where $\mathcal{G}$ is taken from the prior knowledge. It can be seen that the group-structured sparse CCA recovers the true **v** much better while the simple ℓ_1-regularized sparse CCA leads to an over-sparsified **v** vector.

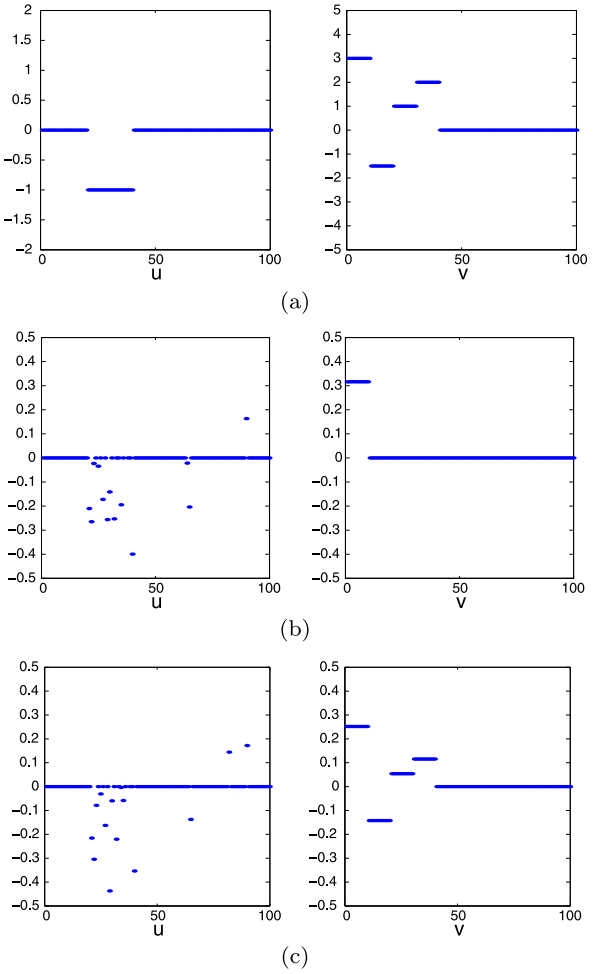

Fig. 3 (**a**) True **u** and **v**; (**b**) Estimated **u** and **v** using the ℓ_1-regularized sparse CCA; (**c**) Estimated **u** and **v** using the network-structured sparse CCA

5.2.2 Network-Structured Sparse CCA for Group Pursuit

In this simulation we assume that the group structure over **v** is unknown and the goal is to uncover the group structure using the network-structured sparse CCA. We generate the data **X** and **Y** with $n = 50$ and $p = d = 100$ as follows. Let **u** be a vector of length d with 20 0s, 20 -1s, and 60 0s as in the previous simulation study; and **v** be a vector of length p with 10 3s, 10 -1.5s, 10 1s, 10 2s and 60 0s. In addition, we randomly generate a latent vector **z** of length n and normalize it to unit length.

We generate **X** with each sample $\mathbf{x}_i \sim N(z_i\mathbf{u}, 0.1I_{d\times d})$; and **Y** with each sample $\mathbf{y}_i \sim N(z_i\mathbf{v}, 0.1\Sigma_y)$ where $(\Sigma_y)_{jk} = \exp^{-|v_j - v_k|}$. We conduct the group pursuit via the network-structured sparse CCA in (31), where we add the fusion penalty for each pair of variables in **v**, i.e., E is the edge set of the complete graph. The estimated vector **u** and **v** are presented in Fig. 3(c). It can be easily seen that the network-structured sparse CCA correctly captures the group structure in the **v** vector. This

Table 2 List of pathways with at least 2 selected genes in the pathway: the first two columns are the pathway ID and annotation from KEGG, the third column is the number of selected genes in this pathway; the fourth column is the ratio of the number of selected genes in the pathway (third column) over the number of genes in the dataset in the pathway; the last column gives the p-values which is calculated as the hypergeometric probability to get so many genes for a KEGG pathway annotation

ID	Annotation	No. Genes	Ratio	p-value
00072	Synthesis and degradation of ketone bodies	2	100.0	7.65E-4**
00280	Valine, leucine and isoleucine degradation	2	18.18	3.58E-2
00620	Pyruvate metabolism	2	6.06	2.35E-1
00640	Propanoate metabolism	2	18.18	3.58E-2
00650	Butanoate metabolism	2	10.00	1.05E-2
00900	Terpenoid backbone biosynthesis	7	53.85	1.26E-8**
01100	Metabolic pathways	29	4.58	1.00E-3*
01110	Biosynthesis of secondary metabolites	15	6.64	7.24E-4**
00100	Steroid biosynthesis	10	66.67	2.84E-13**
00190	Oxidative phosphorylation	4	5.26	1.59E-1
00514	O-Mannosyl glycan biosynthesis	3	23.07	4.81E-3
00600	Sphingolipid metabolism	2	15.38	4.91E-2
03050	Proteasome	2	5.71	2.56E-1
04144	Endocytosis	2	5.56	2.66E-1

experiment demonstrates that network-structured sparse CCA could be a useful tool for conducting group pursuit in the CCA framework.

5.3 Real eQTL Data

5.3.1 Group-Structured Sparse CCA: Pathway Selection

In this section, we report experiment results on a yeast eQTL data [5, 32]. In particular, we have two data matrices, $\mathbf{X}$ and $\mathbf{Y}$. $\mathbf{X}$ contains $d = 1260$ SNPs from the chromosomes 1–16 for $n = 124$ yeast strains. $\mathbf{Y}$ is the gene expression data of $p = 1{,}155$ genes for the same 124 yeast strains. All these $p = 1{,}155$ genes are from the KEGG database [13]. According to KEGG, these genes belong to 92 pathways. We treat each pathway as a group. The statistics of the 92 pathways are summarized as follows: the average number of genes in each group is 25.78; and the largest group has 475 genes. One thing to note is that there are a lot of overlapping genes among the 92 pathways and the average appearance frequency of each gene in the pathways is 2.05. To achieve more refined resolution of gene selection, besides the 92 pathway groups, we also add in $p = 1{,}155$ groups where each group only has one singleton gene. Therefore, instead of just selecting genes at the pathway level, we could also select genes within each pathway.

Using the group-structured sparse CCA, we selected altogether 121 SNPs (i.e. the number of nonzero elements in estimated $\mathbf{u}$) and 47 genes (i.e. the number of nonzero elements in estimated $\mathbf{v}$). These 47 genes spread over 32 pathways. Such a high coverage of pathways is mainly due to the fact that several selected genes

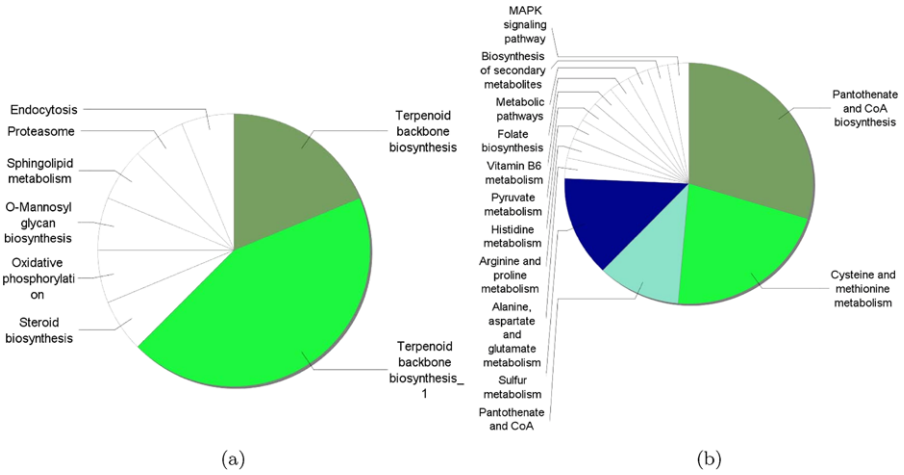

Fig. 4 Overview chart of KEGG functional enrichment using (**a**) the group-structured sparse CCA; (**b**) ℓ_1-regularized sparse CCA

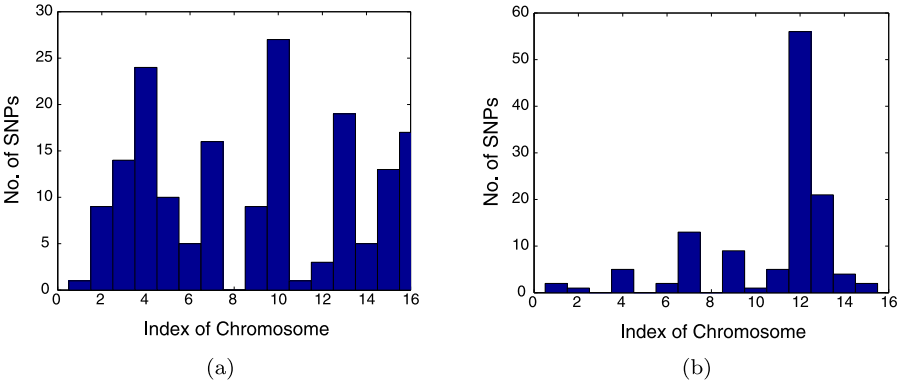

Fig. 5 The number of selected SNPs in each chromosome using (**a**) the ℓ_1-regularized sparse CCA; (**b**) the group-structured sparse CCA

are important in different biological processes and hence each of them belongs to multiple pathways. There are 14 pathways that contain at least two selected genes as listed in Table 2. Among these 14 pathways, 5 of them are highly significant with the p-value less than 0.001. Using the tool ClueGo [3], the overview chart of KEGG enrichment on the selected genes is presented in Fig. 4(a). We can see from Fig. 4(a) that the *Terpenoid backbone biosynthesis* is the most important functional group, which is a large class of natural products consisting of isoprene (C5) units. In fact, the first 8 pathways in Table 2 are all closely related to *Terpenoid backbone biosynthesis*. As a comparison, the ℓ_1-regularized sparse CCA selects 173 SNPs and 71 genes and these 71 genes belong to 50 pathways. The KEGG enrichment for ℓ_1-regularized sparse CCA is presented Fig. 4(b).

Table 3 GO enrichment analysis for the selected genes using the group-structured sparse CCA: the first two columns are the GO ID (category) and annotation, the third column is the number of selected genes having the GO annotation, the fourth column is the GO cluster size and the last column gives the p-value. The rows are ranked according to the increasing order of p-values

GO ID	GO Attribute	N	X	p-value
0006696	ergosterol biosynthetic process	17	25	2.71E-32
0008204	ergosterol metabolic process	17	27	2.09E-31
0006694	steroid biosynthetic process	17	34	5.61E-29
0016126	sterol biosynthetic process	17	34	5.61E-29
0016125	sterol metabolic process	17	45	2.52E-26
0008202	steroid metabolic process	17	49	1.46E-25
0008610	lipid biosynthetic process	20	149	4.71E-21
0006066	cellular alcohol metabolic process	21	210	2.09E-19
0044255	cellular lipid metabolic process	21	259	1.71E-17
0006629	lipid metabolic process	21	279	8.00E-17
0003824	catalytic activity	42	2195	9.20E-15
0006720	isoprenoid metabolic process	7	13	1.35E-12
0008299	isoprenoid biosynthetic process	7	13	1.35E-12
0005783	endoplasmic reticulum	20	410	2.28E-12
0043094	cellular metabolic compound salvage	12	159	1.08E-09
0005789	endoplasmic reticulum membrane	15	293	1.42E-09
0044432	endoplasmic reticulum part	15	317	4.23E-09
0006695	cholesterol biosynthetic process	4	5	1.36E-08
0008203	cholesterol metabolic process	4	5	1.36E-08
0031090	organelle membrane	23	954	4.61E-08
0044444	cytoplasmic part	39	2810	6.43E-08
0044237	cellular metabolic process	46	4187	1.08E-07
0044238	primary metabolic process	43	3581	1.86E-07
0055114	oxidation reduction	13	309	2.32E-07
0016491	oxidoreductase activity	13	318	3.23E-07
0008152	metabolic process	46	4321	4.35E-07

In addition to the above pathway analysis, we also perform the enrichment analysis on the selected genes according to the standard Gene Ontology (GO). The enrichment is carried out using the standard annotation tool from [2] and the result is shown in Table 3 with the cutoff point set to 1E-3. We see that most of clusters are significantly enriched and many of them are known to be explicitly relevant. For example, *isoprenoid metabolic process* and *isoprenoid biosynthetic process* are very closely related to *terpenoid backbone biosynthesis*.

The group-structured sparsity-inducing penalty on genes will also affect the selection of SNPs. As a comparison, the ℓ_1-regularized sparse CCA selects 173 SNPs. while the group-structured sparse CCA selects only 121 SNPs. The number of selected SNPs in each chromosome is presented in Fig. 5. As we can see, most of the

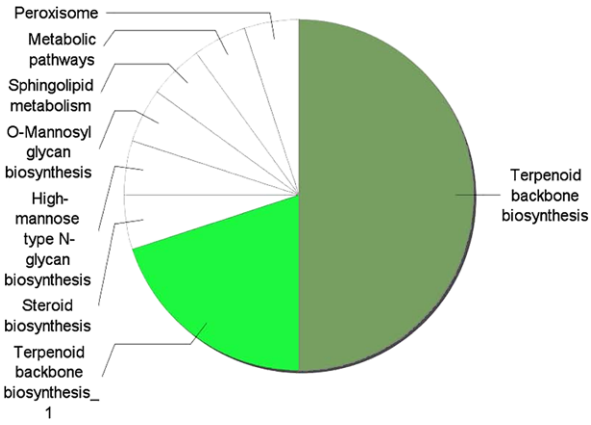

Fig. 6 Overview chart of KEGG functional enrichment using the tree-structured sparse CCA

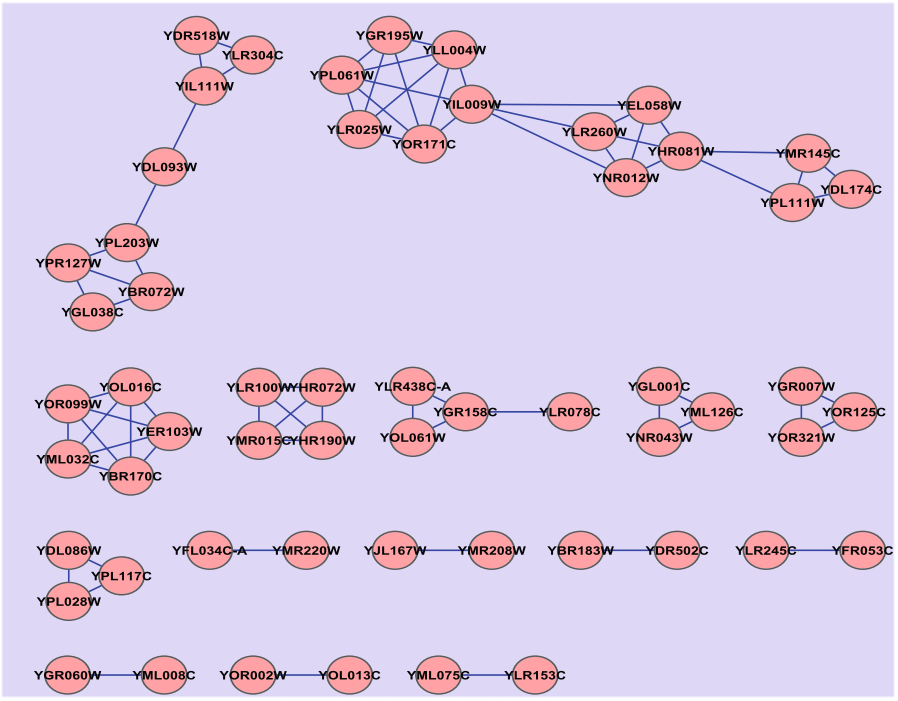

Fig. 7 Selected genes and their relationship estimated by the network-structured sparse CCA

selected SNPs using the group-structured sparse CCA belong to Chromosome 12 and 13.

5.3.2 Tree-Structured Sparse CCA

In this experiment, rather than utilizing the group information extracted from the KEGG pathways [13], we learn a hierarchical tree structure on the same yeast dataset

and then use the learned tree structure to define the groups. In more details, we run the hierarchical agglomerative clustering on the $p \times p$ correlation matrix of $\mathbf{Y}$ where each leaf node of the tree corresponds to a single gene. We discard the tree nodes for weak correlations near the root of the tree. In particularly, we calculate the distance of each tree node to the root; normalize them by the maximum distance and discard those nodes with the distance less than 0.3. Each node of the tree defines a group which contains the genes represented by its leaf children nodes. Finally, we obtain 973 groups corresponding to internal nodes and 1,155 singleton group corresponding to leaf nodes. After obtaining the group structure induced from the clustering tree, we then apply the group-structured sparse CCA as in the last section. We name the obtained model as *tree-structured sparse CCA*.

The tree-structured sparse CCA selects altogether 123 SNPs and 66 genes. An overview chart of functional enrichment using the KEGG pathways is presented in Fig. 6. As we can see, most of the functions are identical to those learned by the group-structured sparse CCA with the group information obtained from KEGG, e.g. *Terpenoid backbone biosynthesis*, *Steroid biosynthesis*, *O-Mannosyl glycan biosynthesis*, *Sphingolipid*, etc. We also perform the GO enrichment analysis on the selected genes and obtained 30 GO clusters. All of these clusters are significantly enriched and 25 of them are the same as the GO clusters obtained by the group-structured sparse CCA with the groups from KEGG. This experiment suggests that, even without any prior knowledge of the pathway information as group structure, our correlation-based tree-structured sparse CCA can also select the relevant genes and provide the similar enrichment results.

5.3.3 Network-Structured Sparse CCA

In this section, we apply the network-structured sparse CCA on the yeast eQTL data based on the correlation network. More specifically, we compute the pairwise correlation among gene expression levels and construct the edge set E by those pairs of variables with the absolute value of the correlation greater than 0.8. With the edge set E, we apply the network-based sparse CCA and select 67 genes. To visualize the network induced by the estimated $\widehat{\mathbf{v}}$, we should connect the (i, j) pair if $\widehat{v}_i = \widehat{v}_j$ which indicates that the ith gene and jth gene are in the same group. However, due to the numerical error introduced during the computation, $\widehat{v}_i$ cannot be exactly the same as $\widehat{v}_j$. Therefore, we connect two selected genes (nodes) if $|\widehat{v}_i - \widehat{v}_j| \leq 10^{-3}$ and present the learned network structure in Fig. 7 (singleton nodes are not plotted). We observe that there are two obvious clusters. With the learned clustering structure, we can study the functional enrichment of each cluster separately, which could lead to more elaborate enrichment analysis as compared to the analysis of the selected genes all together.

6 Conclusions

The sparse CCA model [28, 29] provides an appealing framework to investigate genome-wide association study. In this paper, we further extend this model so that it

can exploit either the group or the network structure information among variables via the structured sparsity-inducing penalty. Moreover, we provide an efficient optimization algorithm that can solve large-scale structured sparse CCA problems efficiently. Compared to the previous sparse CCA methods, our structured sparse CCA models allow for more rich structure information.

Acknowledgements We would like to thank Seyoung Kim for pointing out and helping us prepare the yeast data. We also thank Jaime G. Carbonell for the helpful discussion on the excessive gap methods and related first-order methods. We would also like to thank anonymous reviewers and the associate editor for their constructive comments on improving the quality of the paper.

References

1. Argyriou A, Evgeniou T, Pontil M (2008) Convex multi-task feature learning. Mach Learn 73(3):243–272
2. Berriz G, Beaver J, Cenik C, Tasan M, Roth F (2009) Next generation software for functional trend analysis. Bioinformatics 25(22):3043–3044
3. Bindea G et al (2009) Cluego: a cytoscape plug-in to decipher functionally grouped gene ontology and pathway annotation networks. Bioinformatics 25(8):1091–1093
4. Borwein J, Lewis AS (2000) Convex analysis and nonlinear optimization: theory and examples. Springer, Berlin
5. Brem RB, Krulyak L (2005) The landscape of genetic complexity across 5,700 gene expression traits in yeast. Proc Natl Acad Sci 102(5):1572–1577
6. Cao KL, Pascal M, Robert-Cranie C, Philippe B (2009) Sparse canonical methods for biological data integration: application to a cross-platform study. Bioinformatics 10
7. Chen X, Lin Q, Kim S, Carbonell J, Xing E (2011) Smoothing proximal gradient method for general structured sparse learning. In: Uncertainty in artificial intelligence
8. Duchi J, Singer Y (2009) Efficient online and batch learning using forward backward splitting. J Mach Learn Res 10:2899–2934
9. Hiriart-Urruty JB, Lemarechal C (2001) Fundamentals of convex analysis. Springer, Berlin
10. Jacob L, Obozinski G, Vert JP (2009) Group lasso with overlap and graph lasso. In: ICML
11. Jenatton R, Audibert J, Bach F (2009) Structured variable selection with sparsity-inducing norms. Tech rep, INRIA
12. Jenatton R, Mairal J, Obozinski G, Bach F (2010) Proximal methods for sparse hierarchical dictionary learning. In: ICML
13. Kanehisa M, Goto S (2000) Kegg: Kyoto encyclopedia of genes and genomes. Nucleic Acids Res 28:27–30
14. Kim S, Xing E (2009) Statistical estimation of correlated genome associations to a quantitative trait network. PLoS Genet 5(8)
15. Kim S, Xing EP (2010) Tree-guided group lasso for multi-task regression with structured sparsity. In: ICML
16. Mairal J, Jenatton R, Obozinski G, Bach F (2010) Network flow algorithms for structured sparsity. In: NIPS
17. Mol D, Vito D, Rosasco L (2009) Elastic net regularization in learning theory. J Complex 25:201–230
18. Naylor M, Lin X, Weiss S, Raby B, Lange C (2010) Using canonical correlation analysis to discover genetic regulatory variants. PLoS One
19. Nesterov Y (2003) Excessive gap technique in non-smooth convex minimization. Tech rep, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE)
20. Nesterov Y (2003) Introductory lectures on convex optimization: a basic course. Kluwer Academic, Dordrecht
21. Nesterov Y (2005) Smooth minimization of non-smooth functions. Math Program 103(1):127–152
22. Parkhomenko E, Tritchler D, Beyene J (2009) Sparse canonical correlation analysis with application to genomic data integration. Stat Appl Genet Mol Biol 8:1–34
23. Shen X, Huang HC (2010) Grouping pursuit through a regularization solution surface. J Am Stat Assoc 105(490):727–739

24. Tibshirani R, Saunders M (2005) Sparsity and smoothness via the fused lasso. J R Stat Soc B 67(1):91–108
25. Tütüncü RH, Toh KC, Todd MJ (2003) Solving semidefinite-quadratic-linear programs using sdpt3. Math Program 95:189–217
26. Waaijenborg S, de Witt Hamer PV, Zwinderman A (2008) Quantifying the association between gene expressions and DNA-markers by penalized canonical correlation analysis. Stat Appl Genet Mol Biol 7
27. Wen Z, Goldfarb D, Yin W (2009) Alternating direction augmented Lagrangian methods for semidefinite programming. Tech rep, Dept of IEOR, Columbia University
28. Witten D, Tibshirani R (2009) Extensions of sparse canonical correlation analysis with applications to genomic data. Stat Appl Genet Mol Biol 8(1):1–27
29. Witten D, Tibshirani R, Hastie T (2009) A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics 10:515–534
30. Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. J R Stat Soc B 68:49–67
31. Zhao P, Rocha G, Yu B (2009) Grouped and hierarchical model selection through composite absolute penalties. Ann Stat 37(6A):3468–3497
32. Zhu J et al (2008) Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. Nat Genet 40:854–861
33. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc B 67:301–320