Efficient Computation for Differential Network Analysis with Applications to Quadratic Discriminant Analysis

Yuqing Pan[a], Qing Mai[b,*]

[a]*Microsoft, Redmond, WA 98052*
[b]*Department of Statistics, Florida State University, Tallahassee, FL, 32306*

Abstract

Differential network analysis is an important statistical problem with wide applications. Many statisticians focus on binary problems and propose to perform such analysis by obtaining sparse estimates of the difference between precision matrices. These methods are supported by excellent theoretical properties and practical performance. However, efficient computation for these methods remains a challenging problem. A novel algorithm referred to as the SMORE algorithm is proposed for differential network analysis. The SMORE algorithm has low storage cost and high computation speed, especially in the presence of strong sparsity. In the meantime, the SMORE algorithm provides a unified framework for binary and multiple network problems. In addition, the SMORE algorithm can be applied in high-dimensional quadratic discriminant analysis problems as well, leading to a new approach for multiclass high-dimensional quadratic discriminant analysis. Numerical studies confirm the stability and the efficiency of the proposed SMORE algorithm in both differential network analysis and quadratic discriminant analysis.

Keywords: Coordinate descent, Dantzig selector, differential network analysis, LASSO, quadratic discriminant analysis

1. Introduction

Due to its wide applications in scientific problems, differential network analysis has been receiving increasing attention in the literature. For example, researchers often perform such analysis to understand the interactions among proteins or genes in biology studies [1, 2, 3, 4, 5, 6]. Differential network analysis aims to detect changes of networks under different conditions. Consider variables $\mathbf{X} \in \mathbb{R}^p$ and the condition label $Y = 1, \ldots, K$. In differential network analysis, it is often assumed that

$$\mathbf{X} \mid (Y = k) \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad k = 1, \ldots, K, \tag{1.1}$$

*Corresponding author at: Department of Statistics, Florida State University, Tallahassee, FL, 32306
Email address: mai@stat.fsu.edu (Qing Mai)

where $\boldsymbol{\mu}_k \in \mathbb{R}^p$, $\boldsymbol{\Sigma}_k \in \mathbb{R}^{p\times p}$. Within the level $Y = k$, $\mathbf{X}$ follows the Gaussian graphical model. Let $\boldsymbol{\Omega}_k = \boldsymbol{\Sigma}_k^{-1}$. Then $\boldsymbol{\Omega}_k$ characterizes the conditional dependence structure among $\mathbf{X}$ at Level k. Our goal is estimating the differences among $\boldsymbol{\Omega}_k$'s. It is often assumed that the differences are sparse, indicating that most pairs (X_i, X_j) have the constant interactions in all the levels, while we strive to identify the pairs with different interactions across levels [7, 8, e.g].

An intuitive approach to differential network analysis is to first estimate $\boldsymbol{\Omega}_k$, and then take the differences among the estimated $\boldsymbol{\Omega}_k$'s. The estimation of $\boldsymbol{\Omega}_k$ can be performed either individually [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] or jointly [21, 22, 23, 24]. However, this approach requires stronger assumptions than we desire. In order to estimate $\boldsymbol{\Omega}_k$ in high dimensions by the above methods, we need to assume that each $\boldsymbol{\Omega}_k$ is sparse. This assumption can be stringent in practice. For example, [25, 26] showed that transcriptional networks are often not sparse on their own. From the statistical perspective, sparsity in the individual $\boldsymbol{\Omega}_k$'s is much stronger than sparsity in their differences. For example, consider two precision matrices $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$. If both of them are sparse, then apparently their difference $\boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_1$ has to be sparse as well. However, when $\boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_1$ is sparse, we do not necessarily have sparse $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$. By directly assuming that $\boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_1$ is sparse, we can avoid sparsity assumptions on $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ individually.

Consequently, efforts have been spent on direct estimation of the differential networks [7, 8, 27, 28]. These methods directly estimate the differential networks, without imposing assumptions on the individual precision matrices. Thus, they are particularly suitable when we are reluctant to make sparsity assumptions on each network, as is the case in the transcriptional networks. These methodological developments greatly enrich the literature of differential network analysis. New statistical theories have been derived for these methods, and researchers have applied them on real datasets to obtain new findings.

However, the computation of differential network analysis remains a critical issue. Some of the existing methods can be demanding on the storage, while others have room for improvement on the computation efficiency. Note that we aim to estimate parameters of dimensions $p \times p$, where p can be very large in practice. Therefore, it is essential to accompany the methods with algorithms that scale well with p. To this end, we propose an algorithm with cheaper and faster computation for the two promising differential network analysis methods in [7, 28]. Complexity analysis shows that the improvements are particularly significant when the truth is very sparse. Moreover, most existing methods focus on binary problems, but our algorithm easily accommodates multiple network problems.

Further, our algorithm can be applied to quadratic discriminant analysis (QDA) as well. QDA is one of the most classical classification methods [29, e.g]. It provides nonlinear separation of the data by modeling heterogeneity. There have been considerable interests in its high-dimensional generalizations [30, 31, 28]. Our algorithm can be applied in high-dimensional QDA. The QDA classifier includes quadratic and linear terms. Similar to differential

network analysis, the quadratic terms are differences among precision matrices, and our algorithm easily provides estimates for them. On the other hand, by supplying appropriate initial estimates to our algorithm, we are able to obtain estimates for the linear terms as well. Moreover, unlike existing methods that focus on binary problems, our algorithm provides a framework for multiclass QDA problems.

In our development, we first introduce a generic optimization problem called the SMORE problem, in short for Second-order Matrix Optimization with REgularization. The SMORE problem includes differential network analysis and QDA as special cases. Then we develop the SMORE algorithm to solve the SMORE problem. The SMORE algorithm has low storage and computation costs. Theories confirm the convergence of the SMORE algorithm under mild conditions. We then apply this algorithm in differential network analysis and QDA. Numerical studies demonstrate the computational advantages of the SMORE algorithm in both applications. The rest of the paper is organized as follows. In Section 2 we briefly review differential network analysis. In Section 3 we introduce the SMORE problem and the SMORE algorithm. Convergence analysis is provided as well. Section 4 contains the application of the SMORE algorithm in binary differential network analysis. The extension to multiple network problems is discussed in Section 5. We review QDA and consider its high-dimensional extension in Section 6. In Sections 7 & 8, we present numerical studies on simulated and real data. A discussion is given in Section 9. All the technical proofs are relegated to the appendix.

2. Background on differential network analysis

We review the differential network (DN) analysis methods by [7] and [28] that motivate our algorithmic developments. For simplicity, we distinguish them by the penalties they use. We refer to the proposal by [7] as the Dantzig-DN method, because it employs the Dantzig selector. [28] applies the LASSO penalty and we refer to their method as LASSO-DN. Both methods assume the model in (1.1) restricted to $K = 2$. The goal is to estimate $\boldsymbol{\Delta} = \boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_1$. Straightforward calculation shows that

$$\boldsymbol{\Sigma}_2 \boldsymbol{\Delta} \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2. \tag{2.1}$$

Let vec denote the vectorization operator that converts a $p \times p$ matrix to a p^2-dimensional vector, and $\otimes$ denote the Kronecker product. Then (2.1) implies that

$$(\boldsymbol{\Sigma}_1 \otimes \boldsymbol{\Sigma}_2)\mathrm{vec}(\boldsymbol{\Delta}) = \mathrm{vec}(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2). \tag{2.2}$$

Hence, the Dantzig-DN method estimates $\boldsymbol{\Delta}$ by replacing $\boldsymbol{\Sigma}_k$'s with their sample estimates $\widehat{\boldsymbol{\Sigma}}_k$ and applying the Dantzig selector [32]:

$$\widehat{\boldsymbol{\Delta}} = \arg\min_{\boldsymbol{\Delta}} \|\mathrm{vec}(\boldsymbol{\Delta})\|_1, \quad \text{s.t. } \|(\widehat{\boldsymbol{\Sigma}}_1 \otimes \widehat{\boldsymbol{\Sigma}}_2)\mathrm{vec}(\boldsymbol{\Delta}) - \mathrm{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2)\|_\infty \le \lambda, \tag{2.3}$$

where $\lambda > 0$ is a tuning parameter. The constraint $\|(\widehat{\mathbf{\Sigma}}_1 \otimes \widehat{\mathbf{\Sigma}}_2)\text{vec}(\mathbf{\Delta}) - \text{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2)\|_\infty \leq \lambda$ gurantees that (2.1) is roughly true, while the objective function $\|\text{vec}(\mathbf{\Delta})\|_1$ imposes sparsity. Dantzig-DN directly estimates the difference $\mathbf{\Delta}$, without imposing additional assumptions on $\mathbf{\Sigma}_k$ or $\mathbf{\Omega}_k$. It also has impressive theoretical properties and numerical performance.

However, a major challenge in Dantzig-DN is the computation. In the original paper, the authors proposed two algorithms to solve (2.3). The first is to calculate $\widehat{\mathbf{\Sigma}} = \widehat{\mathbf{\Sigma}}_1 \otimes \widehat{\mathbf{\Sigma}}_2$, and then use a generic algorithm for Dantzig selector [32, 33, 34]. However, in order to compute $\widehat{\mathbf{\Sigma}}$, the storage and the computation costs are both at the prohibitive order of p^4. To resolve this issue, the authors devoted a section to a second algorithm. They exploited the symmetry of $\widehat{\mathbf{\Sigma}}$ and reduced the storage cost by half. This algorithm alleviates the issue of storage to some extent, but the storage cost still grows at the order of p^4. We will show in Section 4.2 that by honoring the Kronecker product structure in $\widehat{\mathbf{\Sigma}}$, we can reduce the storage cost to $O(p^2)$ and the computation cost to $O(p^3)$.

On the other hand, the LASSO-DN method was proposed as a computationally efficient alternative to Dantzig-DN by adopting a different formulation, although it was developed under the context of quadratic discriminant analysis. With some algebraic derivation, it can be shown that (2.1) implies

$$\mathbf{\Delta} = \arg \min_{\mathbf{\Delta} \in \mathbb{R}^{p \times p}} \left[Tr(\mathbf{\Delta}^T \mathbf{\Sigma}_1 \mathbf{\Delta} \mathbf{\Sigma}_2) - 2Tr\{\mathbf{\Delta}(\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2)\} \right], \tag{2.4}$$

where for any matrix $\mathbf{M} \in \mathbb{R}^{q \times q}$, $Tr(\mathbf{M}) = \sum_{i=1}^{q} m_{ii}$ is the summation of all its diagonal elements. Taking derivative of (2.4) with respect to $\mathbf{\Delta}$, we recover (2.1).

Hence, to obtain a sparse estimate of $\mathbf{\Delta}$, LASSO-DN solves for

$$\widehat{\mathbf{\Delta}} = \arg \min_{\mathbf{\Delta} \in \mathbb{R}^{p \times p}} [Tr(\mathbf{\Delta}^T \widehat{\mathbf{\Sigma}}_1 \mathbf{\Delta} \widehat{\mathbf{\Sigma}}_2) - 2Tr\{\mathbf{\Delta}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2)\} + \lambda \sum_{i,j} |\Delta_{ij}|], \tag{2.5}$$

where $\lambda > 0$ is a tuning parameter and $\sum_{i,j} |\Delta_{ij}|$ is the well-known LASSO penalty [35]. As Dantzig-DN, LASSO-DN directly estimates $\mathbf{\Delta}$, and does not need sparsity assumptions on the individual precision matrices. Theories confirm that LASSO-DN is consistent under proper conditions, and numerical studies demonstrate its outstanding performance.

For computation, an ADMM algorithm was proposed for LASSO-DN with the storage cost of $O(p^2)$, and the computation cost of $O(p^3)$. In contrast to Dantzig-DN, LASSO-DN never computes the big matrix $\widehat{\mathbf{\Sigma}} = \widehat{\mathbf{\Sigma}}_1 \otimes \widehat{\mathbf{\Sigma}}_2$. All the operations are performed on the individual matrices $\widehat{\mathbf{\Sigma}}_1$ and $\widehat{\mathbf{\Sigma}}_2$. As a result, LASSO-DN is computationally more efficient than Dantzig-DN, but we will see that its computation cost can be further lowered with our algorithm in the presence of strong sparsity.

Dantzig-DN and LASSO-DN motivate us to consider a generic optimization problem we refer to as the SMORE problem. LASSO-DN is a special case of the SMORE problem, while Dantzig-DN can be performed by iteratively solving a sequence of SMORE problems. Moreover, LASSO-DN and Dantzig-DN focus on binary

problems with $K = 2$, while the SMORE problem lays the foundation to deal with multiple network problems. We introduce this problem and an efficient algorithm in the next section. Its connection to differential network analysis will be explained in detail in Sections 4 & 5.

3. The SMORE algorithm and its convergence analysis

3.1. The SMORE problem

We start with introducing the SMORE problem as a generic optimization problem, without regards to statistical models. We use the following generic notation. Let $\mathbf{U}_k \in \mathbb{R}^{q_1 \times q_1}, \mathbf{V}_k \in \mathbb{R}^{q_2 \times q_2}, k = 2, \ldots, K$ be symmetric matrices, $\mathbf{G}_k \in \mathbb{R}^{q_1 \times q_2}$ and $\lambda > 0$ be a tuning parameter. Note that the indices start from 2 such that we can keep using the same numbering system when we describe the applications of our algorithm in differential network analysis and QDA. We are interested in the following problem:

$$(\widehat{\mathbf{B}}_2, \ldots, \widehat{\mathbf{B}}_K) = \arg \min_{\mathbf{B}_2, \ldots, \mathbf{B}_K \in \mathbb{R}^{q_2 \times q_1}} \left\{ \sum_{k=2}^{K} Tr(\mathbf{B}_k^T \mathbf{V}_k \mathbf{B}_k \mathbf{U}_k) - 2Tr(\mathbf{B}_k \mathbf{G}_k) + \lambda \sum_{i,j} \sqrt{\sum_{k=2}^{K} B_{k,ij}^2} \right\}. \quad (3.1)$$

The objective function in (3.1) is a penalized quadratic function with matrices as arguments. Hence, we refer to (3.1) as the second-order matrix optimization with regularization (SMORE) problem. The penalty function $\sum_{i,j} \sqrt{\sum_{k=2}^{K} B_{k,ij}^2}$ is the group lasso penalty [36] that treats $B_{2,ij}, \ldots, B_{K,ij}$ as a group, but it reduces to the lasso penalty when $K = 2$.

The SMORE problem includes various statistical problems as special cases when we substitute $\mathbf{U}_k, \mathbf{V}_k, \mathbf{G}_k$ with appropriate estimates. For example, when $K = 2, \mathbf{V}_2 = \widehat{\mathbf{\Sigma}}_1, \mathbf{U}_2 = \widehat{\mathbf{\Sigma}}_2, \mathbf{G}_2 = \widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2$, (3.1) reduces to LASSO-DN defined in (2.5). If we make other appropriate substitutions of $\mathbf{V}_k, \mathbf{U}_k, \mathbf{G}_k$, the SMORE problem reduces to a sub-problem of Dantzig-DN or high-dimensional QDA. Indeed, these applications of the SMORE problem are discussed in Sections 4–6. Also, see Table B.8 in the appendix for the choices of $\mathbf{U}_k, \mathbf{V}_k, \mathbf{G}_k$ in all the statistical applications in this paper.

3.2. The SMORE algorithm

We solve the SMORE problem with a blockwise coordinate descent algorithm. The algorithm can be most concisely described when we vectorize $\mathbf{B}_k$. Let $\boldsymbol{\beta}_k = \text{vec}(\mathbf{B}_k) \in \mathbb{R}^{q_1 q_2}$ and $\mathbf{g}_k = \text{vec}(\mathbf{G}_k)$. Equation (3.1) can be rewritten as

$$(\widehat{\boldsymbol{\beta}}_2, \ldots, \widehat{\boldsymbol{\beta}}_K) = \arg \min_{\boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_K} \sum_{k=2}^{K} \{\boldsymbol{\beta}_k^T (\mathbf{U}_k \otimes \mathbf{V}_k) \boldsymbol{\beta}_k - 2\mathbf{g}_k^T \boldsymbol{\beta}_k\} + \lambda \sum_{j} \sqrt{\sum_{k=2}^{K} \beta_{kj}^2}. \quad (3.2)$$

Equation (3.2) is a penalized quadratic problem in $\boldsymbol{\beta}_k$, where the quadratic coefficients $\mathbf{U}_k \otimes \mathbf{V}_k$ have a Kronecker product structure. In principle we could let $\mathbf{W}_k = \mathbf{U}_k \otimes \mathbf{V}_k$ and use generic algorithms to solve (3.2), but the storage and computation of $\mathbf{W}_k$ can be very expensive. Hence, it is more desirable to develop an algorithm honoring the Kronecker product structure. To this end, we derive a lemma concerning the solution of $\boldsymbol{\beta}_{\cdot j} = (\beta_{2j}, \ldots, \beta_{Kj})$ when $\boldsymbol{\beta}_{\cdot j'}$ is given for all $j' \neq j$.

We let $\mathbf{B}_k^j \in \mathbb{R}^{q_2 \times q_1}$ such that the j'-th element of $\mathrm{vec}(\mathbf{B}_k^j)$ coincides with $\beta_{kj'}$ for $j' \neq j$, and the j-th element of $\mathrm{vec}(\mathbf{B}_k^j)$ is zero. The vector $\mathbf{U}_{k,\cdot j}$ is the j-th column of $\mathbf{U}_k$. We denote $\%$ as modulo operation, and the operator $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ if $x \leq 0$. For given q_1, q_2, j, define indices j_1, j_2 as

$$j_1 = \lfloor (j-1)/q_2 \rfloor + 1 \quad \text{and} \quad j_2 = (j-1)\%q_2 + 1. \tag{3.3}$$

For a matrix $\mathbf{U} \in \mathbb{R}^{q \times q}$, define $\mathrm{diag}(\mathbf{U}) \in \mathbb{R}^q$ as a vector containing all the diagonal elements in $\mathbf{U}$. We consider the following condition on $\mathbf{V}_k, \mathbf{U}_k$.

(C1) $\mathrm{diag}(\mathbf{U}_2 \otimes \mathbf{V}_2) = \cdots = \mathrm{diag}(\mathbf{U}_K \otimes \mathbf{V}_K)$.

Lemma 1. *For each $j = 1, \ldots, q_1 q_2$, the solution to $\boldsymbol{\beta}_{\cdot j}$ of (3.2) given $\boldsymbol{\beta}_{\cdot j'}, j' \neq j$, is the same as*

$$\arg\min_{\boldsymbol{\beta}_{\cdot j}} \sum_{k=1}^{K} (\beta_{kj} - \widetilde{\beta}_{kj})^2 + \frac{\lambda}{U_{k,j_1 j_1} V_{k,j_2 j_2}} \|\boldsymbol{\beta}_{\cdot j}\|, \tag{3.4}$$

where

$$\widetilde{\beta}_{kj} = \frac{g_{kj} - \mathbf{V}_{k,\cdot j_2}^T \mathbf{B}_k^j \mathbf{U}_{k,\cdot j_1}}{U_{k,j_1 j_1} V_{k,j_2 j_2}}. \tag{3.5}$$

Further assume that Condition (C1) is true. The solution to (3.4) *is*

$$\widehat{\boldsymbol{\beta}}_{\cdot j} = \widetilde{\boldsymbol{\beta}}_{\cdot j} \left(1 - \frac{\lambda}{\|\widetilde{\boldsymbol{\beta}}_{\cdot j}\|}\right)_+. \tag{3.6}$$

Lemma 1 indicates that, under Condition (C1), we have an explicit solution to $\widehat{\boldsymbol{\beta}}_{\cdot j}$ given all the other elements. Condition (C1) is a mild condition. When $K = 2$ and we only have one pair of $\mathbf{U}_k, \mathbf{V}_k$, Condition (C1) is trivially true. When $K > 2$, we will show in Sections 5 & 6 that Condition (C1) can be satisfied either with proper standardization or reparametrization.

Motivated by Lemma 1, we propose to solve (3.2) by iterating over $\widetilde{\boldsymbol{\beta}}_k$ and $\widehat{\boldsymbol{\beta}}_k$ defined in (3.5) and (3.6) until convergence. We refer to the resulting algorithm as the SMORE algorithm, which is summarized in Algorithm 1. At convergence, we reshape $\widehat{\boldsymbol{\beta}}_k$ to obtain $\widehat{\mathbf{B}}_k$. In the SMORE algorithm, we only need the j_2-th column of $\mathbf{V}_k$ and j_1-th column of $\mathbf{U}_k$ when we update $\widetilde{\beta}_{kj}$. The index relationship between j_1, j_2 and j comes from the Kronecker product. Since the SMORE algorithm has element-wise operations and we update each element by (3.7), there is

Algorithm 1 The SMORE algorithm

1. Input $\widehat{\mathbf{U}}_k$, $\widehat{\mathbf{V}}_k$ and $\widehat{\mathbf{g}}_k, k = 1, \ldots, K$. Initialize $\widehat{\boldsymbol{\beta}}_k^{(0)} = 0$ for $k = 1, \ldots, K$.

2. For steps $w = 1, 2, \ldots$, do the following until convergence:

   for each element $j = 1, \ldots, q_1 q_2$,

   (a) Update $\mathbf{B}_k^j$ based on current $\widehat{\boldsymbol{\beta}}_{\cdot j'}^{(w-1)}$ for all $j' \neq j$ and compute

   $$\widetilde{\beta}_{kj}^{(w-1)} = \frac{\widehat{g}_{kj} - [\mathbf{V}_{k,\cdot j_2}^T \mathbf{B}_k^j \mathbf{U}_{k,\cdot j_1} - V_{k,j_2 j_2} \beta_{k,j}^{(w-1)} U_{k,j_1 j_1}]}{\widehat{U}_{k,j_1 j_1} \widehat{V}_{k,j_2 j_2}}, \tag{3.7}$$

   (b) Compute $\widehat{\beta}_{kj}^{(w)} \longleftarrow \widetilde{\beta}_{kj}^{(w-1)} \left(1 - \frac{\lambda}{\sqrt{\sum_{k=1}^K (\widetilde{\beta}_{kj}^{(w-1)})^2}}\right)_+$ for $k = 1, \ldots, K$.

3. Output $\widehat{\boldsymbol{\beta}}_k, k = 2, \ldots, K$ at convergence.

no need to calculate $\mathbf{U}_k \otimes \mathbf{V}_k$ directly or perform matrix operations on $\mathbf{V}_k$ and $\mathbf{U}_k$. This fact drastically reduces the computational complexity in both time and memory.

Furthermore, we note that the most expensive step in the SMORE algorithm is the calculation of $\mathbf{V}_{k,\cdot j_2}^T \mathbf{B}_k^j \mathbf{U}_{k,\cdot j_1}$ in (3.7). To speed up this step, we apply the active-set strategy to exploit the sparsity of $\mathbf{B}_k^j$ [37, 38]. Let $\mathbf{D}_{k1}, \mathbf{D}_{k2}$ denote the indices of the columns and rows with nonzero elements in $\mathbf{B}_k$, respectively. These sets are updated in each iteration. Then in calculating $\mathbf{V}_{k,\cdot j_2}^T \mathbf{B}_k^j \mathbf{U}_{k,\cdot j_1}$, we restrict our attention to $\mathbf{D}_{k1}, \mathbf{D}_{k2}$ and only calculate $\mathbf{V}_{k,\mathbf{D}_2 j_2}^T \mathbf{B}_{k,\mathbf{D}_2\mathbf{D}_1}^j \mathbf{U}_{k,\mathbf{D}_1 j_1}$. This strategy further improves the computation speed.

3.3. Convergence and computation complexity

We investigate the convergence and the per-iteration complexity of the SMORE algorithm in this section. Since the SMORE algorithm is a coordinate descent algorithm, in each step we minimize the objective function over one coordinate with others fixed. Hence, the objective function monotonically decreases through iterations, and the algorithm is guaranteed to converge. We further examine under what conditions the SMORE algorithm converges to the global minimizer. For simplicity, we make the following assumption:

(C2) The matrices $\mathbf{U}_k, \mathbf{V}_k$ are positive definite.

Theorem 1. *Under Conditions (C1) & (C2), the SMORE algorithm converges to the global minimizer of* (3.2).

Theorem 1 guarantees that the output of the SMORE algorithm coincides with the global minimizer at convergence under the additional Condition (C2). Condition (C2) implies that (3.1) is strictly convex, which is a

technical condition to facilitate the proof. In practice, we can easily construct estimates to satisfy Condition (C2). We also tested our algorithm in cases where $\mathbf{U}_k$ and $\mathbf{V}_k$ were only positive semidefinite, and observed that the SMORE algorithm continued to converge to an accurate estimator. The relaxation of Condition (C2) will be an interesting research topic in the future.

As for the storage and computation costs, it is easy to see that the storage cost is $O(K\{q_1^2 + q_2^2\})$, which is the same order of saving the matrices $\mathbf{U}_k, \mathbf{V}_k$. For the per-iteration computational complexity of the SMORE algorithm, we have the following lemma.

Lemma 1. *The computational cost for each iteration is $O(Kq_1q_2d_{\max})$, where $d_{\max}$ is the maximum number of nonzero elements in $\widehat{\boldsymbol{\beta}}_k$ throughout the iterations.*

By Lemma 1, the per-iteration computational cost is an increasing function of K, q_1, q_2 and $d_{\max}$. It is intuitive that larger values of K, q_1 and q_2 lead to more time-consuming computation, as we need to estimate more parameters. But the computation cost grows linearly with respect to each of them, indicating that the SMORE algorithm scales well with q_1, q_2 and K. The number $d_{\max}$ represents the sparsity level of $\widehat{\boldsymbol{\beta}}_k$ across the iterations. When $\widehat{\boldsymbol{\beta}}_k$ remains very sparse across iterations, the computation cost is low.

4. Binary differential network analysis

4.1. The SMORE algorithm in LASSO-DN

LASSO-DN is a special case of the SMORE problem. Recall that, in LASSO-DN, we assume the differential network analysis model in (1.1) with $K = 2$. We want to identify the sparsity pattern in $\boldsymbol{\Delta} = \boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_1$, where $\boldsymbol{\Omega}_k, k = 1, 2$ are precision matrices.

LASSO-DN estimates $\boldsymbol{\Delta}$ with (2.5). In the SMORE problem, if we let $K = 2$, $\mathbf{U}_2 = \widehat{\boldsymbol{\Sigma}}_2$, $\mathbf{V}_2 = \widehat{\boldsymbol{\Sigma}}_1$ and $\mathbf{G}_2 = \widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2$, it reduces to the LASSO-DN problem defined in (2.5), with the output $\widehat{\mathbf{B}}_2$ being our estimate $\widehat{\boldsymbol{\Delta}}$. Also, since $K = 2$, Condition (C1) is trivially true. Hence, the SMORE algorithm can be used as an implementation for LASSO-DN. When we apply the SMORE algorithm to solve (2.5), we refer to the resulting method as SMORE-LDN.

The storage cost of SMORE-LDN matches that of LASSO-DN; both of them have the storage cost of $O(p^2)$. However, SMORE-LDN is more computationally efficient if $\boldsymbol{\Delta}$ is very sparse. The computation cost for LASSO-DN is $O(p^3)$, while the computation cost for SMORE-LDN is $O(p^2 d_{\max})$ by Lemma 1, where $d_{\max}$ is the maximum number of nonzero elements in the iterations. The comparison between these two methods apparently depends on the magnitude of $d_{\max}$. In general, $d_{\max}$ is comparable to the number of nonzero elements in $\boldsymbol{\Delta}$. Therefore, in what follows we discuss the comparison of computation costs under the assumption that

$d_{\max} = O(\|\mathbf{\Delta}\|_0)$. [7] considered the case where $\mathbf{\Delta}$ is very sparse with $\|\mathbf{\Delta}\|_0 = o(p)$. In this case, SMORE-LDN has a computation cost of $o(p^3)$, which is lower than LASSO-DN. In contrast, [28] showed that LASSO-DN is consistent if $\|\mathbf{\Delta}\|_0 = o\left(p\sqrt[4]{\frac{n}{\log p}}\right)$. Hence, if $\mathbf{\Delta}$ is relatively dense, i.e, $p \ll \|\mathbf{\Delta}\|_0 = o\left(p\sqrt[4]{\frac{n}{\log p}}\right)$, SMORE-LDN may have a slightly higher computation complexity than LASSO-DN by a factor smaller than $\sqrt[4]{\frac{n}{\log p}}$. However, this factor grows slowly, and in our numerical studies we observe that SMORE-LDN is in general very efficient, with faster or comparable computation time as LASSO-DN. Moreover, SMORE-LDN can be applied to multiple network problems, while LASSO-DN cannot.

As for the convergence properties, SMORE-LDN is always guaranteed to converge, since the objective function monotonically decreases in each iteration. If one further desires some assurance that SMORE-LDN produces the global minimizer, we can slightly perturb the sample covariance $\widehat{\mathbf{\Sigma}}_k$ to make them satisfy Condition (C2). In other words, we can supply $\widetilde{\mathbf{\Sigma}}_k$ instead of $\widehat{\mathbf{\Sigma}}_k$ to the SMORE algorithm, where

$$\widetilde{\mathbf{\Sigma}}_k = \frac{1}{n_k - 1} \sum_{Y_i = k} (\mathbf{X}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{X}_i - \widehat{\boldsymbol{\mu}}_k)^T + \xi \mathbf{I}, \tag{4.1}$$

with $\xi > 0$ being a small constant, such as 10^{-2}. The estimate $\widetilde{\mathbf{\Sigma}}_k$ is also known as the Ledoit-Wolf estimator [39] that has been widely applied in statistics. By controlling ξ to be small, we only introduce a tiny amount of bias, but $\widetilde{\mathbf{\Sigma}}_k$ is guaranteed to be positive definite and Condition (C2) is satisfied. By Theorem 1, SMORE-LDN with $\widetilde{\mathbf{\Sigma}}_k$ converges to the global minimizer.

4.2. The SMORE algorithm in Dantzig-DN

The SMORE algorithm can also be applied to reduce the storage and computation cost of Dantzig-DN in (2.3), because Dantzig-DN involves iteratively solving a series of SMORE problems. To see this, we first briefly review the algorithm for Dantzig-DN. Let $\boldsymbol{\delta} = \mathrm{vec}(\mathbf{\Delta})$. Rewrite the optimization problem in (2.3) as

$$\widehat{\boldsymbol{\delta}} = \arg\min f(\mathbf{r}) + \|\boldsymbol{\delta}\|_1, \quad \text{s.t.} \quad \mathbf{r} + (\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)\boldsymbol{\delta} = 2\mathrm{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2), \tag{4.2}$$

where $f(\mathbf{r}) = \infty$ if $\|\mathbf{r}\|_\infty > \lambda$ and $f(\mathbf{r}) = 0$ otherwise. Dantzig-DN alternatively updates $\mathbf{r}$, $\boldsymbol{\delta}$ and a dual variable $\mathbf{w}$ given all the other arguments in each iteration. For completeness, we include this algorithm in Appendix A as Algorithm 3. The updating rules of $\mathbf{r}$ and $\mathbf{w}$ are of closed forms. The most challenging step in Dantzig-DN is updating $\boldsymbol{\delta}$ by

$$\boldsymbol{\delta}^{(t+1)} = \arg\min \| \mathbf{w}^{(t)}/\rho - \mathbf{r}^{(t+1)} + 2\mathrm{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2) - (\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)\boldsymbol{\delta} \|_2^2 /2 + \|\boldsymbol{\delta}\|_1/\rho, \tag{4.3}$$

where $\rho > 0$ is the tuning parameter in the ADMM algorithm.

Equation (4.3) is a special case of the SMORE problem, as shown by the following lemma.

Lemma 2. *Define*

$$\boldsymbol{\Delta}^{(t+1)} = \arg\min_{\boldsymbol{\Delta}} \{Tr(\boldsymbol{\Delta}^T \widehat{\boldsymbol{\Sigma}}_1^2 \boldsymbol{\Delta} \widehat{\boldsymbol{\Sigma}}_2^2) - 2Tr(\boldsymbol{\Delta}\boldsymbol{\Psi}) + \|\boldsymbol{\Delta}\|_1/\rho\}, \tag{4.4}$$

where $\boldsymbol{\Psi} = \widehat{\boldsymbol{\Sigma}}_2\{2(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2) + \mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t+1)}\}\widehat{\boldsymbol{\Sigma}}_1$, *with* $\mathrm{vec}(\mathbf{W}^{(t)}) = \mathbf{w}^{(t)}$, $\mathrm{vec}(\mathbf{R}^{(t+1)}) = \mathbf{r}^{(t+1)}$. *Then* $\mathrm{vec}(\boldsymbol{\Delta}^{(t+1)}) = \boldsymbol{\delta}^{(t+1)}$, *where* $\boldsymbol{\delta}^{(t+1)}$ *is defined as in* (4.3).

Lemma 2 reveals that (4.3) is equivalent to (4.4). We further note that (4.4) is again a special case of the SMORE problem in (3.1). For $K = 2$, let $\mathbf{U}_2 = \widehat{\boldsymbol{\Sigma}}_2^2$, $\mathbf{V}_2 = \widehat{\boldsymbol{\Sigma}}_1^2$ and $\mathbf{G}_2 = \boldsymbol{\Psi}$. Then (3.1) reduces to (4.4). We can incorporate the SMORE algorithm in Dantzig-DN to solve (4.3). Such an algorithm is proposed as SMORE-DDN in Algorithm 2, in which (4.4) is solved by the SMORE algorithm. In addition, in Algorithm 2, we also rewrite the updating formulas for $\mathbf{r}, \mathbf{w}$ in a form that steers clear of $\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1$. See the comparisons between (4.5) and (4.6) and (A.1) and (A.4), respectively.

Algorithm 2 The SMORE-DDN algorithm

1. Input $\widehat{\boldsymbol{\Sigma}}_1$, $\widehat{\boldsymbol{\Sigma}}_2$. Initialize $\mathbf{w}^{(0)} = 0$.
2. Iteratively do the following until convergence.
   (a) Update
   $$\mathbf{r}^{(t+1)} = \mathcal{W}_\lambda[\mathbf{w}^{(t)}/\rho + 2\mathrm{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2) - \mathrm{vec}(\widehat{\boldsymbol{\Sigma}}_1 \boldsymbol{\Delta}^{(t)} \widehat{\boldsymbol{\Sigma}}_2)], \tag{4.5}$$
   where given d-dimensional vector $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_d)^T$, $\mathcal{W}_\lambda$ is defined as $\mathcal{W}_\lambda(\boldsymbol{\nu}) = [\mathrm{sign}(\nu_j) \cdot \min\{|\nu_j|, \lambda\}]_{j=1}^d$.
   (b) Solve (4.4) with the SMORE algorithm, in which we let $K = 2, \mathbf{U}_2 = \widehat{\boldsymbol{\Sigma}}_2^2, \mathbf{V}_2 = \widehat{\boldsymbol{\Sigma}}_1^2$ and $\mathbf{G}_2 = \widehat{\boldsymbol{\Sigma}}_2\{2(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2) + \mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t+1)}\}\widehat{\boldsymbol{\Sigma}}_1$.
   (c) Update
   $$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \rho(2\mathrm{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2) - \mathbf{r}^{(t+1)} - \mathrm{vec}(\widehat{\boldsymbol{\Sigma}}_1 \boldsymbol{\Delta}^{(t+1)} \widehat{\boldsymbol{\Sigma}}_2)). \tag{4.6}$$
3. Output $\boldsymbol{\Delta}^{(t+1)}$ where $\mathrm{vec}(\boldsymbol{\Delta}^{(t+1)}) = \boldsymbol{\delta}^{(t+1)}$ at convergence.

The proposed SMORE-DDN algorithm has significant advantages in both the storage and the computation costs. The Dantzig-DN method has a storage cost of $O(p^4)$, as it needs to calculate at least half of the elements in $(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)$. However, in SMORE-DDN we never calculate this Kronecker product, which brings down the storage cost to $O(p^2)$. For the computation cost, we note that the Dantzig-DN needs $\|\boldsymbol{\Delta}\|_0 = o(p)$ to be consistent. Hence, the computation cost for SMORE-DDN is $O(p^3)$, where the most expensive part is in computing $\widehat{\boldsymbol{\Sigma}}_k^2$. It is easy to see that SMORE-DDN has lower computation complexity than Dantzig-DN.

5. Multiple network analysis

We further discuss the application of the SMORE algorithm in multiple network problems. For the more general case $K \geq 2$, we assume that the data $\{\mathbf{X}_i, Y_i\}_{i=1}^n$ have been standardized within each level, so that the sample variance of X_j within each level is 1. If one does not wish to standardize the data, an alternative is given in Section 6.3.1.

We aim to identify the elements in $\mathbf{\Omega}_k$ that are non-constant across $k = 1, \ldots, K$. In other words, our goal is to identify the pairs of variables whose interactions are affected by the condition Y. To this end, we let the level $k = 1$ be the baseline, and $\mathbf{\Delta}_k = \mathbf{\Omega}_k - \mathbf{\Omega}_1$. The parameters $\mathbf{\Delta}_k$ capture the changes of the network structure across levels. In particular, if a pair (X_i, X_j) has the same interaction across levels, we must have $\omega_{1,ij} = \cdots = \omega_{K,ij}$, which implies

$$\delta_{2,ij} = \cdots = \delta_{K,ij} = 0. \tag{5.1}$$

Therefore, we seek sparse estimates for $\mathbf{\Delta}_k$. Note that, $\mathbf{\Delta}_k$ are solutions to the following estimating equation:

$$(\mathbf{\Delta}_2, \ldots, \mathbf{\Delta}_K) = \arg \min_{\mathbf{\Delta}_2, \ldots, \mathbf{\Delta}_K} \sum_{k=2}^{K} \{Tr(\mathbf{\Delta}_k^T \mathbf{\Sigma}_1 \mathbf{\Delta}_k \mathbf{\Sigma}_k) - 2Tr(\mathbf{\Delta}_k(\mathbf{\Sigma}_1 - \mathbf{\Sigma}_k))\}. \tag{5.2}$$

Sparse estimates of $\mathbf{\Delta}_k$ can be obtained by replacing $\mathbf{\Sigma}_k$ with the sample covariance $\widehat{\mathbf{\Sigma}}_k$ and adding an appropriate penalty in (5.2). By the sparsity criterion in (5.1), $\delta_{k,ij}$ has the group sparsity structure across k, as they all quantify the interactions between the same pair. We enforce the group sparsity structure with the group lasso penalty and obtain the following formula for multiple differential networks:

$$\begin{aligned} &(\widehat{\mathbf{\Delta}}_2, \ldots, \widehat{\mathbf{\Delta}}_K) \\ = \ &\arg \min_{\mathbf{\Delta}_2, \ldots, \mathbf{\Delta}_K} \left\{ \sum_{k=2}^{K} Tr(\mathbf{\Delta}_k^T \widehat{\mathbf{\Sigma}}_1 \mathbf{\Delta}_k \widehat{\mathbf{\Sigma}}_k) - 2Tr(\mathbf{\Delta}_k(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_k)) + \lambda \sum_{i,j} \sqrt{\sum_{k=2}^{K} \delta_{k,ij}^2} \right\}. \end{aligned} \tag{5.3}$$

The estimators $\{\widehat{\mathbf{\Delta}}_2, \cdots, \widehat{\mathbf{\Delta}}_K\}$ tell us about the change of dependence structure across levels. If $\hat{\delta}_{2,ij} = \ldots = \hat{\delta}_{K,ij} = 0$ for some (i, j), the dependence between X_i, X_j is constant given all the other variables. Otherwise, the dependence between X_i, X_j depends on Y.

Equation (5.3) is a special case of the SMORE problem. If we let $\mathbf{U}_k = \widehat{\mathbf{\Sigma}}_k, \mathbf{V}_k = \widehat{\mathbf{\Sigma}}_1, \mathbf{G}_k = \widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_k, k = 2, \ldots, K$, the SMORE problem in (3.1) coincides with (5.3), with the output $\mathbf{B}_k$ being $\widehat{\mathbf{\Delta}}_k$. Because we assume that data are standardized within each class, Condition (C1) is met. Hence, we again employ the SMORE algorithm to solve (5.3). Note that (5.3) reduces to SMORE-LDN in the special case of $K = 2$. Therefore, for simplicity, we refer to (5.3) with $K > 2$ as SMORE-LDN as well. For general K, SMORE-LDN has the storage cost of $O(Kp^2)$ and the per-iteration computation complexity of $O(Kp^2 d_{\max})$. Because the storage and computation costs are linear in K, SMORE-LDN is expected to be efficient as well if K is very large. Just as in binary problems,

SMORE-LDN is guaranteed to converge and is observed to be accurate in numerical studies. But if one desires theoretical guarantee for the convergence to the global minimizer, we can perturb $\widehat{\mathbf{\Sigma}}_k$ by adding $\xi\mathbf{I}$, where $\xi > 0$ is a small constant.

6. High-dimensional quadratic discriminant analysis

6.1. Background on quadratic discriminant analysis

Another application of the SMORE algorithm is in high-dimensional QDA. For ease of presentation, we first briefly review QDA. Consider the predictors $\mathbf{X} \in \mathbb{R}^p$ and the class label $Y \in \{1, \ldots, K\}$. Our goal is to build a classifier that predicts Y based on $\mathbf{X}$. Define the prior probability $\pi_k = \Pr(Y = k)$. It is known that the optimal classifier is

$$\widehat{Y} = \arg\max_k \{\log \Pr(Y = k \mid \mathbf{X} = \mathrm{x})\} = \arg\max_{k=1}^{K} \{\log \pi_k + \log f_k(\mathrm{x})\}, \tag{6.1}$$

where f_k is the conditional probability density function for $\mathbf{X}$ given $Y = k$. This classifier is often known as the Bayes' rule, which achieves the lowest classification error rate possible [29]. We hope to estimate the Bayes' rule from data. However, fully nonparametric estimation of f_k is usually difficult, especially in high dimensions. Therefore, parametric assumptions are often imposed on f_k to ease the challenges in estimation.

The QDA model assumes that f_k is the normal distribution with parameters $\boldsymbol{\mu}_k, \mathbf{\Sigma}_k$. In other words, the QDA model is

$$\Pr(Y = k) = \pi_k, \ \mathbf{X}|(Y = k) \sim N(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k), k = 1, \cdots, K, \tag{6.2}$$

where $0 < \pi_k < 1, \sum_{k=1}^{K} \pi_k = 1$ are the prior probabilities, $\boldsymbol{\mu}_k \in \mathbb{R}^p$ is the mean of $\mathbf{X}$ within Class k and $\mathbf{\Sigma}_k \in \mathbb{R}^{p\times p}$ is the covariance matrix within Class k. Without loss of generality, we assume $\boldsymbol{\mu}_1 = 0$. In practice, we can subtract the sample mean of $\mathbf{X}$ within Class 1 from all the observations to make this assumption true. Under the QDA model, the Bayes' rule is simplified to be a parametric classifier. Define $\boldsymbol{\gamma}_k = \mathbf{\Omega}_k\boldsymbol{\mu}_k \in \mathbb{R}^p$, and $\mathbf{\Delta}_k = \mathbf{\Omega}_k - \mathbf{\Omega}_1 \in \mathbb{R}^{p\times p}$, where $\mathbf{\Omega}_k = \mathbf{\Sigma}_k^{-1}$. The Bayes' rule under the QDA model (6.2) is

$$\widehat{Y} = \arg\max_k \{a_k - \mathbf{X}^T\mathbf{\Delta}_k\mathbf{X} + 2\mathbf{X}^T\boldsymbol{\gamma}_k\}, \tag{6.3}$$

where

$$a_k = 2\log\pi_k - 2\log\pi_1 + \log(|\mathbf{\Sigma}_k^{-1}|) - \log(|\mathbf{\Sigma}_1^{-1}|) - \boldsymbol{\gamma}_k^T\boldsymbol{\mu}_k, \tag{6.4}$$

is a constant.

QDA is regarded as a flexible alternative to the linear discriminant analysis (LDA) model by relaxing the equal covariance assumption. There is a large body of literature on high-dimensional LDA methods [40, 16, 41, 42, 43, 44, 45, 46]. But relatively few high-dimensional QDA methods have been proposed, because the estimation,

computation, and theoretical analysis are more challenging under the QDA model. Some existing works include [30, 31, 28].

Our work is in the same line as the DA-QDA method proposed by [28], which imposes sparsity on the Bayes' rule. Direct regularization on the Bayes' rule avoids additional assumptions on the nuisance parameters, and is thus often more efficient and accurate. Note that (6.3) is fully determined by the parameters $a_k, \boldsymbol{\Delta}_k, \boldsymbol{\gamma}_k$. In high dimensions where p is much larger than n, we assume that the two high-dimensional parameters $\boldsymbol{\Delta}_k, \boldsymbol{\gamma}_k$ are sparse and construct their estimates accordingly. The parameter $\boldsymbol{\Delta}_k$ is the difference between two precision matrices. Hence, the estimation of $\boldsymbol{\Delta}_k$ is similar to differential network analysis. Indeed, when $K = 2$, DA-QDA estimates $\boldsymbol{\Delta}_2$ with the formula in (2.5). In the next a few sections, we show that the SMORE algorithm can be used to speed up the computation of DA-QDA in binary problems. Moreover, SMORE algorithm can be applied to estimate $\boldsymbol{\gamma}_k$ as well, and offers a generalization of DA-QDA to multiclass problems. We first briefly discuss the binary problem with $K = 2$ in Section 6.2, and then proceed to multiclass problems with $K > 2$ in Section 6.3.

6.2. Binary high-dimensional quadratic discriminant analysis

In binary problems, we need to estimate $\boldsymbol{\Delta}_2$, $\boldsymbol{\gamma}_2$ and a_2. To estimate $\boldsymbol{\Delta}_2$, we incorporate the SMORE algorithm with the proposal of DA-QDA. DA-QDA estimates $\boldsymbol{\Delta}_2$ by (2.5), which is a SMORE problem with $K = 2$, $\mathbf{U}_2 = \widehat{\boldsymbol{\Sigma}}_2$, $\mathbf{V}_2 = \widehat{\boldsymbol{\Sigma}}_1$ and $\mathbf{G}_2 = \widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2$. Since $K = 2$, Condition (C1) is always true, and we apply the SMORE algorithm to solve for $\widehat{\boldsymbol{\Delta}}_2$.

Our estimates for $\boldsymbol{\gamma}_2$ and a_2 are slightly different from DA-QDA, though, for the sake of easier extension to multiclass problems. For $\boldsymbol{\gamma}_2 = \boldsymbol{\Omega}_2\boldsymbol{\mu}_2$, we note that $\boldsymbol{\gamma}_2 = \arg\min_{\boldsymbol{\gamma}_2}\{\boldsymbol{\gamma}_2^T\boldsymbol{\Sigma}_2\boldsymbol{\gamma}_2 - 2\boldsymbol{\gamma}_2^T\boldsymbol{\mu}_2\}$. It follows that a sparse estimate of $\boldsymbol{\gamma}_2$ can be obtained by

$$\widehat{\boldsymbol{\gamma}}_2 = \arg\min_{\boldsymbol{\gamma}_2}\{\boldsymbol{\gamma}_2^T\widehat{\boldsymbol{\Sigma}}_2\boldsymbol{\gamma}_2 - 2\boldsymbol{\gamma}_2^T\widehat{\boldsymbol{\mu}}_2 + \lambda\|\boldsymbol{\gamma}_2\|_1\}, \tag{6.5}$$

where $\widehat{\boldsymbol{\Sigma}}_2, \widehat{\boldsymbol{\mu}}_2$ are sample estimates. Equation (6.5) is also a SMORE problem with $\mathbf{U}_2 = \widehat{\boldsymbol{\Sigma}}_2$, $\mathbf{V}_2 = 1 \in \mathbb{R}$, and $\mathbf{G}_k = \widehat{\boldsymbol{\mu}}_2$. Condition (C1) is true because $K = 2$, and we employ the SMORE algorithm to solve for $\widehat{\boldsymbol{\gamma}}_k$.

To estimate a_2, we note that the Bayes' rule is a linear function in $\boldsymbol{\Delta}_2, \boldsymbol{\gamma}_2$. Therefore, after we obtain $\widehat{\boldsymbol{\Delta}}_2, \widehat{\boldsymbol{\gamma}}_2$, we estimate a_2 by performing logistic regression on $\{Y, \mathbf{X}^T\widehat{\boldsymbol{\Delta}}_2\mathbf{X}, \mathbf{X}^T\boldsymbol{\gamma}_2\}$ to determine a_2. This is different from DA-QDA, which searches for a_2 with cross validation. Our approach is faster than DA-QDA in the estimation of a_k, and the margin sharply increases as we move on to multiclass problems. We refer to our proposal as SMORE-QDA.

6.3. Multiclass sparse quadratic discriminant analysis

6.3.1. Estimation of $\mathbf{\Delta}_k$ for $K > 2$

From now on, we consider multiclass problems with $K > 2$. In this section, we focus on the estimation of the quadratic terms $\mathbf{\Delta}_k, k = 2, \ldots, K$. There is no need to estimate $\mathbf{\Delta}_1$, because it is always zero by definition. We start by rigorously investigating the sparsity assumption for $\mathbf{\Delta}_k, k = 2, \ldots, K$, in the context of QDA. According to the Bayes' rule in (6.3), $\delta_{k,ij}, k = 1, \ldots, K$ are the coefficients for $X_i X_j$. Therefore, an interaction term $X_i X_j$ is irrelevant to the classification if and only if $\delta_{1,ij} = \cdots = \delta_{K,ij} = 0$. Consequently, if we assume that the quadratic term in QDA is sparse, it is equivalent to assuming that most interactions $X_i X_j$ are unimportant, with $\delta_{2,ij} = \cdots = \delta_{K,ij} = 0$.

Now we discuss the construction of sparse estimates of $\mathbf{\Delta}_k$. One appealing approach may be using the same method in Section 5, since $\mathbf{\Delta}_k$ are differences among precision matrices, and are the solutions to (5.3). However, in QDA with $K > 2$, the SMORE algorithm is generally not directly applicable in solving (5.3) because Condition (C1) is usually not met. Recall that the SMORE algorithm requires Condition (C1), which reduces to $\text{diag}(\widehat{\mathbf{\Sigma}}_2) = \cdots = \text{diag}(\widehat{\mathbf{\Sigma}}_K)$ in (5.3). In differential network analysis, we can guarantee this assumption by standardizing the data within class. However, in QDA our ultimate goal is to predict Y for future observations. We cannot standardize observations within class, as it involves the knowledge of Y. In general, we expect $\widehat{\mathbf{\Sigma}}_k$ to have different diagonal elements. Because of the violation of Condition (C1), the SMORE algorithm cannot be directly applied in QDA.

To resolve this issue, we propose a reparametrization of $\mathbf{\Delta}_k$. Let $\mathbf{\Delta}_k^* = \mathbf{\Lambda}_1^{1/2} \mathbf{\Delta}_k \mathbf{\Lambda}_k^{1/2}$, where $\mathbf{\Lambda}_k$ is a diagonal matrix with $\Lambda_{k,jj}$ being the variance of X_j within Class k. Once we have estimates for $\mathbf{\Delta}_k^*$, we can easily rescale them to obtain $\widehat{\mathbf{\Delta}}_k$. Moreover, the SMORE algorithm easily obtains a sparse estimate for $\mathbf{\Delta}_k^*$ with the SMORE algorithm, as a consequence of the following lemma.

Lemma 3. *Let* $\mathbf{R}_k = \mathbf{\Lambda}_k^{-1/2} \mathbf{\Sigma}_k \mathbf{\Lambda}_k^{-1/2}$ *and* $\mathbf{Z}_k = \mathbf{\Lambda}_k^{-1/2} \mathbf{\Lambda}_1^{1/2} \mathbf{R}_1 - \mathbf{R}_k \mathbf{\Lambda}_k^{1/2} \mathbf{\Lambda}_1^{-1/2}$. *We have*

$$(\mathbf{\Delta}_2^*, \ldots, \mathbf{\Delta}_K^*) = \arg \min_{\mathbf{\Phi}_2^*, \ldots, \mathbf{\Phi}_K^*} \sum_{k=2}^{K} \{Tr((\mathbf{\Phi}_k^*)^T \mathbf{R}_1 \mathbf{\Phi}_k^* \mathbf{R}_k) - 2Tr(\mathbf{\Phi}_k^* \mathbf{Z}_k)\}. \tag{6.6}$$

The problem in (6.6) is an unpenalized SMORE problem with $\mathbf{U}_k = \mathbf{R}_k$, $\mathbf{V}_k = \mathbf{R}_1$. Moreover, because $\mathbf{R}_k$ is the correlation matrix with all the diagonal elements equal to 1, Condition (C1) holds. However, the parameters $\mathbf{R}_k, \mathbf{Z}_k$ in (6.6) need to be estimated from data. Let $\widehat{\mathbf{R}}_k^0$ be the sample correlation of $\mathbf{X}$ within Class k and $\widehat{\mathbf{R}}_k = \widehat{\mathbf{R}}_k^0 + \xi \mathbf{I}$, where $\xi > 0$ is a small constant. We use $\widehat{\mathbf{R}}_k$ as sample estimates of $\mathbf{R}_k$. Note that $\widehat{\mathbf{R}}_k$ are positive definite with constant diagonal elements, satisfying both Conditions (C1) & (C2). In addition, we define $\widehat{\mathbf{\Lambda}}_k$ to be a diagonal matrix with $\widehat{\Lambda}_{k,jj}$ being the sample variance of X_j within Class k. We estimate $\mathbf{Z}_k$ by $\widehat{\mathbf{Z}}_k = \widehat{\mathbf{\Lambda}}_k^{-1/2} \widehat{\mathbf{\Lambda}}_1^{1/2} \widehat{\mathbf{R}}_1 - \widehat{\mathbf{R}}_k \widehat{\mathbf{\Lambda}}_k^{1/2} \widehat{\mathbf{\Lambda}}_1^{-1/2}$.

To enforce sparsity in the estimation of $\mathbf{\Delta}_k^*$, we note that $\mathbf{\Delta}_k^*$ has the same sparsity pattern as $\mathbf{\Delta}_k$ (i.e, $\delta_{k,ij} = 0$ if and only if $\delta_{k,ij}^* = 0$). Therefore, sparsity in $\mathbf{\Delta}_k$ implies that for most (i, j), we have $\delta_{2,ij}^* = \cdots = \delta_{K,ij}^* = 0$. In other words, the sparsity structure of $\mathbf{\Delta}_k^*$ is grouped across k. Hence, we estimate $\mathbf{\Delta}_k^*$ by

$$\widehat{\mathbf{\Delta}}_k^* = \arg\min_{\mathbf{\Delta}_2^*,\ldots,\mathbf{\Delta}_K^*} \left[\sum_{k=2}^{K} \{Tr((\mathbf{\Delta}_k^*)^T \widehat{\mathbf{R}}_1 \mathbf{\Delta}_k^* \widehat{\mathbf{R}}_k) - 2Tr((\mathbf{\Delta}_k^*)^T \widehat{\mathbf{Z}}_k)\} + \lambda \sum_{i,j} \sqrt{\sum_{k=2}^{K} (\Delta_{k,ij}^*)^2} \right]. \tag{6.7}$$

If we set $\mathbf{U}_k = \widehat{\mathbf{R}}_k, \mathbf{V}_k = \widehat{\mathbf{R}}_1$ and $\mathbf{G}_k = \widehat{\mathbf{Z}}_k$, (6.7) is a SMORE problem with Condition (C1) satisfied. Therefore, we apply the SMORE algorithm to solve (6.7). Because Condition (C2) also holds, the SMORE algorithm converges to the global minimizer. The storage cost is $O(Kp^2)$, while the computation cost is $O(Kp^2 d_{\max})$. As a result, the computation is especially efficient when the solution remains sparse across iterations. Moreover, the computation scales well with K. With $\widehat{\mathbf{\Delta}}_k^*$, we estimate $\mathbf{\Delta}_k$ by $\widehat{\mathbf{\Delta}}_k = \widehat{\mathbf{\Lambda}}_1^{-1/2} \widehat{\mathbf{\Delta}}_k^* \widehat{\mathbf{\Lambda}}_K^{-1/2}$. The estimate $\widehat{\mathbf{\Delta}}_k$ is plugged into (6.3) for prediction, along with the estimators of γ_k and a_k discussed in the next section.

6.3.2. *Estimation of γ_k and a_k for $K > 2$*

We further discuss the estimation of γ_k and a_k for $K > 2$. The parameter γ_k is the coefficient in the linear term $\mathbf{X}^T \gamma_k$. In high dimensions, we assume that the linear term is sparse in the sense that most predictors do not affect it. Specifically, a predictor X_j is unimportant in the linear term if and only if $\gamma_{2j} = \cdots = \gamma_{Kj} = 0$. Therefore, sparsity in the linear term indicates that for most j, we have $\gamma_{kj} = 0, k = 2, \ldots, K$. For easy computation, we again reparametrize γ_k. Let $\gamma_k^* = \mathbf{\Lambda}_k^{1/2} \gamma_k = \mathbf{R}_k^{-1} (\mathbf{\Lambda}_k^{-1/2} \boldsymbol{\mu}_k)$. Obviously, $\gamma_{kj}^* = 0$ if and only if $\gamma_{kj} = 0$. Hence, the sparsity in γ_k implies that, for most j, we have

$$\gamma_{2j}^* = \cdots = \gamma_{Kj}^* = 0. \tag{6.8}$$

Moreover, it is easy to see that

$$(\gamma_2^*, \ldots, \gamma_K^*) = \arg\min_{\gamma_2^*,\ldots,\gamma_K^*} \{\sum_{k=2}^{K} (\gamma_k^*)^T \mathbf{R}_k \gamma_k^* - \frac{1}{2} (\gamma_k^*)^T \mathbf{\Lambda}_k^{-1/2} \boldsymbol{\mu}_k\}. \tag{6.9}$$

Motivated by (6.8) and (6.9), we estimate γ_k^* by

$$(\widehat{\gamma}_2^*, \ldots, \widehat{\gamma}_K^*) = \arg\min_{\gamma_2^*,\ldots,\gamma_K^*} \left\{ \sum_{k=2}^{K} (\gamma_k^*)^T \widehat{\mathbf{R}}_k \gamma_k^* - \frac{1}{2} (\gamma_k^*)^T \widehat{\mathbf{\Lambda}}_k^{-1/2} \widehat{\boldsymbol{\mu}}_k + \lambda \sum_{j=1}^{p} \sqrt{\sum_{k=2}^{K} (\gamma_{kj}^*)^2} \right\}. \tag{6.10}$$

The problem in (6.10) is again a special case of (3.1), with $\mathbf{U}_k = \mathbf{R}_k, \mathbf{V}_k = 1 \in \mathbb{R}$ and $\mathbf{G}_k = \widehat{\mathbf{\Lambda}}_k^{-1/2} \widehat{\boldsymbol{\mu}}_k$. It follows that (6.10) can be solved by the SMORE algorithm. With $\widehat{\gamma}_k^*$, we estimate γ_k by $\widehat{\gamma}_k = \widehat{\mathbf{\Lambda}}_k^{-1/2} \widehat{\gamma}_k^*$.

Finally, we turn to the estimation of a_k. The Bayes' rule (6.3) is a linear function in $\{\mathbf{X}^T \mathbf{\Delta}_k \mathbf{X}, \mathbf{X}^T \gamma_k\}$. Hence, we apply the multinomial regression on the pseudo data $\{Y, \mathbf{X}^T \widehat{\mathbf{\Delta}}_k \mathbf{X}, \mathbf{X}^T \widehat{\gamma}_k\}$ to determine a_k. This is

different from DA-QDA. Recall that, in binary problems, DA-QDA searches for the best intercept over an interval with cross validation. If we continue to use cross validation to search for a_k in multiclass problems, we will have to search for a_k in a $(K-1)$-dimensional cube, which may be computationally prohibitive. In contrast, by fitting the multinomial regression model on the pseudo data, SMORE-QDA finds a_k more efficiently.

Alternatively, if $p < n_k$ for $k = 1, \ldots, K$, where n_k is the sample size within Class k, we could also estimate a_k by (6.4). More specifically, $\mathbf{\Sigma}_1^{-1}$ and $\boldsymbol{\gamma}_1$ are estimated directly by matrix inverse $\widehat{\mathbf{\Sigma}}_1^{-1}$ and $\widehat{\mathbf{\Sigma}}_1^{-1}\widehat{\boldsymbol{\mu}}_1$, respectively. Other precision matrices are calculated by $\widehat{\mathbf{\Sigma}}_k^{-1} = \widehat{\mathbf{\Sigma}}_1^{-1} + \widehat{\mathbf{\Delta}}_k$, where $\widehat{\mathbf{\Delta}}_k$ is obtained from (5.3), and $\widehat{\gamma}_k$ is obtained as discussed above for $k = 2, \cdots, K$. All these estimates are plugged in (6.4) to form estimates for a_k. In practice, one could choose between these two ways of estimating a_k on a validation set.

For simplicity, we refer to our proposal for multiclass high-dimensional QDA as SMORE-QDA as well. In contrast to DA-QDA, SMORE-QDA solves all the optimization problem with the SMORE algorithm, and can handle multiclass problems. In binary problems, both methods have the storage cost of $O(p^2)$, while the computation cost is $O(p^2 d_{\max})$ for SMORE-QDA, and $O(p^3)$ for DA-QDA. Therefore, SMORE-QDA is more efficient when strong sparsity exists.

7. Numerical studies

7.1. Differential network analysis

In this section, we compare the SMORE methods with LASSO-DN and Dantzig-DN in differential network analysis. Our comparison concerns the accuracy and the computation cost. In binary problems, SMORE-LDN and LASSO-DN solve the same optimization problem. Hence, we aim to confirm that they have roughly the same level of accuracy. Similarly, we hope to see that SMORE-DDN performs similarly to Dantzig-DN. In the comparison of computation cost, we will show that the SMORE methods are more efficient than their counterparts.

We consider both binary and multiple differential network cases. For each model, we consider dimensions $p = 40, 60$ and sample sizes $n = 100, 300$. In all the models, we start by determining the set $\mathbf{D}_0$ that contains all the edges in $\mathbf{\Sigma}_1^{-1}$. We obtain $\mathbf{D}_0$ using the power law degree distribution similar to that in [7]. Let $\alpha = 2$ and $c = (\sum_{k=1}^{\infty} \frac{1}{k^\alpha})^{-1}$ be a normalization constant. We calculate a sequence of constants $\{h_i\}_{i=1}^p$ by $h_i = (\frac{c}{\alpha-1})^{\frac{1}{\alpha-1}} (\frac{p}{i})^{\frac{1}{\alpha-1}}$. Then for each pair of nodes (i, j), we generate a Bernoulli random variable $Z_{ij} \sim \text{Bernoulli}(\frac{h_i h_j}{\sum_{k=1}^p h_k})$. If $Z_{ij} = 1$, $(i, j) \in \mathbf{D}_0$; otherwise, $(i, j) \notin \mathbf{D}_0$. After we obtain $\mathbf{D}_0$, the elements $(\mathbf{\Sigma}_1^{-1})_{ij}$ for each model are specified as in Models (B1)–(B3) & (M1)–(M3) in Sections 7.1.1 & 7.1.2. We set the diagonals of $\mathbf{\Sigma}_1^{-1}$ to be 1 and finally symmetrize $\mathbf{\Sigma}_1^{-1}$ by averaging it with its transpose. With $\mathbf{\Sigma}_1^{-1}$, we let $\mathbf{D}$ be the index set of the pairs of top $\rho\%$ largest absolute values on the M nodes with most connections. Then the other networks $\mathbf{\Sigma}_k^{-1}$ are different from $\mathbf{\Sigma}_1^{-1}$ only on the set $\mathbf{D}$.

7.1.1. *Binary networks*

In binary problems, we let $(\mathbf{\Sigma}_2^{-1})_{ij} = a_1(\mathbf{\Sigma}_1^{-1})_{ij}$ for $(i,j) \in \mathbf{D}$ and $(\mathbf{\Sigma}_2^{-1})_{ij} = (\mathbf{\Sigma}_1^{-1})_{ij}$ for $(i,j) \in \mathbf{D}^c$. The parameters in the three models we consider are as follows.

Model (B1): If $(i,j) \in \mathbf{D}_0$, $(\mathbf{\Sigma}_1^{-1})_{ij} = 0.7^{|i-j|}/2$ when $p = 40$ and $(\mathbf{\Sigma}_1^{-1})_{ij} = 0.7^{|i-j|}/3$ when $p = 60$. Then $a_1 = -1$, $\rho\% = 10\%$, $M = 4$.

Model (B2): If $(i,j) \in \mathbf{D}_0$, $(\mathbf{\Sigma}_1^{-1})_{ij} = 0.7^{|i-j|}/2$ when $p = 40$ and $(\mathbf{\Sigma}_1^{-1})_{ij} = 0.7^{|i-j|}/3$ when $p = 60$. Then $a_1 = -1$, $\rho\% = 20\%$, $M = 2$.

Model (B3): Each nonzero entry of $\mathbf{\Sigma}_1^{-1}$ follows a uniform distribution with support $[-1/6, -1/15] \cup [1/15, 1/6]$. Then $a_1 = -0.8$, $\rho\% = 20\%$, $M = 2$.

We apply SMORE-LDN, SMORE-DDN, LASSO-DN and Dantzig-DN on the generated data. We compare the Frobenius norm of $\widehat{\mathbf{\Delta}} - \mathbf{\Delta}$, where $\mathbf{\Delta}$ is the differential network, and $\widehat{\mathbf{\Delta}}$ is its estimate. In each replicate, we record the estimation error $\|\widehat{\mathbf{\Delta}} - \mathbf{\Delta}\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm. The means and standard errors of 100 replicates are reported in Table 1. It can be seen that the accuracy of SMORE-DDN is comparable to LASSO-DN, since they solve the same optimization problem. Similarly, SMORE-DDN gives results close to Dantzig-DN. Such results confirm that the SMORE methods reproduce the results of their counterparts. We will later see that the SMORE methods have computational advantages.

7.1.2. *Multiple networks*

In multiple-network problems, we generate Models (M1)–(M3) with $\mathbf{\Sigma}_1^{-1}, \mathbf{\Sigma}_2^{-1}$ the same as those in Models (B1)–(B3), respectively. Then we further specify $\mathbf{\Sigma}_3^{-1}$ by setting $\Sigma_{3,ij} = -0.5 \times \Sigma_{1,ij}$ for $(i,j) \in \mathbf{D}$ and $\Sigma_{3,ij} = \Sigma_{1,ij}$ for $(i,j) \in \mathbf{D}^c$. We aim to estimate $\mathbf{\Delta}_k = \mathbf{\Sigma}_k^{-1} - \mathbf{\Sigma}_1^{-1}$ for $k = 2, 3$. Note that by our design $\mathbf{\Sigma}_3^{-1}$ is closer to $\mathbf{\Sigma}_1^{-1}$ than $\mathbf{\Sigma}_2^{-1}$. Hence, $\mathbf{\Delta}_3$ has weaker signals and is expected to be more difficult to estimate.

We again apply the four methods considered in binary cases. SMORE-LDN simultaneously estimates $\mathbf{\Delta}_k$, while SMORE-DDN, LASSO-DN and Dantzig-DN estimate $\mathbf{\Delta}_k$ individually. We report $\|\widehat{\mathbf{\Delta}}_k - \mathbf{\Delta}_k\|_F$ in Table 2. It can be seen that SMORE-DDN again achieves accuracy similar to Dantzig-DN, as they solve the same optimization problem. On the other hand, SMORE-LDN employs the group lasso penalty in multiple network problems, which distinguishes it from all the other methods. We can see that SMORE-LDN is similar to the competitors in estimating $\mathbf{\Delta}_2$, but uniformly outperforms all the competitors in estimating $\mathbf{\Delta}_3$. Recall that $\mathbf{\Delta}_3$ has weaker signals and is more difficult to estimate. In SMORE-LDN, we use the group lasso penalty to pool the information from $\mathbf{\Delta}_2$ to facilitate the estimation of $\mathbf{\Delta}_3$, which leads to its superior performance.

7.1.3. *Computation cost*

We further compare the computation costs of the four methods. For simplicity, we focus on the most challenging case with $n = 100, p = 60$. We record the computation time on one single tuning parameter for 20 replicates.

Model	p	n	min(SE)	SMORE-LDN	LASSO-DN	Dantzig-DN	SMORE-DDN
Model B1	40	100	0.0172	1.2840	1.2742	1.2952	1.2818
		300	0.0094	0.7982	0.7923	0.8008	0.7882
	60	100	0.0156	1.4251	1.4385	1.4450	1.4420
		300	0.0103	0.9764	0.9764	0.9914	0.9852
Model B2	40	100	0.0184	1.1294	1.1227	1.1415	1.1271
		300	0.0097	0.7375	0.7274	0.7450	0.7320
	60	100	0.0179	1.0879	1.0896	1.1082	1.1053
		300	0.0110	0.7785	0.7753	0.7923	0.7877
Model B3	40	100	0.0074	1.1866	1.1898	1.1951	1.1923
		300	0.0081	0.8478	0.8409	0.8400	0.8348
	60	100	0.0095	1.3861	1.3662	1.3926	1.3850
		300	0.0092	0.9454	0.9046	0.9232	0.9130

Table 1: Simulation results for binary differential networks. We report the means of $\|\widehat{\mathbf{\Delta}} - \mathbf{\Delta}\|_F$ based on 100 replicates. Within each model, the standard errors of all the methods are very close, so we only report the minimum standard error in each model as min(SE). Based on the standard errors, all the comparisons between LASSO-DN and SMORE-LDN are insignificant, and so are the comparisons between Dantzig-DN and SMORE-DDN. In other words, LASSO-DN and SMORE-LDN produce approximately identical estimates, and so do Dantzig-DN and SMORE-DDN.

The tuning parameter is pre-chosen to minimize the norm of difference. The results are in Table 3. SMORE-LDN is the fastest, followed by LASSO-DN. The other two methods are much slower, as they use the Dantzig selector. Nevertheless, SMORE-DDN manages to greatly reduce the computation time of Dantzig-DN. Therefore, the SMORE methods are more efficient than their counterparts in the models we considered.

7.2. High-dimensional quadratic discriminant analysis

In this section, we demonstrate the application of our proposed SMORE-QDA as a high-dimensional QDA method. In all the models, we consider two scenarios with $p = 50, 200$. We generated training sets according to (6.2), with the sample size within each class being 100. We further generated validation sets to choose the tuning parameters, and testing sets to evaluate the performance of the classifiers. Both the validation sets and the testing sets are of the same sample size as the training data. When we describe the model settings, we say $\mathbf{\Omega} = AR(\rho)$ if $\Omega_{ij} = \rho^{|i-j|}$ and $\mathbf{\Omega} = CS(\rho)$ if $\Omega_{ii} = 1$ and $\Omega_{ij} = \rho$ for $i \neq j$.

p	n	$\| \Delta_2 - \widehat{\Delta}_2 \|_F$				$\| \Delta_3 - \widehat{\Delta}_3 \|_F$			
		SMORE LDN	LASSO DN	Dantzig DN	SMORE DDN	SMORE LDN	LASSO DN	Dantzig DN	SMORE DDN
M1									
40	100	1.3418	1.2742	1.2952	1.2818	1.0590	1.2246	1.2322	1.2270
		(0.0168)	(0.0175)	(0.0173)	(0.0174)	(0.0131)	(0.0165)	(0.0161)	(0.0163)
40	300	0.8294	0.7923	0.8008	0.7882	0.6680	0.8062	0.8040	0.7980
		(0.0091)	(0.0104)	(0.0098)	(0.0099)	(0.0077)	(0.0103)	(0.0096)	(0.0096)
60	100	1.4484	1.4385	1.4450	1.4420	1.1167	1.2016	1.2073	1.2046
		(0.0169)	(0.0158)	(0.0156)	(0.0157)	(0.0134)	(0.0143)	(0.0143)	(0.0142)
60	300	0.9884	0.9764	0.9914	0.9852	0.7779	0.9177	0.9294	0.9267
		(0.0102)	(0.0108)	(0.0106)	(0.0105)	(0.0079)	(0.0095)	(0.0092)	(0.0092)
M2									
40	100	1.1970	1.1227	1.1415	1.1271	0.9320	1.0473	1.0560	1.0482
		(0.0182)	(0.0186)	(0.0188)	(0.0188)	(0.0144)	(0.0173)	(0.0169)	(0.0173)
40	300	0.7690	0.7274	0.7450	0.7320	0.6107	0.7136	0.7228	0.7180
		(0.0094)	(0.0101)	(0.0097)	(0.0097)	(0.0083)	(0.0105)	(0.0101)	(0.0102)
60	100	1.1074	1.0896	1.1082	1.1053	0.8464	0.8959	0.9046	0.9020
		(0.0182)	(0.0181)	(0.0181)	(0.0181)	(0.0143)	(0.0155)	(0.0155)	(0.0154)
60	300	0.7861	0.7753	0.7923	0.7877	0.6086	0.7087	0.7177	0.7150
		(0.0118)	(0.0110)	(0.0113)	(0.0113)	(0.0093)	(0.0108)	(0.0110)	(0.0111)
M3									
40	100	1.2107	1.1898	1.1951	1.1923	1.0189	1.0396	1.0442	1.0432
		(0.0065)	(0.0074)	(0.0076)	(0.0051)	(0.0057)	(0.0053)	(0.0078)	(0.0052)
40	300	0.8403	0.8409	0.8400	0.8348	0.7207	0.7991	0.7962	0.7954
		(0.0075)	(0.0088)	(0.0082)	(0.0084)	(0.0071)	(0.0085)	(0.0085)	(0.0085)
60	100	1.4623	1.3662	1.3926	1.3850	1.2361	1.2475	1.2545	1.2572
		(0.0080)	(0.0103)	(0.0068)	(0.0067)	(0.0065)	(0.0070)	(0.0068)	(0.0067)
60	300	0.9922	0.9046	0.9232	0.9130	0.8511	0.9144	0.9248	0.9279
		(0.0086)	(0.0092)	(0.0100)	(0.0098)	(0.0070)	(0.0083)	(0.0083)	(0.0083)

Table 2: Multiple differential networks. Mean and standard error of the minimum norm of differences based on 100 replicates are reported.

Model	SMORE-LDN	LASSO-DN	Dantzig-DN	SMORE-DDN
B1	0.6813	1.0138	202.87	25.97
B2	0.3757	1.0549	200.81	18.75
B3	1.0500	1.0724	239.37	79.03
M1	0.8161	1.9213	402.11	40.33
M2	0.4355	1.9971	399.88	30.64
M3	1.1575	2.1398	468.67	129.27

Table 3: Time (seconds) of model fitting on a given parameter of 20 replicates. The parameter λ is chosen to minimize the norm of difference.

7.2.1. Binary simulations

We consider the following four models, which were also used in [28]. For each model, we let $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\mu}_2 = \boldsymbol{\Sigma}_2\boldsymbol{\gamma}$, where $\boldsymbol{\gamma} = (0.6, 0.8, 0, \ldots, 0)^T$. The parameters $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are set as follows.

Model (B1): $\boldsymbol{\Sigma}_2^{-1}$ is a band matrix with diagonal elements being 1 and $\boldsymbol{\Sigma}_{2,ij}^{-1} = 0.3$ for $|i-j| = 1$. $\boldsymbol{\Sigma}_1^{-1} = \boldsymbol{\Sigma}_2^{-1} + \boldsymbol{\Sigma}^{-1}$, where $\boldsymbol{\Sigma}^{-1}$ is a symmetric and sparse matrix with $\boldsymbol{\Sigma}_{10,10}^{-1} = -0.3758$, $\boldsymbol{\Sigma}_{10,30}^{-1} = \boldsymbol{\Sigma}_{30,10}^{-1} = 0.0616$, $\boldsymbol{\Sigma}_{10,50}^{-1} = \boldsymbol{\Sigma}_{50,10}^{-1} = 0.2037$, $\boldsymbol{\Sigma}_{30,30}^{-1} = -0.5482$, $\boldsymbol{\Sigma}_{30,50}^{-1} = \boldsymbol{\Sigma}_{50,30}^{-1} = 0.0286$, and $\boldsymbol{\Sigma}_{50,50}^{-1} = -0.4614$.

Model (B2): $\boldsymbol{\Sigma}_2^{-1} = AR(0.5)$. $\boldsymbol{\Sigma}_1^{-1} = \boldsymbol{\Sigma}_2^{-1} + \mathbf{I}$.

Model (B3): $\boldsymbol{\Sigma}_2^{-1} = \boldsymbol{\Sigma}_1^{-1} = AR(0.5)$.

Model (B4): $\boldsymbol{\Sigma}_2^{-1} = AR(0.5)$. $\boldsymbol{\Sigma}_1^{-1} = \boldsymbol{\Sigma}_2^{-1} + \boldsymbol{\Sigma}^{-1}$ where $\boldsymbol{\Sigma}^{-1}$ is a band matrix with diagonal elements being 1 and $\boldsymbol{\Sigma}_{ij}^{-1} = 0.5$ for $|i-j| = 1$.

We compare SMORE-QDA with DA-QDA to confirm that they produce similar results, as they solve the same optimization problem with different algorithms. The prediction error and variable selection results are reported in Table 4. In SMORE-QDA, we perturb the covariance matrices by $\xi = 0.05$ for $p = 200$ cases. The error rates of the two methods are comparable overall, although SMORE-QDA often appears to be less accurate. This is because DA-QDA does a grid search for the optimal intercept a_k, while in SMORE-QDA we refit the logistic regression model on the reduced data. We further plot the solution path of $\boldsymbol{\Delta}$ in Model (B2) in Figure S1 in the supplement. The two algorithms yield similar solution paths.

Finally, we compare the computational cost of SMORE-QDA and DA-QDA. We record the computation time of estimating $\boldsymbol{\Delta}$ in Models (B1) and (B2) for the same pre-tuned parameter, which is chosen to yield the best classifier. The reported time in Figure 7.1 is averaged from 20 replicates. As we have shown, DA-QDA has computation complexity of order $O(p^3)$ and our method of $O(p^2 d_{\max})$. Model 1 is a very sparse case where $\boldsymbol{\Delta}$ has less than p nonzero elements. Therefore, our method is significantly faster than DA-QDA. In Model (B2), the number of important variables in $\boldsymbol{\Delta}$ is exactly p, and two algorithms have comparable computation time.

Model	p=50				p=200			
	SMORE-QDA		DA-QDA		SMORE-QDA		DA-QDA	
	Mean	SE	Mean	SE	Mean	SE	Mean	SE
Model B1	25.45	(0.29)	26.08	(0.30)	25.05	(0.35)	24.68	(0.31)
Model B2	2.00	(0.11)	1.85	(0.11)	0.51	(0.07)	0.13	(0.03)
Model B3	35.09	(0.42)	34.07	(0.39)	36.30	(0.44)	33.82	(0.36)
Model B4	16.34	(0.29)	15.85	(0.32)	8.03	(0.20)	7.91	(0.23)

Table 4: Binary simulations. The means and standard errors of 100 replicates are reported.

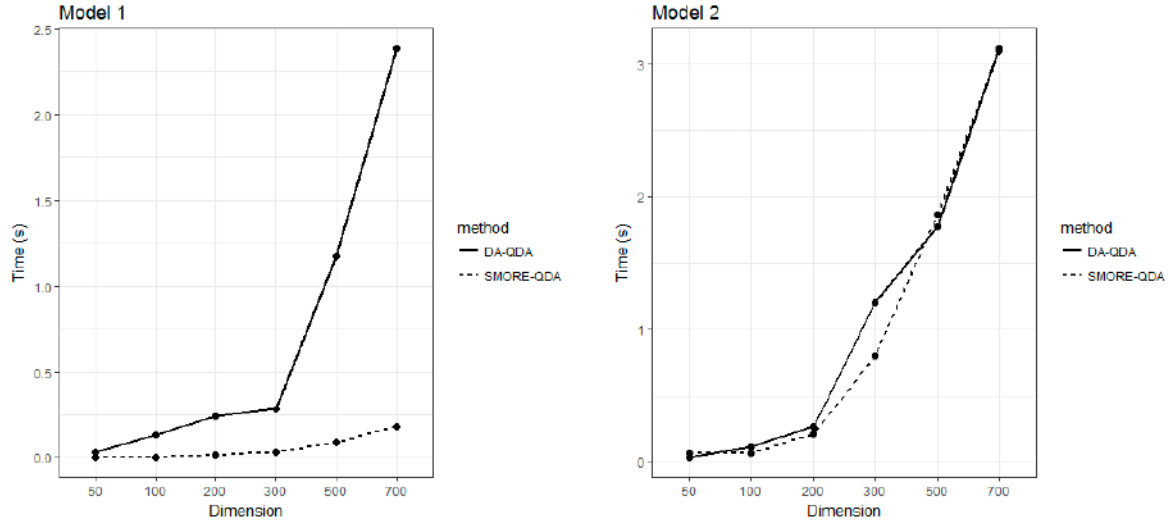

Figure 7.1: Computation time (seconds) from 20 replicates. We record the time of estimating $\mathbf{\Delta}$ in Model (B1) and Model (B2).

7.2.2. Multi-class simulations

We further include multi-class simulations when $K = 3$. We adopt all the parameters $\mathbf{\Sigma}_1^{-1}, \mathbf{\Sigma}_2^{-1}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ in binary class problems. For the third class, we let $\boldsymbol{\mu}_3 = (1.2, 1.6, 0, \ldots, 0)$ and specify the $\mathbf{\Sigma}_3$ according to the following settings.

Model M1: $\mathbf{\Sigma}_1^{-1} = \mathbf{\Sigma}_2^{-1} + \mathbf{\Sigma}^{-1}$, $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_2^{-1} - 2\mathbf{\Sigma}^{-1}$.

Model M2: $\mathbf{\Sigma}_1^{-1} = \mathbf{\Sigma}_2^{-1} + \mathbf{I}$, $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_2^{-1} + 2\mathbf{I}$.

Model M3: $\mathbf{\Sigma}_1^{-1} = \mathbf{\Sigma}_2^{-1} = \mathbf{\Sigma}_3^{-1}$.

Model M4: $\mathbf{\Sigma}_1^{-1} = \mathbf{\Sigma}_2^{-1} + \mathbf{\Sigma}^{-1}$, $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_2^{-1} + 1.6\mathbf{\Sigma}^{-1}$.

In multiclass problems, DA-QDA is no longer applicable, but we consider a wide range of other competitors. We include two versions of penalized multinomial regression (PMR) methods. The first PMR method, referred

to as PMR1, only includes the main effects, while the second PMR method, PMR2, fits a model with both the main effects and the two-way interactions. Popular machine learning methods such as random forest [47, 48], linear support vector machine and kernel support vector machine [49, 50] are also included for comparison. The classification error of the Bayes' rule is also reported as a baseline. In SMORE-QDA, we perturb the covariance matrices by $\xi = 0.01$ for $p = 200$ cases. The classification error rates are recorded in Table 5. It can be seen that SMORE-QDA uniformly outperforms all the competitors. Hence, SMORE-QDA has competitive performance as a classifier as well.

Error (%)	p	Bayes	SMORE-QDA	SVM-l	SVM-k	RF	PMR1	PMR2
Model M1	50	30.74	32.75	51.54	48.35	41.33	44.93	38.17
		(0.29)	(0.30)	(0.30)	(0.33)	(0.30)	(0.26)	(0.31)
	200	29.53	31.64	57.87	56.50	43.93	45.14	38.43
		(0.24)	(0.28)	(0.27)	(0.28)	(0.28)	(0.27)	(0.29)
Model M2	50	7.53	9.51	53.98	11.26	18.01	48.30	19.21
		(0.14)	(0.19)	(0.33)	(0.19)	(0.23)	(0.33)	(0.27)
	200	0.60	1.49	58.18	2.40	9.97	49.06	17.91
		(0.04)	(0.08)	(0.27)	(0.09)	(0.21)	(0.31)	(0.28)
Model M3	50	42.14	43.3	51.09	52.03	47.88	43.62	44.85
		(0.22)	(0.24)	(0.28)	(0.29)	(0.27)	(0.22)	(0.25)
	200	42.41	44.56	57.71	57.43	51.56	44.95	46.51
		(0.29)	(0.30)	(0.29)	(0.31)	(0.29)	(0.32)	(0.28)
Model M4	50	15.23	34.46	55.18	30.49	38.03	51.35	35.57
		(0.19)	(0.32)	(0.30)	(0.32)	(0.31)	(0.31)	(0.28)
	200	5.64	29.66	60.42	22.6	33.24	52.54	34.62
		(0.12)	(0.31)	(0.30)	(0.25)	(0.30)	(0.31)	(0.3)

Table 5: Multiclass simulations. The average classification errors in 100 replicates are reported. Standard errors are in parentheses.

8. Real data

8.1. The TCGA gloiblastoma data

We compare all the differential network analysis methods considered in simulations on the modified TCGA glioblastoma data provided in R package *DINGO* and studied in differential network approach DINGO [51]. The data was collected in a study on glioblastoma multiforme, which is a primary brain tumor for adults, with 233

subjets in total. We standardize expressions for 18 genes and 156 observations. The observations are separated into two sub-groups based on their survival times, LTSs and STSs. The 83 patients in LTS group are the top 45% in the study with survival time longer than 407 days, and the 73 patients in STS group are the bottom 45% with less than 341 days surviving.

We applied SMORE-LDN, LASSO-DN, Dantzig-DN and SMORE-DDN on the gene expressions and plotted the solution paths in Figure 8.1. The range of tuning parameters was chosen such that all the four paths would give $\widehat{\mathbf{\Delta}}$ with d nonzero elements, where d ranged from $O(p^2/\log(p))$ to 0. It can be seen that the solution path of SMORE-LDN is similar to LASSO-DN, and the solution path of SMORE-DDN is similar to Dantzig-DN. The computation time of the solution paths are also reported in Table 6. SMORE methods are faster than their counterparts.

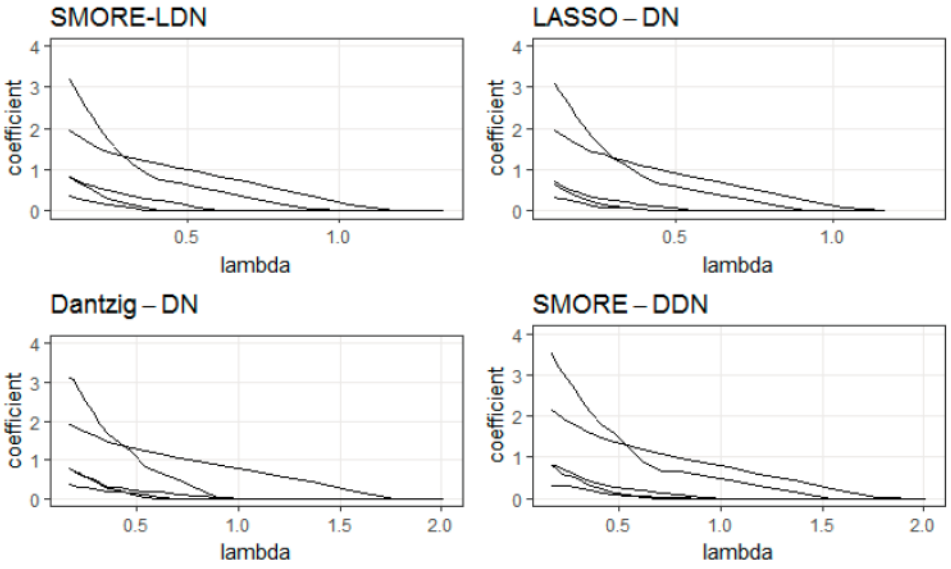

Figure 8.1: Solution path on the TCGA glioblastoma data. Displayed are the coefficients of the following pairs: (11,11), (14,14), (17,5), (8,11), (15,14).

Method	SMORE-LDN	LASSO-DN	Dantzig-DN	SMORE-DDN
Time (s)	0.1035	0.1082	3.3830	0.6539

Table 6: Computation time of a solution path.

8.2. The Vehicle Silhouettes data

We further consider a multi-class classification problem on the Vehicle Silhouettes data set. The data set is available on UCI Machine Learning Repository (`https://archive.ics.uci.edu/ml/datasets/`

Statlog+(Vehicle+Silhouettes)). It records 18 features extracted from the silhouettes of four types of vehicles. All the four classes have a relatively balanced sample size among 846 subjects in total. We want to build a classifier to predict the types of vehicles based on the silhouettes. The data set is randomly partitioned into training, validation and testing sets with the ratio 1:2:2, i.e, 170 training samples, 348 validation samples and 348 testing samples. All the tuning parameters are chosen by minimizing the validation error.

Similar to Section 7.2.2, we compare SMORE-QDA on this dataset with five successful classifiers, including random forest, linear and kernel support vector machines, and two versions of penalized multinomial regression methods, PMR1 and PMR2. PMR1 only includes the main effects, but PMR2 further includes all the two-way interactions. Note that DA-QDA is not applicable on this dataset because we have a multiclass problem instead of a two-class problem. Before fitting the classifiers, each variable is standardized. In SMORE-QDA, we slightly perturb the sample covariance matrices by 0.01 to speed up the computation. Means and standard errors of classification error rates on testing data over 100 replicates are reported in Table 7. It can be seen that SMORE-QDA is significantly more accurate than all the competitors, which supports its application in classification.

Error rate (%)	SMORE-QDA	PMR1	PMR2	SVM-l	SVM-k	RF
Mean	22.34	23.80	24.32	25.16	44.30	28.60
SE	0.25	0.24	0.24	0.26	2.03	0.22

Table 7: Testing errors on the Vehicle Silhouettes data set. Means and standard errors from 100 replicates are reported.

9. Discussion

In this paper, we develop the SMORE algorithm that can be applied to differential network analysis and QDA. See Table B.8 in the appendix for a summary of these applications. In all the applications, the SMORE algorithm has low storage and computation costs in comparison to LASSO-DN and Dantzig-DN, which we list in Table B.9 in the appendix. The SMORE methods are particularly favorable in the presence of strong sparsity. Under mild conditions, the SMORE algorithm is theoretically shown to have nice convergence results. Further, this algorithm provides a natural approach to generalize existing methods for binary problems to multiclass problems. The superior performance of the SMORE algorithm is confirmed by numerical studies.

Our SMORE algorithm is a coordinate descent algorithm. Coordinate descent algorithms are known to be efficient and stable in high-dimensional problems [52, 38, 53]. However, in comparison with existing coordinate descent algorithms, the SMORE algorithm is carefully tailored for the SMORE problem to take advantage of its special form and drastically reduce the storage and computation costs. Some of the techniques we use to develop the SMORE algorithm are similar to those in [54], but the SMORE algorithm targets a more complicated problem

and has much broader applications. We acknowledge though that the objective function of the SMORE problem is convex, and other algorithms may be applicable as well, such as the alternating direction method of multipliers (ADMM) [33, 55], Bregman-based approaches [56, 57], stagewise method [58], minorize-maximization (MM) [59, 60, 61], ODE-based methods [62, 63] and proximal gradient descent algorithms [64]. A thorough study of these algorithms in solving the SMORE problem is left for future study.

Finally, we note that the SMORE algorithm enforces sparsity in estimation. As pointed out by a referee, sparsity may not be reasonable in some real-life problems. If the sparsity assumption is deemed inappropriate for a particular dataset, one possible alternative is the low-rank assumption; see [65, 66, 67, 68, 69, 70, 71] for example. In these works, it is assumed that the variation across all the covariance or precision matrices is fully captured by a common low-rank subspace. These methods are strongly supported by theoretical and numerical results. However, they often involve non-convex optimization, and thus the SMORE algorithm cannot be directly applied to implement them. It will be interesting to see in the future if the low-rank methods can be reformulated as convex optimization problems and if algorithms similar to the SMORE algorithm can be developed for them.

Acknowledgement

The authors are grateful to the editor, the associate editor and three reviewers, whose comments greatly improved the quality of this paper. Mai's research is partially supported by CCF-1617691 and CCF-1908969, National Science Foundation.

Appendix A. The Dantzig algorithm

We describe the Dantzig algorithm proposed by [7] for completeness. The Dantzig algorithm solves (4.2) as follows.

Algorithm 3 The Dantzig algorithm

1. Input $\widehat{\mathbf{\Sigma}}_1$, $\widehat{\mathbf{\Sigma}}_2$. Initialize $\mathbf{w}^{(0)} = 0$.

2. For steps $t = 1, 2, \ldots$, iteratively do the following until convergence:

   (a) Update

$$\mathbf{r}^{(t+1)} = \arg\min_{\mathbf{r}} \| \mathbf{w}^{(t)}/\rho + 2\text{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2) - (\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)\boldsymbol{\delta}^{(t)} - \mathbf{r} \|_2^2 /2 + f(r)/\rho. \tag{A.1}$$

   (b) Update

$$\boldsymbol{\delta}^{(t+1)} = \arg\min_{\boldsymbol{\delta}} \| \mathbf{w}^{(t)}/\rho - \mathbf{r}^{(t+1)} + 2\text{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2) - (\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)\boldsymbol{\delta} \|_2^2 /2 + \|\boldsymbol{\delta}\|_1/\rho. \tag{A.2}$$

   In the implementation, this is approximated with closed form

$$\boldsymbol{\delta}^{(t+1)} = \mathcal{S}_{\frac{1}{\rho}}(\boldsymbol{\delta}^{(t)} - (\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)^T[(\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)\widehat{\boldsymbol{\delta}}^{(t)} - \mathbf{w}^{(t)}/\rho + \mathbf{r}^{(t+1)} - 2\text{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2)]). \tag{A.3}$$

   (c) Update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \rho(2\text{vec}(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2) - \mathbf{r}^{(t+1)} - (\widehat{\mathbf{\Sigma}}_2 \otimes \widehat{\mathbf{\Sigma}}_1)\boldsymbol{\delta}^{(t+1)}). \tag{A.4}$$

3. Output $\boldsymbol{\delta}^{(t+1)}$ at convergence.

Appendix B. Additional Tables and Figures

Table B.8 summarizes the choices of $\mathbf{U}_k, \mathbf{V}_k, \mathbf{G}_k$ in all the applications.

Method		$\mathbf{U}_k$	$\mathbf{V}_k$	$\mathbf{G}_k$
SMORE-LDN		$\widehat{\mathbf{\Sigma}}_k$	$\widehat{\mathbf{\Sigma}}_1$	$\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_k$
SMORE-DDN		$\widehat{\mathbf{\Sigma}}_2^2$	$\widehat{\mathbf{\Sigma}}_1^2$	$\widehat{\mathbf{\Sigma}}_2\{2(\widehat{\mathbf{\Sigma}}_1 - \widehat{\mathbf{\Sigma}}_2) + \mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t+1)}\}\widehat{\mathbf{\Sigma}}_1$
SMORE-QDA	Quadratic term	$\widehat{\mathbf{R}}_k$	$\widehat{\mathbf{R}}_1$	$\widehat{\mathbf{\Lambda}}_k^{-\frac{1}{2}}\widehat{\mathbf{\Lambda}}_1^{\frac{1}{2}}\widehat{\mathbf{R}}_1 - \widehat{\mathbf{R}}_k\widehat{\mathbf{\Lambda}}_k^{\frac{1}{2}}\widehat{\mathbf{\Lambda}}_1^{-\frac{1}{2}}$
	Linear term	$\widehat{\mathbf{R}}_k$	1	$\widehat{\mathbf{\Lambda}}_k^{-\frac{1}{2}}\widehat{\boldsymbol{\mu}}_k$

Table B.8: Choices of $\mathbf{U}_k, \mathbf{V}_k, \mathbf{G}_k$. The matrix $\widehat{\mathbf{\Sigma}}_k$ is the sample covariance matrix, $\widehat{\mathbf{R}}_k$ is the perturbed sample correlation matrix within class k, and $\mathbf{\Lambda}_k = \text{diag}(\mathbf{\Sigma}_k)$ is the diagonal element of covariance matrix.

A comparison on storage and computation costs for differential network analysis is given in Table B.9. Our SMORE algorithm has low storage and computation costs in all applications.

	LASSO-DN	SMORE-LDN	Dantzig-DN	SMORE-DDN
Storage	$O(p^2)$	$O(p^2)$	$O(p^4)$	$O(p^2)$
Computation	$O(p^3)$	$O(p^2 d_{\max})$	$O(p^4)$	$O(p^3)$

Table B.9: Comparison of storage and computation costs for the SMORE methods, LASSO-DN and Dantzig-DN. The quantity $d_{\max}$ is the maximum number of nonzero elements in the iterations.

To further compare the solution path of SMORE-QDA and DA-QDA, we consider the QDA Model (B2) in Section 7.2 as an example. Figure B.1 shows that the solution paths of SMORE-QDA and DA-QDA are similar to each other.

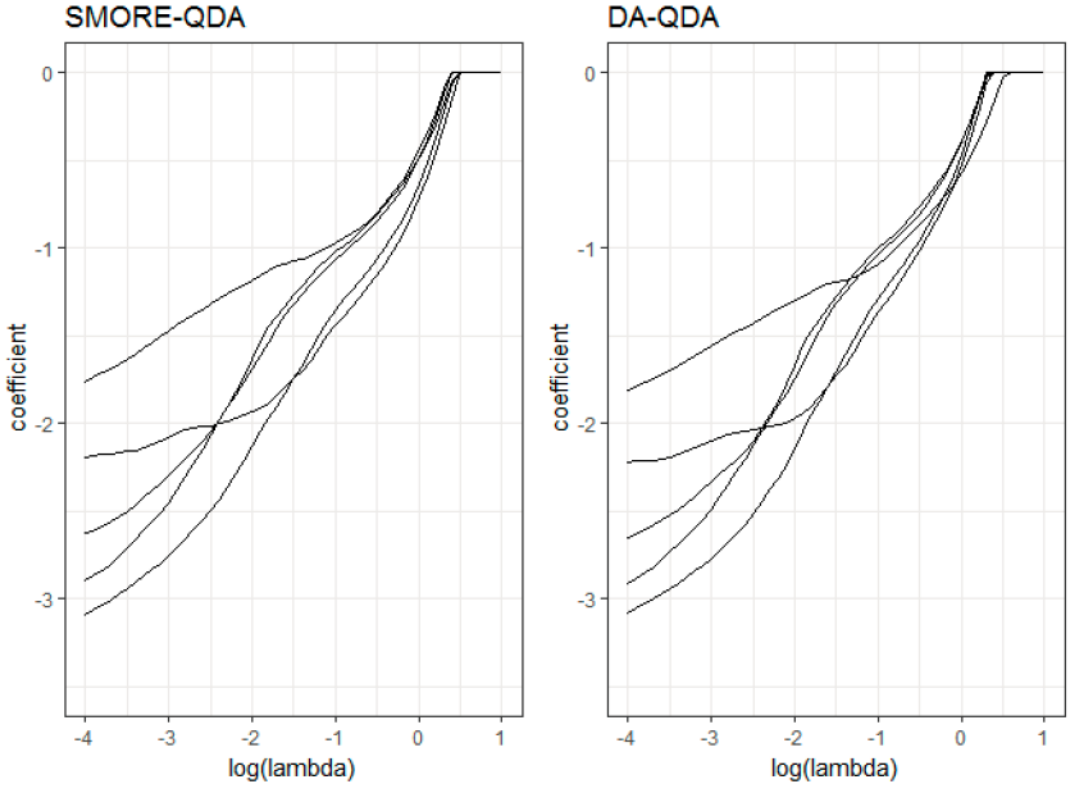

Figure B.1: Solution path of $\boldsymbol{\Delta}$ in Model (B2) defined in Section 7.2. Displayed are the coefficients of the following pairs: (6,6), (13,13), (33,33), (36,36), (43,43).

Appendix C. Proofs for Lemmas

Proof of Lemma 1. First, define $\mathbf{W}_k = \mathbf{U}_k \otimes \mathbf{V}_k$. For fix $j = 1, \cdots, p$, note that

$$\boldsymbol{\beta}_k^T \mathbf{W}_k \boldsymbol{\beta}_k = \sum_{l,m} W_{k,lm} \beta_{kl} \beta_{km} \tag{C.1}$$

$$= \beta_{kj}^2 W_{k,jj} + 2 \sum_{l \neq j} W_{k,lj} \beta_{kl} \beta_{kj} + \sum_{l \neq j, m \neq j} W_{k,lm} \beta_{kl} \beta_{km}, \tag{C.2}$$

$$\mathbf{g}_k^T \boldsymbol{\beta}_k = g_{kj} \beta_{kj} + \sum_{l \neq j} g_{kl} \beta_{kl}. \tag{C.3}$$

Therefore,

$$\boldsymbol{\beta}_k^T \mathbf{W}_k \boldsymbol{\beta}_k - 2 \mathbf{g}_k^T \boldsymbol{\beta}_k = \beta_{kj}^2 W_{k,jj} + 2(\sum_{l \neq j} W_{k,lj} \beta_{kl} - g_{kj}) \beta_{kj} + s_k, \tag{C.4}$$

where s_k is the sum of terms that do not involve β_{kj}. Hence given $\boldsymbol{\beta}_{\cdot j'}, j' \neq j$, the solution of $\boldsymbol{\beta}_{\cdot j}$ to problem (3.2) is equivalent to solving

$$\min_{\boldsymbol{\beta}_{\cdot j}} \sum_{k=1}^{K} (\beta_{kj} - \widetilde{\beta}_{kj})^2 + \frac{\lambda}{\widehat{W}_{k,jj}} \|\boldsymbol{\beta}_{\cdot j}\|, \tag{C.5}$$

where

$$\widetilde{\beta}_{kj} = \frac{\widehat{g}_{kj} - \sum_{l \neq j} \widehat{W}_{k,lj} \beta_{kj}}{\widehat{W}_{k,jj}}. \tag{C.6}$$

By the definition of Kronecker product, we have

$$\widehat{W}_{k,jj} = \widehat{U}_{k,j_1 j_1} \widehat{V}_{k,j_2 j_2}, \tag{C.7}$$

$$\sum_{l \neq j} \widehat{W}_{k,lj} \beta_{kj} = \widehat{\mathbf{V}}_{k,\cdot j_2}^T \mathbf{B}_k^j \widehat{\mathbf{U}}_{k,\cdot j_1}, \tag{C.8}$$

with j_1, j_2 defined as in (3.3). Hence, (3.4) holds. By checking the KKT condition we have (3.6).

□

For the proof of Lemma 2, we recall two useful facts. First, for two matrices $\mathbf{A} \in \mathbb{R}^{p_1 \times p_2}, \mathbf{B} \in \mathbb{R}^{p_2 \times p_1}$, we have $\text{vec}^T(\mathbf{A})\text{vec}(\mathbf{B}) = Tr(\mathbf{A}^T \mathbf{B})$. Second, for matrices $\mathbf{A} \in \mathbb{R}^{p_1 \times p_2}, \mathbf{B} \in \mathbb{R}^{p_2 \times p_3}, \mathbf{C} \in \mathbb{R}^{p_3 \times p_4}$, we have that $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$.

Proof of Lemma 2. Given $\mathbf{w}^{(t)}$ and $\mathbf{r}^{(t+1)}$ fixed, equation (4.4) is equivalent to

$$\begin{aligned} \boldsymbol{\delta}^{(t+1)} = \arg\min \Big\{ & \boldsymbol{\delta}^T (\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1) \boldsymbol{\delta} \\ & - 2\boldsymbol{\delta}^T (\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)^T [\mathbf{w}^{(t)}/\rho - \mathbf{r}^{(t+1)} + 2\text{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2)] + \frac{\|\boldsymbol{\delta}\|_1}{\rho} \Big\} \\ \equiv \arg\min & \left\{ L_1 - 2L_2 + \frac{\|\boldsymbol{\delta}\|_1}{\rho} \right\}. \end{aligned} \tag{C.9}$$

where

$$\begin{aligned} L_1 &= \boldsymbol{\delta}^T(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)\boldsymbol{\delta}, \\ L_2 &= \boldsymbol{\delta}^T(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)^T[\mathbf{w}^{(t)}/\rho - \mathbf{r}^{(t+1)} + 2\text{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2)]. \end{aligned}$$

For L_1, we have

$$\begin{aligned} L_1 &= \boldsymbol{\delta}^T(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)\boldsymbol{\delta} = \text{vec}^T(\boldsymbol{\Delta})(\widehat{\boldsymbol{\Sigma}}_2^2 \otimes \widehat{\boldsymbol{\Sigma}}_1^2)\text{vec}(\boldsymbol{\Delta}) = \{(\widehat{\boldsymbol{\Sigma}}_2^2 \otimes \widehat{\boldsymbol{\Sigma}}_1^2)\text{vec}(\boldsymbol{\Delta})\}^T\text{vec}(\boldsymbol{\Delta}) \\ &= \text{vec}^T(\widehat{\boldsymbol{\Sigma}}_1^2\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2^2)\text{vec}(\boldsymbol{\Delta}) = Tr(\widehat{\boldsymbol{\Sigma}}_2^2\boldsymbol{\Delta}^T\widehat{\boldsymbol{\Sigma}}_1^2\boldsymbol{\Delta}) = Tr(\boldsymbol{\Delta}^T\widehat{\boldsymbol{\Sigma}}_1^2\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2^2). \end{aligned}$$

For L_2, note that

$$\begin{aligned} \boldsymbol{\delta}^T(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)^T\text{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2) &= \text{vec}^T(\widehat{\boldsymbol{\Sigma}}_1\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2)\text{vec}(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2) = Tr(\widehat{\boldsymbol{\Sigma}}_2\boldsymbol{\Delta}^T\widehat{\boldsymbol{\Sigma}}_1(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2)) \\ &= Tr((\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2)\widehat{\boldsymbol{\Sigma}}_1\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2) = Tr(\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2(\widehat{\boldsymbol{\Sigma}}_1 - \widehat{\boldsymbol{\Sigma}}_2)\widehat{\boldsymbol{\Sigma}}_1), \\ \boldsymbol{\delta}^T(\widehat{\boldsymbol{\Sigma}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1)^T(\mathbf{w}^{(t)}/\rho - \mathbf{r}^{(t+1)}) &= \text{vec}^T(\widehat{\boldsymbol{\Sigma}}_1\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2)\text{vec}(\mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t-1)}) = Tr[\widehat{\boldsymbol{\Sigma}}_2\boldsymbol{\Delta}^T\widehat{\boldsymbol{\Sigma}}_1\{\mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t-1)}\}] \\ &= Tr[\{\mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t+1)}\}\widehat{\boldsymbol{\Sigma}}_1\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2] = Tr[\boldsymbol{\Delta}\widehat{\boldsymbol{\Sigma}}_2\{\mathbf{W}^{(t)}/\rho - \mathbf{R}^{(t+1)}\}\widehat{\boldsymbol{\Sigma}}_1]. \end{aligned}$$

Hence, $L_2 = \text{vec}^T(\boldsymbol{\Delta})\text{vec}(\boldsymbol{\Psi}) = Tr(\boldsymbol{\Delta}\boldsymbol{\Psi})$, and the conclusion follows.

□

Proof of Lemma 3. Since $\boldsymbol{\Delta}_k = \boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_1^{-1}$, $\boldsymbol{\Delta}_k$ can be solved by

$$\boldsymbol{\Delta}_k = \arg\min_{\boldsymbol{\Phi}_2,\ldots,\boldsymbol{\Phi}_K} \sum_{k=2}^{K}\Big[Tr(\boldsymbol{\Phi}_k^T\boldsymbol{\Sigma}_1\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_k) - 2Tr(\boldsymbol{\Phi}_k(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_k))\Big]. \tag{C.10}$$

For any $\boldsymbol{\Phi}_k$, define $\boldsymbol{\Phi}_k^* = \boldsymbol{\Lambda}_1^{1/2}\boldsymbol{\Phi}_k\boldsymbol{\Lambda}_k^{1/2}$. Then we have

$$\begin{aligned} &\sum_{k=2}^{K}\Big[Tr(\boldsymbol{\Phi}_k^T\boldsymbol{\Sigma}_1\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_k) - 2Tr(\boldsymbol{\Phi}_k(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_k))\Big] && \text{(C.11)} \\ =\ &\sum_{k=2}^{K}[Tr\{(\boldsymbol{\Lambda}_1^{-\frac{1}{2}}\boldsymbol{\Phi}_k^*\boldsymbol{\Lambda}_k^{-\frac{1}{2}})^T\boldsymbol{\Lambda}_1^{\frac{1}{2}}\mathbf{R}_1\boldsymbol{\Lambda}_1^{\frac{1}{2}}(\boldsymbol{\Lambda}_1^{-\frac{1}{2}}\boldsymbol{\Phi}_k^*\boldsymbol{\Lambda}_k^{-\frac{1}{2}})\boldsymbol{\Lambda}_k^{\frac{1}{2}}\mathbf{R}_k\boldsymbol{\Lambda}_k^{\frac{1}{2}}\} && \text{(C.12)} \\ &-2Tr\{\boldsymbol{\Lambda}_1^{-\frac{1}{2}}\boldsymbol{\Phi}_k^*\boldsymbol{\Lambda}_k^{-\frac{1}{2}}(\boldsymbol{\Lambda}_1^{\frac{1}{2}}\mathbf{R}_1\boldsymbol{\Lambda}_1^{\frac{1}{2}} - \boldsymbol{\Lambda}_k^{\frac{1}{2}}\mathbf{R}_k\boldsymbol{\Lambda}_k^{\frac{1}{2}})\}] && \text{(C.13)} \\ =\ &\sum_{k=2}^{K}\{Tr((\boldsymbol{\Phi}_k^*)^T\mathbf{R}_k\boldsymbol{\Phi}_k^*\mathbf{R}_1) - 2Tr(\boldsymbol{\Phi}_k^*\mathbf{Z}_k)\}. && \text{(C.14)} \end{aligned}$$

It follows that, for any $\boldsymbol{\Phi}^*$ and its corresponding $\boldsymbol{\Phi}$,

$$\begin{aligned} &\sum_{k=2}^{K}\{Tr((\boldsymbol{\Delta}_k^*)^T\mathbf{R}_k\boldsymbol{\Delta}_k^*\mathbf{R}_1) - 2Tr(\boldsymbol{\Delta}_k^*\mathbf{Z}_k)\} = \sum_{k=2}^{K}\Big[Tr(\boldsymbol{\Delta}_k^T\boldsymbol{\Sigma}_1\boldsymbol{\Delta}_k\boldsymbol{\Sigma}_k) - 2Tr(\boldsymbol{\Delta}_k(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_k))\Big] \\ \leq\ &\sum_{k=2}^{K}\Big[Tr(\boldsymbol{\Phi}_k^T\boldsymbol{\Sigma}_1\boldsymbol{\Phi}_k\boldsymbol{\Sigma}_k) - 2Tr(\boldsymbol{\Phi}_k(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_k))\Big] = \sum_{k=2}^{K}Tr((\boldsymbol{\Phi}_k^*)^T\mathbf{R}_k\boldsymbol{\Phi}_k^*\mathbf{R}_1) - 2Tr(\boldsymbol{\Phi}_k^*\mathbf{Z}_k). \end{aligned}$$

And the conclusion follows.

□

Appendix D. Proofs for Theorem 1 (Convergence analysis for Algorithm 1)

In this section, we prove the results in Theorem 1. First, we present the following definitions and propositions in [72]. For an objective function

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_N) = f_0(\mathbf{x}_1, \ldots, \mathbf{x}_N) + \sum_{k=1}^{N} f_k(\mathbf{x}_k), \tag{D.1}$$

where $f_0 : \mathbb{R}^{n_1+\cdots+n_N} \mapsto \mathbb{R} \cup \{\infty\}$ and some $f_k : \mathbb{R}^{n_k} \mapsto \mathbb{R} \cup \{\infty\}, k = 1, \ldots, N$. We have the following definitions:

Definition 1 (Gâteaux-differentiable [73]). *For a function $F : \mathbb{R}^p \mapsto \mathbb{R}$, define Gâteaux derivative as*

$$F'(\mathbf{x}; \mathbf{y}) = \lim_{\lambda \searrow 0} \frac{F(\mathbf{x} + \lambda \mathbf{y}) - F(\mathbf{x})}{\lambda}. \tag{D.2}$$

If $F'(\mathbf{x}; \mathbf{y})$ exists for all $\mathbf{y}$ at $\mathbf{x}$, then F is Gâteaux-differentiable at $\mathbf{x}$.

Definition 2 (Stationary points [72]). *A point $\mathbf{z}$ is stationary for function h if $g(\mathbf{z}; \boldsymbol{\nu}) \geq 0$ for any $\boldsymbol{\nu}$, where*

$$g(\mathbf{z}; \boldsymbol{\nu}) = \liminf_{\lambda \searrow 0} \frac{h(\mathbf{x} + \lambda \boldsymbol{\nu}) - h(\mathbf{x})}{\lambda}. \tag{D.3}$$

Definition 3 (Regular [72]). *A function f is regular at $\mathbf{z}$ if*

$$f'(\mathbf{z}; \boldsymbol{\nu}) \geq 0, \text{ for any } \boldsymbol{\nu} = (\nu_1, \ldots, \nu_N) \text{ such that } f'(\mathbf{z}; (0, \ldots, \nu_k, \cdots, 0)) \geq 0, k = 1, \ldots, N. \tag{D.4}$$

We will also use the following propositions.

Proposition 1 (A simplified version of Lemma 3.1 in [72]). *If f_0 is Gâteaux-differentiable, f is regular at each $\mathbf{z}$.*

Proposition 2 (A simplified version of Theorem 4.1 in [72]). *Assume that the level set $\mathbf{X}^0 = \{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}^0)\}$ is compact, where $\mathbf{x}^0$ is the initial value of the algorithm, and that f is continuous on $\mathbf{X}^0$. Further assume that $f(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is pseudoconvex in $(\mathbf{x}_k, \mathbf{x}_i)$ for all $i, k \in \{1, \ldots, N\}$, and if f is regular at every $\mathbf{x}$, then the solution generated by the cyclic coordinate descent method converges to a stationary point of f.*

We make use of these results to prove Theorem 1.

Lemma 2. *If $\mathbf{W}_k$ is positive definite, the set $\mathcal{B}^0 = \{\boldsymbol{\beta} : L(\boldsymbol{\beta}) \leq L(\boldsymbol{\beta}^0)\}$ is compact for any $\boldsymbol{\beta}^0$, where*

$$L(\boldsymbol{\beta}) = \sum_{k=1}^{K} \boldsymbol{\beta}_k^{\mathrm{T}} \mathbf{W}_k \boldsymbol{\beta}_k - 2\boldsymbol{\beta}_k^{\mathrm{T}} \mathbf{g}_k + \lambda \sum_{j=1}^{p} \|\boldsymbol{\beta}_{\cdot j}\|. \tag{D.5}$$

Proof. Firstly, $\mathcal{B}^0$ must be closed since $L(\boldsymbol{\beta})$ is continuous. Then we show that $\mathcal{B}^0$ is bounded. When $\widehat{\boldsymbol{\Sigma}}_k$ is invertible, for any $\boldsymbol{\beta}_k \in \mathbb{R}^p$, we have

$$\boldsymbol{\beta}_k^{\intercal}\mathbf{W}_k\boldsymbol{\beta}_k - 2\mathbf{g}_k^{\intercal}\boldsymbol{\beta}_k \geq -\mathbf{g}_k^{\intercal}\mathbf{W}_k^{-1}\mathbf{g}_k. \tag{D.6}$$

Take summation over k and it follows that, for any $\boldsymbol{\beta} \in \mathcal{B}^0$,

$$-\sum_{k=1}^{K}\mathbf{g}_k^{\intercal}\mathbf{W}_k^{-1}\mathbf{g}_k + \lambda\sum_{j=1}^{p}\|\boldsymbol{\beta}_{\cdot j}\| \leq L(\boldsymbol{\beta}) \leq L(\boldsymbol{\beta}^0). \tag{D.7}$$

which suggests that $\mathcal{B}^0$ is bounded by

$$\sum_{j=1}^{p}\|\boldsymbol{\beta}_{\cdot j}\| \leq \frac{1}{\lambda}\sum_{k=1}^{K}\mathbf{g}_k^{\intercal}\widehat{\mathbf{W}}_k^{-1}\mathbf{g}_k + L(\boldsymbol{\beta}^0) \text{ for any } \boldsymbol{\beta} \in \mathcal{B}^0. \tag{D.8}$$

Therefore, $\mathcal{B}^0$ is compact. □

Proof of Theorem 1. Let $f_j(\boldsymbol{\beta}_{\cdot j}) = \lambda\|\boldsymbol{\beta}_{\cdot j}\|, j = 1, \ldots, p$ and

$$f_0(\boldsymbol{\beta}) = \sum_{k=1}^{K}\{\boldsymbol{\beta}_k^{\intercal}\mathbf{W}_K\boldsymbol{\beta}_k - 2\mathbf{g}_k^{\intercal}\boldsymbol{\beta}_k\}, \tag{D.9}$$

where $\mathbf{W}_k = \mathbf{U}_k \otimes \mathbf{V}_k$ and $\text{vec}(\mathbf{G}_k) = \mathbf{g}_k$. Obviously $f_0(\boldsymbol{\beta})$ is differentiable. The SMORE problem is equivalent to minimizing $L(\boldsymbol{\beta}) = f_0(\boldsymbol{\beta}) + \sum_{j=1}^{p} f_j(\boldsymbol{\beta}_{\cdot j})$. Then by Proposition 1, we have that $L(\boldsymbol{\beta})$ is regular at each $\boldsymbol{\beta}$. Since $\mathbf{W}_k = \mathbf{U}_k \otimes \mathbf{V}_k$ is positive definite, with Lemma 2, we have compact set $\mathcal{B}^0 = \{\boldsymbol{\beta} : L(\boldsymbol{\beta}) \leq L(\boldsymbol{\beta}^0)\}$. Since $L(\boldsymbol{\beta})$ is continuous and convex, by Proposition 2, the coordinate descent algorithm converges to a stationary point of $L(\boldsymbol{\beta})$, which is exactly the global minimizer as $L(\boldsymbol{\beta})$ is strictly convex. □

References

[1] J. B. Pereira-Leal, A. J. Enright, C. A. Ouzounis, Detection of functional modules from protein interaction networks, Proteins: Structure, Function, and Bioinformatics 54 (1) (2004) 49–57.

[2] K. Basso, A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera, A. Califano, Reverse engineering of regulatory networks in human b cells, Nature genetics 37 (2005) 382–90.

[3] R. Bonneau, M. Facciotti, D. Reiss, A. Schmid, M. Pan, A. Kaur, V. Thorsson, P. Shannon, M. Johnson, J. C. Bare, W. Longabaugh, M. Vuthoori, K. Whitehead, A. Madar, L. Suzuki, T. Mori, D.-E. Chang, J. Diruggiero, C. H Johnson, N. Baliga, A predictive model for transcriptional control of physiology in a free living cell, Cell 131 (2008) 1354–65.

[4] N. J. Hudson, A. Reverter, B. P. Dalrymple, A differential wiring analysis of expression data correctly identifies the gene containing the causal mutation, PLoS Computational Biology 5 (2009) 840 – 845.

[5] S. Bandyopadhyay, M. Mehta, D. Kuo, M.-K. Sung, R. Chuang, E. J. Jaehnig, B. Bodenmiller, K. Licon, W. Copeland, M. Shales, D. Fiedler, J. Dutkowski, A. Guénolé, H. van Attikum, K. M. Shokat, R. D. Kolodner, W.-K. Huh, R. Aebersold, M.-C. Keogh, N. J. Krogan, T. Ideker, Rewiring of genetic networks in response to dna damage, Science 330 (6009) (2010) 1385–1389.

[6] M. Leiserson, F. Vandin, H.-T. Wu, J. Dobson, J. V Eldridge, J. L Thomas, A. Papoutsaki, Y. Kim, B. Niu, M. Mclellan, M. Lawrence, A. Gonzalez-Perez, D. Tamborero, Y. Cheng, G. A Ryslik, N. L'øpez-Bigas, G. Getz, L. Ding, B. Raphael, Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes, Nature Genetics.

[7] S. D. Zhao, T. T. Cai, H. Li, Direct estimation of differential networks, Biometrika 101 (2) (2014) 253–268.

[8] H. Yuan, R. Xi, C. Chen, M. Deng, Differential network analysis via lasso penalized d-trace loss, Biometrika 104 (4) (2017) 755–770.

[9] A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelić, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem, P. Bühlmann, Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana, Genome Biology 5 (11) (2004) R92.

[10] N. Meinshausen, P. Bühlmann, High-dimensional graphs and variable selection with the lasso, The Annals of Statistics 34 (3) (2006) 1436–1462. doi:10.1214/009053606000000281.

[11] M. Yuan, Y. Lin, Model selection and estimation in the gaussian graphical model, Biometrika 94 (1) (2007) 19–35.

[12] J. Friedman, T. Hastie, R. Tibshirani, Sparse inverse covariance estimation with the graphical lasso, Biostatistics (Oxford, England) 9 (2008) 432–41.

[13] O. Banerjee, L. El Ghaoui, A. d'Aspremont, Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data, Journal of Machine Learning Research 9 (2008) 485–516.

[14] C. Lam, J. Fan, Sparsistency and rates of convergence in large covariance matrix estimation, The Annals of Statistics 37 (6B) (2009) 4254–4278. doi:10.1214/09-AOS720.

[15] M. Yuan, High dimensional inverse covariance matrix estimation via linear programming, Journal of Machine Learning Research 11 (2010) 2261–2286.

[16] T. T. Cai, W. Liu, X. Luo, A constrained ℓ_1 minimization approach to sparse precision matrix estimation, Journal of the American Statistical Association 106 (494) (2011) 594–607.

[17] T. Cai, W. Liu, Adaptive thresholding for sparse covariance matrix estimation, Journal of the American Statistical Association 106 (494) (2011) 672–684. doi:10.1198/jasa.2011.tm10560.

[18] W. Liu, X. Luo, High-dimensional sparse precision matrix estimation via sparse column inverse operator, arXiv preprint.

[19] T. Sun, C.-H. Zhang, Sparse matrix inversion with scaled lasso, Journal of Machine Learning Research 14 (1) (2013) 3385–3418.

[20] T. Zhang, H. Zou, Sparse precision matrix estimation via lasso penalized d-trace loss, Biometrika 101 (1) (2014) 103–120.

[21] J. Chiquet, Y. Grandvalet, C. Ambroise, Inferring multiple graphical structures, Statistics and Computing 21 (4) (2011) 537–553.

[22] J. Guo, E. Levina, G. Michailidis, J. Zhu, Joint estimation of multiple graphical models, Biometrika 98 (1) (2011) 1–15.

[23] K. Mohan, M. Chung, S. Han, D. Witten, S. in Lee, M. Fazel, Structured learning of gaussian graphical models, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 620–628.

[24] D. Patrick, W. Pei, W. D. M., The joint graphical lasso for inverse covariance estimation across multiple classes, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76 (2) (2013) 373–397. doi:10.1111/rssb.12033.

[25] A.-L. Barabasi, Z. N. Oltvai, Network biology: understanding the cell's functional organization, Nature reviews genetics 5 (2) (2004) 101.

[26] A.-L. Barabási, N. Gulbahce, J. Loscalzo, Network medicine: a network-based approach to human disease, Nature reviews genetics 12 (1) (2011) 56.

[27] W. Liu, Structural similarity and difference testing on multiple sparse gaussian graphical models, The Annals of Statistics 45 (6) (2017) 2680–2707. doi:10.1214/17-AOS1539.

[28] B. Jiang, X. Wang, C. Leng, A direct approach for sparse quadratic discriminant analysis, The Journal of Machine Learning Research 19 (1) (2018) 1098–1134.

[29] J. Friedman, T. Hastie, R. Tibshirani, The elements of statistical learning, Vol. 1, Springer series in statistics New York, NY, USA:, 2009.

[30] J. Fan, Z. T. Ke, H. Liu, L. Xia, Quadro: A supervised dimension reduction method via rayleigh quotient optimization, Annals of statistics 43 (4) (2015) 1498.

[31] Q. Li, J. Shao, Sparse quadratic discriminant analysis for high dimensional data, Statistica Sinica 25 (2) (2015) 457–473.

[32] E. Candes, T. Tao, The dantzig selector: Statistical estimation when p is much larger than n, The Annals of Statistics 35 (6) (2007) 2313–2351. doi:10.1214/009053606000001523.

[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning. 3 (1) (2011) 1–122.

[34] X. Li, T. Zhao, X. Yuan, H. Liu, The flare package for high dimensional linear regression and precision matrix estimation in r, Journal of Machine Learning Research 16 (2015) 553–557.

[35] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society, Series B 58 (1996) 267–288.

[36] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68 (1) (2006) 49–67.

[37] L. Meier, S. Van De Geer, P. BÃŒhlmann, The group lasso for logistic regression, Journal of the Royal Statistical Society: Series B (Statistical Methodology).

[38] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software 33 (1) (2010) 1–22.

[39] O. Ledoit, M. Wolf, A well-conditioned estimator for large-dimensional covariance matrices, Journal of multivariate analysis 88 (2) (2004) 365–411.

[40] D. M. Witten, R. Tibshirani, T. Hastie, A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis, Biostatistics 10 (3) (2009) 515–534.

[41] L. Clemmensen, T. Hastie, D. Witten, B. Ersbøll, Sparse discriminant analysis, Technometrics 53 (4) (2011) 406–413.

[42] J. Shao, Y. Wang, X. Deng, S. Wang, Sparse linear discriminant analysis by thresholding for high dimensional data, The Annals of Statistics 39 (2) (2011) 1241–1265.

[43] J. Fan, Y. Feng, X. Tong, A road to classification in high dimensional space: the regularized optimal affine discriminant, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 74 (4) (2012) 745–771.

[44] Q. Mai, H. Zou, M. Yuan, A direct approach to sparse discriminant analysis in ultra-high dimensions, Biometrika 99 (1) (2012) 29–42.

[45] P. Xu, J. Zhu, L. Zhu, Y. Li, Covariance-enhanced discriminant analysis, Biometrica 102 (1) (2015) 33–45.

[46] Q. Mai, Y. Yang, H. Zou, Multiclass sparse discriminant analysis, Statistica Sinica In press.

[47] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[48] A. Liaw, M. Wiener, Package 'randomforest': Breiman and cutler's random forests for classification and regression, R Development Core Team 4 (2014) 6–10.

[49] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297.

[50] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, A. Weingessel, E1071: Misc functions of the department of statistics (e1071), tu wien, R package version 1.5-24 1.

[51] M. J. Ha, V. Baladandayuthapani, K.-A. Do, Dingo: differential network analysis in genomics, Bioinformatics 31 (21) (2015) 3413–3420.

[52] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, Pathwise coordinate optimization, The Annals of Applied Statistics 1 (2) (2007) 302–332.

[53] S. Shalev-Shwartz, A. Tewari, Stochastic methods for l1-regularized loss minimization, Journal of Machine Learning Research. 12 (2011) 1865–1892.

[54] Y. Pan, Q. Mai, X. Zhang, Covariate-adjusted tensor classification in high-dimensions, Journal of the American Statistical Association In press.

[55] J. Yang, Y. Zhang, Alternating direction algorithms for ℓ_1-problems in compressive sensing, SIAM Journal on Scientific Computing 33 (1) (2011) 250–278.

[56] W. Yin, S. Osher, D. Goldfarb, J. Darbon, Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing, SIAM Journal on Imaging Sciences 1 (1) (2008) 143–168.

[57] T. Goldstein, S. Osher, The split bregman method for l1-regularized problems, SIAM Journal on Imaging Sciences 2 (2) (2009) 323–343.

[58] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of Statistics 32 (2) (2004) 407–499. doi:10.1214/009053604000000067.

[59] T. T. Wu, K. Lange, Coordinate descent algorithms for lasso penalized regression, The Annals of Applied Statistics 2 (1) (2008) 224–244. doi:10.1214/07-AOAS147.

[60] J. M. Ortega, W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[61] F. Sha, L. K. Saul, D. D. Lee, Multiplicative updates for nonnegative quadratic programming in support vector machines, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Advances in Neural Information Processing Systems 15, MIT Press, 2003, pp. 1065–1072.

[62] Y. Wu, An ordinary differential equation-based solution path algorithm, Journal of nonparametric statistics 23 (1) (2011) 185–199.

[63] H. Zhou, Y. Wu, A generic path algorithm for regularized statistical estimation, Journal of the American Statistical Association 109 (506) (2014) 686–699.

[64] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM Journal on Imaging Sciences 2 (1) (2009) 183–202.

[65] B. N. Flury, Common principal components in k groups, Journal of the American Statistical Association 79 (388) (1984) 892–898.

[66] B. K. Flury, Two generalizations of the common principal component model, Biometrika 74 (1) (1987) 59–69.

[67] J. R. Schott, Partial common principal component subspaces, Biometrika 86 (4) (1999) 899–908.

[68] R. J. Boik, Spectral models for covariance matrices, Biometrika 89 (1) (2002) 159–182.

[69] R. D. Cook, L. Forzani, Covariance reducing models: An alternative to spectral modelling of covariance matrices, Biometrika 95 (4) (2008) 799–812.

[70] A. Franks, P. Hoff, Shared subspace models for multi-group covariance estimation, arXiv preprint arXiv:1607.03045.

[71] W. Wang, X. Zhang, L. Li, Common reducing subspace model and network alternation analysis, Biometrics.

[72] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, Journal of Optimization Theory and Applications 109 (2001) 475–494.

[73] D. P. Bertsekas, Nonlinear Programming, 3rd Edition, Athena Scientific, 2016.