HHS Public Access

Author manuscript

KDD. Author manuscript; available in PMC 2017 June 02.

Published in final edited form as:

KDD. 2016 August ; 2016: 1585–1594. doi:10.1145/2939672.2939851.

Fast Component Pursuit for Large-Scale Inverse Covariance Estimation

Lei Han[†], **Yu Zhang**[‡,*], and **Tong Zhang**[†,§]

[†]Department of Statistics, Rutgers University

[‡]Department of Computer Science and Engineering, Hong Kong University of Science and Technology

[§]Baidu Inc. Beijing, China

Abstract

The maximum likelihood estimation (MLE) for the Gaussian graphical model, which is also known as the inverse covariance estimation problem, has gained increasing interest recently. Most existing works assume that inverse covariance estimators contain sparse structure and then construct models with the ℓ_1 regularization. In this paper, different from existing works, we study the inverse covariance estimation problem from another perspective by efficiently modeling the low-rank structure in the inverse covariance, which is assumed to be a combination of a low-rank part and a diagonal matrix. One motivation for this assumption is that the low-rank structure is common in many applications including the climate and financial analysis, and another one is that such assumption can reduce the computational complexity when computing its inverse. Specifically, we propose an efficient COmponent Pursuit (COP) method to obtain the low-rank part, where each component can be sparse. For optimization, the COP method greedily learns a rank-one component in each iteration by maximizing the log-likelihood. Moreover, the COP algorithm enjoys several appealing properties including the existence of an efficient solution in each iteration and the theoretical guarantee on the convergence of this greedy approach. Experiments on large-scale synthetic and real-world datasets including thousands of millions variables show that the COP method is faster than the state-of-the-art techniques for the inverse covariance estimation problem when achieving comparable log-likelihood on test data.

Keywords

Inverse Covariance Estimation; Component Pursuit; Large-Scale Data; Greedy Algorithm

1. INTRODUCTION

Suppose there are n instances $\{\mathbf{x}_1,..., \mathbf{x}_n\}$ sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where each instance $\mathbf{x}_i \in \mathbb{R}^p$ $(1 \leq i \leq n)$ lies in a p-dimensional space, $\boldsymbol{\mu} \in \mathbb{R}^p$ is the mean, and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ is the covariance matrix. An important and challenging problem is to recover

*The first two authors contributed equally.

$\boldsymbol{\Sigma}$ or its inverse $\boldsymbol{\Sigma}^{-1}$ in a high-dimensional setting where $n \ll p$. Estimating the inverse covariance matrix has attracted a lot of interests in several fields including machine learning, signal processing, computational biology and so on, since it can reveal the dependence among the p attributes [28, 11, 3]. The inverse covariance matrix is estimated by maximizing the log-likelihood as

$$\max_{\boldsymbol{\Theta} \succ \mathbf{0}} \quad \log|\boldsymbol{\Theta}| - \langle \mathbf{S}, \boldsymbol{\Theta} \rangle,$$

or equivalently minimizing the negative log-likelihood (NLL):

$$\min_{\boldsymbol{\Theta} \succ \mathbf{0}} \quad -\log|\boldsymbol{\Theta}| + \langle \mathbf{S}, \boldsymbol{\Theta} \rangle, \tag{1}$$

where $\boldsymbol{\Theta}$ is the inverse covariance estimator, $\boldsymbol{\Theta} \succ \mathbf{0}$ indicates that $\boldsymbol{\Theta}$ is positive definite, $|\cdot|$ denotes the determinant of a square matrix, $\langle \cdot, \cdot \rangle$ denotes the dot product between two matrices or vectors, $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ is the mean of the samples, and $\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$ is the sample covariance matrix. If directly solving problem (1), we can obtain an analytical solution for $\boldsymbol{\Theta}$ as $\boldsymbol{\Theta} = \mathbf{S}^{-1}$. Under the high-dimensional setting where $n \ll p$, $\mathbf{S}$ is rank-deficient and hence this analytical solution is ill-posed. In order to make the problem well-defined, some regularizers are used to constrain $\boldsymbol{\Theta}$ and a widely used one is the ℓ_1 regularization [28, 11, 3] which assumes that $\boldsymbol{\Theta}$ is sparse with the objective function formulated as

$$\min_{\boldsymbol{\Theta} \succ \mathbf{0}} \quad g(\boldsymbol{\Theta}) = -\log|\boldsymbol{\Theta}| + \langle \mathbf{S}, \boldsymbol{\Theta} \rangle + \rho \|\boldsymbol{\Theta}\|_1, \tag{2}$$

where ρ is a regularization parameter that controls the trade-off between the sparseness of $\boldsymbol{\Theta}$ and the fitness to the data.

A large body of works have been devoted to solving problem (2) recently [4, 6, 11, 23, 24, 25, 16, 17, 26], among which the state-of-the-art methods including the QUIC [16], Big-QUIC [17] and BCDIC [26] methods can handle $\boldsymbol{\Theta}$ with billions of entries under the assumption that $\boldsymbol{\Theta}$ is sparse. Those methods commonly use Newton proximal approaches, where a quadratic approximation is made and one key step is to calculate the inverse of $\boldsymbol{\Theta}$, to minimize the NLL. Obviously, the computational bottleneck in those methods is that they need to compute the matrix inverse $\boldsymbol{\Theta}^{-1}$ in each iteration, which is computationally heavy when p is very large. Although the Big-QUIC and BCDIC methods alleviate this problem by splitting the huge matrix $\boldsymbol{\Theta}$ into blocks and use some cheaper operations, e.g., solving some linear systems, to update the corresponding blocks in $\boldsymbol{\Theta}^{-1}$, the matrix inverse operation, whose complexity is $O(p^3)$, is still unavoidable. Actually, almost all the existing methods to solve problem (2) have this problem. Moreover, in the QUIC, Big-QUIC and BCDIC

methods, an operation used to largely improve the efficiency is to restrict the number of updated positions in $\mathbf{\Theta}$ and this operation works well when the optimal $\mathbf{\Theta}$ is very sparse, corresponding to a situation that the regularization parameter ρ in problem (2) has a large value. To see this, empirical studies in those works [16, 17, 26] choose ρ such that the optimal $\mathbf{\Theta}$ has only $10p$ non-zero entries out of a total number of p^2 entries and so only a very small fraction (i.e., $10/p$) in the optimal $\mathbf{\Theta}$ can have non-zero values. Therefore, although those works claim that their methods can handle a covariance matrix with billions of entries, only a small number of non-zero values are actually computed. Empirically we find that the QUIC, Big-QUIC, and BCDIC methods are not very efficient when ρ has a smaller value. Moreover, an extremely sparse $\mathbf{\Theta}$ learned in those works may fail to recover the true relations between attributes.

In this paper, we investigate the inverse covariance estimation problem from another perspective by modeling the low-rank structure contained in $\mathbf{\Theta}$. One motivation for learning the low-rank structure in $\mathbf{\Theta}$ is that the low-rank structure is common in many applications. For example, in climate research, spatially close locations usually exhibit strong dependencies in the climate attributes and such geographical consistency usually leads to low-rank structure in the data [14, 2, 27]. Similarly, in traffic analysis, strong local correlations have been detected on large-scale traffic networks and hence low-rank structure exists [13]. Moreover, in computational finance, a large body of works have focused on estimating nearly low-rank covariance or precision matrices for economy and stock analysis [9, 10, 7, 8]. Moreover, in addition to the generality of the low-rank structure in various applications, this assumption can bring the computational benefit since the matrix inverse $\mathbf{\Theta}^{-1}$ required in each iteration can be computed in lower complexity.

Specifically, we propose a COmponent Pursuit (COP) method which assumes that the inverse covariance is a combination of a diagonal matrix and a low-rank matrix which can be sparse. In order to obtain the low-rank part in $\mathbf{\Theta}$, the COP method greedily learns a rank-one component in each iteration by maximizing the log-likelihood, where each rank-one component can be sparse. The subproblem associated with each rank-one component is shown to be non-convex under the high-dimensional setting but due to the special structure of the subproblem, we can prove that all its local optimums have the globally optimal objective value, making the optimization easier. We further show that the greedy COP algorithm inherently enjoys several appealing properties including the existence of an efficient solution for each subproblem and the theoretical guarantee on the convergence of this greedy approach. Compared with most existing methods whose complexity is $O(p^3)$, the proposed COP method only takes $O(p^2)$ operations. Experiments on large-scale synthetic and real-world datasets show that the proposed COP method is faster than the state-of-the-art methods for large-scale inverse covariance estimation when achieving comparable log-likelihood on test data.

Notations—We use lower-case letters for scalars, bold-face and lower-case letters for vectors, and bold-face and upper case letters for matrices. tr(·) denotes the trace of a square matrix. rank(·) denotes the rank of a matrix. diag(·) converts a vector to a diagonal matrix or extracts the diagonal entries in a square matrix to form a vector. $\|\cdot\|_2$ denotes the ℓ_2 norm of a

vector. $\|\cdot\|_*$ denotes the ℓ_2 norm of a matrix, which equals the maximum eigenvalue of a square matrix.

2. RELATED WORK

Most of the previous studies [4, 6, 11, 23, 24, 25, 16, 17, 26] assume that the inverse covariance matrix is sparse and propose different optimization algorithms to solve problem (2). Different from those approaches, we aim to learn low-rank structure in the inverse covariance matrix.

Similar to our work, some recent methods investigate other structures of the inverse covariance instead of learning with pure sparsity. For example, in [23, 14, 17], the inverse covariance matrix is assumed to have diagonal block structure, where each diagonal block matrix is sparse, when the attributes can be divided into several groups with each one containing similar attributes. Moreover, the latent Gaussian graphical model (LGGM) proposed in [5] assumes that the inverse covariance is equal to the difference between a sparse matrix and a low-rank matrix, and two algorithms [22, 15] including the alternating direction method and Newton proximal method have been proposed for the LGGM method. However, these methods [23, 14, 17] still learn the sparse inverse covariance and the LGGM method treats the sparse part as a dominate part. Moreover, computing the matrix inverse with $O(p^3)$ complexity is unavoidable in the LGGM method and even worse, it has to recover the low-rank part via the eigen-decomposition in each iteration, which also costs $O(p^3)$. Different from these algorithms, the proposed COP method focuses on learning the low-rank part and greedily pursuits a rank-one component in each iteration, whose complexity is $O(p^2)$.

The proposed COP method seems related to the principal component analysis (PCA) [18] but they are different, since the PCA assumes the covariance matrix is a sum of a low-rank part and a diagonal one but in the proposed COP method, the *inverse* covariance matrix is a combination of a low-rank part and a diagonal one, implying that the covariance matrix equals the difference between a diagonal part and a low-rank one.

3. MOTIVATION AND PROBLEM SETUP

In this section, we formally present the motivation and define the problem. In order to make the inverse covariance $\boldsymbol{\Theta}$ positive definite to satisfy the constraint of problem (1), we assume that $\boldsymbol{\Theta}$ is combination of two matrices, i.e., $\boldsymbol{\Theta} = \mathbf{L} + \mathbf{P}$, where $\mathbf{L}$ is a low-rank positive semidefinite matrix and $\mathbf{P}$ is a positive definite diagonal matrix. Such assumption on the structure of $\mathbf{L}$ and $\mathbf{P}$ is motivated by the solution of problem (1) as revealed in the following corollary.

Corollary 1—*The optimal solution* $\boldsymbol{\Theta}^*$ *of problem (*1*) satisfies the following condition:*

$$\boldsymbol{\Theta}^* \succcurlyeq \frac{1}{\|\mathbf{S}\|_\star}\mathbf{I},$$

where $\mathbf{I}$ *is an identity matrix with appropriate size and* $\mathbf{A} \succeq \mathbf{B}$ *implies that* $\mathbf{A} - \mathbf{B}$ *is positive semidefinite for two square matrices* $\mathbf{A}$ *and* $\mathbf{B}$.

Corollary 1 can be directly proved by theorems in [3, 21] and thus its proof is omitted here.

From Corollary 1, $\boldsymbol{\Theta}^*$ can be rewritten as $\boldsymbol{\Theta}^* = \left(\boldsymbol{\Theta}^* - \frac{1}{\|\mathbf{S}\|_\star}\mathbf{I}\right) + \frac{1}{\|\mathbf{S}\|_\star}\mathbf{I}$ where $\frac{1}{\|\mathbf{S}\|_\star}\mathbf{I}$ is diagonal and $\boldsymbol{\Theta}^* - \frac{1}{\|\mathbf{S}\|_\star}\mathbf{I}$ can be assumed to capture the low-rank structure. Inspired by this decomposition, we just assume that $\mathbf{L}$ is a low-rank positive definite matrix and $\mathbf{P} = \mathrm{diag}(\boldsymbol{\eta})$ is a diagonal matrix where $\boldsymbol{\eta} \in \mathbb{R}^p$ with each entry, i.e., η_i, positive. As we will see later, such assumption on the structure of $\boldsymbol{\Theta}$ can bring computational benefit since the complexity to compute $\boldsymbol{\Theta}^{-1}$ reduces from $O(p^3)$ to $O(p^2)$.

Then we are ready to present the problem formulation. Given the sample covariance matrix $\mathbf{S} \in \mathbb{R}^{p \times p}$, we consider the inverse covariance estimation problem by assuming a low-rank plus diagonal structure as

$$\begin{aligned} &\min_{\boldsymbol{\Theta}} \mathscr{L}(\boldsymbol{\Theta}) = -\log|\boldsymbol{\Theta}| + \langle \mathbf{S}, \boldsymbol{\Theta} \rangle \\ &\text{s.t. } \boldsymbol{\Theta} = \mathbf{L} + \mathbf{P},\ \mathbf{P} = \mathrm{diag}(\boldsymbol{\eta}),\ \eta_i > 0,\ \mathbf{L} \succcurlyeq \mathbf{0},\ \mathrm{rank}(\mathbf{L}) \leq r, \end{aligned} \quad (3)$$

where $r \ll p$ is a pre-defined rank. In the next section, we propose the efficient COP method to solve problem (3).

4. THE COP METHOD

In this section, we show how to solve problem (3) efficiently. Since there are two parts, $\mathbf{L}$ and $\mathbf{P}$, in problem (3), we use an alternating method to solve it. That is, in each iteration, we first optimize problem (3) with respect to (w.r.t.) $\mathbf{P}$ by fixing $\mathbf{L}$ and then estimate $\mathbf{L}$ with $\mathbf{P}$ fixed, where $\mathbf{L}$ is learned by pursuing its rank-one components greedily.

4.1 Learning Diagonal Part

When the low-rank component $\mathbf{L}$ is fixed, the problem w.r.t. the diagonal part $\mathbf{P}$ is

$$\begin{aligned} &\min_{\mathbf{P}} h(\mathbf{P}) = -\log|\mathbf{L} + \mathbf{P}| + \langle \mathbf{S}, \mathbf{P} \rangle \\ &\text{s.t. } \mathbf{P} = \mathrm{diag}(\boldsymbol{\eta}), \eta_i > 0. \end{aligned} \quad (4)$$

It is easy to prove that problem (4) is convex w.r.t. $\mathbf{P}$ or $\boldsymbol{\eta}$ and we can use some gradient descent method to solve it directly, where the gradient of the objective function in problem (4) is

$$\nabla_\eta h(\mathbf{P}) = -\mathrm{diag}((\mathbf{L} + \mathbf{P})^{-1}) + \mathrm{diag}(\mathbf{S}),$$

where $(\mathbf{L}+\mathbf{P})^{-1} = \mathbf{P}^{-1} - \mathbf{P}^{-1}\mathbf{U}(\mathbf{I}+\mathbf{U}^T\mathbf{P}^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{P}^{-1}$ by utilizing the low-rank structure of $\mathbf{L}$ that $\mathbf{L}$ equals $\mathbf{U}\mathbf{U}^T$ for some low-rank $\mathbf{U}$ and hence it can be computed efficiently. Then, with a carefully chosen step size as [16, 17, 15], the positiveness of η_i's can be guaranteed in each iteration.

Moreover, at the beginning of the COP algorithm, $\mathbf{L}$ is set to be a zero matrix and the problem for $\mathbf{P}$ is formulated as

$$\min_{\mathbf{P}} \left\{ -\log|\mathbf{P}| + \langle \mathbf{S}, \mathbf{P} \rangle = -\sum_{i=1}^{p} \log\eta_i + \langle \mathrm{diag}(\mathbf{S}), \boldsymbol{\eta} \rangle \right\},$$

which has an analytical solution $\eta_i = \frac{1}{s_{ii}}$ for $1 \le i \le p$, where s_{ij} denotes the (i, j)th element in $\mathbf{S}$ and s_{ii} is positive since $\mathbf{S}$ is a covariance matrix. We use this analytical solution as the initialization for $\mathbf{P}$.

4.2 Component Pursuit for Low-Rank Part

With a fixed $\mathbf{P}$, we aim to learn the low-rank part $\mathbf{L}$ efficiently. We propose to pursue its rank-one components of $\mathbf{L}$ iteratively. When $\mathbf{P}$ is fixed, the problem w.r.t. $\mathbf{L}$ can be formulated as

$$\begin{aligned} \min_{\mathbf{L}} \quad & -\log|\mathbf{L}+\mathbf{P}| + \langle \mathbf{S}, \mathbf{L} \rangle \\ \text{s.t.} \quad & \mathbf{L} \succcurlyeq \mathbf{0},\ \mathrm{rank}(\mathbf{L}) \le r. \end{aligned} \tag{5}$$

In order to make the whole algorithm efficient, we aim to learn the rank-one components in $\mathbf{L}$ greedily and hence in the $(k+1)$th iteration we formulate the estimation $\mathbf{L}_{k+1}$ as $\mathbf{L}_{k+1} = \mathbf{L}_k + \mathbf{u}_{k+1}\mathbf{u}_{k+1}^T$ where $\mathbf{L}_k$ is the low-rank estimation obtained until the kth iteration and $\mathbf{u}_{k+1}$ is the rank-one component to be learned in the $(k+1)$th iteration. By defining $\mathbf{M}_k = \mathbf{L}_k + \mathbf{P}_k$, the subproblem w.r.t. $\mathbf{u}_{k+1}$ in the $(k+1)$th iteration can be formulated as

$$\min_{\mathbf{u}} F(\mathbf{u}\mathbf{u}^T) \triangleq -\log|\mathbf{M}_k + \mathbf{u}\mathbf{u}^T| + \langle \mathbf{S}, \mathbf{u}\mathbf{u}^T \rangle, \tag{6}$$

which can be simplified by omitting some constant terms as

$$\min_{\mathbf{u}} f(\mathbf{u}) \triangleq -\log\left(1 + \mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}\right) + \langle \mathbf{S}, \mathbf{u}\mathbf{u}^T \rangle. \tag{7}$$

Based on problem (7), we are also interested in learning structured components. For example, in many situations, the rank-one component in the low-rank structure can be sparse

[29]. To obtain sparse components via the ℓ_1 regularization, a simple variant of problem (7) can be formulated as

$$\min_{\mathbf{u}} -\log\left(1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}\right)+\langle \mathbf{S}, \mathbf{u}\mathbf{u}^T\rangle+\gamma\|\mathbf{u}\|_1, \quad (8)$$

where $\gamma > 0$ is a regularization parameter controlling sparsity in the rank-one component vector $\mathbf{u}$.

Here we investigate both problems (7) and (8). For the two problems, the following theorem with its proof in the appendix shows that they are non-convex under the high-dimensional setting.

Theorem 1—*When $n \ll p$, problems (7) and (8) are non-convex w.r.t.* $\mathbf{u}$.

According to Theorem 1, we could only find a local optimum of $\mathbf{u}_{k+1}$, making the greedy algorithm hard to learn a globally optimal rank-one component of $\mathbf{L}$ in each iteration. Fortunately, we find that all the local optimums of problem (7) have the same globally optimal objective value of problem (6) according to the following theorem.

Theorem 2—*For problem (7), if* $\mathbf{u}$ *is a rank deficient local minimum of* $f(\mathbf{u})$*, then* $\mathbf{U} = \mathbf{u}\mathbf{u}^T$ *is a global minimum of* $F(\mathbf{U})$*, i.e., all the local optimums have the same globally optimal objective value in problem (6).*

Theorem 2 allows us to use any optimization method, which can find a local optimum, to solve problem (7). Generally, we can use gradient descent algorithms since the objective function $f(\cdot)$ is differentiable and its gradient can be computed as

$$\nabla_{\mathbf{u}} f(\mathbf{u}) = -\frac{2\mathbf{M}_k^{-1}\mathbf{u}}{1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}} + 2\mathbf{S}\mathbf{u}.$$

For problem (8), there is no result similar to Theorem 2. However, we can use general proximal gradient (GPG) methods [12, 20] to solve it efficiently by using the optimal solution of problem (7) as the initialization to speedup the convergence. The entire greedy COP algorithm is depicted in Algorithm 1.

Algorithm 1

The COP algorithm.

Input: $\mathbf{S}$, r;

Output: $\hat{\boldsymbol{\Theta}}$;

1: Initialize $\mathbf{P}_0$ and set $\mathbf{M}_0 = \mathbf{P}_0$, $k = 0$;

2: **repeat**

3: Solve problem (7) or (8) with fixed $\mathbf{P}_k$;

4: $\mathbf{L}_{k+1}=\mathbf{L}_k+\mathbf{u}_{k+1}^{*}\mathbf{u}_{k+1}^{*T}$;

5: $\mathbf{M}_k = \mathbf{L}_k + \mathbf{P}_k$;

6: Compute $\mathbf{M}_{k+1}^{-1}$;

7: Update $\mathbf{P}_k$ with fixed $\mathbf{L}_k$;

8: $k := k + 1$;

9: **until** $k > r$ or some convergence criterion is satisfied

10: $\hat{\boldsymbol{\theta}} = \mathbf{M}_k$;

5. THEORETICAL ANALYSIS

In this section, we theoretically analyze the COP method, where we derive an efficiently analytical solution for problem (7) and prove the convergence of the COP algorithm in Algorithm 1.

We first present some interesting properties, which set the stage for the introduce of our main results, of the COP method.

Proposition 1—*Assume* $\mathbf{M}_k$ *is the matrix obtained in the kth iteration of* Algorithm 1. *If there exists a vector* $\mathbf{a} \in \mathbb{R}^p$ *that*

$$\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}>\mathbf{a}^T\mathbf{S}\mathbf{a}>\mathbf{0}, \quad (9)$$

then by defining

$$\alpha=\sqrt{\frac{1}{\mathbf{a}^T\mathbf{S}\mathbf{a}}-\frac{1}{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}} \ and \ \mathbf{u}=\alpha\mathbf{a}, \quad (10)$$

we have $\mathcal{L}(\mathbf{M}_k + \mathbf{u}\mathbf{u}^T) < \mathcal{L}(\mathbf{M}_k)$. *Otherwise, adding any rank-one component to* $\mathbf{M}_k$ *will not decrease the NLL, implying that* Algorithm 1 *will stop at the kth iteration.*

Proposition 1 provides the necessary condition, i.e., Eq. (9), for the convergence of the COP method. Note that Proposition 1 does not require that **u** should be a local optimum of problem (7) or (8).

Proposition 2—*Suppose a vector* **a** *satisfies* Eq. (9) *and define* $c=\frac{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}{\mathbf{a}^T\mathbf{S}\mathbf{a}}>1$. *Then, using the definitions in* Eq. (10), *the decrease of the NLL in the two successive iterations, i.e.* $\mathcal{L}(\mathbf{M}_k) - \mathcal{L}(\mathbf{M}_k + \mathbf{u}\mathbf{u}^T)$, *is a monotone increasing function w.r.t. c:*

$$\mathscr{L}(\mathbf{M}_k)-\mathscr{L}(\mathbf{M}_k+\mathbf{u}\mathbf{u}^T) \triangleq q(c)=\log c+\frac{1}{c}-1, \quad (11)$$

where $c > 1$.

Proposition 2 implies that in order to achieve fast decrease in the NLL by adding a rank-one component to $\mathbf{M}_k$, we need to choose the maximum value of c. Until now, both the Propositions 1 and 2 hold for Algorithm 1 when solving either problem (7) or (8), since those results are obtained by analyzing the difference of the NLL values in two successive iterations. When we solve problem (7) based on the COP algorithm, we can obtain an analytical solution for it with the detailed result shown in the following proposition.

Proposition 3—*If there exists a vector* $\mathbf{a}$ *satisfying* Eq. (9), *then problem (7) is equivalent to the following Rayleigh quotient problem:*

$$\max_{\mathbf{a}} c \triangleq \frac{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}{\mathbf{a}^T\mathbf{S}\mathbf{a}} \quad \text{s.t. } \mathbf{a}^T\mathbf{S}\mathbf{a}>0, \quad (12)$$

which admits an analytical solution by solving the generalized eigen-decomposition problem $\mathbf{M}_k^{-1}\mathbf{a}^*=\lambda^*\mathbf{S}\mathbf{a}^*$ *with* λ^* *and* $\mathbf{a}^*$ *as the largest eigenvalue and the corresponding eigenvector. Moreover,* $c_{\max}^{(k+1)}$*, i.e., the maximum value that c can reach in the* $(k+1)$*th iteration of* Algorithm 1, *can be computed as*

$$c_{\max}^{(k+1)}=\max_{\mathbf{a}}\frac{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}{\mathbf{a}^T\mathbf{S}\mathbf{a}}=\frac{(\mathbf{a}^*)^T\mathbf{M}_k^{-1}\mathbf{a}^*}{(\mathbf{a}^*)^T\mathbf{S}\mathbf{a}^*}=\lambda^*. \quad (13)$$

In Proposition 3, the largest eigenvalue λ^* and eigenvector $\mathbf{a}^*$ of the generalized eigen-decomposition problem can be computed efficiently by the power method [19]. Moreover, Proposition 3 implies that solving the Rayleigh quotient problem also provides a way to check whether Eq. (9) can be satisfied in the $(k+1)$th iteration by testing whether $c_{\max}^{(k+1)}=\lambda^*>1$ holds or not. When solving problem (8) instead, we directly check Eq. (9) based on the component obtained by the GPG method to determine whether the COP algorithm needs to be terminated.

In the following theorems, we present the analytical solution for problem (7) and prove the convergence of the COP algorithm in Algorithm 1.

Theorem 3 (Analytical Solution)—*Let* $\mathbf{M}_k$ *be the matrix defined in step 5 of* Algorithm 1 *in the kth iteration and denote by* λ^* *and* $\mathbf{a}^*$ *the largest eigenvalue and the corresponding eigenvector of the generalized eigen-decomposition problem* $\mathbf{M}_k^{-1}\mathbf{a}^*=\lambda^*\mathbf{S}\mathbf{a}^*$. *Then* $\mathbf{u}^*$*, which is defined as*

$$\mathbf{u}^* = \begin{cases} \sqrt{\frac{1}{(\mathbf{a}^*)^T \mathbf{S}\mathbf{a}^*}\left(1-\frac{1}{\lambda^*}\right)} \cdot \mathbf{a}^*, & if\ \lambda^* > 1, \\ \mathbf{0}, & otherwise. \end{cases} \quad (14)$$

is a local optimum of problem (7) in the $(k+1)$*th iteration.*

Theorem 4 (Convergence)—*In the COP algorithm shown in* Algorithm 1, *which solves either problem (7) or (8), the NLL decreases iteratively until convergence.*

Theorems 3 and 4 provide important guarantees for the proposed COP method.

6. SPEEDUP IN HIGH DIMENSIONS

According to Proposition 3 and Theorem 3, a key step in the COP method is solving the Rayleigh quotient problem (12) if we want to adopt the analytical solution for problem (7) or use it to initialize the estimator in problem (8). Both Eq. (9) and problem (12) require that $(\mathbf{a}^*)^T \mathbf{S}\mathbf{a}^* > 0$ for the optimal $\mathbf{a}^*$ or equivalently $\mathbf{a}^*$ lies in the range space of $\mathbf{S}$. We can rewrite $\mathbf{S}$ as $\mathbf{S} = \mathbf{X}^T\mathbf{X}$ if we assume that the data samples are normalized to have zero sample mean and based on this reformulation, we can see that the range space of $\mathbf{S}$ is spanned by $\mathbf{X}$, implying that $\mathbf{a}^*$ lies in the row space of $\mathbf{X}$. Hence we can represent $\mathbf{a}$ as $\mathbf{a} = \mathbf{X}^T\mathbf{b}$ where $\mathbf{b} \in \mathbb{R}^n$ contains the spanning coefficients. Accordingly problem (12) can be reformulated as

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \frac{\mathbf{b}^T \left(\mathbf{X}\mathbf{M}_k^{-1}\mathbf{X}^T\right) \mathbf{b}}{\mathbf{b}^T \left(\mathbf{X}\mathbf{S}\mathbf{X}^T\right) \mathbf{b}}. \quad (15)$$

Problem (15) is still a Rayleigh quotient problem which can be solved by the power method. One advantage to solve problem (15) instead of problem (12) is that the size of matrices in the generalized eigen-decomposition problem (15) is $n \times n$ which is much smaller than that of problem (12) under the high-dimensional setting where $n \ll p$, leading to a much more efficient implementation and a significant speedup. Moreover, when solving problem (15), we only need to store the data matrix $\mathbf{X}$ instead of the sample covariance matrix $\mathbf{S}$ as in problem (12), which can largely reduce the storage requirement. For the case where $n > p$, we still solve problem (12) since in this situation the null space of $\mathbf{S}$ is empty with a large probability.

7. COMPLEXITY ANALYSIS

In this section, we discuss the complexity of the proposed COP method in Algorithm 1 and compare with existing approaches.

In each iteration of Algorithm 1, the matrix inverse $\mathbf{M}_{k+1}^{-1}$ is needed in step 6. Since $\mathbf{M}_{k+1} = \mathbf{M}_k + \mathbf{u}_{k+1}^{*T}\mathbf{u}_{k+1}^*$ where $\mathbf{u}_{k+1}^*$ is a vector, we can efficiently compute $\mathbf{M}_{k+1}^{-1}$ as

$$\mathbf{M}_{k+1}^{-1}=\mathbf{M}_k^{-1}-\frac{\mathbf{M}_k^{-1}\mathbf{u}_{k+1}^*\mathbf{u}_{k+1}^{*T}\mathbf{M}_k^{-1}}{1+\mathbf{u}_{k+1}^{*T}\mathbf{M}_k^{-1}\mathbf{u}_{k+1}^*},$$

which only needs $O(p^2)$ operations because $\mathbf{M}_k^{-1}$ has already been stored during the previous iteration. Step 3 in Algorithm 1 when we consider problem (7) needs to solve problem (12) or (15), whose complexity is $O(\min(p^2, n^2))$. When solving problem (8), the complexity of the GPG method is no more than $O(p^2)$, and when we adopt the optimal solution of problem (7) as the initialization, the GPG method will converge fast in considerably few iterations. Moreover, updating the diagonal matrix $\mathbf{P}_k$ in each iteration costs $O(p^2)$. In a word, the overall time complexity of the COP algorithm is $O(rp^2)$ where r is the pre-defined rank satisfying $r \ll p$. Moreover, the storage requirement for the the two matrices (i.e., $\mathbf{L}$ and $\mathbf{P}$) in the COP algorithm is a linear function w.r.t. p, since we only need to keep the diagonal elements in $\mathbf{P}_k$ and the component vectors $\{\mathbf{u}_1^*, \cdots, \mathbf{u}_k^*\}$.

All the sparse inverse covariance estimation methods including [4, 6, 11, 24, 25, 16] use the first-order or second-order proximal methods to solve problem (2) where computing the inverse of a $p \times p$ matrix $\mathbf{\Theta}_k^{-1}$ is needed and costs $O(p^3)$. So the computational complexity of the COP method is lower than those of the aforementioned approaches. For the LGGM methods [5, 22, 15], which assume the inverse covariance has a sparse minus low-rank structure, they need to compute the inverse of $p \times p$ matrices with $O(p^3)$ cost and also need additional $O(p^3)$ operations for the eigen-decomposition to update the low-rank part, making it have higher complexity than the proposed COP method. Moreover, as discussed before, the storage complexity of the COP algorithm is $O(p)$ but those of the above approaches depend on the number of non-zero entries in $\mathbf{\Theta}$, which could be $O(p^2)$ in the worst case.

8. EXPERIMENTS

In this section, we conduct experiments on both synthetic and real-world datasets to evaluate the proposed COP method and the ℓ_1-regularized COP method (COP-ℓ_1).

8.1 Experimental Settings

We compare with a number of state-of-the-art methods for the inverse covariance estimation problem, including the QUIC [16], Big-QUIC [17], BCDIC [26] and QUIC&Dirty [15] methods.[1] Among those methods, the QUIC, Big-QUIC and BCDIC methods are the state-of-the-art sparse inverse covariance estimation methods, while the QUIC&Dirty method is currently the most efficient algorithm for the LGGM problem. The implementations for the QUIC, Big-QUIC, BCDIC and QUIC&Dirty methods adopt the recommended settings as provided in their works and the Big-QUIC and BCDIC methods are parallelized with multiple cores. All the experiments are performed on a machine with dual 6-core Intel Xeon X5650 2.66GHz processor and 32GB RAM.

[1]The codes for the QUIC, Big-QUIC and QUIC&Dirty methods can be downloaded from http://www.stat.ucdavis.edu/~chohsieh/ and that for the BCDIC method can be obtained at http://www.javierturek.com/software/.

In the experiments, we split the data into a training set containing 90% of the samples and a test set with the rest samples. We use $\mathbf{S}_{train}$ to denote the sample covariance matrix on the training set and $\mathbf{S}_{test}$ as the sample covariance matrix on the test set. All the data are normalized such that the diagonal elements in both $\mathbf{S}_{train}$ and $\mathbf{S}_{test}$ are all ones. By following [16, 17, 26], we choose the regularization parameter ρ in problem (2) for the QUIC, Big-QUIC and BCDIC methods such that the estimated $\hat{\boldsymbol{\Theta}}$ contains approximately $10p$ non-zero elements. For the QUIC&Dirty method, we set its regularization parameter ρ_1 for the sparse part to be ρ in problem (2) and choose another regularization parameter ρ_2 for the low-rank part from $\{0.1, 1, 10\}$. For the COP-ℓ_1' method, we choose the best γ from the candidate set $\{10^{-5}, 10^{-4}, \cdots, 10^{-1}\}$.

8.2 Experiments on Synthetic Data

In this section, we conduct experiments on synthetic data to test the performance of the proposed COP and COP-ℓ_1' methods.

8.2.1 Results on Small-Scale Data—We first generate a dataset of small scale to test the correctness of the theoretical results presented in Section 5. In order to do this, we generate a matrix $\mathbf{A} \in \mathbb{R}^{r^* \times p}$, where $r^* = 20$ and $p = 100$. Each entry in $\mathbf{A}$ is sampled from the standard normal distribution $\mathcal{N}(0, 1)$. Then we generate the sample covariance matrix $\mathbf{S}^*$ as $(\mathbf{S}^*)^{-1} = \mathbf{A}^T\mathbf{A} + \mathbf{I}$. We estimate $\hat{\boldsymbol{\Theta}}$ by solving problem (1) and compare the estimated $\hat{\boldsymbol{\Theta}}$ and $(\mathbf{S}^*)^{-1}$ to see whether they are exactly the same. In order to see the learned $\mathbf{L}$, we suppose that the diagonal part $\mathbf{P}$ is known and set to be the ground truth, i.e., the identity matrix $\mathbf{I}$.

Fig. 1 shows detailed results of the COP method. Fig. 1(a) depicts the change of the NLL values when increasing the the number of components or equivalently the iterations. Since we are aware of the ground truth of the inverse covariance, we can calculate the ground truth of the NLL value which is illustrated by the red dashed line. We see that the NLL of the COP algorithm decreases in almost a linear rate, and when the rank or equivalently the number of components reaches 20, which is the ground truth for the rank, the COP method exactly recovers the ground truth of the inverse covariance and the corresponding NLL value is equal to the ground truth. Hence, the COP algorithm stops at the 21st iteration by perfectly recovering the ground truth of the inverse covariance. Fig. 1(b) plots change of the λ^* against the rank. As expected, the value of λ^* decreases when increasing the rank and when the number of iterations reaches 21, λ^* becomes 1, which implies that Eq. (9) is no longer satisfied, leading to the termination of the COP algorithm. These observed results well match the theoretical results in Section 5.

8.2.2 Results on Large-Scale Data—Similar to the previous section, we generate a matrix $\mathbf{A} \in \mathbb{R}^{r^* \times p}$, where $r^* = 100$ and each entry in $\mathbf{A}$ is sampled from the standard normal distribution $\mathcal{N}(0, 1)$. The true covariance $\mathbf{S}^*$ is generated in the same way as $(\mathbf{S})^{-1} = \mathbf{A}^T\mathbf{A} + \mathbf{I}$. In this case, we generate $n = 1000$ samples, which are stored in the data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, from $\mathcal{N}(\mathbf{0}, \mathbf{S}^*)$. We vary p from 5, 000 to 25, 000 at an interval of 5, 000 to evaluate the performance of all the methods. Since the sparse inverse covariance estimation methods including the QUIC, Big-QUIC and BCDIC methods solve problem (2) and the

QUIC&Dirty method solve the LGGM problem, the comparison among them is not straightforward. In order to make fair evaluations, we compare the running time of different methods when they achieve the same or comparable NLL on the test dataset and the method with the lowest running time is the most efficient one. Moreover, we compare the COP and COP-ℓ_1 methods with the sparse inverse covariance estimation methods and the QUIC&Dirty method separately.

Table 1 shows the results by comparing the proposed COP and COP-ℓ_1 methods with the sparse inverse covariance estimation methods. In Table 1, there are seven groups of columns. The first group of columns denotes different settings for p. The second group shows the value of the regularization parameter ρ in problem (2), the number of non-zero (NNZ) entries in the estimated $\hat{\Theta}$ which is around $10p$ by following experimental settings in the original works, and the NLL on the test data denoted by NLL_{te} for the QUIC, Big-QUIC and BCDIC methods. Since all the three methods solve the same problem (i.e., problem (2)), their NNZ's and NLL_{te}'s are nearly the same and thus we only report the results obtained from the BCDIC method. The third group reports the running time for the QUIC, Big-QUIC and BCDIC methods. The forth group of columns shows the learned rank r and the NLL_{te} of the COP method, and the fifth group reports its running time. Similarly, the last two columns show the results for the COP-ℓ_1 method. From the results, the COP and COP-ℓ_1 methods usually needs 10 to 20 components to achieve comparable NLL_{te} with those of the QUIC, Big-QUIC and BCDIC methods on all the synthetic datasets and the COP and COP-ℓ_1 methods are always faster than other methods under all the settings.

The comparison results among the QUIC&Dirty, COP and COP-ℓ_1 methods are recorded in Table 2. Table 2 has a similar format to Table 1 and it shows all the detailed settings and the running time of the three methods. When p is lower than 20, 000, the COP and COP-ℓ_1 methods are much more efficient than the QUIC&Dirty method when they achieve similar NLL_{te}. When p becomes larger, the QUIC&Dirty method cannot provide the estimation in reasonable time (i.e., 5 hours) and hence it can only handle medium-scale datasets. For the COP and COP-ℓ_1 methods, we try larger ranks, i.e., 50, and they still obtain lower NLL_{te} in reasonable time.

By comparing Table 1 and Table 2, we find that the QUIC&Dirty method has slightly better testing NLL values than the sparse inverse covariance estimation methods when their regularization parameters, which control the sparsity, are set to the same value. This observation reveals that considering both the low-rank and sparse structure can fit the data better than purely sparse inverse covariance in these synthetic datasets. However, training the QUIC&Dirty method is much more computational expensive and hence it can hardly process large-scale data as shown in Table 2.

In Tables 1 and 2, the COP-ℓ_1 method does not perform better than the non-regularized one, and it generally needs more ranks and running time to obtain comparable NLL_{te} with the COP method. This is probably because under the synthetic setting, the ground truth does not contain sparse components.

In addition, we provide more details for the COP method in Fig. 2 which plots iterative results of the COP method on synthetic data with $p = 5,000$. We set the total number of ranks to be 101 in this case. Fig. 2(a) plots the change of the NLL on the training data w.r.t. the number of iterations. Again, we find that the NLL on the training set decreases in a linear rate against the number of iterations. Fig. 2(b) shows the value of λ^* in each iteration and we see that λ^* becomes smaller iteratively while it is always larger than 1 even at the 101st iteration, implying that the algorithm can further proceed. Note that in this situation, the ground truth of the rank is 100 but the COP algorithm does not terminate at the 101st iteration. This is reasonable since under this setting where $n \ll p$, the sampling bias exists in the training data and hence the estimated components are not exactly the true components.

8.3 Experiments on Real-World Datasets

In this section, we conduct experiments on large-scale real-world datasets. We use four datasets from the Gridded Climate Data[2] and one stock dataset collected from the Yahoo finance[3], which are also studied in [14]. The four climate datasets are: (1) the Northern Hemisphere EASE-Grid Weekly Snow Cover and Sea Ice Extent (Snow), which records the weekly snow cover in northern hemisphere on 1.0 latitude $\times$ 1.0 longitude grids from January, 1971 to December, 1995. Each grid is treated as an attribute. By removing invalid observations, the number of attributes p is 9,148, and the number of samples n is 297; (2) the NCEP/NCAR Re-analysis air data (Air), which contains daily air temperature on the earth with 2.5 latitude $\times$ 2.5 longitude global grids. The number of attributes p is 10,512 and by following [14] we use $n = 1460$ records in year 2001; (3) the CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS (Precip), which focuses on the daily precipitation in USA. The valid data contains $p = 13,610$ attributes and we use $n = 3652$ observations from year 1997 to year 2006; (4) the NOAA's Outgoing Longwave Radiation (OLR) Daily Climate Data Record, which provides the OLR records on the earth. In this dataset, p equals 21,720 and n is equal to 2903. For the Stock dataset, we collect $p = 21,602$ stocks with daily closing price recorded in latest 300 days before Dec. 31, 2015.

Table 3 reports experimental results for the QUIC, Big-QUIC, BCDIC and QUIC&Dirty methods on all the datasets, while Table 4 gives the results of the COP and COP-ℓ_1 methods. In Table 3, the QUIC, Big-QUIC and BCDIC methods, whose Θ's have about $10p$ non-zero entries, have much larger NLL's on the test data especially for the Snow, Air, and OLR datasets when comparing with the COP and COP-ℓ_1 methods in Table 4. The QUIC&Dirty method has better NLL_{te} than the sparse inverse covariance estimation methods on the Snow and Air datasets, but it fails to learn the model on the larger Precip, OLR and Stock datasets in reasonable time. Under all the settings, we set the rank of the COP method to be 5, which is good enough to obtain lower NLL's on all the test data, and we choose the model parameters for the COP-ℓ_1 method to obtain similar testing NLL's to the COP method. According to Table 4, in most settings, the COP method not only has better NLL_{te} than the COP-ℓ_1 method but also performs faster. The exceptions are that on the OLR dataset, the

[2] http://www.esrl.noaa.gov/psd/data/gridded/
[3] http://finance.yahoo.com/

COP-ℓ_1 method has better predictive performance and that it is slightly faster on the Stock dataset.

Moreover, we provide some additional results for the QUIC, Big-QUIC and BCDIC methods on the Snow, Air, OLR and Stock datasets in Table 5, where their regularization parameters ρ's are selected such that the resulting estimators can achieve comparable NLL's on the test data with those of the COP and COP-ℓ_1 methods.[4] Table 5 does not include the Precip data, because the result reported in Table 3 is already comparable to those of the COP and COP-ℓ_1 methods. Under this setting, the QUIC&Dirty method still can not return any result in 5 hours and so it is not included. From the results, we can see that in order to achieve lower NLL's on the test set of the four datasets, the numbers of the non-zero entries in their estimators become larger and as a consequence, the running time of the three methods significantly increases, which again demonstrates the efficiency of the proposed COP and COP-ℓ_1 methods.

9. CONCLUSION

In this paper, we proposed an efficient component pursuit (COP) method and its ℓ_1-regularized variant for the large-scale inverse co-variance estimation problem by assuming that the inverse covariance is a combination of a low-rank matrix and a diagonal matrix. Both theoretical analysis and empirical evaluations demonstrate the effectiveness and efficiency of the proposed methods when compared with the state-of-the-art methods.

As a future direction, we are interested in applying the COP methods to more large-scale applications, e.g., the gene expression data, where there exists inherent low-rank structure among the features. Another future direction is to extend the COP method to deal with more complex structure in the inverse covariance estimation problem, e.g., the low-rank plus block diagonal structure, since in many applications such as financial analysis, the group information among features is available as a priori information.

Acknowledgments

This research was partially supported by NSF IIS-1250985, NSF IIS-1407939, NIH R01AI116744 and NSFC 61305071.

References

1. Bach F, Mairal J, Ponce J. Convex sparse matrix factorizations. 2008 arXiv preprint arXiv: 0812.1869.
2. Bahadori MT, Yu QR, Liu Y. Fast multivariate spatio-temporal analysis via low rank tensor learning. Advances in Neural Information Processing Systems. 2014:3491–3499.
3. Banerjee O, El Ghaoui L, d'Aspremont A. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. Journal of Machine Learning Research. 2008; 9:485–516.

[4] ρ with a value smaller than 0.95 on the Snow, OLR and Stock datasets will lead to memory exceeded problem for the three methods and hence we just set ρ to be 0.95.

4. Banerjee, O., Ghaoui, LE., d'Aspremont, A., Natsoulis, G. Convex optimization techniques for fitting sparse Gaussian graphical models. Proceedings of the International Conference on Machine learning; 2006. p. 89-96.
5. Chandrasekaran V, Parrilo PA, Willsky AS. Latent variable graphical model selection via convex optimization. The Annals of Statistics. 2012; 40(4):1935–1967.
6. d'Aspremont A, Banerjee O, El Ghaoui L. First-order methods for sparse covariance selection. SIAM Journal on Matrix Analysis and Applications. 2008; 30(1):56–66.
7. Fan, J., Han, F., Liu, H. Technical report, Technical report. Princeton University; 2014. Page: Robust pattern guided estimation of large covariance matrix.
8. Fan J, Liao Y, Liu H. An overview on the estimation of large covariance and precision matrices. 2015 arXiv preprint arXiv:1504.02995.
9. Fan J, Liao Y, Mincheva M. High dimensional covariance matrix estimation in approximate factor models. Annals of Statistics. 2011; 39(6):3320. [PubMed: 22661790]
10. Fan J, Liao Y, Mincheva M. Large covariance estimation by thresholding principal orthogonal complements. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2013; 75(4):603–680.
11. Friedman J, Hastie T, Tibshirani R. Sparse inverse covariance estimation with the graphical lasso. Biostatistics. 2008; 9(3):432–441. [PubMed: 18079126]
12. Gong, P., Zhang, C., Lu, Z., Huang, JZ., Ye, J. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. Proceedings of the International Conference on Machine Learning; 2013;
13. Han, L., Song, G., Cong, G., Xie, K. Overlapping decomposition for causal graphical modeling. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2012. p. 114-122.
14. Hsieh C-J, Dhillon IS, Ravikumar P, Banerjee A. A divide-and-conquer procedure for sparse inverse covariance estimation. Advances in Neural Information Processing Systems. 2012
15. Hsieh C-J, Dhillon IS, Ravikumar PK, Becker S, Olsen PA. QUIC & DIRTY: A quadratic approximation approach for dirty statistical models. Advances in Neural Information Processing Systems. 2014:2006–2014.
16. Hsieh C-J, Dhillon IS, Ravikumar PK, Sustik MA. Sparse inverse covariance matrix estimation using quadratic approximation. Advances in Neural Information Processing Systems. 2011:2330–2338.
17. Hsieh C-J, Sustik MA, Dhillon IS, Ravikumar PK, Poldrack R. BIG & QUIC: Sparse inverse covariance estimation for a million variables. Advances in Neural Information Processing Systems. 2013:3165–3173.
18. Jolliffe, I. Principal Component Analysis. Wiley Online Library; 2002.
19. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Journal of Research of the National Bureau of Standards. 1950; 45(4):255–282.
20. Li H, Lin Z. Accelerated proximal gradient methods for nonconvex programming. Advances in Neural Information Processing Systems. 2015:379–387.
21. Lu Z. Smooth optimization approach for sparse covariance selection. SIAM Journal on Optimization. 2009; 19(4):1807–1827.
22. Ma S, Xue L, Zou H. Alternating direction methods for latent variable gaussian graphical model selection. Neural Computation. 2013; 25(8):2172–2198. [PubMed: 23607561]
23. Mazumder R, Hastie T. Exact covariance thresholding into connected components for large-scale graphical lasso. Journal of Machine Learning Research. 2012; 13(1):781–794. [PubMed: 25392704]
24. Oztoprak F, Nocedal J, Rennie S, Olsen PA. Newton-like methods for sparse inverse covariance estimation. Advances in Neural Information Processing Systems. 2012:755–763.
25. Rolfs B, Rajaratnam B, Guillot D, Wong I, Maleki A. Iterative thresholding algorithm for sparse inverse covariance estimation. Advances in Neural Information Processing Systems. 2012:1574–1582.

26. Treister E, Turek JS. A block-coordinate descent approach for large-scale sparse inverse covariance estimation. Advances in Neural Information Processing Systems. 2014:927–935.
27. Yu, R., Cheng, D., Liu, Y. Accelerated online low rank tensor learning for multivariate spatiotemporal streams. Proceedings of the 32nd International Conference on Machine Learning; 2015. p. 238-247.
28. Yuan M, Lin Y. Model selection and estimation in the Gaussian graphical model. Biometrika. 2007; 94(1):19–35.
29. Zou H, Hastie T, Tibshirani R. Sparse principal component analysis. Journal of Computational and Graphical Statistics. 2006; 15(2):265–286.

APPENDIX

A. PROOF OF THEOREM 1

Proof

For problem (7), the derivative and the Hessian of $f(\mathbf{u})$ can be calculated as

$$\nabla_{\mathrm{u}} f = -\frac{2\mathbf{M}_k^{-1}\mathbf{u}}{1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}} + 2\mathbf{S}\mathbf{u} \tag{16}$$

$$\begin{aligned}\nabla_{\mathbf{u}}^2 f &= -\frac{2(1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u})\mathbf{M}_k^{-1} - 4\mathbf{M}_k^{-1}\mathbf{u}\mathbf{u}^T\mathbf{M}_k^{-1}}{\left(1+\mathbf{u}_k^T\mathbf{M}_k^{-1}\mathbf{u}_k\right)^2} + 2\mathbf{S} \\ &= \frac{4\mathbf{M}_k^{-1}\mathbf{u}\mathbf{u}^T\mathbf{M}_k^{-1}}{\left(1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}\right)^2} + 2\mathbf{S} - \frac{2}{1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}}\mathbf{M}_k^{-1}.\end{aligned} \tag{17}$$

It is easy to see that $\frac{2}{1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}} < 2$, since $\mathbf{M}_k^{-1}$ is positive-definite and $\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u} > 0$ for any vector $\mathbf{u}$. If $\mathbf{S} \succcurlyeq \mathbf{M}_k^{-1}$, it is easy to see that the hessian $\nabla_{\mathbf{u}}^2 f \succcurlyeq 0$ and $f(\mathbf{u})$ is convex w.r.t. $\mathbf{u}$. However, under the high-dimensional case where $\mathbf{S}$ is positive semidefinite but not positive definite due to $n \ll p$, $2\mathbf{S} - \frac{2}{1+\mathbf{u}^T\mathbf{M}_k^{-1}\mathbf{u}}\mathbf{M}_k^{-1}$ is not positive semi-definite and it contains at least $p - n$ negative eigenvalues. Moreover, the first term in the right-hand side of Eq. (17) is only a rank-one matrix. So $\nabla_{\mathbf{u}}^2 f$ is not positive semidefinite and hence $f(\mathbf{u})$ is a non-convex function w.r.t. $\mathbf{u}$.

B. PROOF OF THEOREM 2

Proof

The proof of Theorem 2 follows directly from the Proposition 4 in [1].

C. PROOF OF PROPOSITION 1

Proof

By considering the difference between $\mathcal{L}(\mathbf{M}_k)$ and $\mathcal{L}(\mathbf{M}_k + \mathbf{u}\mathbf{u}^T)$, we have

$$\begin{aligned}&\mathscr{L}(\mathbf{M}_k)-\mathscr{L}(\mathbf{M}_k+\mathbf{u}\mathbf{u}^T)\\=&-\log|\mathbf{M}_k|+\langle\mathbf{S},\mathbf{M}_k\rangle+\log|\mathbf{M}_k+\alpha^2\mathbf{a}\mathbf{a}^T|-\langle\mathbf{S},\mathbf{M}_k+\alpha^2\mathbf{a}\mathbf{a}^T\rangle\\=&\log\left(1+\alpha^2\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}\right)-\alpha^2\mathbf{a}^T\mathbf{S}\mathbf{a}.\end{aligned}$$

Define $l(\alpha,\mathbf{a})=\log(1+\alpha^2\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a})-\alpha^2\mathbf{a}^T\mathbf{S}\mathbf{a}$. We investigate whether there exists some pair (α, **a**) such that

$$\max_{\alpha,\mathbf{a}} l(\alpha,\mathbf{a})>0.$$

When **a** is fixed, we define $c_1=\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}$ and $c_2 = \mathbf{a}^T\mathbf{S}\mathbf{a}$, where obviously $c_1 > 0$ and $c_2 \geq 0$ since $\mathbf{M}_k^{-1}$ is positive definite and **S** is positive semidefinite. Then $l(\alpha,$ **a**) can be formulated as a function w.r.t. α as

$$l(\alpha)=\log(1+c_1\alpha^2)-c_2\alpha^2. \quad (18)$$

The convexity and the extreme value of $l(\alpha)$ depends on the two scalars c_1 and c_2. Fig. 3(a) plots some examples of the function $l(\alpha)$ when adopting different values for c_1 and c_2. By setting $\frac{\partial l}{\partial\alpha}=0$ we obtain the maximizer of $l(\alpha)$ as

$$\alpha=\begin{cases}\sqrt{\frac{1}{c_2}-\frac{1}{c_1}}, & \text{if } c_1>c_2,\\ 0, & \text{otherwise.}\end{cases}. \quad (19)$$

From Eq. (19), if $c_1 > c_2$, plugging Eq. (19) into $l(\alpha,$ **a**) gives

$$\begin{aligned}&\max_{\mathbf{u}}\mathscr{L}(\mathbf{M}_k)-\mathscr{L}(\mathbf{M}_k+\mathbf{u}\mathbf{u}^T)\\=&\max_{\mathbf{a}}\log\frac{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}{\mathbf{a}^T\mathbf{S}\mathbf{a}}+\frac{\mathbf{a}^T\mathbf{S}\mathbf{a}}{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}-1.\end{aligned}$$

By defining $c=\frac{\mathbf{a}^T\mathbf{M}_k^{-1}\mathbf{a}}{\mathbf{a}^T\mathbf{S}\mathbf{a}}$, we have

$$\max_{\mathbf{u}}\mathscr{L}(\mathbf{M}_k)-\mathscr{L}(\mathbf{M}_k+\mathbf{u}\mathbf{u}^T)=\max_c\log c+\frac{1}{c}-1, \quad (20)$$

where $q(c)=\log c+\frac{1}{c}-1$. Then we get $\frac{\partial q}{\partial c}=\frac{1}{c}-\frac{1}{c^2}$. It is easy to see that the function $q(c)$ is monotonically increasing when $c > 1$, because $\frac{\partial q}{\partial c}>0$. Moreover, since $q(1) = 0$, we have $q(c)$

>0 for any $c>1$. Fig. 3(b) plots the curve of $q(c)$ for $c>1$. Therefore, we can get that $\mathcal{L}(\mathbf{M}_k + \mathbf{u}\mathbf{u}^T) < \mathcal{L}(\mathbf{M}_k)$.

When $c_1 \leq c_2$, based on Eq. (19), we have $\mathbf{u} = a\mathbf{a} = \mathbf{0}$ and hence $\mathcal{L}(\mathbf{M}_k) - \mathcal{L}(\mathbf{M}_k + \mathbf{u}\mathbf{u}^T) = 0$, which implies that adding an additional rank-one component is not helpful to decrease the NLL. If this happens, the greedy algorithm stops.

D. PROOF OF PROPOSITION 2

Proof

When condition (9) is satisfied, combing Eqs. (10), (19) and (20), we have

$$\mathscr{L}(\mathbf{M}_k) - \mathscr{L}(\mathbf{M}_k + \mathbf{u}\mathbf{u}^T) = q(c) = \log c + \frac{1}{c} - 1.$$

Based on the proof of Proposition 1, $q(c)$ is a monotonically increasing function w.r.t. c for $c>1$.

E. PROOF OF PROPOSITION 3

Proof

According to Eq. (20), the decrease in the NLL becomes faster if c is larger, since $q(c)$ is monotonically increasing when $c>1$. Therefore, based on the analysis on the Rayleigh quotient problem (12), we reach the conclusion.

F. PROOF OF THEOREM 3

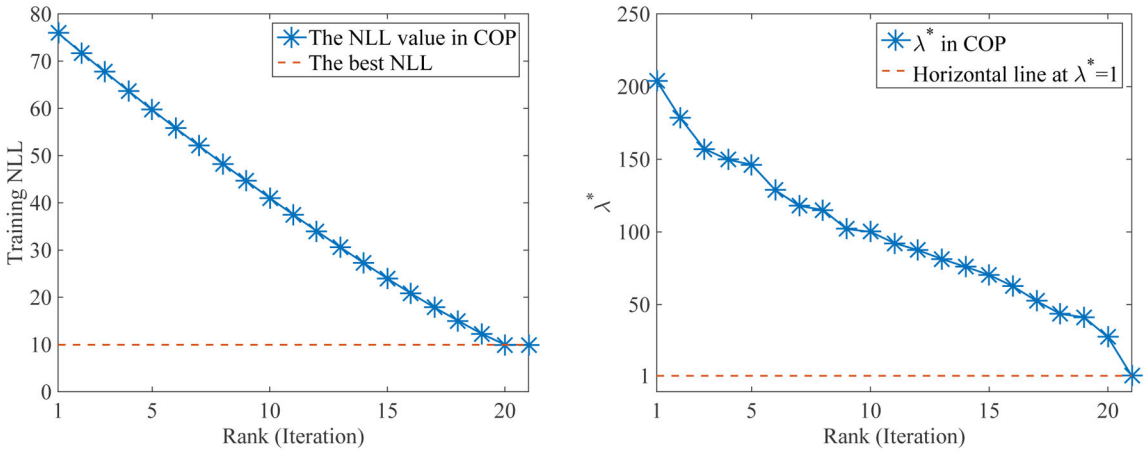
Proof

Based on propositions 1–3, we only need to check whether $\mathbf{u}^* = a^*\mathbf{a}^*$ is a local optimum of problem (7). Since $\nabla f(\mathbf{u}^*) = 0$ and $\nabla^2_{\mathbf{u}} f(\mathbf{u}^*) \succcurlyeq 0$, $\mathbf{u}^*$ is a local optimum and we reach the conclusion.

G. PROOF OF THEOREM 4

Proof

Proposition 1 implies that adding a rank-one component will lead to a lower NLL in the current iteration, if the component vector **a** satisfies Eq. (9). Moreover, after adding a rank-one component, updating the diagonal part **P** solves a convex function w.r.t. **P**, and therefore the NLL will not increase after the updating. So the NLL is guaranteed to decrease during iterations in Algorithm 1 until there is no vector satisfying Eq. (9) and then the algorithm converges.

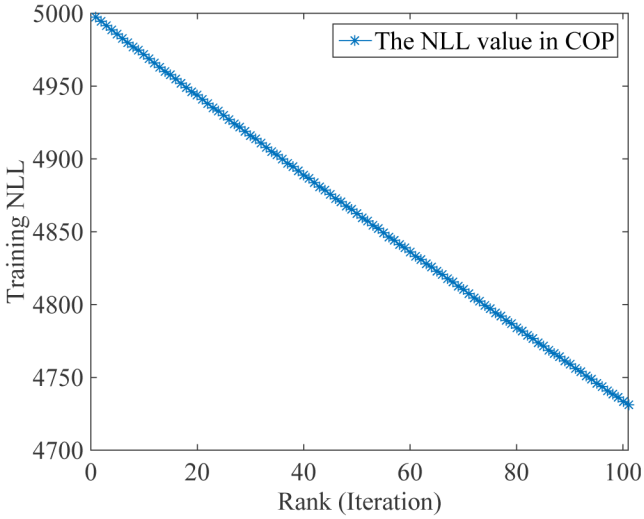


(a) The NLL for training v.s. iteration.

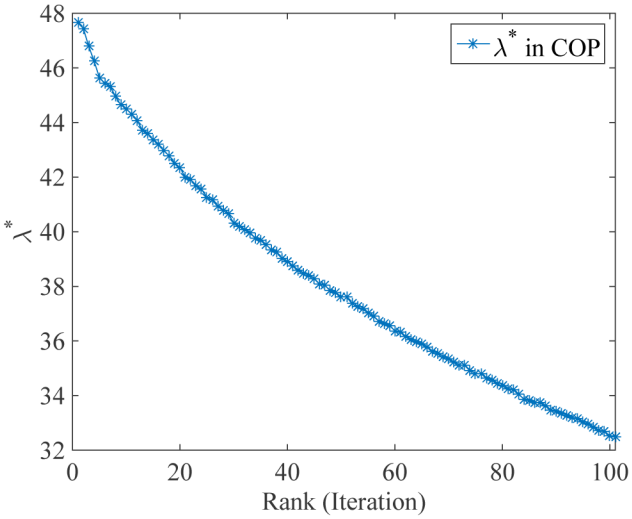
(b) λ^* v.s. iteration.

Figure 1.

The detailed results of the COP method on a small-scale synthetic data.

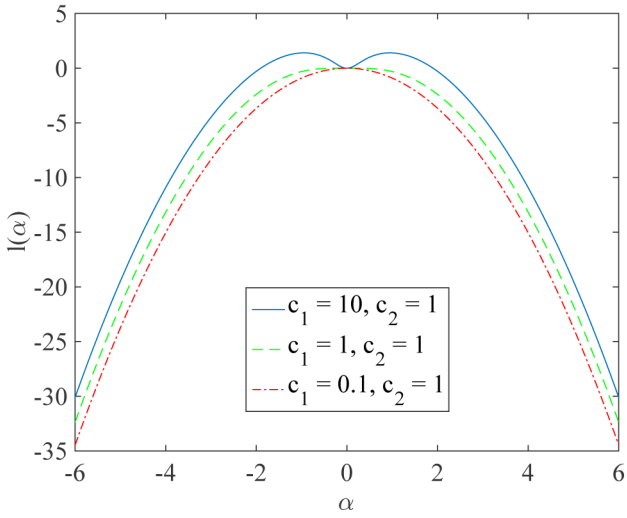


(a) The NLL for training v.s. iteration.

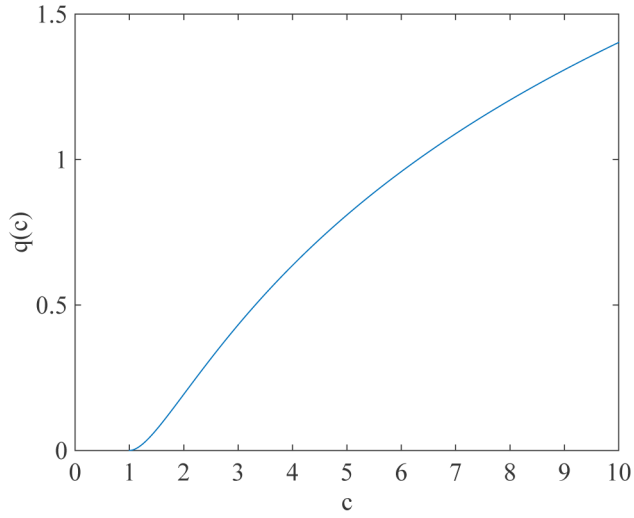


(b) λ^* v.s. iteration.

Figure 2.
Detailed results on synthetic data when $p = 5,000$.



(a) The curve of $l(\alpha)$



(b) The curve of $q(c)$ for $c > 1$

Figure 3.
Illustrations of two functions.

Table 1

Comparison of the running time (in seconds) between the sparse inverse covariance estimation methods, i.e., QUIC, Big-QUIC, BCDIC, and the proposed methods, i.e., COP and COP-ℓ_1, on synthetic data. The detailed settings for various methods are reported, including the setting of ρ, the number of non-zeros (NNZ), the rank r, the parameter γ, and the NLL on the test data denoted by NLL_{te}.

Data	ρ	NNZ	NLL_{te}	QUIC	Big-QUIC	BCDIC	r	NLL_{te}	COP	r	γ	NLL_{te}	COP-ℓ_1
$p = 5,000$	0.218	51,296	4967.5	79.8	64.9	50.7	11	4965.3	**15.0**	14	10^{-5}	4965.9	26.1
$p = 10,000$	0.231	103,422	9959.3	271.8	234.8	191.4	16	9959.2	**87.9**	21	10^{-5}	9959.4	136.7
$p = 15,000$	0.239	153,454	14966.4	798.0	622.7	404.3	16	14966.0	**200.9**	20	10^{-5}	14966.2	312.5
$p = 20,000$	0.244	207,306	19970.1	1821.6	1349.0	696.3	17	19969.2	**376.8**	22	10^{-5}	19968.9	584.3
$p = 25,000$	0.248	258,850	24980.3	4709.2	3828.2	1087.3	12	24980.1	**401.0**	15	10^{-5}	24980.5	560.8

Table 2

Comparison of the running time (in seconds) between the QUIC&Dirty method and the proposed methods, i.e. COP and COP-ℓ_1, on synthetic data. The detailed settings for various methods are reported, including the setting of ρ_1 and ρ_2, the rank r, the parameter γ, and the NLL on the test data denoted by NLL_{te}.

Data	ρ_1	ρ_2	NLL_{te}	QUIC&Dirty	r	NLL_{te}	COP	r	γ	NLL_{te}	COP-ℓ_1
$p = 5,000$	0.218	1	4953.8	881.9	16	4951.9	**23.4**	21	10^{-5}	4942.3	40.3
$p = 10,000$	0.231	1	9945.8	2795.0	24	9945.1	**132.3**	30	10^{-5}	9946.1	197.5
$p = 15,000$	0.239	1	14941.3	8527.9	33	14940.9	**413.7**	41	10^{-5}	14941.5	645.6
$p = 20,000$	0.244	–	–	–	50	19910.6	**1121.1**	66	10^{-5}	19911.7	1757.7
$p = 25,000$	0.248	–	–	–	50	24918.0	**1423.0**	68	10^{-5}	24917.6	2202.9

'–' indicates that the QUIC&Dirty method does not return a result after running over 5 hours for all choices of $\rho_2 \in \{0.1, 1, 10\}$.

Table 3

Comparison of the running time (in seconds) for the QUIC, Big-QUIC, BCDIC and QUIC&Dirty methods on real-world data. The NNZ's for the QUIC, Big-QUIC and BCDIC methods are around $10p$ by choosing their regularization parameter ρ.

Data	p	ρ	NNZ	NLL_{te}	QUIC	Big-QUIC	BCDIC	ρ_1	ρ_2	NLL_{te}	QUIC&Dirty
Snow	9, 148	0.982	108,246	10639.1	282.3	243.5	72.0	0.982	10	10439.0	3174.2
Air	10, 512	0.975	105,636	12250.3	660.5	539.4	129.1	0.975	10	11965.7	7060.8
Precip	13, 610	0.820	132,200	13992.1	1680.9	1213.6	483.0	0.820	–	–	–
OLR	21, 720	0.988	223,936	25647.9	3650.2	2989.0	544.2	0.988	–	–	–
Stock	21, 602	0.985	221,053	24814.7	3990.6	3212.7	634.1	0.985	–	–	–

'–' indicates that the corresponding methods do not return a result after running over 5 hours for all choices of $\rho_2 \in \{0.1, 1, 10\}$.

Table 4

Comparison of the running time (in seconds) for the COP and COP-ℓ_1 methods on real-world data.

Data	p	r	NLL_{te}	COP	r	γ	NLL_{te}	COP-ℓ_1
Snow	9, 148	5	**9162.3**	**20.7**	6	10^{-5}	9167.9	30.2
Air	10, 512	5	**10635.2**	**28.2**	6	10^{-5}	10651.6	41.5
Precip	13, 610	5	**13901.5**	**50.6**	6	10^{-5}	13921.9	65.7
OLR	21, 720	5	21754.0	**125.1**	5	10^{-4}	**21753.9**	148.5
Stock	21, 602	5	**21632.6**	122.8	4	10^{-4}	21652.1	**120.3**

Table 5

Comparison of the running time (in seconds) for the QUIC, Big-QUIC and BCDIC methods on the Snow, Air, OLR and Stock datasets when decreasing ρ to obtain more non-zero elements.

Data	ρ	NNZ	NLL_{te}	QUIC	Big-QUIC	BCDIC
Snow	0.950	568,402 ($\approx 50p$)	9404.0	1604.3	1253.6	429.8
Air	0.910	406,888 ($\approx 40p$)	10646.7	2117.0	1542.3	594.9
OLR	0.950	940,100 ($\approx 45p$)	24180.9	15557.2	13725.0	3248.2
Stock	0.950	928,147 ($\approx 45p$)	23990.6	17209.1	14632.4	3785.5