EiGLasso: Scalable Estimation of Cartesian Product of Sparse Inverse Covariance Matrices

Jun Ho Yoon
Computational Biology Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Seyoung Kim
Computational Biology Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we address the problem of jointly modeling dependencies across samples and across multiple features, where each set of dependencies is modeled as an inverse covariance matrix. In particular, we study a matrix Gaussian distribution with the Kronecker sum of sample-wise and feature-wise inverse covariances. While the Kronecker sum has been used as an intuitively more appealing convex alternative to the Kronecker product of two inverse covariances, the existing methods do not scale to large datasets. We introduce EiGLasso, a highly efficient optimization method for estimating the Kronecker-sum-structured inverse covariance matrix from matrix-variate data. We describe an alternative simpler approach for handling the non-identifiability of parameters than the one used in previous work. Using simulated and real data, we show our approach leads to one or two orders of magnitude speedup.

1 INTRODUCTION

One common assumption in statistical models for multivariate data is that samples, each consisting of multiple and often high-dimensional features, are drawn identically and independently from a multivariate distribution. These models typically focus on modeling dependencies across features, assuming samples are independent of each other. However, often samples as well as features are correlated. For example, in tumor expression data collected from many cancer patients for tens of thousands of genes, genes in the same pathways have correlated expression patterns and patients with the same or similar subtypes of cancer exhibit correlated gene expression patterns (Dai et al., 2015). Another example can be found in multivariate time-series data such as temporally correlated stock prices for related companies (King, 1966) or a sequence of images in video data (Kalchbrenner et al., 2017). In order to fully explain the correlation structure in matrix data, it is necessary to model row-wise correlation across samples and column-wise correlation across features jointly.

The matrix Gaussian distribution has been widely used for jointly modeling row-wise and column-wise dependencies in data, as it is a natural generalization of multivariate Gaussian distribution to matrix-variate data (Dawid, 1981; Dutilleul, 1999; Gupta and Nagar, 1999). It uses the Kronecker product of two inverse covariance matrices, each corresponding to graphs for sample-wise and feature-wise dependencies, as the inverse covariance of multivariate Gaussian distribution (Leng and Tang, 2012; Tsiligkaridis and Hero, 2013; Zhou, 2014). The data log-likelihood based on this model is not jointly convex with respect to the two inverse covariance parameters but bi-convex. Thus, a flip-flop style optimization has been used to estimate the row-wise parameter fixing the column-wise parameter and vice versa until convergence.

More recently, a different matrix Gaussian distribution based on the Kronecker sum of row and column graphs has been introduced as a sparse and convex substitute for Kronecker product (Kalaitzis et al., 2013; Greenewald et al., 2019). The Kronecker sum of two graphs has an intuitively more appealing structure than the Kronecker-product counterpart, as it represents a sparse Cartesian product of the two graphs (Hammack et al., 2011). BiGLasso (Kalaitzis et al., 2013) estimated this model in a flip-flop manner using GLasso (Friedman et al., 2007) as a subroutine and alternately optimized for one of the two graphs fixing the other until convergence. TeraLasso (Greenewald et al., 2019) significantly improved the computation time of BiGLasso using a gradient-based method (Beck and Teboulle, 2009); however, it is still significantly slower than the state-of-the-art optimization methods such as QUIC (Hsieh et al., 2014) and

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

BIG&QUIC (Hsieh et al., 2013) for a Gaussian graphical model for a single graph. One of the key challenges in the estimation of this model arises from the non-identifiability of the diagonal elements of row and column graphs. While BiGLasso did not estimate these diagonal elements, TeraLasso proposed a strategy for estimating them through identifiable reparameterization.

Here, we introduce a new optimization method, called eigen graphical Lasso (EiGLasso), for estimating the matrix Gaussian distribution with a Kronecker-sum-structured inverse covariance matrix. EiGLasso improves upon TeraLasso with a significantly more efficient optimization method and with a simpler strategy for estimating the non-identifiable diagonal elements of row and column graphs. Toward efficient optimization, EiGLasso combines the simplicity of the flip-flop style optimization of the Kronecker-product models and the efficiency of QUIC for estimating sparse Gaussian graphical models (Hsieh et al., 2014). EiGLasso simplifies each flip-flop optimization problem by exploiting the inflated structure in the Cartesian product of two graphs and further reduces computation time by exploiting the eigen-structure of the gradient and Hessian of the log-likelihood function. This leads to an algorithm that is significantly more efficient than BiGLasso and TeraLasso.

In addition, we provide results that shed new insights into the non-identifiability of the diagonal elements of row and column graphs in Kronecker-sum model. While TeraLasso introduced an identifiable reparameterization and projected the gradient into this reparameterized space in each iteration, in EiGLasso we show it is sufficient to use a far simpler strategy of adjusting the diagonal elements to an identifiable representation once after convergence, a strategy similar to the one used for Kronecker-product models (Yin and Li, 2012). Because the non-identifiability of these parameters does not affect the gradient and Hessian required for optimization, EiGLasso proceeds without being concerned with the non-identifiability during estimation. In our experiments, we demonstrate EiGLasso has one or two orders-of-magnitude speed-up compared to TeraLasso across problems of different sizes and recovers the true parameters with higher accuracy than TeraLasso.

2 RELATED WORK

2.1 KRONECKER-PRODUCT INVERSE COVARIANCE

The matrix Gaussian distribution generalizes a multivariate Gaussian distribution. It models the distribution of a $q \times p$ matrix random variable $\boldsymbol{Y} \sim \mathcal{MN}(\boldsymbol{M}, \boldsymbol{\Psi}, \boldsymbol{\Theta})$, where $\boldsymbol{M}$ is a $q \times p$ mean, $\boldsymbol{\Theta}$ is a $p \times p$ positive definite matrix modeling column-wise dependencies in $\boldsymbol{Y}$, and $\boldsymbol{\Psi}$ is a $q \times q$ positive definite matrices representing row-wise dependencies in $\boldsymbol{Y}$. This distribution can be also written as multivariate Gaussian with Kronecker product:

$$\mathrm{vec}(\boldsymbol{Y}) \sim \mathcal{N}\left(\mathrm{vec}(\boldsymbol{M}), (\boldsymbol{\Theta} \otimes \boldsymbol{\Psi})^{-1}\right),$$

where $\otimes$ is the Kronecker-product operator, $\mathrm{vec}(\cdot)$ is the *vec*-operator that stacks the columns of a matrix into a vector. Throughout the paper we assume zero mean as we focus on the covariance structure.

Maximum likelihood estimation has been widely used to estimate the parameters $\boldsymbol{\Psi}$ and $\boldsymbol{\Theta}$ of this model. The data log-likelihood is not jointly convex in $\boldsymbol{\Psi}$ and $\boldsymbol{\Theta}$ but bi-convex with respect to one parameter given the other. The log-likelihood is maximized in a flip-flop fashion for $\boldsymbol{\Psi}$ given $\boldsymbol{\Theta}$ and for $\boldsymbol{\Theta}$ given $\boldsymbol{\Psi}$ in each iteration until convergence. The parameters are not fully identifiable as $\boldsymbol{\Theta} \otimes \boldsymbol{\Psi} = (c\boldsymbol{\Theta}) \otimes (\frac{1}{c}\boldsymbol{\Psi})$ for any positive constant c. Thus, after convergence, $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ are rescaled as $\boldsymbol{\Theta} \leftarrow c\boldsymbol{\Theta}$ and $\boldsymbol{\Psi} \leftarrow \frac{1}{c}\boldsymbol{\Psi}$ with some constant c such that the (1,1) element $\boldsymbol{\Theta}_{11}$ of $\boldsymbol{\Theta}$ is equal to 1 (Yin and Li, 2012).

2.2 KRONECKER-SUM INVERSE COVARIANCE

A Gaussian distribution for a matrix variable with Kronecker-sum inverse covariance has been proposed to model a sparse lattice structure in the Cartesian product of two graphs $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ (Kalaitzis et al., 2013). In the Kronecker-product model, even when $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ are sparse, $\boldsymbol{\Theta} \otimes \boldsymbol{\Psi}$ is often dense; for example, an edge $\boldsymbol{\Psi}_{ij}$ in graph $\boldsymbol{\Psi}$ induces edges in $\boldsymbol{\Theta} \otimes \boldsymbol{\Psi}$ between the same two column nodes even if they arise from different row nodes in graph $\boldsymbol{\Theta}$. In contrast, the Kronecker sum of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$, defined as

$$\boldsymbol{\Theta} \oplus \boldsymbol{\Psi} \triangleq \boldsymbol{\Theta} \otimes \boldsymbol{I}_q + \boldsymbol{I}_p \otimes \boldsymbol{\Psi},$$

results in a network over pq nodes in the Cartesian product of the two graphs, where an edge $\boldsymbol{\Psi}_{ij}$ in $\boldsymbol{\Psi}$ induces edges in $\boldsymbol{\Theta} \oplus \boldsymbol{\Psi}$ only between the same two column nodes arising from the same row node in $\boldsymbol{\Theta}$. The Gaussian distribution with Kronecker-sum-structured inverse covariance is given as

$$\mathrm{vec}(\boldsymbol{Y}) \sim \mathcal{N}\left(\mathrm{vec}(\boldsymbol{M}), (\boldsymbol{\Theta} \oplus \boldsymbol{\Psi})^{-1}\right). \quad (1)$$

A generalization of this model into the Kronecker sum of M inverse covariance matrices for $M \geq 2$ has been discussed (see Section 1.2 in Greenewald et al. (2019)).

BiGLasso (Kalaitzis et al., 2013) and TeraLasso (Greenewald et al., 2019) have considered the problem of obtaining a sparse estimate of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ in Eq. (1) by minimizing the ℓ_1-regularized negative log-likelihood of data. Given data $\{\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^n\}$, where $\boldsymbol{Y}^i \in \mathbb{R}^{q \times p}$ for

$i = 1, \ldots, n$ for n observations, BiGLasso and TeraLasso solve the following convex optimization problem:

$$\underset{\boldsymbol{\Theta} \oplus \boldsymbol{\Psi} \succ 0}{\operatorname{argmin}} g(\boldsymbol{\Theta}, \boldsymbol{\Psi}) + h_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}) + h_{\boldsymbol{\Psi}}(\boldsymbol{\Psi}), \tag{2}$$

where

$$g(\boldsymbol{\Theta}, \boldsymbol{\Psi}) = q \operatorname{tr}(\boldsymbol{S}\boldsymbol{\Theta}) + p \operatorname{tr}(\boldsymbol{T}\boldsymbol{\Psi}) - \log |\boldsymbol{\Theta} \oplus \boldsymbol{\Psi}|$$

$$h_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}) = q\gamma_{\boldsymbol{\Theta}} \|\boldsymbol{\Theta}\|_{1,\text{off}}, \quad h_{\boldsymbol{\Psi}}(\boldsymbol{\Psi}) = p\gamma_{\boldsymbol{\Psi}} \|\boldsymbol{\Psi}\|_{1,\text{off}},$$

given the sample covariances $\boldsymbol{S} \triangleq \frac{1}{nq} \sum_{i=1}^{n} \boldsymbol{Y}^{iT} \boldsymbol{Y}^{i}$ and $\boldsymbol{T} \triangleq \frac{1}{np} \sum_{i=1}^{n} \boldsymbol{Y}^{i} \boldsymbol{Y}^{iT}$.

One of the key challenges in estimating this model stems from the non-identifiability of diagonals of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$: for an arbitrary constant c we have

$$\boldsymbol{\Theta} \oplus \boldsymbol{\Psi} = (\boldsymbol{\Theta} - c\boldsymbol{I}_p) \oplus (\boldsymbol{\Psi} + c\boldsymbol{I}_q).$$

BiGLasso did not estimate the diagonals of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$. TeraLasso formed an identifiable representation

$$\bar{\boldsymbol{\Theta}} \oplus \bar{\boldsymbol{\Psi}} + \tau \boldsymbol{I}_{pq}, \tag{3}$$

where $\tau = \frac{\operatorname{tr}(\boldsymbol{\Theta} \oplus \boldsymbol{\Psi})}{pq}$, $\bar{\boldsymbol{\Theta}} = \boldsymbol{\Theta} - \frac{\operatorname{tr}(\boldsymbol{\Theta})}{p} I_p$, $\bar{\boldsymbol{\Psi}} = \boldsymbol{\Psi} - \frac{\operatorname{tr}(\boldsymbol{\Psi})}{q} I_q$, enforcing the constraint $\operatorname{tr}(\bar{\boldsymbol{\Theta}}) = \operatorname{tr}(\bar{\boldsymbol{\Psi}}) = 0$. Then, during estimation with a gradient-based method, TeraLasso projected the gradient to this reparameterized Kronecker-sum space in each iteration. For efficient computation and projection of the gradient, TeraLasso eigendecomposed $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$: $\boldsymbol{\Theta} = \boldsymbol{Q}_{\boldsymbol{\Theta}} \boldsymbol{\Lambda}_{\boldsymbol{\Theta}} \boldsymbol{Q}_{\boldsymbol{\Theta}}^T$, where $\boldsymbol{Q}_{\boldsymbol{\Theta}} \in \mathbb{R}^{p \times p}$ is an orthonormal matrix whose ith column is the ith eigenvector of $\boldsymbol{\Theta}$ and $\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} \in \mathbb{R}^{p \times p}$ is a diagonal matrix of p eigenvalues, and similarly, $\boldsymbol{\Psi} = \boldsymbol{Q}_{\boldsymbol{\Psi}} \boldsymbol{\Lambda}_{\boldsymbol{\Psi}} \boldsymbol{Q}_{\boldsymbol{\Psi}}^T$. While TeraLasso is significantly faster than BiGLasso, it does not scale to graphs with more than a few hundred nodes. In the next section, we propose a significantly faster algorithm to solve Eq. (2) and a simpler approach to handling the non-identifiability of the diagonal elements of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$.

3 EiGLasso

We introduce EiGLasso for an efficient estimation of sparse $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ in Eq. (1). We perform a flip-flop, optimizing for $\boldsymbol{\Theta}$ fixing $\boldsymbol{\Psi}$ and optimizing for $\boldsymbol{\Psi}$ fixing $\boldsymbol{\Theta}$ until convergence. To estimate one network parameter given the other, we adopt the strategy used in QUIC for estimating sparse Gaussian graphical models. QUIC employed the Newton's method: in each Newton iteration, the Newton direction was found by minimizing the second-order approximation of the objective and the parameter was updated, based on this Newton direction and the step size found by line search. Applying the same strategy to estimate $\boldsymbol{\Theta}$ given $\boldsymbol{\Psi}$ in the Kronecker-sum model, we find the Newton direction by solving the following optimization problem:

$$D_{\boldsymbol{\Theta}} = \underset{\Delta_{\boldsymbol{\Theta}}}{\operatorname{argmin}} \hat{g}(\Delta_{\boldsymbol{\Theta}}) + h_{\boldsymbol{\Theta}}(\boldsymbol{\Theta} + \Delta_{\boldsymbol{\Theta}}), \tag{4}$$

where $\hat{g}(\Delta_{\boldsymbol{\Theta}})$ is the second-order approximation of $g(\boldsymbol{\Theta}, \boldsymbol{\Psi})$ in Eq. (2) with respect to $\boldsymbol{\Theta}$ fixing $\boldsymbol{\Psi}$

$$\hat{g}(\Delta_{\boldsymbol{\Theta}}) \triangleq \operatorname{tr}\left(\nabla_{\boldsymbol{\Theta}} g \Delta_{\boldsymbol{\Theta}}\right) + \frac{1}{2} \operatorname{vec}(\Delta_{\boldsymbol{\Theta}})^T \nabla^2_{\boldsymbol{\Theta}} g \operatorname{vec}(\Delta_{\boldsymbol{\Theta}})$$

and is given as

$$\begin{aligned} \hat{g}(\Delta_{\boldsymbol{\Theta}}) = {} & q \operatorname{tr}\left(\boldsymbol{S}\Delta_{\boldsymbol{\Theta}}\right) - \operatorname{tr}\left(\boldsymbol{W}(\Delta_{\boldsymbol{\Theta}} \otimes \boldsymbol{I}_q)\right) \\ & + \frac{1}{2} \operatorname{vec}\left(\Delta_{\boldsymbol{\Theta}} \otimes \boldsymbol{I}_q\right)^T \boldsymbol{W} \otimes \boldsymbol{W} \operatorname{vec}\left(\Delta_{\boldsymbol{\Theta}} \otimes \boldsymbol{I}_q\right), \end{aligned} \tag{5}$$

where $\boldsymbol{W} \triangleq (\boldsymbol{\Theta} \oplus \boldsymbol{\Psi})^{-1}$. Eq. (4) can be solved efficiently with coordinate descent. The same strategy can be used to solve the symmetric optimization problem of estimating $\boldsymbol{\Psi}$ given $\boldsymbol{\Theta}$.

The key challenges that arise in a direct application of QUIC to learn the Kronecker-sum model are 1) the need to compute the large matrix $\boldsymbol{W}$ of size $pq \times pq$ in the gradient, which would require prohibitively expensive computation time $\mathcal{O}(p^3q^3)$ to invert $\boldsymbol{\Theta} \oplus \boldsymbol{\Psi}$ and storage $\mathcal{O}(p^2q^2)$, 2) the need to handle the even larger matrix $\boldsymbol{W} \otimes \boldsymbol{W}$ of size $p^2q^2 \times p^2q^2$ in the Hessian, and 3) the need to deal with the non-identifiable diagonals of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$. Below we describe how we address each of these challenges and then present the full algorithm.

3.1 EFFICIENT COMPUTATION OF GRADIENT AND HESSIAN

Collapsed form of gradient and Hessian We re-write the second-order approximation in Eq. (5), which involves the gradient $\boldsymbol{W}$ of size $pq \times pq$ and Hessian $\boldsymbol{W} \otimes \boldsymbol{W}$ of size $p^2q^2 \times p^2q^2$, into a significantly more compact form that involves the gradient $\boldsymbol{W}_{\boldsymbol{\Theta}}$ of size $p \times p$ and Hessian $\boldsymbol{H}_{\boldsymbol{\Theta}}$ of size $p^2 \times p^2$ as follows:

$$\begin{aligned} \hat{g}(\Delta_{\boldsymbol{\Theta}}) = {} & q \operatorname{tr}\left(\boldsymbol{S}\Delta_{\boldsymbol{\Theta}}\right) - \operatorname{tr}\left(\boldsymbol{W}_{\boldsymbol{\Theta}}\Delta_{\boldsymbol{\Theta}}\right) \\ & + \frac{1}{2} \operatorname{vec}(\Delta_{\boldsymbol{\Theta}})^T \boldsymbol{H}_{\boldsymbol{\Theta}} \operatorname{vec}(\Delta_{\boldsymbol{\Theta}}). \end{aligned}$$

$\boldsymbol{W}_{\boldsymbol{\Theta}}$ and $\boldsymbol{H}_{\boldsymbol{\Theta}}$ above are given as

$$\boldsymbol{W}_{\boldsymbol{\Theta}} \triangleq \sum_{i=1}^{q} (\boldsymbol{I}_p \otimes \boldsymbol{e}_i)^T \boldsymbol{W} (\boldsymbol{I}_p \otimes \boldsymbol{e}_i) \tag{6a}$$

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{\Theta}} \triangleq \sum_{i=1}^{q} \sum_{j=1}^{q} \Big[& (\boldsymbol{I}_p \otimes \boldsymbol{e}_i \otimes \boldsymbol{I}_p \otimes \boldsymbol{e}_i)^T \\ & \boldsymbol{W} \otimes \boldsymbol{W} (\boldsymbol{I}_p \otimes \boldsymbol{e}_j \otimes \boldsymbol{I}_p \otimes \boldsymbol{e}_j) \Big], \end{aligned} \tag{6b}$$

where $\boldsymbol{e}_k$ is a vector of length q with 0's except for 1 in the kth element.

Furthermore, while a naive strategy would compute $\boldsymbol{W}_{\boldsymbol{\Theta}}$ and $\boldsymbol{H}_{\boldsymbol{\Theta}}$ by collapsing $\boldsymbol{W}$ and $\boldsymbol{W} \otimes \boldsymbol{W}$ as in Eq. (6), the following theorem states that these collapsed forms can be obtained without constructing $\boldsymbol{W}$ explicitly, using eigendecomposition of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ (proof in Appendix).

Theorem 1. *Given the eigendecomposition* $\boldsymbol{\Theta} = \boldsymbol{Q}_{\boldsymbol{\Theta}} \boldsymbol{\Lambda}_{\boldsymbol{\Theta}} \boldsymbol{Q}_{\boldsymbol{\Theta}}^T$ *and* $\boldsymbol{\Psi} = \boldsymbol{Q}_{\boldsymbol{\Psi}} \boldsymbol{\Lambda}_{\boldsymbol{\Psi}} \boldsymbol{Q}_{\boldsymbol{\Psi}}^T$*, the collapsed form of the gradient* $\boldsymbol{W}_{\boldsymbol{\Theta}}$ *and Hessian* $\boldsymbol{H}_{\boldsymbol{\Theta}}$ *in Eq. (6) can be computed as*

$$\boldsymbol{W}_{\boldsymbol{\Theta}} = \sum_{k=1}^{q} \boldsymbol{V}_k \;\; \textit{and} \;\; \boldsymbol{H}_{\boldsymbol{\Theta}} = \sum_{k=1}^{q} \boldsymbol{V}_k \otimes \boldsymbol{V}_k. \qquad (7)$$

where

$$\boldsymbol{V}_k = \boldsymbol{Q}_{\boldsymbol{\Theta}} \left(\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} + \lambda_{\boldsymbol{\Psi},k} \boldsymbol{I}_p \right)^{-1} \boldsymbol{Q}_{\boldsymbol{\Theta}}^T,$$

given $\lambda_{\boldsymbol{\Psi},k}$*, the* k*th eigenvalue of* $\boldsymbol{\Psi}$*.*

While TeraLasso also employed eigendecomposition of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$, they did so for a different purpose. In TeraLasso, this was done for an efficient projection of the gradient to the reparameterized space in each iteration. In contrast, EiGLasso makes use of the eigendecomposition for an efficient computation of the gradient and Hessian for $\boldsymbol{\Theta}$ given $\boldsymbol{\Psi}$ in the original space. This in turn reveals new insights into the eigen-structure of $\boldsymbol{W}_{\boldsymbol{\Theta}}$ and $\boldsymbol{H}_{\boldsymbol{\Theta}}$ that decomposes into sum over q components arising from the eigen-structure of $\boldsymbol{\Psi}$. In the next section, Theorem 1 is also used to show the identifiable parameterization needs to be considered only once at convergence, not in every iteration.

Efficient computation of gradient and approximate Hessian From Eq. (7), the gradient $\boldsymbol{W}_{\boldsymbol{\Theta}}$ can be computed efficiently as follows:

$$\boldsymbol{W}_{\boldsymbol{\Theta}} = \boldsymbol{Q}_{\boldsymbol{\Theta}} \left(\sum_{k=1}^{q} \left(\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} + \lambda_{\boldsymbol{\Psi},k} \boldsymbol{I}_p \right)^{-1} \right) \boldsymbol{Q}_{\boldsymbol{\Theta}}^T.$$

$\boldsymbol{W}_{\boldsymbol{\Theta}}$ can be pre-computed and stored at the beginning of the coordinate descent optimization to solve Eq. (4); thus, it is not necessary to maintain all $\boldsymbol{V}_k$'s during the coordinate descent updates.

However, for the Hessian $\boldsymbol{H}_{\boldsymbol{\Theta}}$ in Eq. (7), for large p, explicitly precomputing and storing $p^2 \times p^2$ matrix $\boldsymbol{H}_{\boldsymbol{\Theta}}$ is still expensive in terms of memory. An alternative approach of pre-computing and storing $\boldsymbol{V}_k$'s for $k = 1, \ldots, q$ and computing the elements of $\boldsymbol{H}_{\boldsymbol{\Theta}}$ as needed during the coordinate descent optimization in Eq. (4) would be also prohibitively expensive in memory as storing $\boldsymbol{V}_k$'s would require space $\mathcal{O}(p^2 q)$. Hence, we propose to approximate $\boldsymbol{H}_{\boldsymbol{\Theta}}$ based on the observation that the smallest eigenvalues of $\boldsymbol{\Psi}$ contribute the most to the eigenvalues of $\boldsymbol{H}_{\boldsymbol{\Theta}}$, $(\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} + \lambda_{\boldsymbol{\Psi}} \boldsymbol{I}_p)^{-1} \otimes (\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} + \lambda_{\boldsymbol{\Psi}} \boldsymbol{I}_p)^{-1}$, as can be seen in Eq. (7). Specifically, we sort the eigenvalues of $\boldsymbol{\Psi}$ in ascending order and use only $\{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_K\}$, replacing the remaining $q - K$ components with $\boldsymbol{V}_{K+1}$:

$$\hat{\boldsymbol{H}}_{\boldsymbol{\Theta}} = \sum_{k=1}^{K} \boldsymbol{V}_k \otimes \boldsymbol{V}_k + (q - K) \boldsymbol{V}_{K+1} \otimes \boldsymbol{V}_{K+1}. \qquad (8)$$

Note that dropping $(q - K)$ eigenvalues and eigenvectors in $\boldsymbol{H}_{\boldsymbol{\Theta}}$ would amount to assuming that $\boldsymbol{\Psi}$ has $(q - K)$ eigenvalues of infinite magnitude. Thus, instead of dropping these, we replace them with $\boldsymbol{V}_{K+1}$ in Eq. (8). We demonstrate with simulations in Section 4 that often $K = 1$ suffices.

Algorithm 1: EiGLasso

input : Inputs $\boldsymbol{Y} \in \mathbb{R}^{q \times p}$ and regularization parameters $\gamma_{\boldsymbol{\Theta}}, \gamma_{\boldsymbol{\Psi}}$

output : Parameters $\boldsymbol{\Theta}, \boldsymbol{\Psi}$

Initialize $\boldsymbol{\Theta} \leftarrow \boldsymbol{I}_p, \boldsymbol{\Psi} \leftarrow \boldsymbol{I}_q$

for $t = 0, 1, \ldots$ **do**

- *# Estimate* $\boldsymbol{\Theta}$:
- Determine active sets $\mathcal{S}_{\boldsymbol{\Theta}} = \{(i,j) \mid |\nabla_{\boldsymbol{\Theta}_{ij}} g| > q\gamma_{\boldsymbol{\Theta}} \text{ or } \boldsymbol{\Theta}_{ij} \neq 0\}$.
- Compute $D_{\boldsymbol{\Theta}}$ via coordinate descent: $D_{\boldsymbol{\Theta}} = \arg\min_{\Delta_{\boldsymbol{\Theta}}} \hat{g}(\Delta_{\boldsymbol{\Theta}}) + q\gamma_{\boldsymbol{\Theta}} ||\boldsymbol{\Theta} + \Delta_{\boldsymbol{\Theta}}||_{1,\text{off}}$ over $(i,j) \in \mathcal{S}_{\boldsymbol{\Theta}}$.
- Use *Armijo*-rule to compute a step-size $\alpha_{\boldsymbol{\Theta}}$ such that $\boldsymbol{\Theta}^{t+1} \oplus \boldsymbol{\Psi} \succ 0$, where $\boldsymbol{\Theta}^{t+1} = \boldsymbol{\Theta}_t + \alpha_{\boldsymbol{\Theta}} D_{\boldsymbol{\Theta}}$ and update $\boldsymbol{\Theta}^{t+1}$.
- Eigendecompose $\boldsymbol{\Theta}^{t+1}$ and compute the gradient and Hessian for $\boldsymbol{\Psi}$.
- *# Estimate* $\boldsymbol{\Psi}$:
- Proceed as if estimating $\boldsymbol{\Theta}$ but symmetrically.
- Check convergence.

Adjust the diagonal elements of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ according to the preset trace ratio.

3.2 HANDLING THE NON-IDENTIFIABILITY

We describe a simple approach to making the diagonal elements of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ identifiable. We show it is sufficient to identify the diagonals once after the convergence, unlike in TeraLasso that identifies the diagonals in every iteration during optimization.

We claim that assuming a fixed trace ratio $\text{tr}(\boldsymbol{\Psi}) / \text{tr}(\boldsymbol{\Theta}) = r$ is sufficient to make the diagonals identifiable. We make use of the following fact:

$$\text{tr}(\boldsymbol{\Theta} \oplus \boldsymbol{\Psi}) = q \, \text{tr}(\boldsymbol{\Theta}) + p \, \text{tr}(\boldsymbol{\Psi}), \qquad (9)$$

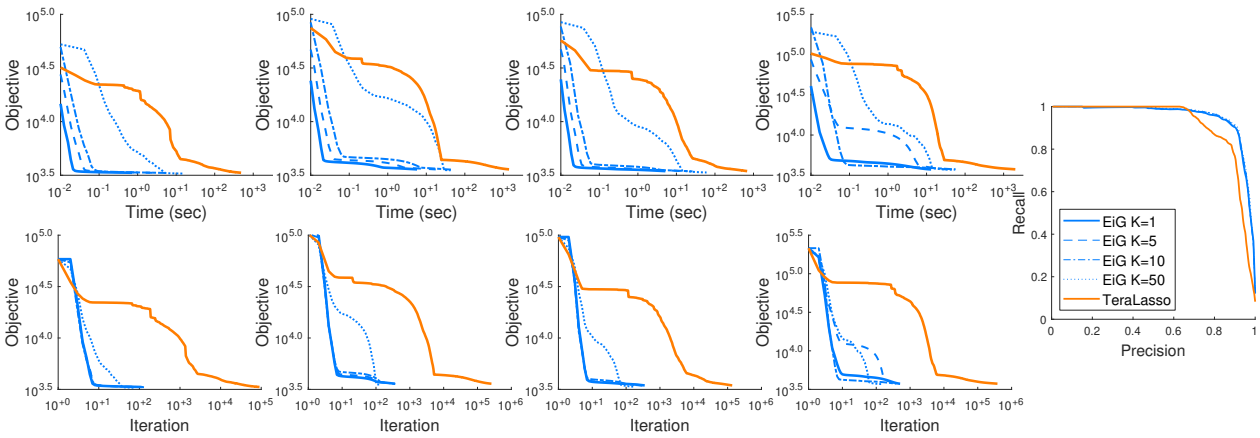

Figure 1: Comparison of EiGLasso and TeraLasso on simulated data with random graphs of size 50×50. Left: objective values over time (top) and over iterations (bottom) for data simulated from four random graphs (four columns). Right: precision-recall curves for the sparse structure recovery of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$ averaged over 10 simulated datasets. In EiGLasso, K eigenvalues were used to approximate Hessian.

which can be directly verified by plugging in the definition of Kronecker sum to the left-hand side of the above equation. Eq. (9) suggests we can distribute $\text{tr}(\boldsymbol{\Theta} \oplus \boldsymbol{\Psi})$ to $\text{tr}(\boldsymbol{\Theta})$ and $\text{tr}(\boldsymbol{\Psi})$ such that $\text{tr}(\boldsymbol{\Psi})/\,\text{tr}(\boldsymbol{\Theta}) = r$ to identify the diagonals of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$. Given the estimates $\boldsymbol{\Theta}^t$ and $\boldsymbol{\Psi}^t$ at iteration t, we adjust the diagonals

$$\boldsymbol{\Theta}^t \leftarrow \boldsymbol{\Theta}^t + c_t \boldsymbol{I}_p \quad \text{and} \quad \boldsymbol{\Psi}^t \leftarrow \boldsymbol{\Psi}^t - c_t \boldsymbol{I}_q, \tag{10}$$

where $c_t = \frac{\text{tr}(\boldsymbol{\Psi}^t) - r\,\text{tr}(\boldsymbol{\Theta}^t)}{q + rp}$ is obtained by solving $\text{tr}(\boldsymbol{\Psi}^t - c_t \boldsymbol{I}_q)/\,\text{tr}(\boldsymbol{\Theta}^t + c_t \boldsymbol{I}_p) = r$ for c_t.

In the special case with $r = q/p$, our strategy above reduces to the identifiable reparameterization used in TeraLasso. At the end of each iteration t, TeraLasso applies the reparameterization in Eq. (3) and evenly distributes $\tau_t = \frac{\text{tr}(\boldsymbol{\Theta}^t \oplus \boldsymbol{\Psi}^t)}{pq}$ in Eq. (3) across the two parameters:

$$\begin{aligned} \boldsymbol{\Theta}^t &\leftarrow \boldsymbol{\Theta}^t - \frac{\text{tr}(\boldsymbol{\Theta}^t)}{p} \boldsymbol{I}_p + \frac{\tau_t}{2} \boldsymbol{I}_p \\ &= \boldsymbol{\Theta}^t + c_t \boldsymbol{I}_p \qquad \text{from Eq. (9)} \end{aligned}$$

and similarly for $\boldsymbol{\Psi}^t$. These updates are identical to our adjustment in Eq. (10).

The adjustments made to the diagonal elements of the parameters in Eq. (10) leave the gradient and Hessian in Eqs. (7) and (8) unchanged. Eq. (10) leaves the eigenvectors of $\boldsymbol{\Theta}^t$ and $\boldsymbol{\Psi}^t$ the same but changes the eigenvalues $\boldsymbol{\Lambda}_{\boldsymbol{\Theta}^t} \leftarrow \boldsymbol{\Lambda}_{\boldsymbol{\Theta}^t} + c_t \boldsymbol{I}$ for $\boldsymbol{\Theta}^t$ and $\boldsymbol{\Lambda}_{\boldsymbol{\Psi}^t} \leftarrow \boldsymbol{\Lambda}_{\boldsymbol{\Psi}^t} - c_t \boldsymbol{I}$ for $\boldsymbol{\Psi}^t$. However, this change in the eigenvalues does not affect the eigenvalues of $\boldsymbol{V}_k$ in the gradient and Hessian in Eq. (7), as can be seen from

$$\begin{aligned} &((\boldsymbol{\Lambda}_{\boldsymbol{\Theta}^t} + c_t \boldsymbol{I}_p) + (\lambda_{\boldsymbol{\Psi}^t,k} - c_t) \boldsymbol{I}_p)^{-1} \\ &= (\boldsymbol{\Lambda}_{\boldsymbol{\Theta}^t} + \lambda_{\boldsymbol{\Psi}^t,k} \boldsymbol{I}_p)^{-1}. \end{aligned}$$

Since the update in Eq. (10) does not affect the gradient and Hessian, it follows that the update in Eq. (10) affects neither the Newton direction found in Eq. (4) nor the step-size found from line search. Thus, unlike in TeraLasso, it is sufficient to perform the adjusment in Eq. (9) only once, after the convergence in EiGLasso.

3.3 THE FULL ALGORITHM

Putting together the time- and memory-efficient strategies for obtaining the gradient and Hessian and the strategies for handling non-identifiability as described in the previous sections, we provide the pseudocode for EiGLasso in Algorithm 1. We adopt other strategies in QUIC for improving computation time, such as updating only the parameters in the active set during the coordinate descent optimization. The details of the coordinate descent update are discussed in Appendix.

4 EXPERIMENTS

We compared the performance of our EiGLasso with that of TeraLasso (Greenewald et al., 2019) on simulated and real data. We implemented EiGLasso in C++ with the sequential version of Intel Math Kernel Library. We downloaded the authors' implementation of TeraLasso and modified it to perform more iterations during line search, when the safe-step approach suggested by the authors failed to find a step-size that satisfies the positive definite condition on $\boldsymbol{\Theta} \oplus \boldsymbol{\Psi}$. All experiments were run on a single core of Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz.

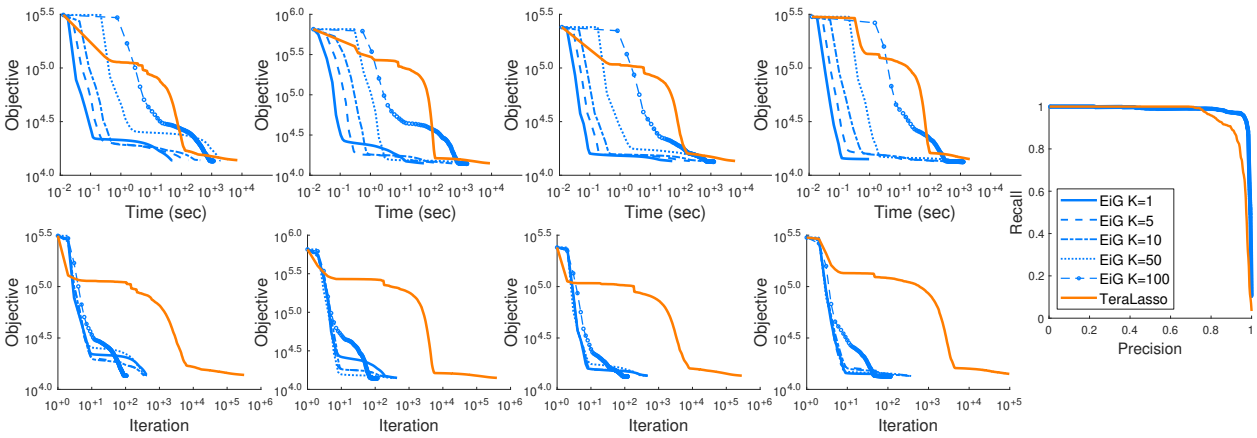

Figure 2: Comparison of EiGLasso and TeraLasso on data simulated from random graphs of size 100×100. Left: objective values over time (top) and over iterations (bottom) for four simulated datasets (four columns). Right: precision-recall curves for the sparse structure recovery of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ averaged over 10 simulated datasets. In EiGLasso, K eigenvalues were used to approximate Hessian.

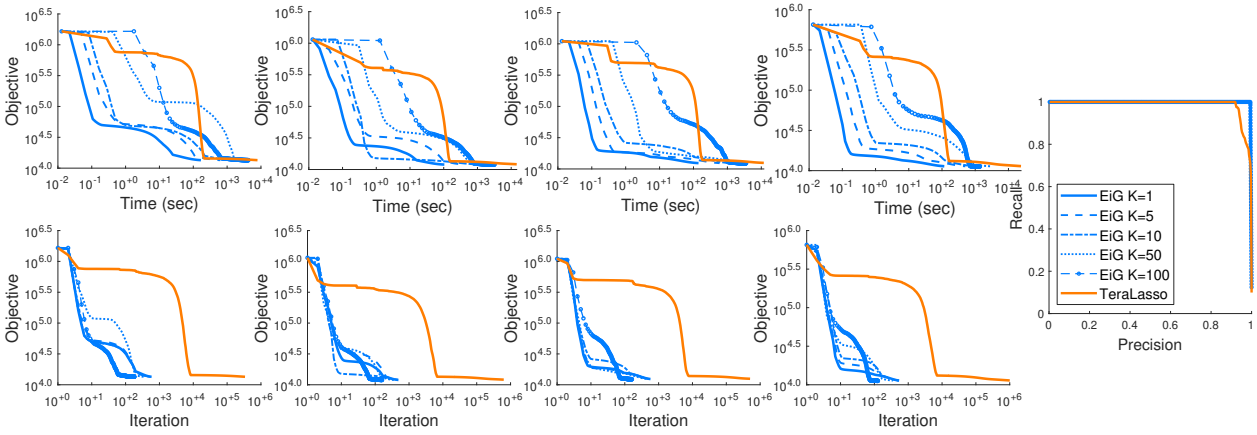

Figure 3: Comparison of EiGLasso and TeraLasso on data simulated from block graphs of size 100×100. Left: objective values over time (top) and over iterations (bottom) for four simulated datasets (four columns). Right: precision-recall curves for the sparse structure recovery of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ averaged over 10 simulated datasets. In EiGLasso, K eigenvalues were used to approximate Hessian.

4.1 SIMULATED DATA

We compared EiGLasso and TeraLasso on data simulated from the known $\mathbf{\Theta}$ and $\mathbf{\Psi}$, assuming two types of graph structures, random graphs and block-diagonal graphs. To set the true $p \times p$ matrix $\mathbf{\Theta}$ with random graph structure, we first generated a sparse $p \times p$ matrix $\boldsymbol{A}$ by assigning $\{-1, 0, 1\}$ to each element with probabilities $\{\frac{1-\rho}{2}, \rho, \frac{1-\rho}{2}\}$. We chose ρ such that the number of off-diagonal non-zero elements is p. To ensure $\mathbf{\Theta}$ is positive definite, we set $\mathbf{\Theta}$ to $\boldsymbol{A}\boldsymbol{A}^T$ after adding $\sigma + 10^{-4}$ with $\sigma \sim \text{Unif}(0, 0.1)$ to each diagonal element. To set $\mathbf{\Theta}$ with block-diagonal structure, we generated 10 blocks of identical size $\frac{p}{10} \times \frac{p}{10}$ in the same way as random graphs and aligned the blocks along the diagonals of $\mathbf{\Theta}$. We set the true $\mathbf{\Psi}$ similarly. Given these true parameters, we simulated matrix-variate data from Gaussian distribution with inverse covariance $\mathbf{\Theta} \oplus \mathbf{\Psi}$.

We evaluated EiGLasso with Hessian approximated with different number of eigenvalues $K \in \{1, 5, 10, 50, 100\}$ and TeraLasso on computation time and on the accuracy of recovering the true graph structures. We ran EiGLasso until the decrease in the objective function satisfies the condition $\left|\frac{f_t - f_{t-1}}{f_t}\right| < \epsilon$. Applying the same convergence criterion terminated TeraLasso at suboptimal objective with far inferior estimate of the parameters, so we let it run until it reached similar objective value of EiGLasso. We evaluated the accuracy of graph structure recovery using precision and recall curves averaged over 10 datasets.

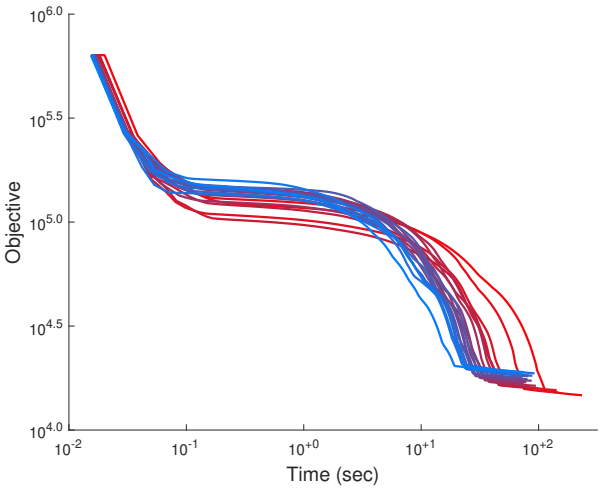

Figure 4: Convergence of EiGLasso for different regularization parameters. EiGLasso was run on a single dataset generated from random graphs with size 100×100 for $\mathbf{\Theta}$ and $\mathbf{\Psi}$, using 20 different regularization parameters ranging from 0.05 to 1.0 (red to blue curves).

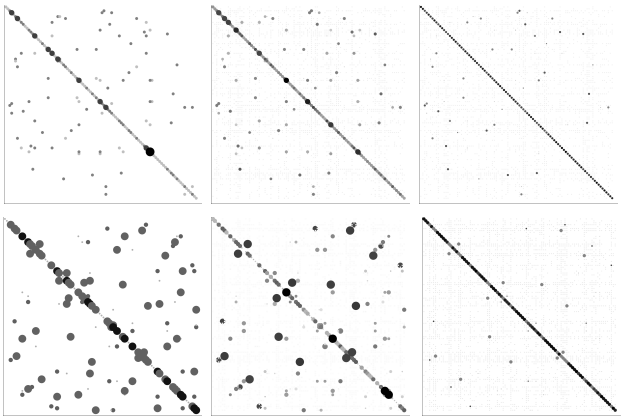

Figure 5: Comparison of the graph structures recoverd by EiGLasso and TeraLasso using simulated data. (a) True 100×100 random graphs for $\mathbf{\Theta}$ in two simulated datasets (top and bottom). Estimated $\mathbf{\Theta}$ are shown for (b) EiGLasso with $K = 1$ eigenvalue and (c) TeraLasso.

Our results show that across different graph types and sizes, our algorithm consistently converges one or two orders of magnitude faster in significantly fewer iterations than TeraLasso (Figs. 1-3, left). We make several observations on how the number of eigenvalues K used in Hessian approximation affects the convergence of EiGLasso. First, EiGLasso is faster than TeraLasso, even when it uses the exact Hessian. Second, the trade-off is that with larger K, EiGLasso converges in fewer iterations but usually takes more time because each iteration is more

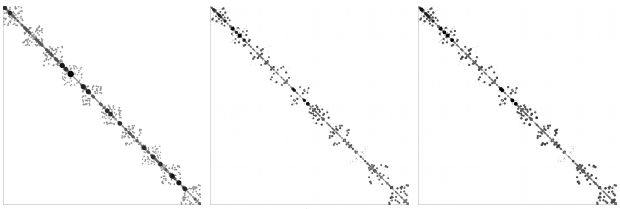

Figure 6: Comparison of graph structures recovered by EiGLasso with different Hessian approximations using simulated data. (a) True 1000×1000 block graph for $\mathbf{\Theta}$. Estimated $\mathbf{\Theta}$ with (b) $K = 1$ and (c) $K = 5$ eigenvalues.

expensive. Overall, we found $K = 1$ suffices and is the best choice for K in practice, because it reached the same solution with less memory and time. The convergence of EiGLasso is consistent as we vary the regularization parameters that control the level of sparsity (Fig. 4).

EiGLasso outperformed TeraLasso even when $K = 1$ on graph structure recovery (Figs. 1-3, right; Fig. 5). In addition, EiGLasso with $K = 1$ provided nearly identical accuracy as EiGLasso with exact Hessian (Figs. 1-3, right) and with $K = 5$ (Fig. 6). Overall, EiGLasso with Hessian approximation with $K = 1$ reaches the optimum significantly faster than EiGLasso with more eigenvalues and TeraLasso without sacrificing accuracy.

For larger graphs with $p = q = 1000$, on which TeraLasso could not run within a day, we applied EiGLasso with $K = 1$ and 5. EiGLasso with $K = 1$ converged substantially faster than EiGLasso with $K = 5$ (Figs. 7 and 8, left), with almost no sacrifice in accuracy as can be seen from the two completely overlapping precision-recall curves (Figs. 7 and 8, right).

4.2 REAL DATA

We obtained historical daily close price data of the S&P 500 constituents from Yahoo! Finance ($q = 306$ companies and $p = 2517$ days). We preprocessed the data and chose two different numbers of days ($p = 100$ and $p = 2516$). We compared EiGLasso and TeraLasso only on the smaller data, because TeraLasso did not converge within 24 hours on the larger one. On the smaller dataset, TeraLasso took 656 seconds and EiGLasso took 13 seconds. On the larger dataset, EiGLasso recovered dependencies among 306 companies that match the human annotation of the clusters of companies (Fig. 9).

5 CONCLUSION

In this paper, we introduced EiGLasso for an efficient estimation of the Kronecker-sum inverse covariance ma-

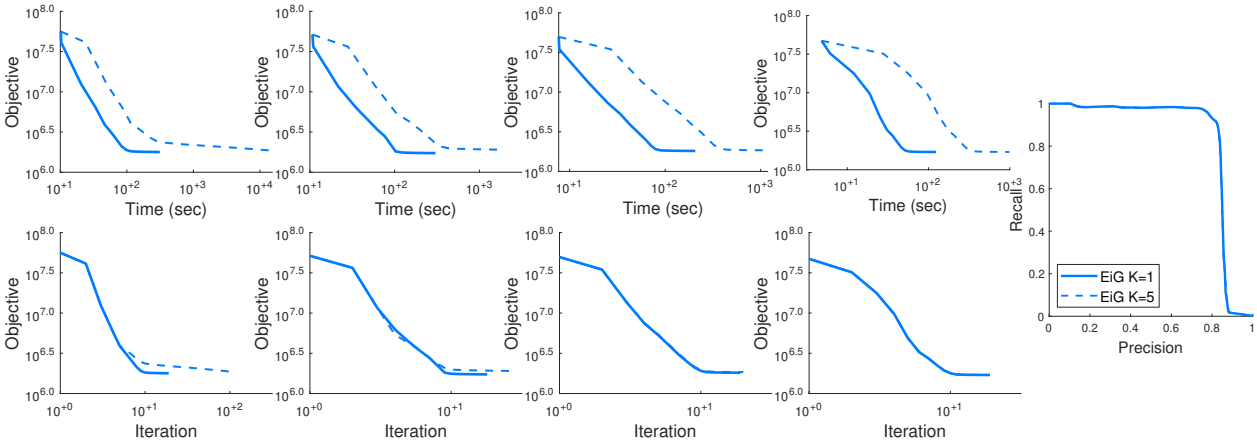

Figure 7: The performance of EiGLasso on simulated data with random graphs of size 1000×1000. Left: objective values over time (top) and over iterations (bottom) for four simulated datasets (four columns). Right: precision-recall curves for the sparse structure recovery of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ averaged over 10 simulated datasets. In EiGLasso, K eigenvalues were used to approximate Hessian.

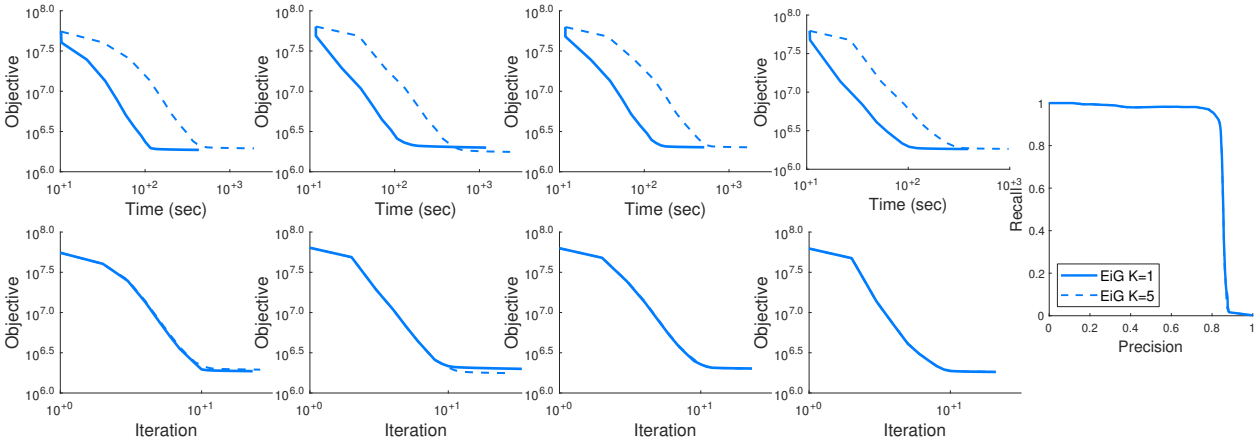

Figure 8: The performance of EiGLasso on simulated data with block graphs of size 1000×1000. Left: objective values over time (top) and over iterations (bottom) for four simulated datasets (four columns). Right: precision-recall curves for the sparse structure recovery of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ averaged over 10 simulated datasets. In EiGLasso, K eigenvalues were used to approximate Hessian.

trix. We described a new approach for handling the non-identifiable diagonal elements of the parameters. Our future work includes providing a theoretical justification of the low-rank approximation of Hessian and extending EiGLasso to incorporate the block-wise optimization used in BIG&QUIC (Hsieh et al., 2013) to lift memory constraint in QUIC (Hsieh et al., 2014).

Acknowledgements

This work was supported by NIH 1R21HG011116 and NSF CAREER Award MCB-1149885. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF ACI-1548562 and ACI-1445606.

Appendix

Proof of Theorem 1

Proof. The Kronecker sum of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ can be written using the eigendecomposition of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ as follows:

$$\mathbf{\Theta} \oplus \mathbf{\Psi} = (\boldsymbol{Q}_{\mathbf{\Theta}} \otimes \boldsymbol{Q}_{\mathbf{\Psi}})(\mathbf{\Lambda}_{\mathbf{\Theta}} \oplus \mathbf{\Lambda}_{\mathbf{\Psi}})(\boldsymbol{Q}_{\mathbf{\Theta}} \otimes \boldsymbol{Q}_{\mathbf{\Psi}})^T.$$

Then, the inverse of $\mathbf{\Theta} \oplus \mathbf{\Psi}$ is given as

$$\boldsymbol{W} = (\mathbf{\Theta} \oplus \mathbf{\Psi})^{-1}$$

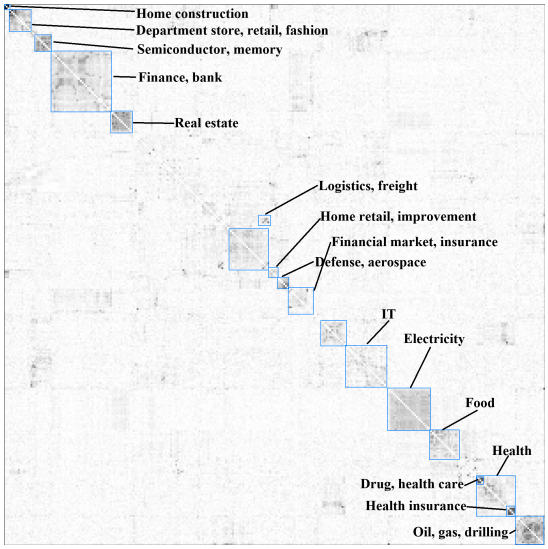

Figure 9: Graph over companies estimated by EiGLasso from S&P 500 close price data. Data from 306 companies over 2,516 days were used. Only the off-diagonals of the estimated graph matrix are shown.

$$= (\boldsymbol{Q}_{\boldsymbol{\Theta}} \otimes \boldsymbol{Q}_{\boldsymbol{\Psi}})(\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} \oplus \boldsymbol{\Lambda}_{\boldsymbol{\Psi}})^{-1}(\boldsymbol{Q}_{\boldsymbol{\Theta}} \otimes \boldsymbol{Q}_{\boldsymbol{\Psi}})^T.$$

Let $\boldsymbol{q}_{\boldsymbol{\Theta},i}$ and $\boldsymbol{q}_{\boldsymbol{\Psi},i}$ be the ith eigenvectors of $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$, given as the ith columns of $\boldsymbol{Q}_{\boldsymbol{\Theta}}$ and $\boldsymbol{Q}_{\boldsymbol{\Psi}}$. Let $\lambda_{W,lk} = \frac{1}{\lambda_{\boldsymbol{\Theta},l}+\lambda_{\boldsymbol{\Psi},k}}$. Then, we can re-write $\boldsymbol{W}$ above as follows:

$$\boldsymbol{W} = \sum_{l=1}^{p}\sum_{k=1}^{q} \lambda_{W,lk}(\boldsymbol{q}_{\boldsymbol{\Theta},l} \otimes \boldsymbol{q}_{\boldsymbol{\Psi},k})(\boldsymbol{q}_{\boldsymbol{\Theta},l} \otimes \boldsymbol{q}_{\boldsymbol{\Psi},k})^T.$$

To prove for the gradient, we begin with Eq. (6a) and substitute $\boldsymbol{W}$ with its eigendecomposition. Let $\boldsymbol{q}_{\boldsymbol{\Psi},ki}$ be the ith element of $\boldsymbol{q}_{\boldsymbol{\Psi},k}$. Then, we have

$$\begin{aligned}
\boldsymbol{W}_{\boldsymbol{\Theta}} &= \sum_{i=1}^{q} (\boldsymbol{I}_p \otimes \boldsymbol{e}_i)^T \boldsymbol{W} (\boldsymbol{I}_p \otimes \boldsymbol{e}_i) \\
&= \sum_{i=1}^{q}\sum_{l=1}^{p}\sum_{k=1}^{q} \lambda_{W,lk}(\boldsymbol{q}_{\boldsymbol{\Psi},ki}\boldsymbol{q}_{\boldsymbol{\Theta},l})(\boldsymbol{q}_{\boldsymbol{\Psi},ki}\boldsymbol{q}_{\boldsymbol{\Theta},l})^T \\
&= \sum_{l=1}^{p}\sum_{k=1}^{q} (\lambda_{\boldsymbol{\Theta},l} + \lambda_{\boldsymbol{\Psi},k})^{-1} \boldsymbol{q}_{\boldsymbol{\Theta},l}\boldsymbol{q}_{\boldsymbol{\Theta},l}^T \\
&= \sum_{k=1}^{q} \boldsymbol{Q}_{\boldsymbol{\Theta}} \left(\boldsymbol{\Lambda}_{\boldsymbol{\Theta}} + \lambda_{\boldsymbol{\Psi},k}\boldsymbol{I}_p\right)^{-1} \boldsymbol{Q}_{\boldsymbol{\Theta}}^T = \sum_{k=1}^{q} \boldsymbol{V}_k.
\end{aligned}$$

The third line follows from the second line above since $\Sigma_i \boldsymbol{q}_{\boldsymbol{\Psi},ki}^2 = 1$ by the orthonormality of eigenvectors.

To prove for the Hessian, we begin with Eq. (6b) and again substitute $\boldsymbol{W}$ with its eigendecomposition:

$$\begin{aligned}
\boldsymbol{H}_{\boldsymbol{\Theta}} &= \sum_{i=1}^{q}\sum_{j=1}^{q} \Big[(\boldsymbol{I}_p \otimes \boldsymbol{e}_i \otimes \boldsymbol{I}_p \otimes \boldsymbol{e}_i)^T \\
&\qquad \boldsymbol{W} \otimes \boldsymbol{W} (\boldsymbol{I}_p \otimes \boldsymbol{e}_j \otimes \boldsymbol{I}_p \otimes \boldsymbol{e}_j)\Big] \\
&= \sum_{i,j} \Big[(\boldsymbol{I}_p \otimes \boldsymbol{e}_i)^T \boldsymbol{W} (\boldsymbol{I}_p \otimes \boldsymbol{e}_j) \\
&\qquad \otimes (\boldsymbol{I}_p \otimes \boldsymbol{e}_i)^T \boldsymbol{W} (\boldsymbol{I}_p \otimes \boldsymbol{e}_j)\Big] \\
&= \sum_{i,j} \Big[\sum_{l,k} \lambda_{W,lk}(\boldsymbol{q}_{\boldsymbol{\Psi},ki}\boldsymbol{q}_{\boldsymbol{\Theta},l})(\boldsymbol{q}_{\boldsymbol{\Psi},kj}\boldsymbol{q}_{\boldsymbol{\Theta},l})^T \\
&\qquad \otimes \sum_{r,s} \lambda_{W,rs}(\boldsymbol{q}_{\boldsymbol{\Psi},si}\boldsymbol{q}_{\boldsymbol{\Theta},r})(\boldsymbol{q}_{\boldsymbol{\Psi},sj}\boldsymbol{q}_{\boldsymbol{\Theta},r})^T\Big] \\
&= \sum_{k} \Big[\sum_{l} \lambda_{W,lk}\boldsymbol{q}_{\boldsymbol{\Theta},l}\boldsymbol{q}_{\boldsymbol{\Theta},l}^T \otimes \sum_{r} \lambda_{W,rk}\boldsymbol{q}_{\boldsymbol{\Theta},r}\boldsymbol{q}_{\boldsymbol{\Theta},r}^T\Big] \\
&= \sum_{k} \boldsymbol{V}_k \otimes \boldsymbol{V}_k.
\end{aligned}$$

The fourth equality above follows from the orthonormality of eigenvectors: $\Sigma_{i,j}\boldsymbol{q}_{\boldsymbol{\Psi},ki}\boldsymbol{q}_{\boldsymbol{\Psi},si}\boldsymbol{q}_{\boldsymbol{\Psi},kj}\boldsymbol{q}_{\boldsymbol{\Psi},sj}$ is 1 if $s = k$ or 0 otherwise. □

Coordinate descent optimization in EiGLasso

To solve the optimization problem in Eq. (4), we adopt the strategy developed for Gaussian graphical models in QUIC (Hsieh et al., 2014). In EiGLasso, solving Eq. (4) for a single (i,j)th element in $D_{\boldsymbol{\Theta}}$ amounts to solving the following optimization problem:

$$\begin{aligned}
\underset{\mu}{\text{argmin}} \quad & \mu\Big(n\boldsymbol{S}_{ij} - \boldsymbol{W}_{\boldsymbol{\Theta},ij} + \sum_{k=1}^{K} \boldsymbol{v}_{k,i}^T D_{\boldsymbol{\Theta}} \boldsymbol{v}_{k,j} \\
& + (q-K)\boldsymbol{v}_{K+1,i}^T D_{\boldsymbol{\Theta}} \boldsymbol{v}_{K+1,j}\Big) \\
& + \frac{\mu^2}{2}\Big(\sum_{k=1}^{K} \boldsymbol{V}_{k,ij}^2 + \boldsymbol{V}_{k,ii}\boldsymbol{V}_{k,jj} \\
& + (q-K)\left(\boldsymbol{V}_{K+1,ij}^2 + \boldsymbol{V}_{K+1,ii}\boldsymbol{V}_{K+1,jj}\right)\Big) \\
& + q\gamma_{\boldsymbol{\Theta}}||\boldsymbol{\Theta}_{ij} + D_{\boldsymbol{\Theta},ij} + \mu||_{1,\text{off}},
\end{aligned}$$

where $\boldsymbol{v}_{k,i}$ is the ith column of $\boldsymbol{V}_k$. The problem above has a closed-form solution

$$\mu = -c + \mathcal{S}\left(c - \frac{b}{a}, \frac{q\gamma_{\boldsymbol{\Theta}}}{a}\right),$$

where $a = \sum_{k=1}^{K} \boldsymbol{V}_{k,ij}^2 + \boldsymbol{V}_{k,ii}\boldsymbol{V}_{k,jj} + (q-K)\left(\boldsymbol{V}_{K+1,ij}^2 + \boldsymbol{V}_{K+1,ii}\boldsymbol{V}_{K+1,jj}\right)$, $b = n\boldsymbol{S}_{ij} - \boldsymbol{W}_{\boldsymbol{\Theta},ij} + \sum_{k=1}^{K} \boldsymbol{v}_{k,i}^T D_{\boldsymbol{\Theta}} \boldsymbol{v}_{k,j}, + (q-K)\boldsymbol{v}_{K+1,i}^T D_{\boldsymbol{\Theta}} \boldsymbol{v}_{K+1,j}$, $c = \boldsymbol{\Theta}_{ij} + D_{\boldsymbol{\Theta},ij}$ and $\mathcal{S}(z,r) = \text{sign}(z)\max\{|z| - r, 0\}$ is the soft-thresholding function.

References

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2:183–202.

Dai, X., Li, T., Bai, Z., Yang, Y., Liu, X., Zhan, J., and Shi, B. (2015). Breast cancer intrinsic subtype classification, clinical use and future trends. *American Journal of Cancer Research*, 5:2929–43.

Dawid, A. P. (1981). Some matrix-variate distribution theory: Notational considerations and a Bayesian application. *Biometrika*, 68(1):265–274.

Dutilleul, P. (1999). The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123.

Friedman, J., Hastie, T., and Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Greenewald, K., Zhou, S., and Hero III, A. (2019). Tensor graphical lasso (TeraLasso). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(5):901–931.

Gupta, A. and Nagar, D. (1999). *Matrix Variate Distributions*. Monographs and Surveys in Pure and Applied Mathematics. Taylor & Francis.

Hammack, R., Imrich, W., and Klavzar, S. (2011). *Handbook of Product Graphs, Second Edition*. CRC Press, Inc., USA, 2nd edition.

Hsieh, C.-J., Sustik, M. A., Dhillon, I. S., and Ravikumar, P. (2014). QUIC: Quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15(83):2911–2947.

Hsieh, C.-J., Sustik, M. A., Dhillon, I. S., Ravikumar, P. K., and Poldrack, R. (2013). BIG & QUIC: Sparse inverse covariance estimation for a million variables. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3165–3173. Curran Associates, Inc.

Kalaitzis, A., Lafferty, J., Lawrence, N. D., and Zhou, S. (2013). The bigraphical lasso. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1229–1237, Atlanta, Georgia, USA. PMLR.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017). Video pixel networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1771–1779, International Convention Centre, Sydney, Australia. PMLR.

King, B. F. (1966). Market and industry factors in stock price behavior. *The Journal of Business*, 39(1):139–190.

Leng, C. and Tang, C. Y. (2012). Sparse matrix graphical models. *Journal of the American Statistical Association*, 107(499):1187–1200.

Tsiligkaridis, T. and Hero, A. O. (2013). Covariance estimation in high dimensions via Kronecker product expansions. *IEEE Transactions on Signal Processing*, 61(21):5347–5360.

Yin, J. and Li, H. (2012). Model selection and estimation in the matrix normal graphical model. *Journal of Multivariate Analysis*, 107:119–140.

Zhou, S. (2014). Gemini: Graph estimation with matrix variate normal instances. *Ann. Statist.*, 42(2):532–562.