Gaussian Graphical Modelling Without Independence Assumptions for Uncentered Data

Bailey Andrew, David R Westhead, Luisa Cutillo

University of Leeds
sceba@leeds.ac.uk, D.R.Westhead@leeds.ac.uk, L.Cutillo@leeds.ac.uk

Abstract

The independence assumption between random variables is a useful tool to increase the tractability of a modelling framework. However, this assumption can be too simplistic; failing to take dependencies into account can cause models to fail dramatically. The field of multi-axis graphical modelling (also called multi-way modelling, Kronecker-separable modelling) has seen growth over the past decade, but these models require that the data have zero mean. In the multi-axis case, inference is typically done in the single sample scenario, making mean inference impossible.

In this paper, we demonstrate how the zero-mean assumption can cause egregious modelling errors for Kronecker-sum-decomposable Gaussian graphical models, as well as propose a relaxation to the zero-mean assumption that allows the avoidance of such errors. Specifically, we propose the "Kronecker-sum-structured mean" assumption, which leads to models with nonconvex-but-unimodal log-likelihoods that can be solved efficiently with coordinate descent.

Code — https://github.com/BaileyAndrew/Noncentral-KS-Normal

Introduction

We often wish to find networks ('graphs') that describe our data. For example, we may be interested in gene regulatory networks in biology, or social interaction networks in epidemiology. In these cases, the graph itself is an object of interest. When the goal of an analysis does not involve a graph, the creation of one can still be useful as a preprocessing step for further insights, such as for clustering. Their use is not limited to clustering; the first step of the popular dimensionality reduction method UMAP, for example, is to create a (typically nearest neighbors) graph (McInnes, Healy, and Melville 2020), but one could also experiment with other graphs.

While there are many types of graphs, this paper will focus on *conditional dependency graphs*. Intuitively, two vertices are connected in such a graph if they are still statistically dependent after conditioning out the other variables. We denote the property that x and y are conditionally independent, given z, as $x \perp\!\!\!\perp y \mid z$. We will focus on the case where we condition over all other variables in the dataset $\mathcal{D}$, i.e. where $z = \mathcal{D}_{\setminus x \setminus y} \overset{def}{=} \mathcal{D} - \{x, y\}$.

$$x \perp\!\!\!\perp y \mid z \overset{def}{\iff} \mathbb{P}[x, y \mid z] = \mathbb{P}[x \mid z]\,\mathbb{P}[y \mid z]$$

Conditional dependency graphs have several convenient properties. They are interpretable, and intuitively capture the 'direct effect' of two variables on each other (rather than 'indirect effects' passing through confounders and mediators). They are typically sparse, and are related to causality[1]. They also have a very useful property in multivariate Gaussian datasets, namely that two Gaussian variables are conditionally independent if and only if the corresponding term in the inverse covariance matrix is zero.

$$x \perp\!\!\!\perp y \mid \mathcal{D}_{\setminus x \setminus y} \overset{Gaussian}{\iff} \mathbf{\Sigma}^{-1}_{xy} = 0$$

The inverse covariance matrix is also called the precision matrix, and we will denote it $\mathbf{\Psi}$. It can be interpreted as an adjacency matrix for the graph of conditional dependencies. Due to this convenient correspondence, and due to the fact that practically all multi-axis work has so far been limited to the Gaussian case, we will assume in this paper that our data is Gaussian. This type of model, without the Gaussian assumption, is more generally known as a Markov random field.

Through tools such as the Nonparanormal Skeptic (Liu et al. 2012), this assumption can be weakened to the Gaussian copula assumption (intuitively, the variables have arbitrary continuous marginals but still interact 'Gaussian-ly').

When making independence assumptions, this problem is solved by well-known methods such as the Graphical Lasso (Friedman, Hastie, and Tibshirani 2008). However, independence assumptions are often false in practice. They become particularly egregious in the case of omics data, such as single cell RNA-sequencing (scRNA-seq). These datasets take the form of a cells-by-genes matrix. We might be interested in a gene regulatory network, in which case we assume the cells are independent. Alternatively, we might be interested

[1]Under modest assumptions, that can be violated in some real-world scenarios, conditional dependency graphs form the 'skeleton' of a causal directed acyclic graph, i.e. the causal graph with directionality removed.

in the cellular microenvironment, in which case we assume the genes are independent. Whichever one we want to learn, we are put in the awkward position of assuming the other does not exist.

To avoid independence assumptions altogether, we can vectorize our dataset $\mathbf{X}$: $\text{vec}[\mathbf{X}] \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Psi}^{-1})$. Instead of n samples of m features, we have 1 sample of nm features (each element of the data matrix is a feature). For scRNA-seq, the precision matrix is of size $O(n^2m^2)$ and represents the dependencies between different (cell, gene) pairs. The problem becomes both computationally and statistically intractable; we quickly run out of space to store such matrices, and have no way of being confident in our results to any degree of statistical certainty. We need to find a middle ground between independence and full dependence.

To do this, we can take advantage of matrix structure. We are not truly interested in a graph of (cell, gene) pairs, but rather a graph of cells and a graph of genes. Thus, we can make a 'graph decomposition assumption'; our (cell, gene) pair graph should be able to be factored (for some definition of factoring ζ) into the cell graph and the gene graph: $\text{vec}[\mathbf{X}] \sim \mathcal{N}(\boldsymbol{\mu}, \zeta(\boldsymbol{\Psi}_{cells}, \boldsymbol{\Psi}_{genes})^{-1})$.

Several choices of decomposition have appeared in the literature, such as the Kronecker product (Dutilleul 1999) and squared Kronecker sum decompositions (Wang, Jang, and Hero 2020); in this paper, we focus on the Kronecker sum decomposition (Kalaitzis et al. 2013) due to its popularity, convexity, relationship to conditional dependence, interpretability as a Cartesian product, and correspondence to a maximum entropy distribution. The Kronecker sum is denoted $\oplus$ and is defined as $\mathbf{A}_{a\times a} \oplus \mathbf{B}_{b\times b} = \mathbf{A}_{a\times a} \otimes \mathbf{I}_{b\times b} + \mathbf{I}_{a\times a} \otimes \mathbf{B}_{b\times b}$, where $\otimes$ is the Kronecker product. The operation is associative, allowing a straightforward generalization to more than two axes (i.e. tensor-variate datasets); this was first explored by Greenewald, Zhou, and Hero (2019). Under the zero-mean and Kronecker sum decomposability assumptions, the model for a dataset $\mathcal{D}$ is as follows:

$$\mathcal{D} \sim \mathcal{N}_{KS}(\mathbf{0}, \{\boldsymbol{\Psi}_\ell\}_\ell)$$
$$\iff \text{vec}[\mathcal{D}] \sim \mathcal{N}\left(\mathbf{0}, \left(\bigoplus_\ell \boldsymbol{\Psi}_\ell\right)^{-1}\right)$$

The zero-mean assumption is conspicuous, but necessary; recall that we transformed our dataset from n samples and m features to a single sample of nm features. Estimating the mean of a single sample is a recipe for disaster; when one centers the dataset, they will be left with a constant zero vector.

We could consider decomposing the mean of our model, just like for our precision matrices. A natural and analogous decomposition would be that our mean takes the form $\boldsymbol{\mu}_{\text{cells}} \widetilde{\oplus} \boldsymbol{\mu}_{\text{genes}} = \boldsymbol{\mu}_{\text{cells}} \otimes \mathbf{1} + \mathbf{1} \otimes \boldsymbol{\mu}_{\text{genes}}$; this operation intermixes the means in the same way the Kronecker sum intermixes the variances. For Kronecker *product* models (not Kronecker sum models), this has been considered before (Allen and Tibshirani 2010). It was shown that the maximum likelihood for μ_ℓ has the following form (in the two-axis case, where d_ℓ represents the number of elements in axis ℓ):

$$\boldsymbol{\mu}_1 = \frac{\mathbf{1}^T\left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}_2^T\right)}{d_2} \qquad \boldsymbol{\mu}_2 = \frac{\mathbf{1}^T\left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}_1^T\right)}{d_1}$$

We will see later that such formulas are not as simple in the Kronecker sum case. First, though, let us note that the estimate of $\boldsymbol{\mu}_\ell$ does not depend on the dependencies in the data ($\boldsymbol{\Psi}_\ell$). This is a mathematically convenient property to have, but is philosophically troubling. Suppose we have a dataset of only two points, both distinct and independent. The true mean value is just their average. Now, suppose we duplicate the first point several times, perhaps with some modest noise added - the mean estimate will lie much closer to the first point.

In fact, we can choose how many times to duplicate the first or second points in order to place the estimated mean anywhere between the original points. However, this estimated mean is an artifact of the dependencies in our data! The true mean, ultimately, still lies at its original location.

Our Contributions

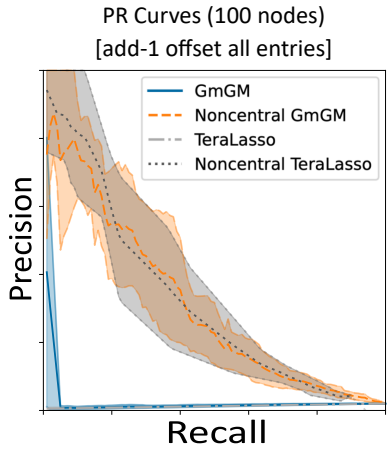

Figure 1: Precision and recall for a synthetic dataset in which data was generated from a Kronecker-sum normal distribution with a constant value of 1 added to every entry of the matrix. Error bars are the best/worst performance over 10 trials; the center line is average performance.

We are naturally led to the following question:

Problem 1. *For the noncentral Kronecker sum model, does using the mean decomposition $\widetilde{\bigoplus}_\ell \boldsymbol{\mu}_\ell$ allow its parameters to be effectively estimated?*

In this paper, we show that the answer to this question is yes. For the Kronecker-sum-structured model, the estimators for $\boldsymbol{\mu}_\ell$ depend on $\boldsymbol{\Psi}_\ell$. However, due to this dependency, our problem is no longer convex; there could be some non-global minima!

Problem 2. *Is our estimator for the parameters of the noncentral Kronecker-sum-structured normal distribution guaranteed to be the global maximum likelihood estimator?*

We also show that the answer to this question is affirmative; in fact, there are no non-global minima, despite the nonconvexity.

One may also wonder if the zero-mean assumption is truly problematic. In fact, the zero-mean assumption can have an extremely dramatic negative effect on performance. In the simple scenario in which we add a constant offset to every element, we can see that methods that do not try to estimate the mean are practically worthless, whereas those that do (which we shall call 'noncentral') maintain reasonable performance (Figure 1).

Notation

In this paper, lowercase letters a will refer to scalars, lowercase bold $\mathbf{v}$ will refer to vectors, uppercase bold $\mathbf{M}$ will refer to matrices, and uppercase calligraphic $\mathcal{T}$ will refer to tensors. We will conceptualize our dataset to be a tensor $\mathcal{D}$ with axes lengths d_ℓ for each axis ℓ. $d_{<\ell}$ and $d_{>\ell}$ are shorthands for the product of lengths of axes before and after ℓ, respectively. $d_{\setminus\ell}$ is the product of all axes other than ℓ; from the perspective of ℓ, this is the number of samples for that axis. $d_\forall$ is the product of all d_ℓ. We typically follow the notation of Kolda and Bader (2009) for tensor operations.

Let $\mathbf{I}_a$ and $\mathbf{1}_a$ be the $a \times a$ identity matrix and length a vector of ones, respectively. We formally define the Kronecker sum of matrices and analogous operation on vectors as follows:

$$\bigoplus_\ell \mathbf{\Psi}_\ell = \sum_\ell \mathbf{I}_{d_{<\ell}} \otimes \mathbf{\Psi}_\ell \otimes \mathbf{I}_{d_{>\ell}}$$
$$\widetilde{\bigoplus}_\ell \boldsymbol{\mu}_\ell = \sum_\ell \mathbf{1}_{d_{<\ell}} \otimes \boldsymbol{\mu}_\ell \otimes \mathbf{1}_{d_{>\ell}}$$

We will let $\mathbf{\Psi}_{\setminus\ell}$ be a shorthand for $\bigoplus_{\ell' \neq \ell} \mathbf{\Psi}_{\ell'}$, with an analogous definition for $\boldsymbol{\mu}_{\setminus\ell}$. Kronecker products, and by extension Kronecker sums, have a convenient permutation-invariance property; we can swap the order of summands without affecting the results, as long as we perform this swap consistently across an equation and permute matrices accordingly; we can typically rewrite equations involving $\bigoplus_\ell \mathbf{\Psi}_\ell$ into those involving $\mathbf{\Psi}_\ell \oplus \mathbf{\Psi}_{\setminus\ell}$. We will define $\mathbf{\Omega} = \bigoplus_\ell \mathbf{\Psi}_\ell$ and $\boldsymbol{\omega} = \widetilde{\bigoplus}_\ell \boldsymbol{\mu}_\ell$. We define noncentral Kronecker sum normal distribution as:

$$\operatorname{vec}[\mathcal{D}] \sim \mathcal{N}\left(\boldsymbol{\omega}, \mathbf{\Omega}^{-1}\right)$$

We let $\mathbb{K}_\mathbf{M}$ be the space of Kronecker-sum-decomposable matrices[2], i.e. $\mathbf{\Omega} \in \mathbb{K}_\mathbf{M}$. Likewise, $\boldsymbol{\omega} \in \mathbb{K}_\mathbf{v}$ is the space of Kronecker-sum-decomposable vectors. Finally, let $\mathbb{K}_{\mathbf{Mv}}$ be the space of vectors of the form $\mathbf{Xy}$, where $\mathbf{X} \in \mathbb{K}_\mathbf{M}, \mathbf{y} \in \mathbb{K}_\mathbf{v}$. All of these spaces are linear subspaces of $\mathbb{R}^{d_\forall}$ or $\mathbb{R}^{d_\forall \times d_\forall}$. We may use $\mathbb{K}$ as a shorthand when discussing properties that apply to both $\mathbb{K}_\mathbf{M}$ and $\mathbb{K}_\mathbf{v}$.

[2]Technically, we need to specify the dimensionality of each term in the decomposition to define this space, but this would be notationally cumbersome and is always clear from context.

It is important to note that the parameterization of $\mathbb{K}$ we have used so far is not identifiable. If $\sum_\ell c_\ell = 0$, then $\bigoplus_\ell (\mathbf{\Psi}_\ell + c_\ell \mathbf{I}) = \bigoplus_\ell \mathbf{\Psi}_\ell$. Greenewald, Zhou, and Hero (2019) use the identifiable representation $\tau \mathbf{I}_\forall + \bigoplus_\ell \widetilde{\mathbf{\Psi}}_\ell$, where $\operatorname{tr}\left[\widetilde{\mathbf{\Psi}}_\ell\right] = 0$. Likewise, we can identifiably represent our mean decomposition as $m\mathbf{1} + \widetilde{\bigoplus}_\ell \widetilde{\boldsymbol{\mu}}_\ell$, where $\mathbf{1}^T \widetilde{\boldsymbol{\mu}}_\ell = 0$.

The last important property we will introduce is the matrix representation of Kronecker products of vectors: note that $\boldsymbol{\mu}_\ell \otimes \mathbf{1}_{d_{\setminus\ell}} = \left(\mathbf{I}_{d_\ell} \otimes \mathbf{1}_{d_{\setminus\ell}}\right) \boldsymbol{\mu}_\ell$.

Estimation Method

As a shorthand, let $\mathbf{x} = \operatorname{vec}[\mathcal{D}]$. Our negative log likelihood looks just like that of our normal distribution, except that the parameters are restricted to lie in $\mathbb{K}$. To ensure existence of $\mathbf{\Omega}$, it is necessary to either add a regularization penalty, as in TeraLasso, or restrict $\mathbf{\Omega}$ to a low-rank subspace, as proposed by Andrew, Westhead, and Cutillo (2024b). These are details of the estimator for $\mathbf{\Omega}$, which do not affect the estimator of $\boldsymbol{\omega}$; we do not focus on these aspects of the problem.

$$\mathrm{p}(\mathcal{D}) = \frac{\sqrt{|\mathbf{\Omega}|}}{(2\pi)^{\frac{d_\forall}{2}}} e^{\frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\omega})^T \mathbf{\Omega}(\boldsymbol{x}-\boldsymbol{\omega})}$$
$$\mathrm{NLL}(\mathcal{D}) \propto \frac{-1}{2} \log|\mathbf{\Omega}| + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\omega})^T \mathbf{\Omega}(\boldsymbol{x}-\boldsymbol{\omega})$$

Despite its ubiquity, the normal distribution is actually fairly poorly behaved from an optimization perspective; its NLL is not convex, nor is it even geodesically convex (Hosseini and Sra 2015). Thankfully, the MLE of the mean (for the unrestricted normal) has a closed form solution, $\omega = \frac{1}{n}\sum_i^n \mathbf{x}_i$, so this nonconvexity does not affect results in practice. Unfortunately, due to our $\mathbb{K}_\mathbf{v}$ restriction, this solution no longer holds.

While it is not convex in $(\mathbf{\Omega}, \boldsymbol{\omega})$ jointly, it is convex in each argument individually - i.e, it is biconvex. This suggests a flip-flop optimization scheme. For fixed $\boldsymbol{\omega}$, the estimation of $\mathbf{\Omega}$ reduces to the noncentral case (by subtracting out $\boldsymbol{\omega}$); several algorithms already exist to solve this. Thus, we focus on optimization when $\mathbf{\Omega}$ is fixed.

Optimization w.r.t. $\boldsymbol{\omega}$ can be framed as a convex constrained quadratic programming (QP) problem. However, it will turn out to be convenient to further break the problem up into its constituent parts $(m, \{\widetilde{\boldsymbol{\mu}}_\ell\})$, and minimize them each individually. Let $\theta_\ell = \mathbf{1}_{d_{\ell'}}^T \mathbf{\Psi}_{\ell'} \mathbf{1}_{d_{\ell'}}$, $\theta_{\setminus\ell} = \sum_{\ell' \neq \ell} \frac{d_\forall}{d_\ell d_{\ell'}} \theta_\ell$, and $\mathbf{x}_\ell$ be the vectorization of $\mathcal{D}$ done in the order as if axis ℓ were the first axis. To preserve space, we will defer the algebraic manipulations to the appendix, and simply note here that the optimal value of m, with all other values fixed, is:

$$m = \frac{\left(\mathbf{x} - \widetilde{\bigoplus}_\ell \widetilde{\boldsymbol{\mu}}_\ell\right)\left(\widetilde{\bigoplus} \mathbf{\Psi}_\ell \mathbf{1}_{d_\ell}\right)}{\sum_\ell d_{\setminus\ell} \theta_\ell} \tag{1}$$

Observe that, when the elements of $\mathbf{x}$ are independent, this formula expresses the average distance of the dataset from 0 (after accounting for the axis-wise means $\widetilde{\boldsymbol{\mu}}_\ell$).

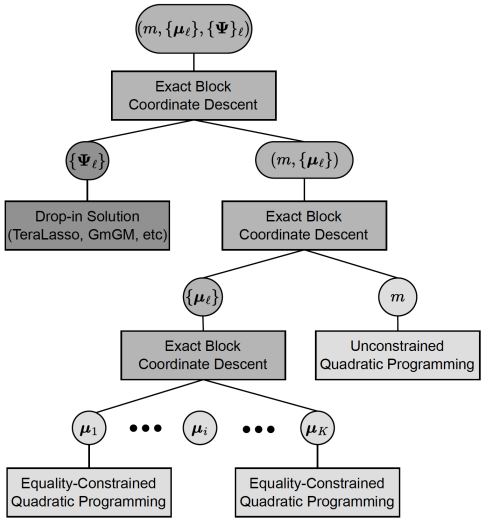

Figure 2: A graphical summary of the manner in which we divide our problem into sub-problems. While the main problem is nonconvex, all sub-problems are convex, and by Theorem 1 the main problem has a unique solution.

For $\widetilde{\boldsymbol{\mu}}_\ell$ (with $\widetilde{\boldsymbol{\mu}}_{\setminus\ell}$ fixed, and again deferring the derivation to the appendix) we have the following QP problem:

$$\begin{aligned}
&\underset{\widetilde{\boldsymbol{\mu}}_\ell}{\operatorname{argmin}} \ \widetilde{\boldsymbol{\mu}}_\ell^T \mathbf{A}_\ell \widetilde{\boldsymbol{\mu}}_\ell + \mathbf{b}_\ell^T \widetilde{\boldsymbol{\mu}}_\ell \\
&\text{where } \mathbf{A}_\ell = d_{\setminus\ell}\boldsymbol{\Psi}_\ell + \theta_{\setminus\ell}\mathbf{I}_{d_\ell} \\
&\qquad \boldsymbol{b}_\ell = m d_{\setminus\ell}\mathbf{1}_{d_\ell}^T\boldsymbol{\Psi}_\ell + m\theta_{\setminus\ell}\mathbf{1}_{d_\ell} + \left[\widetilde{\boldsymbol{\mu}}_{\setminus\ell}^T\boldsymbol{\Psi}_{\setminus\ell}\mathbf{1}_{d_{\setminus\ell}}\right]\mathbf{1}_{d_\ell} \\
&\qquad\quad - \mathbf{x}_\ell^T\left(\boldsymbol{\Psi}_\ell \otimes \mathbf{1}_{d_{\setminus\ell}}\right) - \mathbf{x}_\ell^T\left(\mathbf{1}_\ell \otimes \boldsymbol{\Psi}_{\setminus\ell}\mathbf{1}_{d_{\setminus\ell}}\right) \\
&\qquad \widetilde{\boldsymbol{\mu}}_\ell^T\mathbf{1}_{d_\ell} = \mathbf{0}
\end{aligned}$$

Parts of the definition of $\mathbf{b}_\ell$ can be simplified further, leading to more efficient computations, but such manipulations are notationally complicated to express; we defer them to the appendix, where we also show that $\mathbf{A}_\ell$ is guaranteed to be invertible. QP problems with linear equality constraints have closed-form solutions. In particular, we have that:

$$\widetilde{\boldsymbol{\mu}}_\ell = \frac{\mathbf{1}_{d_\ell}^T\boldsymbol{A}_\ell^{-1}\boldsymbol{b}_\ell}{\mathbf{1}_{d_\ell}^T\mathbf{A}_\ell^{-1}\mathbf{1}_{d_\ell}}\boldsymbol{A}_\ell^{-1}\mathbf{1}_{d_\ell} - \mathbf{A}_\ell^{-1}\boldsymbol{b}_\ell \tag{2}$$

As we have closed-form solutions to all our coordinate-wise minima, block coordinate descent is a natural solution. While typically it is advisable to avoid directly calculating inverses $\mathbf{A}_\ell^{-1}$ in the solution to QP problems, inversion here is likely to be cheap; most solvers for $\boldsymbol{\Psi}_\ell$ require calculating eigendecompositons of $\boldsymbol{\Psi}_\ell$. Thus, the eigendecomposition, and hence inverse, of $\mathbf{A}_\ell$ is readily available. This answers Problem 1.

Some methods, such as GmGM (Andrew, Westhead, and Cutillo 2024a), require only a single eigendecomposition, and then find the optimum while staying in 'eigen-space'. They use the eigenvectors of covariance matrices; when the mean is updated, the new covariance matrices can be expressed in terms of rank-one-updates of the original covariance matrix.

Rank-one updates of eigendecompositions are cheap (comparatively); thus, for each flip-flop of our algorithm, if GmGM is used as the solver for $\boldsymbol{\Psi}_\ell$, then it can stay entirely in eigenspace - we preserve the 'only-one-eigendecomposition' property of GmGM. Other methods, such as TeraLasso, will still require multiple eigendecompositions, but these will remain useful for our calculation of $\mathbf{A}_\ell^{-1}$. It is because of these convenient computational properties that we have chosen a flip-flop route for estimation of $\widetilde{\boldsymbol{\mu}}_\ell$, despite the fact that the optimization problem is jointly convex in $\{\widetilde{\boldsymbol{\mu}}_\ell\}_\ell$.

Algorithm 1: **Noncentral KS-structured Gaussian MLE**

Require: Dataset $\mathcal{D}$ of size $d_1, ..., d_K$
Require: Sub-procedure $\bigoplus \text{GRAPH}$ that estimates $\boldsymbol{\Psi}_\ell$ such as TeraLasso, GmGM
Output: Identifiable MLE $\left(m, \widetilde{\boldsymbol{\mu}}_\ell, \tau, \widetilde{\boldsymbol{\Psi}}_\ell\right)$

```
#Initialize Parameters with Reasonable Guesses
m^(0) ← (1/d_∀) Σ_{i=1}^{d_∀} vec[D]_i
for ℓ ∈ {1, ..., K} do
    Let x_ℓ be vectorized D with ℓ as first axis.
    Initial guess is the mean along axis ℓ:
        μ̃_ℓ^(0) ← (1/d_{\ℓ}) Σ_{i=1}^{d_{\ℓ}} x_{ℓ,i} − m^(0)
end for
#Iterate Until Convergence
while not converged do
    {Ψ_ℓ^(i+1)}_ℓ ← ⊕ GRAPH(D − m^(i) 1 − ⊕̃_{ℓ'} μ_ℓ^(i))
    while not converged do
        #Find Mean Parameters
        m^(i+1) ← (Equation 1)
        for ℓ ∈ {1, ..., K} do
            μ̃_ℓ^(i+1) ← (Equation 2)
        end for
    end while
end while
Map {Ψ_ℓ} to identifiable parameterization ({Ψ̃_ℓ}, τ)
```

See Algorithm 1 for a pseudocode presentation of our algorithm. Note that the pseudocode does not take into account potential speedups that come from sharing information between the precision matrix and mean estimation tasks, such as sharing eigenvectors. We wanted our method to be able to be used as a 'drop-in wrapper' for pre-existing methods. Accordingly, we have presented and implemented our methodology in its most general form. For a graphical overview of how we optimize, see Figure 2.

Global Optimality and Complexity

The optimization procedure we consider is not convex, merely biconvex. How do we know if the solution we find is globally optimal? In this section, we will show that, despite

the non-convexity, the problem still has the property that all local minima are global minima. In fact, when identifiable representations are used, we will see that there is exactly one minimum. As in the last section, we will defer the more mechanical details of the proof to the appendix, and rather give a sketch of the main ideas here.

Theorem 1. *The maximum likelihood of the noncentral Kronecker-sum-structured normal distribution has a unique maximum, which is global. The estimator defined by Algorithm 1 converges to this.*

Proof. See appendix. Sketch given below. □

The general idea is to perform a one-to-one transformation on the problem to turn it into a strictly convex problem in a different set of parameters - specifically, an instance of conic programming with linear constraints. Letting $\mathfrak{A}$ be our original problem, $\mathfrak{A}'$ be a slight modification of it, and $\mathfrak{B}$ be the conic problem, we show the following chain of implications:

$$\begin{aligned} & (\boldsymbol{\omega}, \boldsymbol{\Omega}) \in \text{local minima}\,(\mathfrak{A}) \\ \iff & (\boldsymbol{\omega}, \boldsymbol{\Omega}, s=1) \in \text{local minima}\,(\mathfrak{A}') \\ \iff & (\boldsymbol{\Gamma}) \in \text{local minima}\,(\mathfrak{B}) \\ \iff & (\boldsymbol{\Gamma}) \in \text{global minima}\,(\mathfrak{B}) \qquad \text{(convexity)} \end{aligned}$$

$\boldsymbol{\Gamma}$ is a one-to-one function of $\boldsymbol{\omega}, \boldsymbol{\Omega}$, and s, where s is somewhat like a slack variable. We show that local minima always occur at $s = 1$, and furthermore that at local minima, the value of the objective function of $\mathfrak{A}$ equals the value of the objective function of $\mathfrak{B}$ for the corresponding parameters. Thus, all local minima of $\mathfrak{A}$ must obtain the same value, i.e. they are global minima. In fact, because the transformation is one-to-one, and $\mathfrak{B}$ has a unique global minimum, there is exactly one local minimum of $\mathfrak{A}$.

The transformation we use is:

$$\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\omega}, \boldsymbol{\Omega}^{-1}\right) \to \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Gamma}^{-1}\right)$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\Omega} & -\boldsymbol{\Omega}\boldsymbol{\omega} \\ -\boldsymbol{\omega}^T\boldsymbol{\Omega} & \frac{1}{s} + \boldsymbol{\omega}^T\boldsymbol{\Omega}\boldsymbol{\omega} \end{bmatrix}$$

$$s > 0$$

We first came across a similar transformation in a paper by Hosseini and Sra (2015), although their transformation was for the covariance matrix; this transform here can be derived from theirs using block matrix inversion. The variable $\frac{1}{s}$ is used to allow $\boldsymbol{\Gamma}$ to be any positive definite matrix (subject to constraints on $\boldsymbol{\Omega}, \boldsymbol{\omega}$). Otherwise, the value of the bottom right entry would be determined by the other entries. It should be clear that, as long as $\boldsymbol{\Omega}$ is positive definite, this transformation is one-to-one given a fixed s. If it is merely positive semidefinite, we cannot guarantee the global minimum is unique (but all local minima remain global); typically, solvers for Kronecker-sum-structured $\boldsymbol{\Omega}$ require it to be positive definite.

The strategy used to prove the chain of implications is to show that, when $s = 1$, $\mathfrak{A}$ and $\mathfrak{A}'$ correspond to the same optimization problem. Since $(\boldsymbol{\Omega}, \boldsymbol{\omega}, s) \to \boldsymbol{\Gamma}$ is a one-to-one mapping, as well as the fact that this mapping preserves the objective function's values, and finally the fact that $\mathbb{K}_{\mathbf{Mv}}$ is a linear subspace and hence preserves strict convexity of $\mathfrak{B}$, we have that there is exactly one minimum to $\mathfrak{B}$ which by the chain of implications is also the unique minimum of $\mathfrak{A}$.

All such claims are proven in the appendix. In particular, uniqueness of the global minimum holds when one uses convex regularization penalties on $\boldsymbol{\Omega}$, and furthermore holds (subject to a reasonable constraint on $\boldsymbol{\omega}$) when one restricts $\boldsymbol{\Omega}$ to be a low-rank matrix, such as that considered by a variant of GmGM (Andrew, Westhead, and Cutillo 2024b). The proof in the latter case is considerably more complicated, requiring mathematical machinery particularly from work by Hartwig (1976) and Holbrook (2018); it holds when we restrict $\boldsymbol{\omega}$ to be orthogonal to the nullspace of $\boldsymbol{\Omega}$ (a reasonable assumption since data generated from the distribution should also be orthogonal to the nullspace).

We claimed that our block coordinate descent algorithm converges to this unique optimal value. Block coordinate descent methods are not guaranteed to converge for smooth nonconvex problems. However, when there is a unique minimum for each block, as is the case here, block coordinate descent is known to converge (Tseng 2001; Luenberger and Ye 2008).

For simplicity, we report runtimes in which all dimensions of the input tensor with k dimensions are the same size n. The runtime complexity of our model is $O(I(G + M))$, where I is the number of iterations, G is the cost of graph estimation, and M is the cost of mean estimation. There are several $O(n^3 + n^k)$ algorithms for graph estimation, such as TeraLasso. Our mean estimation step can also naively be done in $O(n^3)$ due to it requiring matrix inverses[3]. Thus, complexity is $O(I(n^3 + n^k))$. It's natural to ask how large I needs to be; in the supplementary material, we show empirically that $I = 2$ or 3 is usually sufficient.

Results

Figure	Mean Distribution
Figure 1	Constant mean of 1
Figure 3	$m + \widetilde{\bigoplus}_\ell \boldsymbol{\mu}_\ell$ $\boldsymbol{\mu}_\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\ m \sim \mathcal{N}(0, 1)$
Figure 4 (left)	$\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \frac{1}{20}\mathbf{I})$
Figure 4 (right)	$\frac{\boldsymbol{\omega}}{14}$ $\boldsymbol{\omega}_{ij} \sim \text{Poisson}(10)$

Table 1: The distributions of the means in each of our synthetic data experiments, and the corresponding figures. Both mean distributions for Figure 4 have a variance of 0.05. This variance had to be smaller for the unstructured distributions, otherwise the noise would overwhelm the signal in all algorithms.

[3]Most graph estimation algorithms produce the eigendecomposition of $\boldsymbol{\Psi}_\ell$, and hence $\mathbf{A}_\ell$, as a subroutine. Assuming this is then passed to the mean estimation step, mean estimation is only $O(n^2)$.

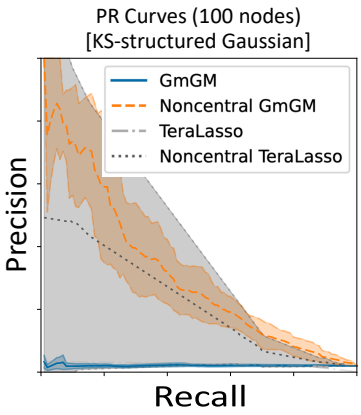

Figure 3: Precision and recall for on a synthetic dataset in which data was generated from a noncentral Kronecker-sum normal distribution. Noncentral TeraLasso's high susceptibility to changes in the regularization parameter led to our grid search being too coarse-grained during some trials.

All experiments were run on a 2020 MacBook Pro with an M1 chip and 8 GB of RAM. Our method was implemented using NumPy 1.25.2 (Harris et al. 2020) and SciPy 1.12.0 (Virtanen et al. 2020); for precision matrix routines, we used Greenewald, Zhou, and Hero's reference implementation of TeraLasso (2019) as well as GmGM 0.5.3. GmGM and TeraLasso are fairly similar algorithms; GmGM outputs positive definite matrices and does not use regularization, while TeraLasso uses regularization and does not require the outputs to be positive definite (as long as the Kronecker sum of the outputs is). By presenting results with each algorithm, we hope to show that our proposed mean estimation algorithm is not specific to any one method of precision estimation. All code is provided in the supplementary material.

In this section, we compare the performance of precision matrix algorithms with and without our mean estimation wrapper across a variety of test suites. We first compare performances on synthetic data, in which the graphs are generated from a Barabasi-Albert (power-law) distribution. We generate means for our synthetic data from a variety of distributions; see Table 1. We also compare them on Erdos-Renyi graphs, achieving similar results, but for conciseness we defer those results to the appendix. Next, we compare performances on the real-world COIL-20 video dataset ((Nene, Nayar, and Murase 1996)), demonstrating clear improvements. Finally, we show that, without properly taking into account the mean, precision matrix estimation will yield clearly wrong results on transcriptomics datasets such as the mouse embryo stem cell 'E-MTAB-2805' dataset (Buettner et al. 2015).

For synthetic data, we saw the dramatic deterioration in performance of algorithms that make the zero-mean assumption in Figure 1, while mean-corrected algorithms still perform well. Zero-mean algorithms also perform poorly when we generate data from the noncentral Kronecker-sum-structured Gaussian distribution; understandably, our correction is immune to this problem (Figure 3).

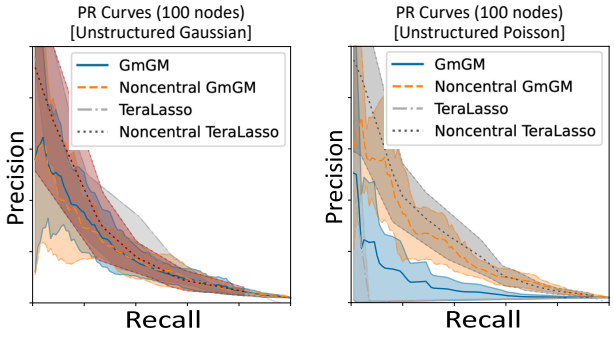

Figure 4: Precision and recall for on a synthetic dataset in which data was generated from a zero-mean Kronecker-sum normal distribution, with independent Gaussian (left) and Poisson (right) noise added to every element. Error bars are the best/worst performance over 10 trials; the center line is average performance.

Finally, we may wonder what happens if our mean has absolutely no structure, with every element of our generated matrix having a different, independent mean. We compared both Gaussian and Poisson-distributed random means (Figure 4) and found that there was still a drop-off in performance for Poisson means, but not nearly as dramatic as that seen in the other cases considered. There was no drop-off in the Gaussian case.

When we increased the variance of our means in our unstructured tests, both the standard algorithms and our mean-corrected algorithms dropped in performance equally. We suspect this is because, for large variances, the signal is overwhelmed by the noise. For small variances, since each row/-column of the mean is uncorrelated, their effects average to zero, and thus our mean-corrected algorithms will also estimate means of zero (making them identical to the uncorrected case). The Poisson distribution behaves slightly differently, as the mean is instead a constant nonzero number, and hence our mean-corrected algorithm is able to correctly account for this, whereas the standard algorithms experience degradation in performance similar to that of the constant-mean offset test (Figure 1).

The unstructured mean experiment has important consequences: if real-world data has an unstructured mean, then it is less important to correct for the means! However, it is likely that the mean does have a structured component, as each axis will have its own latent factors affecting the outcome in addition to any unstructured noise that may or may not exist. To demonstrate this, we run our algorithm on two real-world datasets, showing that our mean correction truly does improve performance in practice.

The COIL-20 dataset consists of 20 videos (each with 72 frames) of objects rotating 360 degrees; it has become somewhat customary for multi-axis methods to present their performance on the rubber duck object of this dataset. The dataset is chosen because of its ease to validate: frames that are close in time (i.e. adjacent) should be connected in the graph. We present our results graphically in Figure 6, from

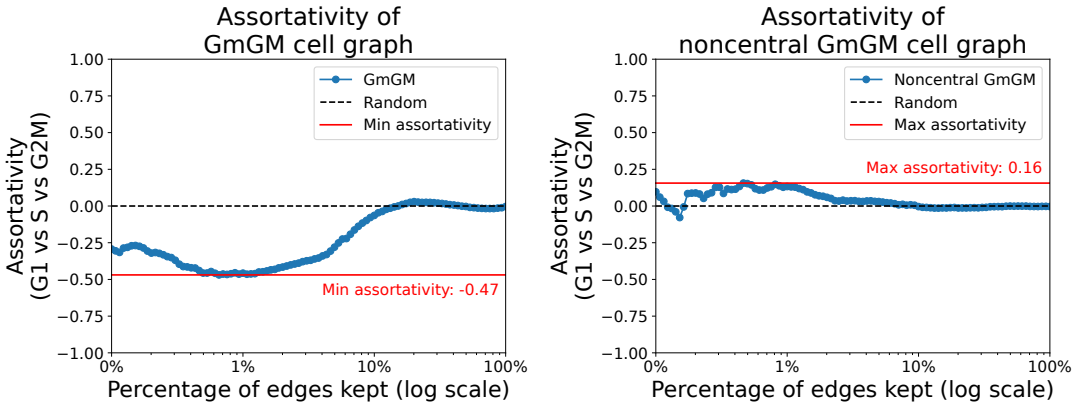

Figure 5: Assortativity of standard GmGM and our mean-corrected GmGM as we vary the number of edges kept, on the E-MTAB-2805 cell-cycle dataset. The cells were labeled by their stage in the cell cycle (G1/S/G2M).

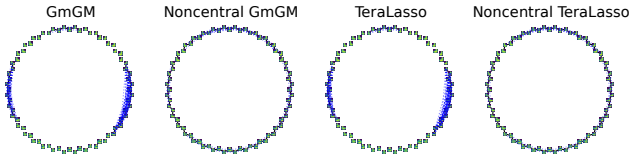

Figure 6: Each circle represents the results of an algorithm on the COIL-20 dataset; 'noncentral' denotes our mean estimation method. Dashed blue lines indicate connections between non-adjacent frames; solid black lines indicate adjacent connections.

which we can easily see the improvement. We pre-processed the data using the nonparanormal skeptic (Liu et al. 2012) to relax the Gaussian assumption of the models, and we chose thresholding/regularization parameters such that there would be approximately 144 edges.

We considered 'correct' edges to be those connecting adjacent frames or frames with a single frame in between. GmGM had 59/147 (40%) correct and TeraLasso had 55/142 (39%) correct. Once we wrap the methods in our mean estimation procedure, this rose to 127/147 (86%) and 126/141 (89%), respectively. Furthermore, all 'wrong' connections of our noncentral algorithms were nearly correct: with one exception, every connection was between frames with at most two frames between them. For the standard algorithms, more than a third of all edges failed this criteria. The noncentral algorithms also had high recall, at 88% for both noncentral GmGM and TeraLasso. The uncorrected algorithms had a recall of 41% and 38%. Overall, the results for this experiment are conclusively in favor of using our correction.

Our final experiment is on the E-MTAB-2805 scRNA-seq dataset. This dataset consists of 288 mouse embryo stem cells and 34,573 unique genes, with each cell being labeled by its stage in the cell cycle (G1, S, or G2M). It has previously been considered by the creators of scBiGLasso (Li et al. 2022), in which they limited it to a hand-selected set of 167 mitosis-related genes - we limited it to the same set of genes. We would expect that cells at the same stage in the same cycle should have some similarities, and hence should have some tendency to cluster together in our learned graphs.

We can use 'assortativity' as a measure for the tendency for cells within a stage to connect: assortativity ranges from -1 to 1. When it is positive, it indicates the tendency of nodes in the same cell cycle stage to connect to each other. If it is negative, cells in differing cell cycle stages tend to be connected to each other. If zero, there is no tendency; the connections are random.

Without correcting for the mean, we find that there is a very strong tendency for cells in the same cell cycle *to not connect* (assortativity = -0.47). This runs contrary to intuition: cells with some factor in common should not be caused by that same factor to look different. When correcting for the means, we see a modest positive tendency (assortativity = 0.16), which is more reasonable. Figure 5 shows the change in assortativity for GmGM, with and without our correction, as we vary the threshold of how many edges to keep.

Conclusion

Current Kronecker-structured models do not take into account the mean of the data. As one might expect, this can lead to wildly incorrect inferences (Figures 1 and 5). In fact, it can result in inferences directly opposite to reality: as thresholding/regularization increased, zero-mean algorithms decreased in both precision and recall, and identified an extremely strong (and incorrect) repulsive force between cells in the same stage of the cell cycle.

In this paper, we have demonstrated that this can be fixed by adding a mean-estimating wrapper around standard algorithms. While the means and precisions are nonseperable in the KS-structured case, and hence coincide with a nonconvex optimization problem, we proved that there is a unique global optimum, to which our algorithm converges. We implemented our algorithm as a 'drop-in' wrapper (for easy use with any standard precision-estimating algorithm), but with minor modifications it can be made to preserve the 'only-one-eigendecomposition' property of GmGM.

References

Allen, G. I.; and Tibshirani, R. 2010. Transposable regularized covariance models with an application to missing data imputation. *The annals of applied statistics*, 4(2): 764–790.

Andrew, B.; Westhead, D. R.; and Cutillo, L. 2024a. GmGM: a fast multi-axis Gaussian graphical model. Conference Name: The 27th International Conference on Artificial Intelligence and Statistics ISSN: 2640-3498 Meeting Name: The 27th International Conference on Artificial Intelligence and Statistics Place: València, Spain Publisher: Proceedings of Machine Learning Research.

Andrew, B.; Westhead, D. R.; and Cutillo, L. 2024b. Making Multi-Axis Gaussian Graphical Models Scalable to Millions of Samples and Features. ArXiv:2407.19892 [cs, q-bio, stat].

Buettner, F.; Natarajan, K. N.; Casale, F. P.; Proserpio, V.; Scialdone, A.; Theis, F. J.; Teichmann, S. A.; Marioni, J. C.; and Stegle, O. 2015. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature Biotechnology*, 33(2): 155–160. Number: 2 Publisher: Nature Publishing Group.

Dutilleul, P. 1999. The mle algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2): 105–123. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00949659908811970.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3): 432–441.

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

Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; and Oliphant, T. E. 2020. Array programming with NumPy. *Nature*, 585(7825): 357–362. Number: 7825 Publisher: Nature Publishing Group.

Hartwig, R. E. 1976. Singular Value Decomposition and the Moore–Penrose Inverse of Bordered Matrices. *SIAM Journal on Applied Mathematics*, 31(1): 31–41. Publisher: Society for Industrial and Applied Mathematics.

Holbrook, A. 2018. Differentiating the pseudo determinant. *Linear Algebra and its Applications*, 548: 293–304.

Hosseini, R.; and Sra, S. 2015. Matrix Manifold Optimization for Gaussian Mixtures. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Kalaitzis, A.; Lafferty, J.; Lawrence, N. D.; and Zhou, S. 2013. The Bigraphical Lasso. In *Proceedings of the 30th International Conference on Machine Learning*, 1229–1237. PMLR. ISSN: 1938-7228.

Kolda, T. G.; and Bader, B. W. 2009. Tensor Decompositions and Applications. *SIAM Review*, 51(3): 455–500. Publisher: Society for Industrial and Applied Mathematics.

Li, S.; López-García, M.; Lawrence, N. D.; and Cutillo, L. 2022. Scalable Bigraphical Lasso: Two-way Sparse Network Inference for Count Data. ArXiv:2203.07912 [cs, stat].

Liu, H.; Han, F.; Yuan, M.; Lafferty, J.; and Wasserman, L. 2012. The nonparanormal SKEPTIC. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 1415–1422.

Luenberger, D. G.; and Ye, Y. 2008. *Linear and Nonlinear Programming*, volume 116 of *International Series in Operations Research & Management Science*. New York, NY: Springer US. ISBN 978-0-387-74502-2 978-0-387-74503-9.

McInnes, L.; Healy, J.; and Melville, J. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv:1802.03426 [cs, stat].

Nene, S. A.; Nayar, S. K.; and Murase, H. 1996. Columbia Object Image Library (COIL-20).

Tseng, P. 2001. Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications*, 109(3): 475–494.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, I.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; and van Mulbregt, P. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3): 261–272. Number: 3 Publisher: Nature Publishing Group.

Wang, Y.; Jang, B.; and Hero, A. 2020. The Sylvester Graphical Lasso (SyGlasso). In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 1943–1953. PMLR. ISSN: 2640-3498.