The Strong Product Model for Network Inference without Independence Assumptions

Bailey Andrew
University of Leeds

David R. Westhead
University of Leeds

Luisa Cutillo
University of Leeds

Abstract

Multi-axis graphical modelling techniques allow us to perform network inference without making independence assumptions. This is done by replacing the independence assumption with a weaker assumption about the interaction between the axes; there are several choices for which assumption to use. In single-cell RNA sequencing data, genes may interact differently depending on whether they are expressed in the same cell, or in different cells. Unfortunately, current methods are not able to make this distinction. In this paper, we address this problem by introducing the strong product model for Gaussian graphical modelling.

1 INTRODUCTION

It is often the case that one wishes to learn a network ('graph') from their data. A common example is learning a gene regulatory network from a single-cell RNA-sequencing (scRNA-seq) dataset. This type of dataset takes the form of a matrix whose rows represent cells and whose columns represent genes, and whose entries quantify how much a given gene is expressed in a given cell. There are numerous methods that perform network inference across a range of scenarios, such as the Graphical Lasso ('*glasso*', Friedman et al., 2008), neighborhood selection (Meinshausen & Bühlmann, 2006), and SIMoNe (Ambroise et al., 2009).

However, such methods make an independence assumption: learning the gene regulatory network requires assuming that the cells in your body do not interact! This assumption is clearly faulty. There are two fundamentally different types of dependencies existing in our dataset - the dependencies between genes, and the dependencies between cells. More generally, matrix-variate datasets have row-dependencies and column-dependencies.

We can replace the independence assumption with a weaker assumption about how these two types of dependencies interact. We can imagine a larger graph of dependencies between matrix elements, which is some product graph of the row-dependencies and column-dependencies - for example, by using the Cartesian product. We will make this more formal in Section 2. In scRNA-seq data, the larger graph corresponds to connections between (cell, gene) pairs.

There are many choices of graph product to use, but none of them make the distinction between within-row and between-row interactions. A gene X might upregulate another gene Y when both are present in the same cell, but not have any affect on the expression of Y in some other cell (due to the gene transcript never leaving the cell membrane). Likewise, gene X may *only* affect the expression of gene Y in different cells; perhaps gene X encodes a surface protein or causes an enzyme to be secreted which affects the behavior of nearby cells.

By taking the differences between within-row and between-row interactions into account, we can arrive at more biologically meaningful results. In this paper, we develop a model, inspired by the 'strong product' graph product, that successfully differentiates and learns these types of dependencies.

2 BACKGROUND

One of the most popular methods of network inference is the Graphical Lasso (Friedman et al., 2008). The general idea is to perform regularized maximum likelihood estimation for the inverse covariance matrix (the 'precision matrix') of a Gaussian distribution. Accordingly, the method assumes your data is Gaussianly distributed, but this can be replaced with the

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

Decomposition	Definition	Justification	Rule
Kronecker Product	$\mathbf{\Psi}_{\text{rows}} \otimes \mathbf{\Psi}_{\text{cols}}$	Both	$\text{row}[x] \sim \text{row}[y] \wedge \text{col}[x] \sim \text{col}[y]$
Kronecker Sum	$\mathbf{\Psi}_{\text{rows}} \oplus \mathbf{\Psi}_{\text{cols}}$	Graph Product	either $\text{row}[x] \sim \text{row}[y] \wedge \text{col}[x] = \text{col}[y]$ or $\text{row}[x] = \text{row}[y] \wedge \text{col}[x] \sim \text{col}[y]$
Sylvester Product	$(\mathbf{\Psi}_{\text{rows}} \oplus \mathbf{\Psi}_{\text{cols}})^2$	Generative Process	N/A

Table 1: A comparison of the three decompositions in common use. The Kronecker product has a long history of use, but its use in graphical modelling comes from Tsiligkaridis et al. (2012); the other two are from Kalaitzis et al. (2013) and Wang et al. (2020), respectively. The 'rule' column represents the procedure by which to derive the connections between matrix elements x and y in the full graph $\mathbf{\Omega}$; $\sim$ is used as a shorthand for 'is connected to'. The Sylvester Product can also be expressed as a graph product, but one needs to use second-order connections of the axis graphs to do so; its primary motivation is the generative process that yields it.

weaker 'Gaussian copula' assumption through preprocessing tools such as the nonparanormal skeptic (Liu et al., 2012). Intuitively, the nonparanormal skeptic allows these techniques to be used on data with arbitrary marginals, as long as relationships between the datapoints 'behave Gaussianly'.

$$\underset{\mathbf{\Psi}}{\text{argmin}} \;\; \log|\mathbf{\Psi}| - \text{tr}\left[\mathbf{\Psi}\mathbf{x}\mathbf{x}^T\right] + \rho\|\mathbf{\Psi}\|_{1,od} \qquad \text{(Graphical Lasso)}$$

The choice to model the data as a Gaussian is not just for computational convenience; it has the convenient theoretical property that $\mathbf{\Psi}$ encodes the *graph of conditional dependencies*. Let x, y be two variables from a dataset $\mathcal{D}$ ($\mathcal{D}_{\backslash x \backslash y} = \mathcal{D} - \{x, y\}$), then we have the following relation for Gaussian data:

$$\mathbb{P}\left(x, y|\mathcal{D}_{\backslash x \backslash y}\right) = \mathbb{P}\left(x|\mathcal{D}_{\backslash x \backslash y}\right)\mathbb{P}\left(y|\mathcal{D}_{\backslash x \backslash y}\right) \iff \mathbf{\Psi}_{xy} = 0$$

Conditional dependencies encode the concept of independence after conditioning out the rest of the dataset. Intuitively, it captures the *direct relation* of two variables on each other - this is in contrast to correlation, which captures indirect relations as well. The classic example is the correlation between shark attacks and ice cream sales due to the confounding factor of it being summer; conditioning out the season, the two variables become independent.

Graphs of conditional dependencies should typically be sparse, by nature of their capturing the direct effect. This forms part of the justification for the use of regularization penalties, but penalties are also necessary since in the small-sample scenario the unregularized maximum likelihood estimate will typically not exist.

A drawback of the Graphical Lasso is that it assumes independence of the samples. In reality, there may be dependencies between rows $\mathbf{\Psi}_{\text{rows}}$ and dependencies between columns $\mathbf{\Psi}_{\text{cols}}$. To solve this, we can 'vectorize' our data and consider the graph of (row, column) pairs (i.e. the graph connecting all matrix elements).

The vectorization of a $d_1 \times d_2$ matrix $\mathbf{X}$, denoted $\text{vec}[\mathbf{X}]$, is the 'flattening' of the matrix into a length $d_1 d_2$ vector. Whether this flattening is done in a rows-first or columns-first manner does not matter as long as it is done consistently. We can view our data as being a set of d_1 length-d_2 samples $\mathbf{x}_i$ or alternatively as a $d_1 \times d_2$ matrix $\mathbf{X}$.

$$\mathbf{x}_i \underset{i.i.d.}{\sim} \mathcal{N}\left(\mathbf{0}, \mathbf{\Psi}_{\text{cols}}^{-1}\right) \quad \text{(independence assumption)}$$
$$\text{vec}[\mathbf{X}] \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Omega}^{-1}\right) \quad \text{(no assumption)}$$

The first model considers each row to be a separate sample, and each column corresponds to an element of this vector. The second model considers our entire dataset to be a single sample, and each element of our dataset is a (row, column) pair. We can then discuss the dependencies of such pairs, $\mathbf{\Omega}$, rather than being limited to column-column dependencies.

While this removes the independence assumption, it comes with its own drawbacks. In the independence scenario, we have d_1 samples of d_2 features, but in the non-independence scenario we have 1 sample of $d_1 d_2$ features! $\mathbf{\Omega}$ has $O(d_1^2 d_2^2)$ parameters to learn, and we must do so with a single sample. Not only is this intractable from a computational complexity perspective, but also from a statistical perspective; how could we hope to have any degree of confidence in our results?

To resolve the problem, we can impose some structure on $\mathbf{\Omega}$, such as it being composed out of the axis-wise graphs: $\mathbf{\Omega} = \zeta(\mathbf{\Psi}_{\text{rows}}, \mathbf{\Psi}_{\text{cols}})$. Several choices for ζ exist, which we describe in Table 1. They can broadly be separated into two categories; those based on graph products and those based on generative processes. They all use the Kronecker product $\otimes$ as a building block, and some further use the Kronecker

sum $\oplus$: $\mathbf{A} \oplus \mathbf{B} = \mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{B}$. Our work exists within the framework of Kronecker-separable modelling; for an overview, see the review by Wang et al. (2022).

Of the decompositions in prior work, two admit a straightforward interpretation as an operation on graphs: the Kronecker product corresponds to the tensor product of graphs, while the Kronecker sum corresponds to the Cartesian product. The Kronecker product only takes into account between-row connections, i.e. to determine if two genes interact, we only consider the relationship of their expressions in different cells. The Kronecker sum only takes into account within-row connections: to determine if two genes interact, we only consider the relationship of their expressions in the same cell. Ideally, we would be able to capture both types of relationships.

The Sylvester product, when forced into the framework of graph products, does capture both types of relationships - but only through 'second-order' connections in the graph (by 'second order', we mean that $x \sim y \sim z$ implies x and z have a second-order connection, where $\sim$ is notation for 'is connected to'). This is difficult to interpret through the lens of conditional dependency, where we are after the direct connections.

3 OUR CONTRIBUTIONS

In this paper, we introduce a new graph decomposition framework for conditional dependency estimation (Section 4.2). It is based on the 'strong product' from graph theory, and hence we call our model the 'strong product model'. We show that it captures and distinguishes between both the within-row and between-row dependencies, which is particularly beneficial in use cases such as scRNA-seq. We demonstrate that one can perform inference with this model efficiently (Section 4.3), and prove that there exists a unique solution to the problem (Section 4.7). Finally, we demonstrate practical utility on real-world datasets (Sections 5.2 and 5.3). These results hold *even in the single-sample case.*

4 METHODOLOGY

This section will focus on the derivation of the strong product model. We will first introduce our notation in Section 4.1. Then, we will motivate the structure of our model in Section 4.2. Next, we will derive an efficient maximum likelihood estimator (MLE) in Section 4.3, and finally in Sections 4.6 and 4.7 we will prove key properties of our algorithm to guarantee that its solutions are reasonable and well-founded.

4.1 Notation

In this paper, lower-case letters a represent scalars, lower-case bold $\mathbf{v}$ represent vectors, and upper-case bold $\mathbf{M}$ represent matrices. $\mathbf{I}_a$ represents an $a \times a$ identity matrix, although we will frequently omit the subscript as the shape will be clear from context. d_1 will represent the number of rows in the dataset, and d_2 will represent the number of columns.

We will denote the precision matrix of the rows as $\mathbf{\Psi}_1$. The precision matrix of the columns will be split into two parts, those that focus on the within-row dependencies $\mathbf{\Psi}_{2,w}$ and the between-row dependencies $\mathbf{\Psi}_{2,b}$. It will often be convenient to consider the matrix $\mathbf{\Theta} = \mathbf{\Psi}_{2,b} + \mathbf{I}$. We will let $\mathbf{\Omega}$ represent the whole precision matrix, which we are assuming can be factorized as $\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w} + \mathbf{\Psi}_1 \otimes \mathbf{\Psi}_{2,b}$.

We also define the empirical gram matrices $\mathbf{S}_1, \mathbf{S}_2$, which will represent sufficient statistics for our model. These are defined as $\mathbf{S}_1 = \mathbf{X}\mathbf{X}^T$ and $\mathbf{S}_2 = \mathbf{X}^T\mathbf{X}$, where $\mathbf{X}$ is the data matrix. Finally, we define the stridewise trace tr^{d_1} and the blockwise trace tr_{d_2}, which show up frequently when working with derivatives involving log-determinants of Kronecker products (below, $\mathbf{J}^{ij}$ is the matrix of zeros except for a 1 at (i, j)).

$$\mathrm{tr}^{d_1}\left[\mathbf{A}\right]_{ij} = \mathrm{tr}\left[\mathbf{A}\left(\mathbf{I} \otimes \mathbf{J}^{ij}\right)\right]$$
$$\mathrm{tr}_{d_2}\left[\mathbf{A}\right]_{ij} = \mathrm{tr}\left[\mathbf{A}\left(\mathbf{J}^{ij} \otimes \mathbf{I}\right)\right]$$

4.2 The Strong Product Model

As mentioned in Section 2, the Kronecker sum $\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_2$ and Kronecker product $\mathbf{\Psi}_1 \otimes \mathbf{\Psi}_2$ only take into account the within-row dependencies (Kronecker sum) and between-row dependencies (Kronecker product), rather than both simultaneously. It is tempting to add these together: $\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_2 + \mathbf{\Psi}_1 \otimes \mathbf{\Psi}_2$, and the instinct turns out to be correct. This corresponds to the *strong product* from graph theory. If we split $\mathbf{\Psi}_2$ into separate matrices, each measuring either the within- or between- row dependencies, then we arrive at $\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w} + \mathbf{\Psi}_1 \otimes \mathbf{\Psi}_{2,b}$. It will turn out to be more convenient to parameterize this in terms of $\mathbf{\Theta} = \mathbf{\Psi}_{2,b} + \mathbf{I}$, i.e. as $\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}$.

$$\mathrm{vec}\left[\mathbf{X}\right] \sim \mathcal{N}\left(\mathbf{0}, \left(\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}\right)^{-1}\right) \quad \text{(Strong product model)}$$

$$\mathrm{NLL} = -\frac{1}{2}\log\left|\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}\right| + \frac{1}{2}\mathrm{tr}\left[\mathbf{\Psi}_{2,w}\mathbf{S}_2\right] + \frac{1}{2}\mathrm{tr}\left[\mathbf{\Psi}_1\mathbf{X}\mathbf{\Theta}\mathbf{X}^T\right] \quad (1)$$

We will restrict all $\mathbf{\Psi}$ to be positive definite. This is a common, but not universal, restriction. Precision matrices must be positive definite (so $\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}$ must be), but in theory we do not have to enforce positive definiteness of the factor matrices. The difference can have theoretical implications - not everything that is representable as a composition of non-pos.-def. matrices is representable as a composition of pos. def. matrices (Song & Hero, 2023). We enforce positive definiteness as it leads to better interpretability (the matrices correspond to real precision matrices).

Positive-definiteness of the factor matrices comes at a price. Any sparsifying regularizer (whether it be Lasso or a non-convex regularizer such as SCAD (Fan & Li, 2001) or MCP (Zhang, 2010)) have proximal operators that output indefinite matrices for positive definite inputs - both in theory (Guillot & Rajaratnam, 2015) and in practice (Xue et al., 2012). Furthermore, the unregularized model is not convex, but it is *geodesically convex* (see Section 4.7 for details), whereas the Lasso penalty is convex, but not geodesically convex (Duembgen & Tyler, 2016). This mismatch makes establishing useful properties such as existence and uniqueness harder. Here, we do not consider sparsifying regularizers.

In this paper, we follow Deng and Tsui (2013) by imposing a Frobenius norm penalty $\|\log \mathbf{\Gamma}\|_F^2$ on the logarithms of our parameters. This penalty corresponds to the geodesic distance between our parameter and the identity matrix (Moakher (2005), Eq. 2.9); it has the effect of normalizing our matrix towards assuming independence, and thus is a natural penalty to impose.

4.3 Derivation of the Method

To minimize the NLL, we will use Riemannian gradient descent. The problem is not convex, but it is geodesically convex. See Algorithm 1 for pseudocode. We chose to parameterize our problem in terms of $\mathbf{\Theta}$, as it consistently led to simpler expressions.

Without any simplification, these gradients require computing the inverse of the strong product. The computation of such a matrix is prohibitively expensive ($O(d_1^3 d_2^3)$ runtime and $O(d_1^2 d_2^2)$ space). For Kronecker-sum methods, this is often circumvented by eigendecomposing the inputs (Greenewald et al., 2019; Li et al., 2022). The eigenvectors can be extracted from the Kronecker sum, reducing the problem to the computation of the inverse of a (large) diagonal matrix.

The strong product is not amenable to the same trick. For other models, there has always been only a single parameter for each axis. In our case, the columns have two parameters, $\mathbf{\Psi}_{2,w}$ and $\mathbf{\Theta}$. They must be *simultaneously diagonalized*. Diagonalization by orthonormal vectors is only possible for commuting matrices, which rarely holds. However, for any pair of positive definite matrices, they can be *simultaneously diagonalized via congruence*, i.e. $\mathbf{\Theta} = \mathbf{P}\mathbf{P}^T$ and $\mathbf{\Psi}_{2,w} = \mathbf{P}\mathbf{D}\mathbf{P}^T$.

$\mathbf{P}^T \neq \mathbf{P}^{-1}$, unlike in the case of eigendecomposition. Regardless, they can still be factored out of the strong product. In fact, by Lemma 2 in Dahl et al. (2013), we can even factor $\mathbf{P}$ and $\mathbf{V}$ (where $\mathbf{\Psi}_1 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$) out of the stridewise-blockwise trace. This yields the following gradients, in which the portion corresponding to the regularizer is shaded grey.

$$\frac{d}{d\mathbf{\Psi}_1}\text{NLL} = -\mathbf{V}\text{tr}_{d_2}\left[(\mathbf{\Lambda} \oplus \mathbf{D})^{-1}\right]\mathbf{V}^T + \mathbf{X}\mathbf{\Theta}\mathbf{X}^T + 2\rho\mathbf{\Psi}_1^{-1}\log\mathbf{\Psi}_1 \quad \text{(grad1)}$$

$$\frac{d}{d\mathbf{\Psi}_{2,w}}\text{NLL} = -\mathbf{P}^{-T}\text{tr}^{d_1}\left[(\mathbf{\Lambda} \oplus \mathbf{D})^{-1}\right]\mathbf{P}^{-1} + \mathbf{S}_2 + 2\rho\mathbf{\Psi}_{2,w}^{-1}\log\mathbf{\Psi}_{2,w} \quad \text{(grad2)}$$

$$\frac{d}{d\mathbf{\Theta}}\text{NLL} = -\mathbf{P}^{-T}\text{tr}^{d_1}\left[(\mathbf{\Lambda} \oplus \mathbf{D})^{-1}(\mathbf{\Lambda} \otimes \mathbf{I})\right]\mathbf{P}^{-1} + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X} + 2\rho\mathbf{\Theta}^{-1}\log\mathbf{\Theta} \quad \text{(grad3)}$$

4.4 Geodesic Convexity

Our optimization problem is not convex, due to the inclusion of a Kronecker product of two parameters. However, it is *geodesically convex*, or *g-convex*. In brief, geodesic convexity is the analogous concept to standard convexity when optimization takes place on a manifold (such as the set of positive-definite matrices), and many useful properties are preserved - in particular, the fact that sums of g-convex functions are g-convex and that all local minima are global.

The field of geodesic convexity is too broad to overview here, so we refer the reader to the relevant literature (Absil et al., 2009; Vishnoi, 2018). There are multiple metrics one could apply to the set of positive definite matrices (see (Han et al., 2021) for a comparison), but by far the most common is the one with geodesic distance $\gamma(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2) = \mathbf{\Sigma}_1^{\frac{1}{2}}\left(\mathbf{\Sigma}_1^{\frac{-1}{2}}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{\frac{-1}{2}}\right)^t\mathbf{\Sigma}_1^{\frac{1}{2}}$. We'll denote it $\mathbb{S}_{++}$, and denote the geometry of tuples of positive definite matrices (each equipped with the same geodesic along the relevant coordinates) as $\mathbb{S}_{++}^{\otimes}$. These geometries have several key properties:

1. Traces $\text{tr}\left[\mathbf{\Sigma}\mathbf{X}\right]$ are g-convex ($\mathbb{S}_{++}$)
2. Log determinants $-\log|\mathbf{\Sigma}|$ are g-convex ($\mathbb{S}_{++}$)
3. Orthogonal invariance and convexity in $\log\mathbf{\Sigma}$ implies g-convexity ($\mathbb{S}_{++}$) (Yi & Tyler, 2020)

4. $\|\log \mathbf{\Sigma}\|_F^2$ is *strictly* g-convex ($\mathbb{S}_{++}$) (Yi & Tyler, 2020)

5. Log determinants of sums $-\log|\sum_i \mathbf{\Sigma}_i|$ are g-convex ($\mathbb{S}_{++}^{\otimes}$) (Wiesel, 2012a)

6. If $f(\mathbf{\Omega})$ is g-convex ($\mathbb{S}_{++}$), then $g(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2) = f(\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2)$ is g-convex ($\mathbb{S}_{++}^{\otimes}$) (Wiesel, 2012b)

Let us parameterize our problem as $\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{\Gamma} \otimes \mathbf{\Psi}_{2,b}$, where $\mathbf{\Gamma}$ is constrained to be the identity matrix. (such a constraint is g-convex). Then, the negative log-likelihood is a sum of traces (g-convex), terms of the form $\|\log \mathbf{\Sigma}\|_F^2$ (strictly g-convex), and a log determinant of a sum of Kronecker products (g-convex). Thus, our problem is g-convex.

In fact, the problem is *strongly* g-convex, as $\|\log \mathbf{\Sigma}\|_F^2$ is strongly g-convex (See Equation 18 of Darmwal and Rajawat (2023)). Accordingly, we can expect fast (linear) convergence from gradient descent. This can be seen in Figure 1.

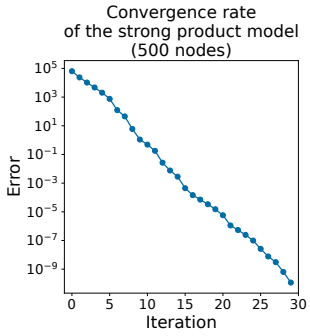

Figure 1: Rate of decrease in error using Algorithm 1. Ran on synthetic data, see Section 5.1 for generation details.

4.5 Riemannian Gradient Descent

Rather than using traditional gradient descent, the geodesic convexity motivates the use of *Riemannian* gradient descent. This is analogous to traditional gradient descent, but follows the direction of steepest descent under the definition of 'steepest' intrinsic to the manifold (in this case, the set of positive semi-definite matrices). We refer the reader to several papers which we found beneficial for understanding how to utilize the method in practice, in particular the following: Hosseini and Sra (2015), Duembgen and Tyler (2016), Jeuris et al. (2012), and Darmwal and Rajawat (2023).

Here, we will only state the changes that need to be made to gradient descent to fit the Riemannian context. Rather than using the 'Euclidean' gradient of f at a point $\mathbf{X}$, $\text{grad}_{\mathbf{X}}^{\mathbb{E}} f = \frac{d}{d\mathbf{X}} f(\mathbf{Y})\big|_{\mathbf{Y}=\mathbf{X}}$,

Algorithm 1 Strong Product Model MLE

Require: Initial guesses $\mathbf{\Psi}_1^{(0)}, \mathbf{\Theta}^{(0)}, \mathbf{\Psi}_{2,w}^{(0)}$, data $\mathbf{X}$
Require: Regularization parameters $\rho_{\psi_1}, \rho_\theta, \rho_{\psi_{2,w}}$
Require: Line search parameters $\eta_{\text{init}}, \beta, \tau$
  while not converged **do**
    #Riemannian Gradient Descent
    $\mathcal{P} \leftarrow \{\mathbf{\Psi}_1, \mathbf{\Theta}, \mathbf{\Psi}_{2,w}\}$
    for parameter $\mathbf{\Gamma} \in \mathcal{P}$ **do**
      $\mathbf{G} \leftarrow$ gradient at $\mathbf{\Gamma}^{(t)}$ (grad1)/(grad2)/(grad3)
      $\overline{\mathbf{G}}_{\mathbf{\Gamma}} \leftarrow \mathbf{\Gamma}^{(t)} \mathbf{G} \mathbf{\Gamma}^{(t)}$
      $\mathbf{\Gamma}^{(t+1)} \leftarrow \mathbf{\Gamma}^{(t)} e^{-\eta_t \mathbf{\Gamma}_{(t)}^{-1} \overline{\mathbf{G}}_{\mathbf{\Gamma}}}$
    end for
    #Riemannian Armijo Line Search
    $\eta_t \leftarrow \eta_{\text{init}}$
    while $f\left(\mathcal{P}^{(t+1)}\right) > f\left(\mathcal{P}^{(t)}\right) - \sum_{\mathbf{\Gamma}} \eta_t \beta \|\overline{\mathbf{G}}_{\mathbf{\Gamma}}\|_F^2$ **do**
      $\eta_t \leftarrow \tau \eta_t$
      $\mathbf{\Gamma}^{(t+1)} \leftarrow \mathbf{\Gamma}^{(t)} e^{-\eta_t \mathbf{\Gamma}_{(t)}^{-1} \overline{\mathbf{G}}}$
    end while
  end while

the correct gradient is $\text{grad}_{\mathbf{X}}^{\mathbb{S}_{++}} f = \mathbf{X}\left(\text{grad}_{\mathbf{X}}^{\mathbb{E}} f\right)\mathbf{X}$. The second required concept is the 'retraction' at $\mathbf{X}$, $\text{R}_{\mathbf{X}}^{\mathbb{S}_{++}}(\boldsymbol{\xi}) = \mathbf{X} e^{\mathbf{X}^{-1}\boldsymbol{\xi}}$. It maps steps in a direction (typically the gradient) to a position on the relevant geodesic. See Sra and Hosseini (2015) for details.

One step of Riemannian gradient descent with stepsize η can be expressed as $\mathbf{X}_{t+1} = \text{R}_{\mathbf{X}_t}^{\mathbb{S}_{++}}\left(-\eta \, \text{grad}_{\mathbf{X}_t}^{\mathbb{S}_{++}} f\right)$. To determine an effective step size, we use backtracking line search with Armijo's rule (Armijo, 1966). Line search is more expensive in the Riemannian case, due to the need to repeatedly compute the retraction, which involves matrix exponentiation; modifications can be made to mitigate this (Yamakawa et al., 2023), but for pedagogical reasons we shall stick with the most basic form.

4.6 Identifiability

Kronecker models often are not identifiable. For our *penalized* problem, this is not an issue; we will see in Section 4.7 that the minimum is unique. However, this is due to the strict g-convexity of our penalty function; it is still of interest to understand the extent to which our original problem is identifiable. Below, we list the non-identifiabilities of common Kronecker decompositions, where c is any constant.

$$\mathbf{\Psi}_1 \otimes \mathbf{\Psi}_2 = \frac{1}{c}\mathbf{\Psi}_1 \otimes c\mathbf{\Psi}_2 \qquad \text{(Kronecker Product)}$$

$$\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_2 = (\mathbf{\Psi}_1 + c\mathbf{I}) \oplus (\mathbf{\Psi}_2 - c\mathbf{I}) \qquad \text{(Kronecker Sum)}$$

$$(\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_2)^2 = ((\mathbf{\Psi}_1 + c\mathbf{I}) \oplus (\mathbf{\Psi}_2 - c\mathbf{I}))^2 \quad \text{(Sylvester Sum)}$$

It is not immediately obvious what the non-identifiabilities of our model are. We summarize them with the following proposition:

Proposition 1. *Let* $(\mathbf{\Psi}_1, \mathbf{\Psi}_{2,w}, \mathbf{\Theta})$ *and* $(\overline{\mathbf{\Psi}}_1, \overline{\mathbf{\Psi}}_{2,w}, \overline{\mathbf{\Theta}})$ *be two different parameterizations. Then, they have the same strong product if and only if one of the following conditions hold for some* c_1, c_2*:*

1. $\mathbf{\Psi}_1 = c_1\mathbf{I}$, $\overline{\mathbf{\Psi}}_1 = c_2\mathbf{I}$, $c_1\mathbf{\Theta} + \mathbf{\Psi}_{2,w} = c_2\overline{\mathbf{\Theta}} + \overline{\mathbf{\Psi}}_{2,w}$
2. $\mathbf{\Psi}_1 = c_1\mathbf{I} + c_2\overline{\mathbf{\Psi}}_1$, $\overline{\mathbf{\Theta}} = c_2\mathbf{\Theta}$, $\overline{\mathbf{\Psi}}_{2,w} = \mathbf{\Psi}_{2,w} + c_1\mathbf{\Theta}$

Case 1 tells us that we cannot identify differences between within-row and between-row column dependencies when the rows are independent (when $\mathbf{\Psi}_1 \propto \mathbf{I}$). This is to be expected; it doesn't make sense to talk of between-row dependencies when the rows are independent. Additionally, when the rows are independent, algorithms that assume independence, such as *glasso*, should be preferred.

On the surface, case 2 seems much more complicated; it is capturing the mixing of Kronecker product and Kronecker sum non-identifiabilities. Consider that, when $c_1 = 0$, this reduces to Kronecker product non-identifiability. When $c_2 = 1$, it's not quite the Kronecker sum non-identifiability but is analogous to it.

Case 2 also has an intuitive interpretation: as the variance of our rows decreases (c_1 is negative), our within-row graph looks more like our between-row graph (since approximately $\overline{\mathbf{\Psi}}_{2,w} \propto \mathbf{\Theta}$ for large negative c_1). This makes sense intuitively: if there were no row-variance (i.e. the columns are all identical), then there is no difference between between-row and within-row relationships; the value in a cell in a given column is the same regardless of which row it is in.

Proof.

$$\begin{aligned}\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w} &= \overline{\mathbf{\Psi}}_1 \otimes \overline{\mathbf{\Theta}} + \mathbf{I} \otimes \overline{\mathbf{\Psi}}_{2,w} \\ \iff \mathbf{\Psi}_1 \otimes \mathbf{\Theta} &= \overline{\mathbf{\Psi}}_1 \otimes \overline{\mathbf{\Theta}} + \mathbf{I} \otimes \left(\overline{\mathbf{\Psi}}_{2,w} - \mathbf{\Psi}_{2,w}\right)\end{aligned}$$

Note that the left hand side of the equation is a rank-1 Kronecker decomposition and the right hand side is a rank-2 Kronecker decomposition (i.e. it is a sum of two Kronecker products). This can only hold if either the left terms $\{\mathbf{\Psi}_1, \overline{\mathbf{\Psi}}_1, \mathbf{I}\}$, or alternatively if the right terms $\{\mathbf{\Theta}, \overline{\mathbf{\Theta}}, \left(\overline{\mathbf{\Psi}}_{2,w} - \mathbf{\Psi}_{2,w}\right)\}$, form a rank-1 linearly dependent set. Let's assume the first set is linearly dependent. Then, they must all be multiples of the identity matrix:

$$\begin{aligned}c_1\mathbf{I} \otimes \mathbf{\Theta} &= c_2\mathbf{I} \otimes \overline{\mathbf{\Theta}} + \mathbf{I} \otimes \left(\overline{\mathbf{\Psi}}_{2,w} - \mathbf{\Psi}_{2,w}\right) \\ \iff c_1\mathbf{\Theta} + \mathbf{\Psi}_{2,w} &= c_2\overline{\mathbf{\Theta}} + \overline{\mathbf{\Psi}}_{2,w}\end{aligned}$$

This yields case 1. Now let's consider what happens when the right terms are linearly dependent, i.e. if $\overline{\mathbf{\Psi}}_{2,w} - \mathbf{\Psi}_{2,w} = c_1\mathbf{\Theta}$, $\overline{\mathbf{\Theta}} = c_2\mathbf{\Theta}$. Then we must have:

$$\mathbf{\Psi}_1 = c_2\overline{\mathbf{\Psi}}_1 + c_1\mathbf{I}$$

□

These non-identifiabilities could be fixed with a trace constraint $\text{tr}[\mathbf{\Gamma}] = \tau_{\mathbf{\Gamma}}$ for some chosen constant $\tau_{\mathbf{\Gamma}}$ for every factor matrix $\mathbf{\Gamma}$. Furthermore, the *support* of $\mathbf{\Psi}_1$ and $\mathbf{\Theta}$ is preserved; they always represents the same graph. Since we are penalizing our model by a strictly g-convex function, **it is not affected by these non-identifiabilities**.

4.7 Existence and Uniqueness

Theorem 1. *There exists a unique solution to the penalized optimization problem given by Equation 1 equipped with the penalty* $\rho\|\mathbf{\Gamma}\|_F^2$ *in each of its parameters.*

Proof. Uniqueness follows from the strict g-convexity of our objective. Existence follows from the fact that our function is *g-coercive*. G-coercitivity guarantees for g-convex functions that there exists a compact set of minimizers (Duembgen & Tyler, 2016; Sra & Hosseini, 2015), and in fact for any g-convex function $f(\mathbf{\Sigma})$, $f(\mathbf{\Sigma}) + \|\log \mathbf{\Sigma}\|_F^2$ is g-coercive (see Lemma 4.7 of Duembgen and Tyler (2016)). Existence and uniqueness holds both coordinate-wise and jointly. □

5 EXPERIMENTAL RESULTS

We will now validate our methodology on synthetic and real-world datasets. In Section 5.1, we will demonstrate the performance of our algorithm on within-distribution synthetic data. Next, in Section 5.2, we will validate our model on a real-world dataset for which the results are easily interpretable without specialized domain knowledge. Finally, in Section 5.3, we will validate our model on a real-world scRNA-seq dataset. For real-world datasets $\mathbf{D}$, we used as input to the models $\mathbf{D} - \frac{1}{d_1 d_2}\mathbf{1}^T\mathbf{D}\mathbf{1}$, to lessen the impact of the zero-mean assumption. Code is available on GitHub (https://github.com/BaileyAndrew/Strong-Product-Model).

All experiments were run on a 2020 M1 Macbook Pro with 8GB of RAM. We implemented our algorithm using Python 3.9.18, NumPy 1.25.2, (Harris et al., 2020) and SciPy 1.12.0 (Virtanen et al., 2020). Our line search parameters were $\eta_{\text{init}} = 0.1$, $\beta = \frac{1}{2}$, and $\tau = \frac{1}{2}$, except in cases where we used a large regularization parameter, in which case we found we needed $\beta = 0$ to prevent early convergence (it is likely that a smaller, nonzero value would suffice). We always initialized our parameters to the identity matrix. $\rho = 0.1$, except for the COIL-20 experiment (Section 5.2), in which we found a slight improvement using $\rho = 10^6$.

For precision-recall (PR) curves, error bars indicate max/min performance over 10 trials. For other curves, they indicate standard deviation. As we are the first to introduce the strong product model, there is no direct prior work to compare against. We compared against GmGM, as it performs similarly to other multi-axis models (Andrew et al., 2024) and is by far the fastest, allowing us to run many high-dimensional trials. It is a Kronecker-sum structured model. PR curves were generated by varying the amount of thresholding applied to the outputs.

Since our model reduces (up to a constant shift in the diagonals) to both the Kronecker product and Kronecker sum models when $\mathbf{\Psi}_{2,b} = \mathbf{I}$ and $\mathbf{\Psi}_{2,w} = \mathbf{I}$, respectively, we also used these restricted models for comparison.

5.1 Synthetic Data

For our synthetic data, we generated our ground truth graphs to have the form $y\mathbf{A} \otimes x\mathbf{B} + y\mathbf{A} \oplus \frac{1}{x}\mathbf{C}$, where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are Barabasi-Albert random graphs with 500 nodes each. Barabasi-Albert is a power-law distribution; many real world datasets (approximately) follow power-law distributions, and hence we felt it was an adequate test case.

Results reported in the Figure 2 assume $y = 1$. We found that the ratio of strength of between-row and within-row column factors, x, had a large effect on performance. In Figure 3, we show how the area under PR curves (AUPR) changes as x varies.

In Figure 2, we report the PR curves on our performance on synthetic data, where $x = 1$ and $x = 2.5$ to both validate our model and demonstrate this effect. Note that we do not demonstrate any loss in performance compared to other models, and are the only model that estimates all three relevant graphs. Other models only estimate one type of column graph, and assume that the other does not exist.

We ran several other tests, but defer these to the supplementary material due to space concerns. The

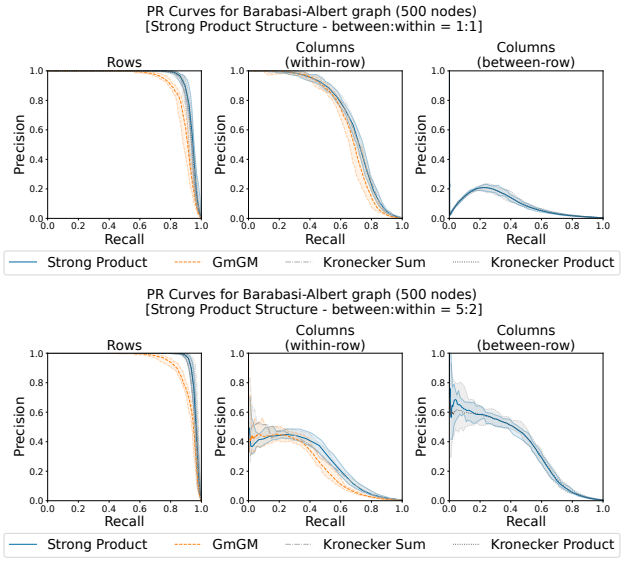

Figure 2: Precision-recall curves on random graphs with structure $\mathbf{A} \otimes x\mathbf{B} + \mathbf{A} \oplus \frac{1}{x}\mathbf{C}$. 'Strong Product' refers to our model; 'Kronecker Sum' and 'Kronecker Product' refer to our model with the relevant factor ($\mathbf{B}$ or $\mathbf{C}$) constrained to an identity matrix. GmGM and 'Kronecker Sum' do not estimate the between-row graph, and 'Kronecker Product' does not estimate the within-row graph; they have been ommitted from the relevant sub-plots. (Top) $x = 1$. (Bottom) $x = 2.5$

supplementary material contains a demonstration that performance improves as the number of nodes increases, an analysis of the effects of changing y, and a demonstration that our algorithm still performs well when run on data that does not follow a strong product structure. We also show that the regularization parameter has minimal effect on the results; its purpose is primarily to ensure existence of solutions.

The results presented here and in the supplementary material show that our model performs well on all graphical models considered by prior work, and is the only option that learns both types of column graph. There is nothing to be gained from assuming a more restricted structure, such as the Kronecker sum or Kronecker product. Of course, our results also show that models developed for such structures also perform well when applied to data generated from our proposed structure, except that they are unable to estimate either the between-row column graph (Kronecker sum) or the within-row column graph (Kronecker product).

5.2 COIL Dataset

The first real-world dataset we test on is the COIL-20 video dataset (Nene et al., 1996). This is somewhat of a 'classic' dataset for Kronecker-structured models,

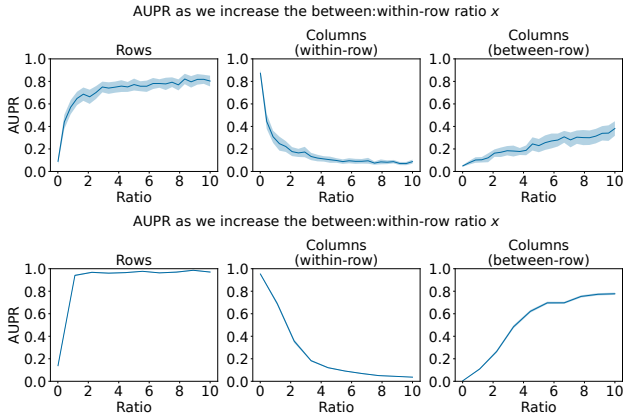

Figure 3: How AUPR varies as the ratio x varies, for data generated following the structure outlined in Section 5.1. (Top) 100-node graphs, 50 trials. (Bottom) 500-node graphs, 25 trials. Note that the deviations are much smaller for larger graphs.

Algorithm	Precision	1-off Precision	Recall
Strong Product	38%	65%	78%
GmGM	33%	59%	67%
TeraLasso	36%	64%	69%
Within-Row	49%	85%	**100%**
Between-Row	**49%**	**94%**	**100%**

Table 2: Metrics for the COIL dataset experiment. 1-off precision refers to frames connected to adjacent frames and frames with one intermediate frame between them. As we chose to keep twice as many edges as exist, the highest possible value for precision is 50%.

with several prior algorithms also demonstrating results on it. Its convenience comes from the fact that there is a very clear 'natural graph' we should expect in the data: a frame in the video should be connected only to the frames immediately before and immediately after itself. The video is that of a rotating duck, and hence we would expect the first and last frames to connect as well.

The video is a $72 \times 128 \times 128$ tensor; we down-sampled the pixels to $72 \times 16 \times 16$, and then flattened to 72×256. We compared our performance against GmGM and TeraLasso (Greenewald et al., 2019). TeraLasso is a Kronecker-sum-structure algorithm that performs similarly to GmGM. Its main differences are that it does not enforce positive-definite factor graphs and uses *glasso* regularization. Due to its similarity and comparatively slow runtime, we did not compare against it for our other, more computationally expensive tests.

We chose thresholding/regularization parameters so that all models output roughly 144 edges (exactly 144 was not always possible). Figure 4 displays a graphical interpretation of our results - an enlarged version of this figure is available in the supplementary material, and Table 2 contains the corresponding numeric metrics. We found that our model performed similarly to prior work. However, when we considered the frames to be the column dimension, the performance was much better than prior work.

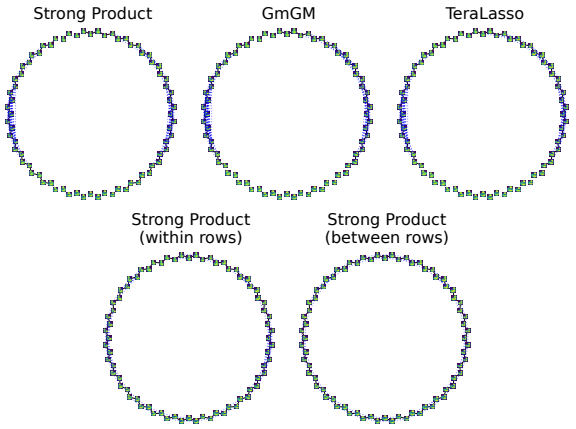

Figure 4: The learned connections between the video frames for various algorithms. Black connections indicate connections between adjacent frames, blue indicate non-adjacent frames. A larger version of this figure is available in the supplementary material. For the bottom row, we transposed the data and investigated the two different returned graphs.

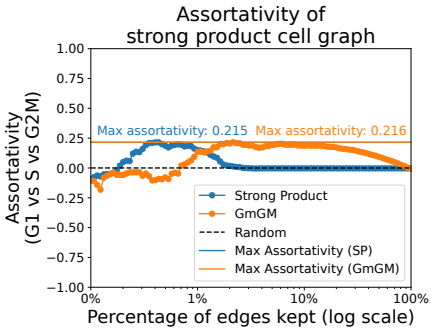

Figure 5: Assortativity as we vary thresholding, for the scRNA-seq dataset.

5.3 scRNA-seq Dataset

While informative, the COIL dataset is somewhat of a toy example. To get a better understanding of how our algorithm will perform on real data, we will use a mouse embryo stem cell scRNA-seq dataset from Buettner et al. (2015), limited to only mitosis-related genes as in Li et al. (2022). The dataset consists of 288 cells and 167 genes, in which each cell has been labelled according to its stage in the cell cycle - either G1, S, or

G2M.

To understand performance, we can use *assortativity* - this measures the tendency of nodes in the same class (cell cycle stage) to connect to each other in a graph, and ranges from -1 (tend not to connect) to 0 (no tendency) to 1 (tend to connect). We should expect there to be some positive assortativity, as there should be similarities between the mitotic gene expressions of cells in the same stage.

Interestingly, we found that our algorithm found assortativities very similar to prior work (GmGM), but on much sparser graphs. GmGM only had high assortativity when we only slightly thresholded the output graph; for sparse graphs, the assortativity was 0. In contrast, our model had high assortativity on strongly thesholded graphs, but had an assortativity of zero otherwise. This can be seen in Figure 5. The sparser regime is the more relevant regime here, as most real-world interaction networks tend to be sparse.

6 CONCLUSION

We have shown that our model is able to perform well on synthetic data generated from both the strong product model, the Kronecker sum model, and the Kronecker product model. Our model, like other Kronecker-structured models, improves in performance as the number of rows and columns increases, even if we only have one sample matrix. On real data, we still get reasonable results, and outperform prior work in some cases.

We are very interested in the performance of our model as the between-row/within-row column ratio changes, and the extent to which this ratio is identifiable in real data. In future work, we intend to prove statistical recovery rates, parameterized by this ratio. Additionally, graphically-focused Kronecker-structured models do not seem to have adopted the geodesic convexity framework for optimization. The fast convergence we experience is encouraging; we would like to incorporate sparsity-inducing regularizers into this framework.

Acknowledgements

There was no direct funding for this research. B.A. is supported by the UKRI Engineering and Physical Sciences Research Council (EPSRC) [EP/S024336/1]. We would like to thank David Marples for their ideas and insightful comments. Additionally, we would like to thank the reviewers for their active input and discussion; it was valuable and very much appreciated.

References

Absil, P.-A., Mahony, R., & Sepulchre, R. (2009, April). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press. https://doi.org/10.1515/9781400830244

Ambroise, C., Chiquet, J., & Matias, C. (2009). Inferring sparse Gaussian graphical models with latent structure [Publisher: Institute of Mathematical Statistics and Bernoulli Society]. *Electronic Journal of Statistics*, *3*(none), 205–238. https://doi.org/10.1214/08-EJS314

Andrew, B., Westhead, D. R., & Cutillo, L. (2024, January). 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]. Retrieved July 16, 2024, from https://eprints.whiterose.ac.uk/208283/

Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. [Publisher: Pacific Journal of Mathematics, A Non-profit Corporation]. *Pacific Journal of Mathematics*, *16*(1), 1–3. Retrieved August 16, 2024, from https://projecteuclid.org/journals/pacific-journal-of-mathematics/volume-16/issue-1/Minimization-of-functions-having-Lipschitz-continuous-first-partial-derivatives/pjm/1102995080.full

Buettner, F., Natarajan, K. N., Casale, F. P., Proserpio, V., Scialdone, A., Theis, F. J., Teichmann, S. A., Marioni, J. C., & Stegle, O. (2015). Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells [Number: 2 Publisher: Nature Publishing Group]. *Nature Biotechnology*, *33*(2), 155–160. https://doi.org/10.1038/nbt.3102

Dahl, A., Hore, V., Iotchkova, V., & Marchini, J. (2013, December). Network inference in matrix-variate Gaussian models with non-independent noise [arXiv:1312.1622 [stat]]. https://doi.org/10.48550/arXiv.1312.1622

Darmwal, Y., & Rajawat, K. (2023, December). Low-complexity subspace-descent over symmetric positive definite manifold [arXiv:2305.02041 [cs, eess, math, stat]]. https://doi.org/10.48550/arXiv.2305.02041

Deng, X., & Tsui, K.-W. (2013). Penalized Covariance Matrix Estimation Using a Matrix-Logarithm Transformation

[Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2012.715556]. *Journal of Computational and Graphical Statistics*, *22*(2), 494–512. https://doi.org/10.1080/10618600.2012.715556

Duembgen, L., & Tyler, D. E. (2016, July). Geodesic Convexity and Regularized Scatter Estimators [arXiv:1607.05455 [stat]]. https://doi.org/10.48550/arXiv.1607.05455

Fan, J., & Li, R. (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties [Publisher: Taylor & Francis _eprint: https://doi.org/10.1198/016214501753382273]. *Journal of the American Statistical Association*, *96*(456), 1348–1360. https://doi.org/10.1198/016214501753382273

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, *9*(3), 432–441. https://doi.org/10.1093/biostatistics/kxm045

Greenewald, K., Zhou, S., & Hero, A. (2019). Tensor graphical lasso (TeraLasso) [Publisher: [Royal Statistical Society, Oxford University Press]]. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, *81*(5), 901–931. Retrieved July 16, 2024, from https://www.jstor.org/stable/26833080

Guillot, D., & Rajaratnam, B. (2015). Functions Preserving Positive Definiteness for Sparse Matrices [Publisher: American Mathematical Society]. *Transactions of the American Mathematical Society*, *367*(1), 627–649. Retrieved August 12, 2024, from https://www.jstor.org/stable/24512825

Han, A., Mishra, B., Jawanpuria, P. K., & Gao, J. (2021). On Riemannian Optimization over Positive Definite Matrices with the Bures-Wasserstein Geometry. *Advances in Neural Information Processing Systems*, *34*, 8940–8953. Retrieved August 14, 2024, from https://proceedings.neurips.cc/paper_files/paper/2021/hash/4b04b0dcd2ade339a3d7ce13252a29d4-Abstract.html

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., … Oliphant, T. E. (2020). Array programming with NumPy [Number: 7825 Publisher: Nature Publishing Group]. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hosseini, R., & Sra, S. (2015). Matrix Manifold Optimization for Gaussian Mixtures. *Advances in Neural Information Processing Systems*, *28*. Retrieved August 15, 2024, from https://papers.nips.cc/paper_files/paper/2015/hash/dbe272bab69f8e13f14b405e038deb64-Abstract.html

Jeuris, B., Vandebril, R., & Vandereycken, B. (2012). A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *Electronic Transactions on Numerical Analysis*, *39*, 379–402. Retrieved August 15, 2024, from https://etna.ricam.oeaw.ac.at/volumes/2011-2020/vol39/abstract.php?vol=39&pages=379-402

Kalaitzis, A., Lafferty, J., Lawrence, N. D., & Zhou, S. (2013). The Bigraphical Lasso [ISSN: 1938-7228]. *Proceedings of the 30th International Conference on Machine Learning*, 1229–1237. Retrieved August 8, 2024, from https://proceedings.mlr.press/v28/kalaitzis13.html

Li, S., López-García, M., Lawrence, N. D., & Cutillo, L. (2022, March). Scalable Bigraphical Lasso: Two-way Sparse Network Inference for Count Data [arXiv:2203.07912 [cs, stat]]. https://doi.org/10.48550/arXiv.2203.07912

Liu, H., Han, F., Yuan, M., Lafferty, J., & Wasserman, L. (2012). The nonparanormal SKEPTIC. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 1415–1422. Retrieved July 16, 2024, from https://collaborate.princeton.edu/en/publications/the-nonparanormal-skeptic

Meinshausen, N., & Bühlmann, P. (2006). High-Dimensional Graphs and Variable Selection with the Lasso [Publisher: Institute of Mathematical Statistics]. *The Annals of Statistics*, *34*(3), 1436–1462. Retrieved February 9, 2025, from https://www.jstor.org/stable/25463463

Moakher, M. (2005). A Differential Geometric Approach to the Geometric Mean of Symmetric Positive-Definite Matrices [Publisher: Society for Industrial and Applied Mathematics]. *SIAM Journal on Matrix Analysis and Applications*, *26*(3), 735–747. https://doi.org/10.1137/S0895479803436937

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

Song, D., & Hero, A. O. (2023, October). On Separability of Covariance in Multiway Data Analysis [arXiv:2302.02415 [math, stat]]. https://doi.org/10.48550/arXiv.2302.02415

Sra, S., & Hosseini, R. (2015). Conic Geometric Optimization on the Manifold of Positive Definite Matrices [Publisher: Society for Industrial and Applied Mathematics]. *SIAM Journal on Optimization*, *25*(1), 713–739. https://doi.org/10.1137/140978168

Tsiligkaridis, T., Hero, A. O., & Zhou, S. (2012). Kronecker graphical lasso [ISSN: 2373-0803]. *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 884–887. https://doi.org/10.1109/SSP.2012.6319849

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., ... van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python [Number: 3 Publisher: Nature Publishing Group]. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vishnoi, N. K. (2018, June). Geodesic Convex Optimization: Differentiation on Manifolds, Geodesics, and Convexity [arXiv:1806.06373 [cs, math]]. https://doi.org/10.48550/arXiv.1806.06373

Wang, Y., Jang, B., & Hero, A. (2020). The Sylvester Graphical Lasso (SyGlasso) [ISSN: 2640-3498]. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 1943–1953. Retrieved August 8, 2024, from https://proceedings.mlr.press/v108/wang20d.html

Wang, Y., Sun, Z., Song, D., & Hero, A. (2022). Kronecker-structured covariance models for multiway data [Publisher: Amer. Statist. Assoc., the Bernoulli Soc., the Inst. Math. Statist., and the Statist. Soc. Canada]. *Statistics Surveys*, *16*(none), 238–270. https://doi.org/10.1214/22-SS139

Wiesel, A. (2012a). On the convexity in Kronecker structured covariance estimation [ISSN: 2373-0803]. *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 880–883. https://doi.org/10.1109/SSP.2012.6319848

Wiesel, A. (2012b). Geodesic Convexity and Covariance Estimation [Conference Name: IEEE Transactions on Signal Processing]. *IEEE Transactions on Signal Processing*, *60*(12), 6182–6189. https://doi.org/10.1109/TSP.2012.2218241

Xue, L., Ma, S., & Zou, H. (2012). Positive-Definite L1-Penalized Estimation of Large Covariance Matrices [Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01621459.2012.725386]. *Journal of the American Statistical Association*, *107*(500), 1480–1491. https://doi.org/10.1080/01621459.2012.725386

Yamakawa, Y., Sato, H., & Aihara, K. (2023, April). Modified Armijo line-search in Riemannian optimization with reduced computational cost [arXiv:2304.02197 [math]]. https://doi.org/10.48550/arXiv.2304.02197

Yi, M., & Tyler, D. E. (2020). Shrinking the Covariance Matrix Using Convex Penalties on the Matrix-Log Transformation [Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2020.1814788]. *Journal of Computational and Graphical Statistics*, *30*(2), 442–451. https://doi.org/10.1080/10618600.2020.1814788

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty [Publisher: Institute of Mathematical Statistics]. *The Annals of Statistics*, *38*(2), 894–942. https://doi.org/10.1214/09-AOS729

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [**Yes**/No/Not Applicable]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [**Yes**/No/Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [**Yes**/No/Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [**Yes**/No/Not Applicable]
   (b) Complete proofs of all theoretical results. [**Yes**/No/Not Applicable]
   (c) Clear explanations of any assumptions. [**Yes**/No/Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [**Yes**/No/Not Applicable]

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [**Yes**/No/Not Applicable]

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [**Yes**/No/Not Applicable]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [**Yes**/No/Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [**Yes**/No/Not Applicable]

   (b) The license information of the assets, if applicable. [Yes/No/**Not Applicable**]

   (c) New assets either in the supplemental material or as a URL, if applicable. [**Yes**/No/Not Applicable]

   (d) Information about consent from data providers/curators. [Yes/No/**Not Applicable**]

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/**Not Applicable**]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Yes/No/**Not Applicable**]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/**Not Applicable**]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/**Not Applicable**]

The Strong Product Model for Network Inference without Independence Assumptions

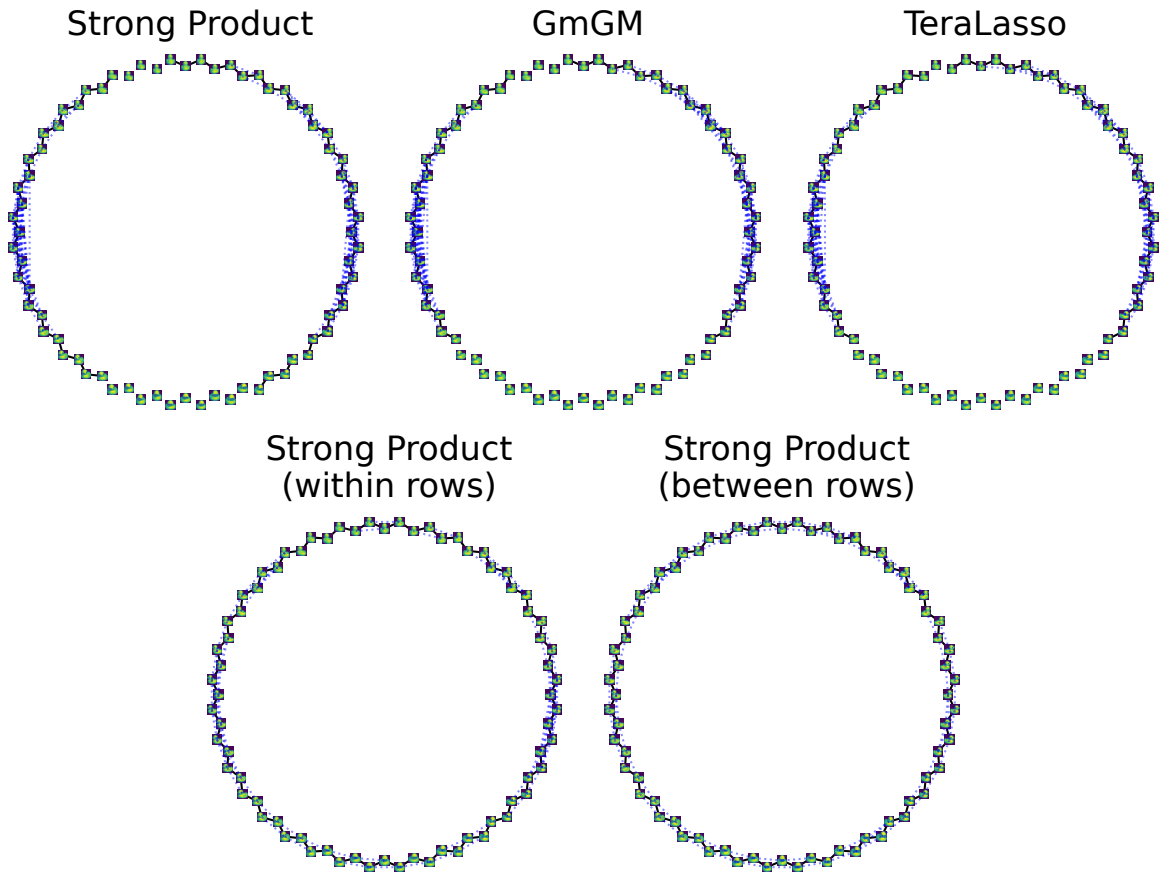

Figure 1: The learned connections between the video frames for various algorithms. Black connections indicate connections between adjacent frames, blue indicate non-adjacent frames. This is an enlarged version of Figure 4 in the main paper.

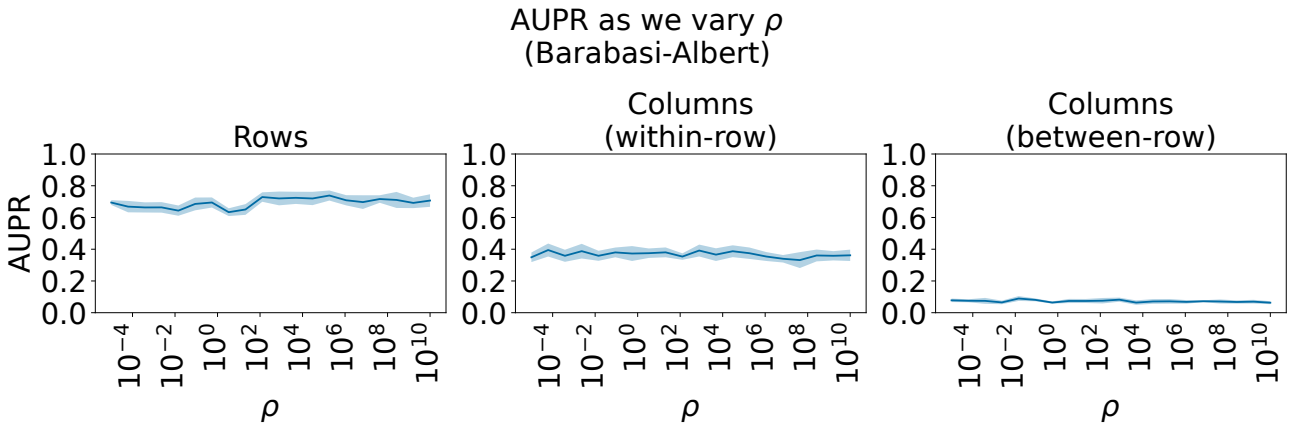

Figure 2: The value of ρ does not significantly affect the results. Experiment performed on 100-node graphs with the standard 1:1 x ratio for data generated from the strong product model, with Barabasi-Albert factor graphs.

1 ADDITIONAL EXPERIMENTS

In this section, we briefly recount the additional experiments we ran. All data is generated identically to that in the main paper, except for the differences we note. Figure 3 demonstrates that we do not experience a degradation in performance when we run our algorithm on Kronecker-sum-structured and Kronecker-product-structured data; we still perform comparably to prior work.

In Figure 4, we demonstrate that performance improves as our factor graphs get larger. We analyze the AUPR as y varies for 100-node and 500-node factor graphs in Figure 5. This latter experiment takes significantly longer to run than other experiments in this paper, so we do not repeat it for 1000-node graphs. Figure 2 shows that performance is not affected by the value of ρ.

2 DERIVATION OF GRADIENTS

In this section, we will derive the gradients of our negative log-likelihood, with respect to our three parameter matrices. The negative log-likelihood is given below, for reference, with irrelevant additive and multiplicative constants removed.

$$\begin{aligned}\text{NLL} &= -\log|\boldsymbol{\Psi}_1 \oplus \boldsymbol{\Psi}_{2,w} + \boldsymbol{\Psi}_1 \otimes \boldsymbol{\Psi}_{2,b}| + \operatorname{tr}\left[\boldsymbol{\Psi}_1 \mathbf{S}_1\right] + \operatorname{tr}\left[\boldsymbol{\Psi}_{2,w} \mathbf{S}_2\right] + \operatorname{tr}\left[\boldsymbol{\Psi}_1 \mathbf{X} \boldsymbol{\Psi}_{2,b} \mathbf{X}^T\right] \\ &= -\log|\boldsymbol{\Psi}_1 \otimes \boldsymbol{\Theta} + \mathbf{I} \otimes \boldsymbol{\Psi}_{2,w}| + \operatorname{tr}\left[\boldsymbol{\Psi}_{2,w} \mathbf{S}_2\right] + \operatorname{tr}\left[\boldsymbol{\Psi}_1 \mathbf{X} \boldsymbol{\Theta} \mathbf{X}^T\right]\end{aligned}$$

To derive the gradients, we will need to take derivatives with respect to matrices. Standard relations that we will use are provided below, for reference.

$$\begin{aligned} d\log|\mathbf{M}| &= \operatorname{tr}\left[\mathbf{M}^{-1} d\mathbf{M}\right] \\ \frac{d}{d\mathbf{M}} \operatorname{tr}\left[\mathbf{M}\mathbf{N}\right] &= \mathbf{N} \\ d\left(\mathbf{M} \otimes \mathbf{N}\right) &= d\mathbf{M} \otimes \mathbf{N} + \mathbf{M} \otimes d\mathbf{N} \\ d\left(\mathbf{M} \oplus \mathbf{N}\right) &= d\mathbf{M} \oplus d\mathbf{N} \end{aligned}$$

The $\oplus$ relation follows directly from the $\otimes$ relation. Using these equations, the derivative of he negative log likelihood is straightforward; we just need an expression for the derivative of the strong product.

$$d\left(\boldsymbol{\Psi}_1 \oplus \boldsymbol{\Psi}_{2,w} + \boldsymbol{\Psi}_1 \otimes \boldsymbol{\Psi}_{2,b}\right) = d\boldsymbol{\Psi}_1 \oplus d\boldsymbol{\Psi}_{2,w} + d\boldsymbol{\Psi}_1 \otimes \boldsymbol{\Psi}_{2,b} + \boldsymbol{\Psi}_1 \otimes d\boldsymbol{\Psi}_{2,b}$$

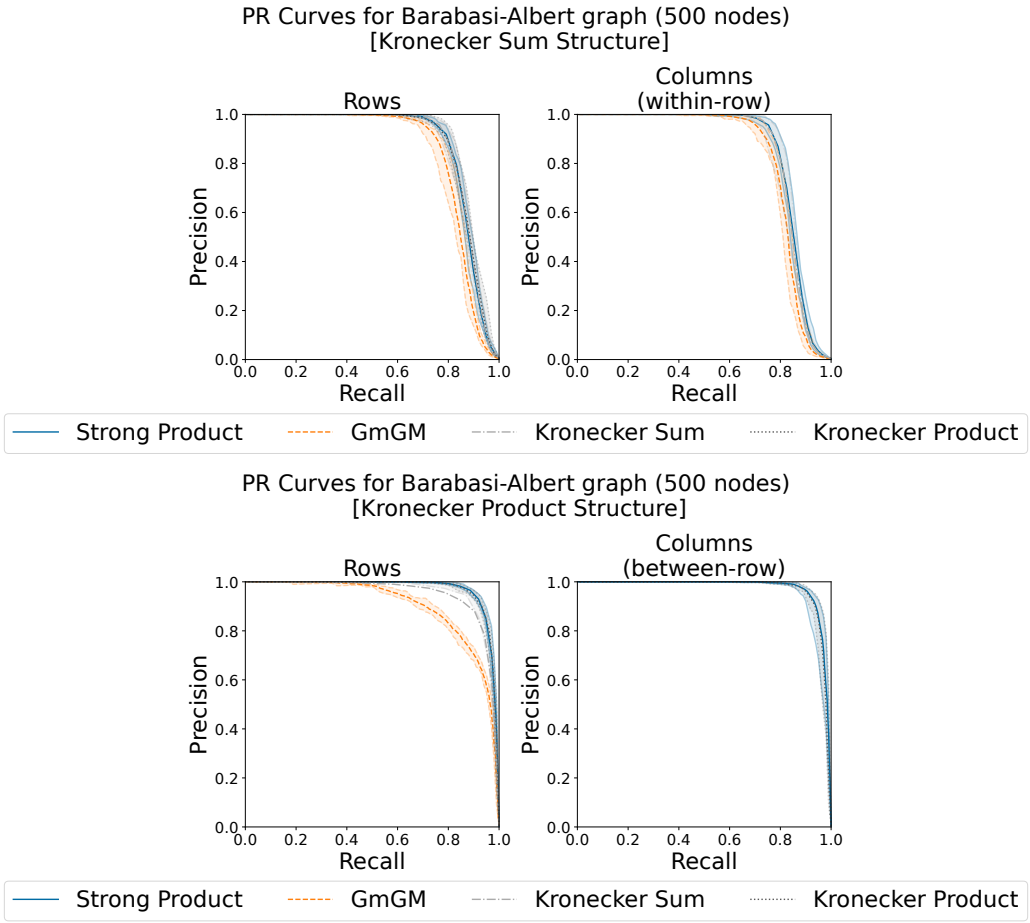

Figure 3: Performance of algorithms when data is generated from the Kronecker sum or Kronecker product structures.

Rather than differentiating with respect to entire matrices, we will differentiate with respect to a single element. We will let $\mathbf{J}^{ij}$ be the matrix of zeros except for a 1 at position (i, j), i.e. $\frac{d}{dx_{ij}}\mathbf{X} = \mathbf{J}^{ij}$.

$$\begin{aligned}\frac{d}{d\psi_{1,ij}}\left(\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w} + \mathbf{\Psi}_1 \otimes \mathbf{\Psi}_{2,b}\right) &= \mathbf{J}^{ij} \oplus \mathbf{0} + \mathbf{J}^{ij} \otimes \mathbf{\Psi}_{2,b} \\ &= \mathbf{J}^{ij} \otimes \mathbf{I} + \mathbf{J}^{ij} \otimes \mathbf{\Psi}_{2,b} \\ &= \mathbf{J}^{ij} \otimes \mathbf{\Theta} \\ &= \left(\mathbf{I} \otimes \mathbf{\Theta}\right)\left(\mathbf{J}^{ij} \otimes \mathbf{I}\right)\end{aligned}$$

From the definition of the stridewise-blockwise trace, and letting *sp* abbreviate the strong product, we have that:

$$\begin{aligned}\frac{d}{d\mathbf{\Psi}_1}\log|sp| &= \mathrm{tr}_{d_2}\left[sp^{-1}\left(\mathbf{I} \otimes \mathbf{\Theta}\right)\right] \\ &= \mathrm{tr}_{d_2}\left[\left(\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w}\mathbf{\Theta}^{-1}\right)^{-1}\right]\end{aligned}$$

Thus, our derivative with respect to $\mathbf{\Psi}_1$ is just:

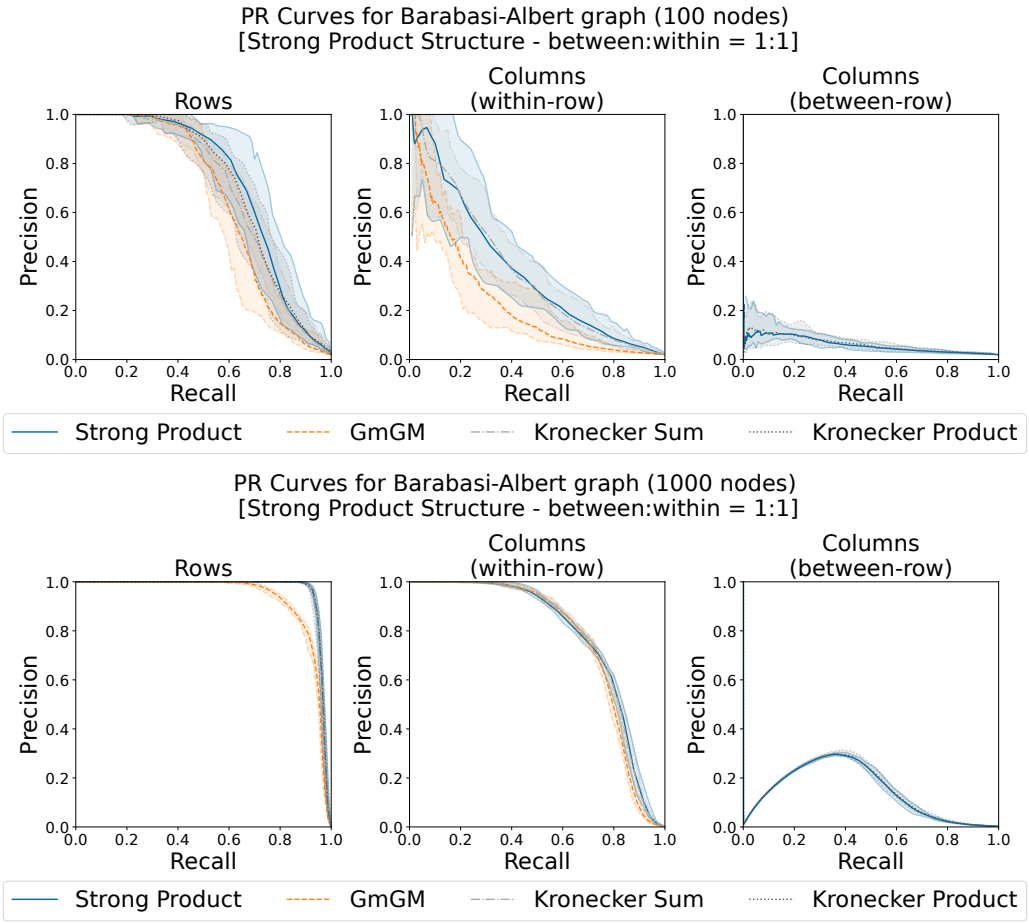

Figure 4: The effect of different graph sizes on performance.

$$\frac{d}{d\mathbf{\Psi}_1}\text{NLL} = -\text{tr}_{d_2}\left[\left(\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w}\mathbf{\Theta}^{-1}\right)^{-1}\right] + \mathbf{X}\mathbf{\Theta}\mathbf{X}^T$$

The gradient with respect to $\mathbf{\Psi}_{2,w}$ is fairly easy to compute.

$$\frac{d}{d\psi_{2,w;ij}}\left(\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w} + \mathbf{\Psi}_1 \otimes \mathbf{\Psi}_{2,b}\right) = \mathbf{I} \otimes \mathbf{J}^{ij}$$

$$\frac{d}{d\mathbf{\Psi}_{2,w}}\text{NLL} = -\text{tr}^{d_1}\left[\left(\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}\right)^{-1}\right] + \mathbf{S}_2$$

We now have a choice to consider our method as a function of $\mathbf{\Psi}_{2,b}$ or $\mathbf{\Theta}$; so far, parameterizing in terms of $\mathbf{\Theta}$ has simplified our equations, so we will continue to do so.

$$\frac{d}{d\theta_{ij}}\left(\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}\right) = \mathbf{\Psi}_1 \otimes \mathbf{J}^{ij}$$

$$\begin{aligned}\frac{d}{d\mathbf{\Theta}}\text{NLL} &= -\text{tr}^{d_1}\left[\left(\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}\right)^{-1}\left(\mathbf{\Psi}_1 \otimes \mathbf{I}\right)\right] + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X} \\ &= -\text{tr}^{d_1}\left[\left(\mathbf{I} \otimes \mathbf{\Theta} + \mathbf{\Psi}_1^{-1} \otimes \mathbf{\Psi}_{2,w}\right)^{-1}\right] + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X}\end{aligned}$$

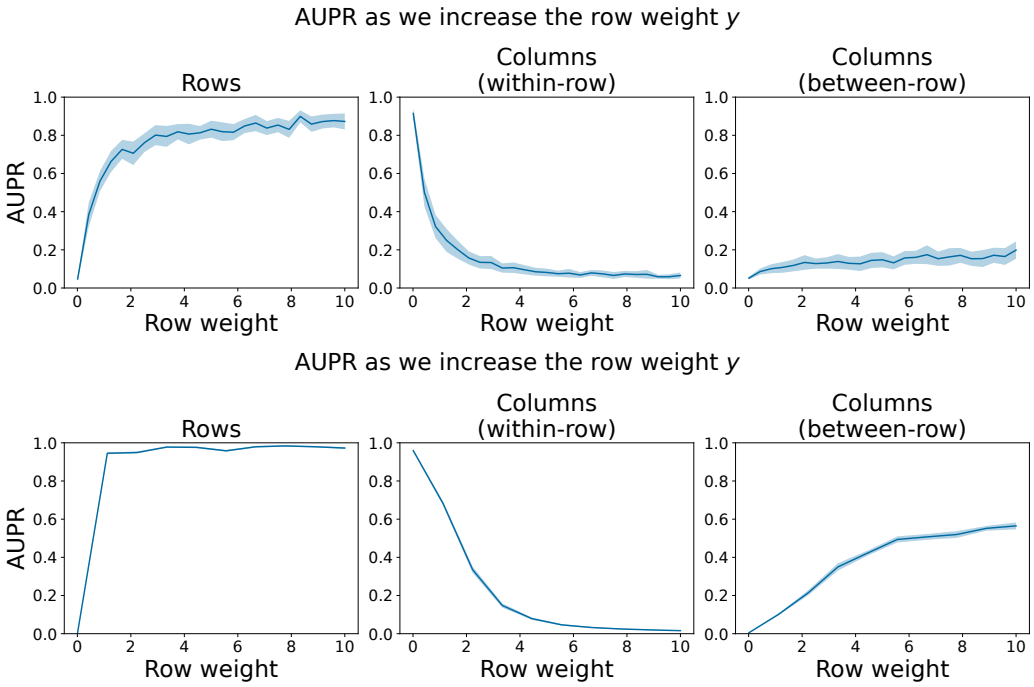

Figure 5: Variance of AUPR as y changes for 100-node (top) and 500-node (bottom) graphs. Error bars represent standard deviations, computed over 50 trials (top) and 25 trials (bottom).

2.1 Efficient Gradients

Earlier, we derived the following gradients:

$$\frac{d}{d\mathbf{\Psi}_1}\text{NLL} = -\text{tr}_{d_2}\left[\left(\mathbf{\Psi}_1 \oplus \mathbf{\Psi}_{2,w}\mathbf{\Theta}^{-1}\right)^{-1}\right] + \mathbf{X}\mathbf{\Theta}\mathbf{X}^T$$
$$\frac{d}{d\mathbf{\Psi}_{2,w}}\text{NLL} = -\text{tr}^{d_1}\left[\left(\mathbf{\Psi}_1 \otimes \mathbf{\Theta} + \mathbf{I} \otimes \mathbf{\Psi}_{2,w}\right)^{-1}\right] + \mathbf{S}_2$$
$$\frac{d}{d\mathbf{\Theta}}\text{NLL} = -\text{tr}^{d_1}\left[\left(\mathbf{I} \otimes \mathbf{\Theta} + \mathbf{\Psi}_1^{-1} \otimes \mathbf{\Psi}_{2,w}\right)^{-1}\right] + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X}$$

These derivatives, while valid, have little practical utility; they require inverting a $d_1d_2 \times d_1d_2$ matrix, which is an $O(d_1^3d_2^3)$ operation that in practice requires $O(d_1^2d_2^2)$ space.

Because all parameters are positive definite, we know two key facts:

1. They each have an eigendecomposition
2. Every pair can be simultaneously diagonalized

In particular, we can express our parameters as follows:

$$\mathbf{\Psi}_1 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$
$$\mathbf{\Psi}_{2,w} = \mathbf{P}\mathbf{D}\mathbf{P}^T$$
$$\mathbf{\Theta} = \mathbf{P}\mathbf{P}^\mathbf{T}$$

$\mathbf{V}$ is orthonormal, and $\mathbf{\Lambda}$ and $\mathbf{D}$ are diagonal. Importantly, $\mathbf{P}$ is **not orthonormal**. Furthermore, note that $\mathbf{\Psi}_{2,w}\mathbf{\Theta}^{-1} = \mathbf{PDP}^{-1}$. Computing these decompositions takes $O(d_1^3 + d_2^3)$ time and $O(d_1^2 + d_2^2)$ space. Thanks to a lemma by Dahl et al. (2013), we have a convenient way of accessing this diagonal core:

$$\begin{aligned}
\mathrm{tr}_{d_2}\left[(\mathbf{M}\otimes\mathbf{I})\,\mathbf{A}\,(\mathbf{N}\otimes\mathbf{I})\right] &= \mathbf{M}\mathrm{tr}_{d_2}\left[\mathbf{A}\right]\mathbf{N} && \text{Lemma 2 (Dahl et al., 2013)}\\
\mathrm{tr}_{d_2}\left[(\mathbf{I}\otimes\mathbf{M})\,\mathbf{A}\,(\mathbf{I}\otimes\mathbf{N})\right] &= \mathrm{tr}_{d_2}\left[(\mathbf{I}\otimes\mathbf{NM})\,\mathbf{A}\right] && \text{(Cyclic Property)}
\end{aligned}$$

The latter fact follows directly from the definition of the stridewise-blockwise trace. Analogous formulas hold for tr^{d_1}, as long as the order of the Kronecker products is reversed. Using these, we can re-express our gradients.

$$\begin{aligned}
\frac{d}{d\mathbf{\Psi}_1}\mathrm{NLL} &= -\mathrm{tr}_{d_2}\left[\left(\mathbf{\Psi}_1\oplus\mathbf{\Psi}_{2,w}\mathbf{\Theta}^{-1}\right)^{-1}\right] + \mathbf{X\Theta X}^T\\
&= -\mathrm{tr}_{d_2}\left[\left(\mathbf{V\Lambda V}^T\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{PDP}^{-1}\right)^{-1}\right] + \mathbf{X\Theta X}^T\\
&= -\mathrm{tr}_{d_2}\left[(\mathbf{V}\otimes\mathbf{P})\left(\mathbf{\Lambda}\otimes\mathbf{V}^T\mathbf{V}+\mathbf{P}^{-1}\mathbf{P}\otimes\mathbf{D}\right)^{-1}(\mathbf{V}\otimes\mathbf{P})^{-1}\right] + \mathbf{X\Theta X}^T\\
&= -\mathbf{V}\mathrm{tr}_{d_2}\left[(\mathbf{\Lambda}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{D})^{-1}\right]\mathbf{V}^T + \mathbf{X\Theta X}^T\\
&= -\mathbf{V}\mathrm{tr}_{d_2}\left[(\mathbf{\Lambda}\oplus\mathbf{D})^{-1}\right]\mathbf{V}^T + \mathbf{X\Theta X}^T\\
\frac{d}{d\mathbf{\Psi}_{2,w}}\mathrm{NLL} &= -\mathrm{tr}^{d_1}\left[(\mathbf{\Psi}_1\otimes\mathbf{\Theta}+\mathbf{I}\otimes\mathbf{\Psi}_{2,w})^{-1}\right] + \mathbf{S}_2\\
&= -\mathrm{tr}^{d_1}\left[\left(\mathbf{I}\otimes\mathbf{P}^{-T}\right)(\mathbf{\Lambda}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{D})^{-1}\left(\mathbf{I}\otimes\mathbf{P}^{-1}\right)\right] + \mathbf{S}_2\\
&= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\mathbf{\Lambda}\oplus\mathbf{D})^{-1}\right]\mathbf{P}^{-1} + \mathbf{S}_2\\
\frac{d}{d\mathbf{\Theta}}\mathrm{NLL} &= -\mathrm{tr}^{d_1}\left[\left(\mathbf{I}\otimes\mathbf{\Theta}+\mathbf{\Psi}_1^{-1}\otimes\mathbf{\Psi}_{2,w}\right)^{-1}\right] + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X}\\
&= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[\left(\mathbf{I}\otimes\mathbf{I}+\mathbf{\Lambda}^{-1}\otimes\mathbf{D}\right)^{-1}\right]\mathbf{P}^{-1} + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X}\\
&= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\mathbf{\Lambda}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{D})^{-1}(\mathbf{\Lambda}\otimes\mathbf{I})\right]\mathbf{P}^{-1} + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X}\\
&= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\mathbf{\Lambda}\oplus\mathbf{D})^{-1}(\mathbf{\Lambda}\otimes\mathbf{I})\right]\mathbf{P}^{-1} + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X}
\end{aligned}$$

Computing the stridewise-blockwise trace can now be done much more efficiently, as its input is a diagonal matrix. It can be done in $O(d_1 d_2)$ space. As a sanity check, we can observe that when $\mathbf{\Theta} = \mathbf{I}$ (i.e. $\mathbf{\Psi}_{2,b} = \mathbf{0}$), the gradients become those of the Kronecker-sum model (note that $\mathbf{P}$ can in that case be chosen to be the eigenvectors of $\mathbf{\Psi}_{2,w}$), and when $\mathbf{\Psi}_1 = \mathbf{I}$ then the other gradients become identical. Both of these should be expected from the structure of the strong product.

2.2 Addition of Frobenius Norm of Logarithm

Since $\frac{d}{d\mathbf{M}}\|\log\mathbf{M}\|_F = 2\mathbf{M}^{-1}\log\mathbf{M}$, we have that our gradients are:

$$\begin{aligned}
\frac{d}{d\mathbf{\Psi}_1}\mathrm{NLL} &= -\mathbf{V}\mathrm{tr}_{d_2}\left[(\mathbf{\Lambda}\oplus\mathbf{D})^{-1}\right]\mathbf{V}^T + \mathbf{X\Theta X}^T + 2\rho\mathbf{\Psi}_1^{-1}\log\mathbf{\Psi}_1\\
\frac{d}{d\mathbf{\Psi}_{2,w}}\mathrm{NLL} &= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\mathbf{\Lambda}\oplus\mathbf{D})^{-1}\right]\mathbf{P}^{-1} + \mathbf{S}_2 + 2\rho\mathbf{\Psi}_{2,w}^{-1}\log\mathbf{\Psi}_{2,w}\\
\frac{d}{d\mathbf{\Theta}}\mathrm{NLL} &= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\mathbf{\Lambda}\oplus\mathbf{D})^{-1}(\mathbf{\Lambda}\otimes\mathbf{I})\right]\mathbf{P}^{-1} + \mathbf{X}^T\mathbf{\Psi}_1\mathbf{X} + 2\rho\mathbf{\Theta}^{-1}\log\mathbf{\Theta}
\end{aligned}$$

2.3 Riemannian Gradients

Recall from the main paper the definition of the Riemannian gradient and retraction.

$$\begin{aligned}
\operatorname{grad}_{\mathbf{M}}^{\mathbb{S}_{++}} f &= \mathbf{M}\left(\operatorname{grad}_{\mathbf{M}}^{\mathbb{E}} f\right)\mathbf{M} && \text{(gradient)}\\
\mathrm{R}_{\mathbf{M}}^{\mathbb{S}_{++}}(\boldsymbol{\xi}) &= \mathbf{M}e^{\mathbf{M}^{-1}\boldsymbol{\xi}} && \text{(retraction)}\\
\mathbf{M}_{t+1} &= \mathrm{R}_{\mathbf{M}_t}^{\mathbb{S}_{++}}\left(-\eta_t \operatorname{grad}_{\mathbf{M}_t}^{\mathbb{S}_{++}} f\right) && \text{(update)}
\end{aligned}$$

There is ample opportunity for simplification. Note that:

$$\begin{aligned}
\mathbf{M}_{t+1} &= \mathbf{M}_t e^{\left(-\eta_t \operatorname{grad}_{\mathbf{M}_t}^{\mathbb{E}} f\right)\mathbf{M}_t}\\
\mathbf{M}^{-1}\log\mathbf{M} &= (\log\mathbf{M})\,\mathbf{M}^{-1}
\end{aligned}$$

The second statement follows from simultaneous diagonalizability. What happens when we right-multiply our Euclidean gradients by $\mathbf{M}$?

$$\begin{aligned}
\left(\frac{d}{d\boldsymbol{\Psi}_1}\mathrm{NLL}\right)\boldsymbol{\Psi}_1 &= -\mathbf{V}\mathrm{tr}_{d_2}\left[(\boldsymbol{\Lambda}\oplus\mathbf{D})^{-1}\right]\boldsymbol{\Lambda}\mathbf{V}^T + \mathbf{X}\boldsymbol{\Theta}\mathbf{X}^T\boldsymbol{\Psi}_1 + 2\rho\log\boldsymbol{\Psi}_1\\
\left(\frac{d}{d\boldsymbol{\Psi}_{2,w}}\mathrm{NLL}\right)\boldsymbol{\Psi}_{2,w} &= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\boldsymbol{\Lambda}\oplus\mathbf{D})^{-1}\right]\mathbf{D}\mathbf{P}^T + \mathbf{S}_2\boldsymbol{\Psi}_{2,w} + 2\rho\log\boldsymbol{\Psi}_{2,w}\\
\left(\frac{d}{d\boldsymbol{\Theta}}\mathrm{NLL}\right)\boldsymbol{\Theta} &= -\mathbf{P}^{-T}\mathrm{tr}^{d_1}\left[(\boldsymbol{\Lambda}\oplus\mathbf{D})^{-1}(\boldsymbol{\Lambda}\otimes\mathbf{I})\right]\mathbf{P}^T + \mathbf{X}^T\boldsymbol{\Psi}_1\mathbf{X}\boldsymbol{\Theta} + 2\rho\log\boldsymbol{\Theta}
\end{aligned}$$

Note that this is actually (slightly) cheaper to compute than the unmodified gradient, since diagonal matrix multiplications are cheap, we removed many matrix multiplications, and most of the few matrix multiplications we did add are duplicates.

3 EFFICIENT EVALUATION OF LOG-LIKELIHOOD

Recall that the negative log likelihood is as follows, ignoring irrelevant constants.

$$\mathrm{NLL} = -\log|\boldsymbol{\Psi}_1\otimes\boldsymbol{\Theta} + \mathbf{I}\otimes\boldsymbol{\Psi}_{2,w}| + \mathrm{tr}\left[\boldsymbol{\Psi}_{2,w}\mathbf{S}_2\right] + \mathrm{tr}\left[\boldsymbol{\Psi}_1\mathbf{X}\boldsymbol{\Theta}\mathbf{X}^T\right]$$

For calculating our gradients, we already used certain decompositions, which we can re-use in the evaluation of our NLL. Evaluating the NLL is useful for convergence checking and line searching.

$$\begin{aligned}
\mathrm{NLL} &= -\log|\mathbf{I}\otimes\mathbf{P}|\,|\boldsymbol{\Lambda}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{D}|\left|\mathbf{I}\otimes\mathbf{P}^T\right| + \mathrm{tr}\left[\boldsymbol{\Psi}_{2,w}\mathbf{S}_2\right] + \mathrm{tr}\left[\boldsymbol{\Psi}_1\mathbf{X}\boldsymbol{\Theta}\mathbf{X}^T\right]\\
&= -\log|\boldsymbol{\Lambda}\otimes\mathbf{I}+\mathbf{I}\otimes\mathbf{D}|\,|\boldsymbol{\Theta}|^{d_1} + \mathrm{tr}\left[\boldsymbol{\Psi}_{2,w}\mathbf{S}_2\right] + \mathrm{tr}\left[\boldsymbol{\Psi}_1\mathbf{X}\boldsymbol{\Theta}\mathbf{X}^T\right]\\
&= -\log|\boldsymbol{\Lambda}\oplus\mathbf{D}| - d_1\log|\boldsymbol{\Theta}| + \mathrm{tr}\left[\boldsymbol{\Psi}_{2,w}\mathbf{S}_2\right] + \mathrm{tr}\left[\boldsymbol{\Psi}_1\mathbf{X}\boldsymbol{\Theta}\mathbf{X}^T\right]
\end{aligned}$$

The two trace terms require taking traces of matrices already computed for the gradients, so could be re-used. The log determinant of a Kronecker sum of diagonal matrices is easy to compute (sum of the logs of the diagonals); thus, the cost reduces to creating the decompositions and finding the determinant of $\boldsymbol{\Theta}$, both of which are expensive operations but are comparable to the cost of gradient evaluation. In fact, the Cholesky factors of $\boldsymbol{\Theta}$ are computed during simultaneous diagonalization, making determinant computation easy.

References

Dahl, A., Hore, V., Iotchkova, V., & Marchini, J. (2013, December). Network inference in matrix-variate Gaussian models with non-independent noise [arXiv:1312.1622 [stat]]. https://doi.org/10.48550/arXiv.1312.1622