antGLasso: An Efficient Tensor Graphical Lasso Algorithm

BAILEY ANDREW, University of Leeds, UK
DAVID WESTHEAD, University of Leeds, UK
LUISA CUTILLO, University of Leeds, UK

The class of bigraphical lasso algorithms (and, more broadly, 'tensor'-graphical lasso algorithms) has been used to estimate dependency structures within matrix and tensor data. However, all current methods to do so take prohibitively long on modestly sized datasets. We present a novel tensor-graphical lasso algorithm that directly estimates the dependency structure, unlike its iterative predecessors. This provides a speedup of multiple orders of magnitude, allowing this class of algorithms to be used on large, real-world datasets. We consider its applicability to video data and single-cell RNA sequencing datasets.

CCS Concepts: • **Theory of computation** → **Design and analysis of algorithms**; • **Mathematics of computing** → *Probability and statistics*; • **Applied computing** → *Computational genomics*; *Transcriptomics.*

Additional Key Words and Phrases: bigraphical lasso, network inference, scRNA sequencing

1 INTRODUCTION

We often make independence assumptions to simplify the modelling of complex data. The strength of an independence assumption exists on a spectrum. One could assume complete independence, independence except on a sparse subset, or independence of samples but not features, among others.

If one focuses on conditional dependencies, there is a natural way to model data under a Gaussian assumption. We say that two datapoints are conditionally independent (with respect to a dataset) if knowing one provides no information about the other that is not already contained in the rest of the dataset. As an example, suppose that for some random variable x we have datapoints $a = x$, $b = 2x$, and $c = 3x$. Clearly b is dependent on a - however, it is not conditionally dependent, as with c we can already predict the value of b. b provides no useful information about the value of a above that which is already contained in the rest of the dataset.

For normally distributed data, conditional dependencies are encoded in the inverse of the covariance matrix (the 'precision' matrix). Specifically, two datapoints are conditionally dependent on each other if and only if their corresponding element in the precision matrix is zero. If our dataset $\mathbf{d}$ were in the form of a vector, we could then model it as $\mathbf{d} \sim \mathcal{N}(\mathbf{0}, \Psi^{-1})$ for precision matrix Ψ.

However, datasets are often not simple vectors - for example, single-cell RNA sequencing data (scRNAseq) comes in the form of a matrix of gene expression counts whose rows are cells and columns are genes. Video data naturally requires a third-order tensor of pixels to represent it - rows, columns, and frames. To address this, we 'vectorize' matrices by stacking the columns vertically. Vectorization of tensors proceeds in much the same way. In fact, all results in this paper extend naturally to tensors.

Our matrix-variate dataset $\mathbf{D}$ can then be represented as $\mathrm{vec}[\mathbf{D}] \sim \mathcal{N}(\mathbf{0}, \Psi^{-1})$. Suppose our matrix were $n \times p$ dimensional - then $\mathrm{vec}[\mathbf{D}]$ has np elements, and its precision matrix has n^2p^2 elements. This poses a problem: the estimation of our precision matrix requires substantially more parameters than we have in our dataset and would require a large amount of memory.

This problem is not fatal - as our dataset was originally a matrix, it contains more inherent structure than its vectorized form suggests. We can make the following reasonable assumption: since the columns of the matrix have interdependencies and the rows of the matrix have interdependencies, the interdependencies of the elements of the matrix can then be computed via some deterministic function of the row-wise and column-wise dependencies. That is, for some function ζ, $\mathrm{vec}[\mathbf{D}] \sim \mathcal{N}(\mathbf{0}, \zeta(\Psi_{\mathrm{row}}, \Psi_{\mathrm{col}})^{-1})$. For ζ, we chose the Kronecker Sum as it has convenient sparsity properties and a natural interpretation as the Cartesian product between graphs that encode the row-wise and column-wise dependencies[6].

In this manner, we can learn a series of statistically informative graphs of dependencies from tensor data. This allows problems involving numerical data to be transferred to a graphicalg domain. When coupled with graph clustering algorithms, BiGLasso algorithms can be used for unsupervised learning.

2 BACKGROUND

The Kronecker sum bigraphical lasso (BiGLasso) model was first considered by Kalaitzis et al[6]. BiGLasso is the multi-axis analog to graphical lasso methods [1, 2], which are used to estimate covariance matrices of data drawn from a multivariate Gaussian distribution. The Kronecker sum of two matrices, $\mathbf{A} \oplus \mathbf{B}$, can be expressed in terms of Kronecker products: $\mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{B}$. When the matrices $\mathbf{A}, \mathbf{B}$ are adjacency matrices of graphs, the Kronecker sum has the interpretation as the Cartesian product of those graphs[15]. This sum is one choice of function ζ to combine the per-axis precision matrices into the precision matrix of the vectorized dataset, $\mathrm{vec}[\mathbf{D}] \sim \mathcal{N}(\mathbf{0}, (\Psi_{\mathrm{row}} \oplus \Psi_{\mathrm{col}})^{-1})$.

Other choices for ζ have been considered, such as using the Kronecker product[17, 22], and the square of the Kronecker sum[19]. Each method has its advantages - the advantages of a Kronecker sum structure are its interpretability as a graph product, stronger sparsity, and its allowance of inter-task transfer (information gained in the estimation of one component precision matrix is informative in the estimation of the other)[6].

The original BiGLasso model was very slow to converge to a solution, in large part due to its non-optimal space complexity of $O(n^2p^2)$. This prohibited its use on large datasets (measuring in a couple hundred samples and/or features). Numerous modifications have been made to the algorithm to improve its speed and achieve an optimal space complexity of $O(n^2 + p^2)$, such as scBiGLasso[9], TeraLasso[3], and EiGLasso[21]. Of these, TeraLasso is notable in that it generalizes to an arbitrary amount of axes, i.e. $\zeta(\mathbf{\Psi}_1, ..., \mathbf{\Psi}_k) = \mathbf{\Psi}_1 \oplus ... \oplus \mathbf{\Psi}_k$. However, all of these algorithms are still iterative. The convergence time can vary greatly depending on the precise values of the data, and in general they struggle to handle datasets with a couple thousand samples and/or features. Our proposed variant, antGLasso (**an**alytic **t**ensor-**g**raphical **lasso**), does not suffer from these issues, preserves optimal memory complexity, and like TeraLasso can work on datasets with an arbitrary amount of axes.

All algorithms listed rely on a normality assumption. One could attempt to manually transform non-normal data to a Gaussian, if they knew the right transform to use. An alternative method is to use the Nonparanormal Skeptic[10], an algorithm which directly estimates the empirical covariance matrices of data that has been transformed to a Gaussian by some unspecified function[9]. Since all previous BiGLasso algorithms mentioned can be framed in terms of accepting these empirical covariance matrices as input, they can circumvent the normality assumption.

3 METHODOLOGY AND RESULTS

In the following we introduce the main theorems and results of our study. For our implementation, we used Python Version 3.9.12, Numpy Version 1.22.2[4], Scipy Version 1.7.3[18], Sklearn Version 1.0.2[13], Matplotlib Version 3.5.1[5], and Pandas Version 1.4.2[12, 20]. All tests were run on a MacBook Pro (13-inch, M1, 2020) with 8GB of RAM, using Jupyter Notebooks[7, 14].

3.1 Theorems

Let s be the number of samples of tensors $(\mathcal{Y}_1, ...\mathcal{Y}_s)$, each with shape $(d_1, ..., d_K)$. Each of these tensors is assumed to be i.i.d from our tensor-variate normal distribution for which we want to estimate the per-axis precision matrices. It is not uncommon for $n = 1$. It will be helpful to define $m_\ell = \frac{\prod_{i=1}^K d_i}{d_\ell}$. Let $\mathbf{\Psi}_\ell$ be the precision matrix of the ℓth axis, and S_ℓ be the empirical covariance matrix for the ℓth axis. These can either be obtained via the Nonparanormal Skeptic, or by the formula $\frac{1}{nm_\ell} \sum_{i=1}^n \mathrm{Y}_{i,(\ell)} \mathrm{Y}_{i,(\ell)^T}$, where $\mathbf{A}_{(\ell)}$ is the 'matricization' of the ℓth axis[1]. Another piece of tensor-specific notation we use is the ℓ-mode product, $\mathcal{Y} \times_\ell \mathbf{M}$, which intuitively is multiplying a tensor by a matrix along its ℓth dimension. When the input is a two-dimensional tensor (a matrix), $\times_1$ and $\times_2$ correspond to left-multiplying by the transpose and right-multiplying, respectively. For an overview of tensor notation, we direct the reader to the comprehensive report by Kolda and Bader[8]. We use $\circ$ to represent the Hadamard product.

The general structure of the algorithm is as follows: we first directly estimate the eigenvectors of the solution. We then use the eigenvectors to diagonalize the covariance matrix of the tensor-variate distribution, such that we can directly estimate the eigenvalues from the variances of the data. Proofs are given in the supplementary material available at https://github.com/BaileyAndrew/antGLasso-Implementation. An implementation of our algorithm is also available at that repository.

Theorem (Eigenvectors of Precision Matrices). *$\mathbf{V}_\ell$, the eigenvectors of $\mathbf{\Psi}_\ell$, are the eigenvectors of $\mathrm{S}_\ell \circ \mathbf{K}_{m_\ell}^{2m_\ell - 1}$, where $\mathbf{K}_b^a$ is the matrix with diagonal elements a and off-diagonal elements b.*

Theorem (Eigenvalues of Precision Matrices). *Let* $\mathbf{a}$ *be the vector of length $\prod_\ell^K d_\ell$ whose $(\sum_\ell i_\ell \prod_{j=1}^{\ell-1} d_j)$th element is $\frac{1}{\mathrm{var}[[\mathcal{Y} \times_1 \mathbf{V}_1^T \times_2 ... \times_K \mathbf{V}_K^T]_{i_1 ... i_K}]}$. Then there is a system of linear equations relating* $\mathbf{a}$ *to the eigenvalues of $(\mathbf{\Psi}_1, ..., \mathbf{\Psi}_K)$. This system has non-optimal space complexity, but can be reworked into series of small, sparse matrix multiplications via a Monte Carlo estimation to preserve space optimality.*

In this way, we estimate the eigenvectors and eigenvalues of each of the per-axis precision matrices, after which it is trivial to reconstruct them. Unfortunately, Theorem 3.1 requires the whole dataset $\mathcal{Y}$, whereas the use of the Nonparanormal Skeptic requires that the algorithm be parametrizable in terms of the empirical covariance matrices. We can circumvent this by introducing a heuristic, $\mathrm{var}\left[\mathcal{X}_{i_1,...,i_K}\right] \approx d_\ell \Delta_{i_\ell} \mathbf{V}_\ell^T \mathrm{S}_\ell \mathbf{V}_\ell \Delta_{i_\ell}^T$, where Δ_{i_ℓ} is a row vector of zeros except for a one at position i_ℓ. We call this variant of the algorithm 'Heuristic antGLasso'.

3.2 Results on Simulated Data

We first tested our algorithms on simulated data, in which the precision matrices were drawn from an inverse Wishart distribution. We found the iterative algorithms tended to converge slowly when the degrees of freedom parameter of the inverse Wishart distribution was small, and quickly when it was large. Additionally, iterative algorithms converged much slower on small sample data (Figure 1). We can see that antGLasso is by far the fastest algorithm, giving up to two orders of magnitude improvement over the next fastest algorithm (EiGLasso).

Not pictured are the results for tensor data, due to space constraints. For three-axis tensors, antGLasso is still markedly faster than TeraLasso in the small sample case - we did not have enough RAM to test the large sample case. For four-axis and larger, the difference between the algorithms becomes negligible - this is because the major runtime constraint becomes the calculation of the empirical covariance matrices, whose complexity grows exponentially with the amount of axes and thus rapidly drowns out the performance gains of antGLasso.

[1]This is similar to vectorization, except instead of stacking all axes into a column vector, we preserve the ℓth axis, reducing our tensor to a matrix instead.

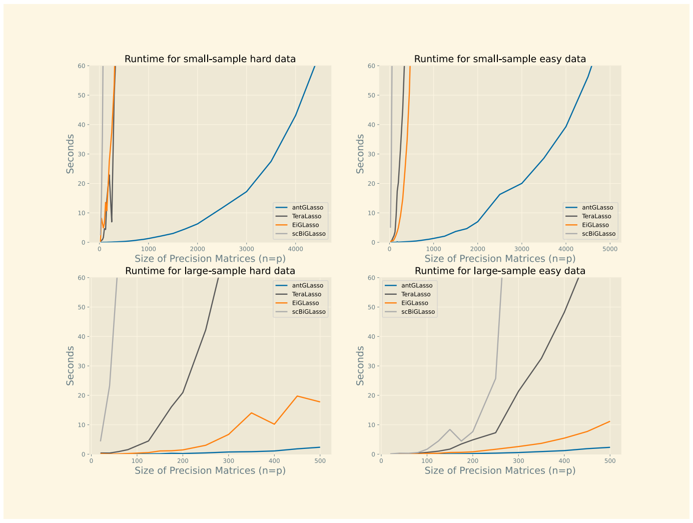

Fig. 1. A comparison of runtimes of four BiGLasso algorithms on simulated data. We consider small samples ($s = 1$) and large samples ($s = 100$), as well as data drawn from distributions that tend to cause algorithms to converge slowly ('hard') and quickly ('easy').

By extrapolating the (cubic) runtime curves to the largest dataset whose precision matrices could fit on our computer's RAM, we can predict that antGLasso would take 2 days to run. Thus, further improvements to speed would be useful, but the constraint limiting its applicability to even larger datasets is likely to be memory. Unfortunately, there is not much room for improving memory usage - perhaps modifications could be made to remove some intermediate products, but even in the ideal case we would be reducing memory usage by a factor of two at most. Improving speed would likely require a fundamentally different approach, as the majority of the runtime is taken up by a single eigendecomposition (per axis).

In terms of practical performance (Figure 2), antGLasso does not perform as well as other algorithms, although this gap drops as the number of samples increase. Interestingly, Heuristic antGLasso performs comparably to the others while preserving the speed of antGLasso. We believe this is because antGLasso directly estimates the variances of the data, which will be inaccurate for small samples. The heuristic does not rely on estimating the variance, so it circumvents this issue. The gap between the two variants drops as the number of samples increases, as the non-heurisitc antGLasso's variance estimate increases in accuracy. When tested with 100 samples the two variants were indistinguishable.

3.3 Results on and Applicability to Real World Data

We have shown that, at least on simulated data, Heuristic antGLasso performs comparably but much quicker than existing algorithms. This enables the application of BiGLasso methods to datasets in the size of thousands. Of particular interest to us is its applicability to single-cell RNA sequencing data (scRNA), which typically comes in the form of a matrix of counts (with one axis representing cells, and another representing genes). One way in which it might be useful is in producing a pseudotime ordering of the cells[2], such as that done by Monocle[16].

[2]Given a batch of cells following some developmental trajectory, some cells may be further along than others. Rather than considering physical time, pseudotime measures the amount of progress along the trajectory.

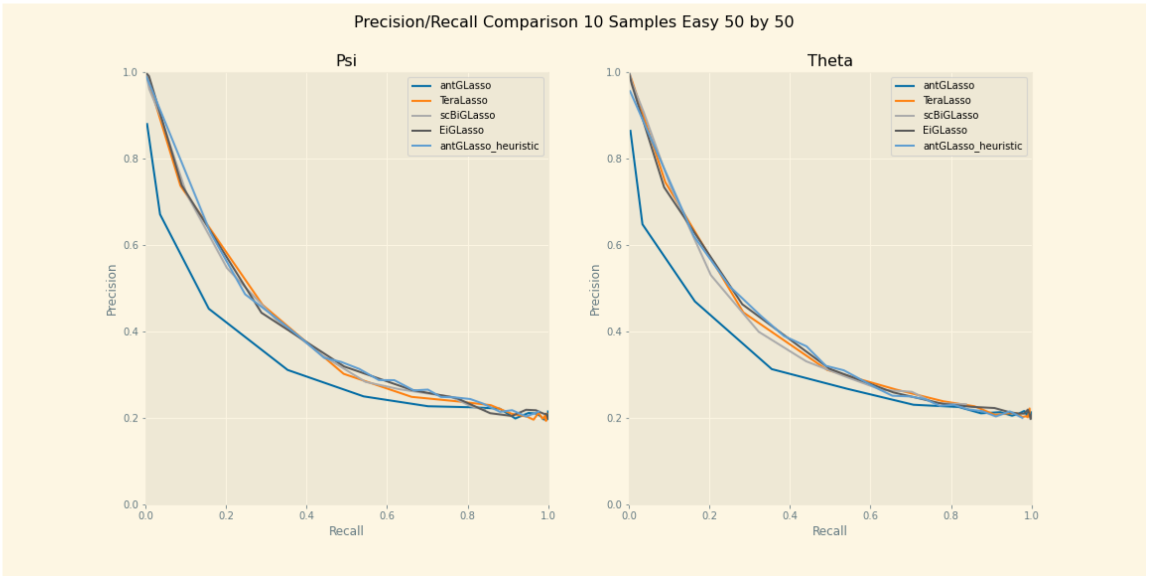

Fig. 2. Precision-recall curves for two-dimensional simulated data from a distribution that converged quickly ('easy'), for 50x50 input matrices. Note that antGLasso does worse than competitors, but Heuristic antGLasso does not.

To see why we might expect this, consider the case of video data. Frame 100 will be dependent on the frames that come before it - however, most useful information that frame 98 contains for prediction of frame 100 will likely be contained in frame 99 as well. Thus, frame 100 is conditionally dependent on frame 99, but not frame 98. Given the graph of conditional dependencies, we might expect to be able to order the frames using this observation. The hope is that pseudotime may also be estimated in an analogous way.

As a proof-of-concept for video data, we consider the video of a rubber duck from the COIL-20 dataset[11], as a downsampled version of this video was also considered by the original BiGLasso paper for a similar purpose. If we scramble the rows, frames, and columns of the video, we can recover the original video from the outputs of antGLasso, as seen in Figure 3. The reconstruction is almost perfect, except for one mistake in the row reconstruction which results in the duck being cut in half - this is due to the difficulty in figuring out which row to start on.

The strong results for video reconstruction suggest that this method would indeed be usable for pseudotime reconstruction. Unfortunately, when we apply it to single cell data - even simulated single cell data with much real-world complexity removed - we do not get usable results. scRNA data is not as highly structured as video data; in fact, we can modify video data a bit to capture some of the complexity of scRNA data, by duplicating frames and adding noise. Due to conditional independence being a global[3] phenomenon, duplicating frames interferes with the dependency graph. This is similar to the scenario we considered in the introduction, with datapoints $\{x, 2x, 3x\}$ - except here we have added some noise. It is reasonable to believe that this makes video data a closer analog to scRNA data, as scRNA data is noisy and there will be many cells at similar positions on their trajectories. When we modify video data in this way, reconstruction accuracy drops rapidly as a function of number of duplications - already when frames are

[3]Since we condition over the whole dataset.

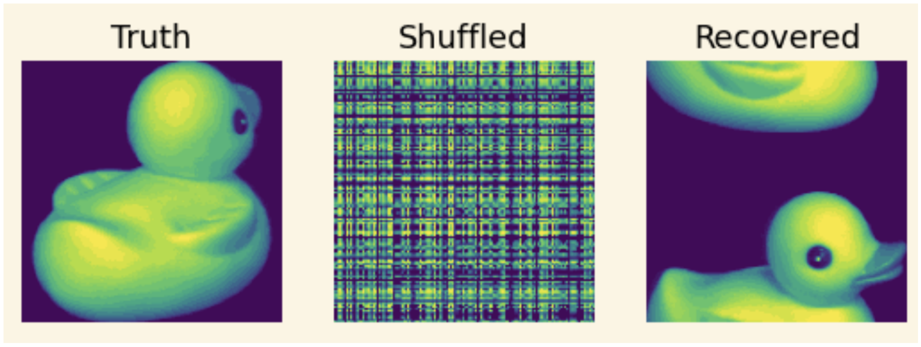

Fig. 3. Temporal recovery from a video of a spinning rubber duck. The left image is a frame from the original video, the middle image is a frame from the shuffled video, and the right image is a frame from the reconstructed video. Not represented in the image is the recovery of the frames of the video, which was perfect.

duplicated five times each the reconstruction accuracy dips below 10%. This suggests that temporal recovery can only be done in highly structured cases with few redundancies - it is unlikely to be applicable to scRNA data.

However, antGLasso is capable of being used for clustering. In fact, the authors of scBiGLasso already considered clustering stages of the cell cycle in mouse embryo stem cell scRNA data[9], although they reduced the size of the dataset to 167 mitosis-related genes using domain knowledge. antGLasso was tested on the same dataset, but with the top 8000 most expressed genes instead, and got similar results.

On simulated examples, we found that the best way to use antGLasso for clustering was to use Heuristic antGLasso without the Nonparanormal Skeptic to produce precision matrices, and then invert them into covariance matrices before performing a clustering algorithm such as spectral clustering on the weighted graph represented by the matrix. Since the algorithm returns an eigendecomposition, this inversion is trivial. For temporal recovery, we found that Heuristic antGLasso with the Nonparanormal Skeptic performed best, and that it did not matter whether we used the precision or covariance matrices. All tests mentioned in this paragraph were also performed on EiGLasso to verify that any deficiencies were not due to our algorithm specifically, but rather a more general feature of BiGLasso as a methodology.

4 CONCLUSION

We have created a new BiGLasso algorithm, antGLasso, which is much faster than current competitors, while preserving (in the heuristic case) their accuracy. It can be applied to tensors of arbitrary dimensions, like TeraLasso, and performs well on simulated data as well as highly-structured real-world data. It falters when applied to more complicated data such as scRNA data. However, this is a systemic problem with BiGLasso algorithms rather than our implementation specifically.

For future work, we would like to understand better when the Nonparanormal Skeptic should be used, as we found that it decreased performance when clustering highly multimodal datasets but increased performance for temporal recovery. Additionally, we would like to investigate how to circumvent the known issues with applying this method to less structured datasets, such as scRNA data.

REFERENCES

[1] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. 2008. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *Journal of Machine Learning Research* 9, 15 (2008), 485–516. http://jmlr.org/papers/v9/banerjee08a.html

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2007. Sparse inverse covariance estimation with the lasso. https://doi.org/10.48550/ARXIV.0708.3517

[3] Kristjan Greenewald, Shuheng Zhou, and Alfred Hero. 2017. Tensor Graphical Lasso (TeraLasso). https://doi.org/10.48550/ARXIV.1705.03983

[4] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[5] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[6] Alfredo Kalaitzis, John Lafferty, Neil D. Lawrence, and Shuheng Zhou. 2013. The Bigraphical Lasso. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 1229–1237. https://proceedings.mlr.press/v28/kalaitzis13.html

[7] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. 2016. Jupyter Notebooks - a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, Fernando Loizides and Birgit Scmidt (Eds.). IOS Press, Netherlands, 87–90. https://eprints.soton.ac.uk/403913/

[8] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (September 2009), 455–500. https://doi.org/10.1137/07070111X

[9] Sijia Li, Martin Lopez-Garcia, Neil Lawrence, and Luisa Cutillo. 2022. Scalable Bigraphical Lasso: Two-way Sparse Network Inference for Count Data. (03 2022).

[10] Han Liu, Fang Han, Ming Yuan, John Lafferty, and Larry Wasserman. 2012. The Nonparanormal SKEPTIC. https://doi.org/10.48550/ARXIV.1206.6488

[11] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. 1996. *Columbia Object Image Library (COIL-20.* Technical Report.

[12] The pandas development team. 2020. *pandas-dev/pandas: Pandas.* https://doi.org/10.5281/zenodo.3509134

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[14] Fernando Pérez and Brian E. Granger. 2007. IPython: a System for Interactive Scientific Computing. *Computing in Science and Engineering* 9, 3 (May 2007), 21–29. https://doi.org/10.1109/MCSE.2007.53

[15] Gert Sabidussi. 1959/60. Graph Multiplication. *Mathematische Zeitschrift* 72 (1959/60), 446–457. http://eudml.org/doc/183624

[16] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall Lennon, Kenneth Livak, Tarjei Mikkelsen, and John Rinn. 2014. Pseudo-temporal ordering of individual cells reveals dynamics and regulators of cell fate decisions. *Nature biotechnology* 32 (03 2014). https://doi.org/10.1038/nbt.2859

[17] Theodoros Tsiligkaridis and Alfred O. Hero. 2013. Covariance Estimation in High Dimensions Via Kronecker Product Expansions. *IEEE Transactions on Signal Processing* 61, 21 (nov 2013), 5347–5360. https://doi.org/10.1109/tsp.2013.2279355

[18] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[19] Yu Wang, Byoungwook Jang, and Alfred Hero. 2020. The Sylvester Graphical Lasso (SyGlasso). In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 108)*, Silvia Chiappa and Roberto Calandra (Eds.). PMLR, 1943–1953. https://proceedings.mlr.press/v108/wang20d.html

[20] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a

[21] Jun Ho Yoon and Seyoung Kim. 2020. EiGLasso: Scalable Estimation of Cartesian Product of Sparse Inverse Covariance Matrices. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI) (Proceedings of Machine Learning Research, Vol. 124)*, Jonas Peters and David Sontag (Eds.). PMLR, 1248–1257. https://proceedings.mlr.press/v124/ho-yoon20a.html

[22] Shuheng Zhou. 2014. Gemini: Graph estimation with matrix variate normal instances. *The Annals of Statistics* 42, 2 (apr 2014). https://doi.org/10.1214/13-aos1187