The Bigraphical Lasso

Alfredo Kalaitzis A.KALAITZIS@UCL.AC.UK
Department of Statistical Science, University College London, Gower Street, London WC1E 6BT, UK

John Lafferty LAFFERTY@GALTON.UCHICAGO.EDU
Department of Statistics, University of Chicago, 5734 S. University Ave., Chicago, IL 60637, USA

Neil D. Lawrence N.LAWRENCE@SHEFFIELD.AC.UK
Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP, UK

Shuheng Zhou SHUHENGZ@UMICH.EDU
Department of Statistics, University of Michigan, 1085 S. University Ave., Ann Arbor, MI 48109, USA

Abstract

The i.i.d. assumption in machine learning is endemic, but often flawed. Complex data sets exhibit partial correlations between both instances and features. A model specifying both types of correlation can have a number of parameters that scales quadratically with the number of features and data points. We introduce the bigraphical lasso, an estimator for precision matrices of matrix-normals based on the Cartesian product of graphs. A prominent product in spectral graph theory, this structure has appealing properties for regression, enhanced sparsity and interpretability. To deal with the parameter explosion we introduce ℓ_1 penalties and fit the model through a flip-flop algorithm that results in a linear number of lasso regressions. We demonstrate the performance of our approach with simulations and an example from the COIL image data set.

1. Introduction

When fitting Gaussian models to data, an independence assumption is usually made across data points and the covariance matrix is fit by penalized likelihood. The number of parameters in the covariance matrix can be reduced by low rank constraints such as factor analysis (see e.g. Tipping & Bishop, 1999) or by constraining the inverse covariance (or precision) to be sparse (e.g. Banerjee et al., 2008). A sparse precision matrix defines a Gaussian Markov random field relationship which is conveniently represented by a weighted undirected graph (Lauritzen, 1996). Nodes that are not neighbors in the graph are conditionally independent given all other nodes. Models specified in this way encode conditional independence structures between features.

An alternative Gaussian modeling approach was introduced by Lawrence (2005) where the i.i.d. assumption is across data *features*. Lawrence (2012) showed that spectral dimensionality reduction methods have an interpretation as sparse graphical models over the instances (rows), with the parameters of the covariance fit by maximum likelihood (or in the case of *local linear embeddings* (Roweis & Saul, 2000) by maximizing a pseudolikelihood).

Feature independence or data point independence is a model choice issue. Both are special cases of a more general framework that models conditional independence relationships between features and data points together. This is the type of model that we study in this paper. Specifically, we are concerned with estimating a sparse graph that interrelates both features and data points. In video data, for instance, both the frames of the data and the image variables (pixels) are of course correlated. In Section 5 we illustrate our modeling approach by estimating the conditional independence structure for simple video in the COIL data set. In gene expression data, as another example, it may be of interest to extract a gene network from the expression values and also estimate ancestral relationships among the samples in a separate network.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

Such problems motivate the models and estimation algorithms studied in this paper.

1.1. Graphical Lasso and the Matrix-Variate Normal

The *graphical lasso* (GLasso, Friedman et al., 2008; Banerjee et al., 2008) is a computationally efficient penalized likelihood algorithm for learning sparse Gaussian Markov random fields (GMRF) over features of i.i.d. vector-variate Gaussian samples.

The matrix variate normal (Dawid, 1981; Gupta & Nagar, 1999) is a Gaussian density which can be applied to a matrix through first taking a vectorized (*vec*) representation[1] of the matrix samples $\mathbf{X} \in \mathbb{R}^{n \times p}$ and assuming the covariance matrix has the form of a **Kronecker product (KP)** between two covariance matrices, separately associated with the rows and columns of the data matrix. The KP assumption for the covariance implies that the precision matrix is also a KP, which is formed from the KP of the precision matrices associated with the rows and columns $(\mathbf{\Psi} \otimes \mathbf{\Theta})$.

One approach to sparse graphical modeling of matrix data is to combine the KP-structured matrix normal with the graphical Lasso. Dutilleul (MLE, 1999) used a flip-flop approach for maximum likelihood estimation of the parameters of the matrix-normal, and much later Zhang & Schneider (2010) used it for MAP estimation with sparsity penalties on the precision matrices. More recently, Leng & Tang (2012) applied the SCAD penalty (Fan & Li, 2001) as well as the Lasso penalty to the matrix normal. Tsiligkaridis et al. (2013) analyzed the asymptotic convergence of Kronecker GLasso and carried out simulations showing significant convergence speedups over GLasso.

While the KP structure arises naturally when considering matrix-normals, it results in relatively dense dependencies between the rows. More precisely, if Ψ_{ij} in $\mathbf{\Psi} \otimes \mathbf{\Theta}$ is non-zero (for example, corresponding to an edge between samples i and j in the design matrix $\mathbf{X}$) then many edges between features of sample i and sample j (as many as in $\mathbf{\Theta}$) will also be active. A *sparser* structure would benefit situations where the connection between a feature of some sample and a different feature of any other sample is of no interest or redundant. For instance in a video, it is reasonable to assume that the neighbors of pixel (i, j) in frame k are conditionally independent of the neighbors of pixel (i, j) in frame $k+1$, conditioned on pixels (i, j) of both frames.

[1] Vectorization of a matrix involves converting the matrix to a vector by stacking the columns of the matrix.

1.2. The Bigraphical Lasso

We propose the *bigraphical lasso* (BiGLasso), a model for matrix-variate data that preserves their column/row structure and, like the KP-structured matrix normal, simultaneously learns two graphs, one over rows and one over columns of the matrix samples. The model is trained in a flip-flop fashion, so the number of lasso regressions reduces to $\mathcal{O}(n + p)$. However, the model preserves the matrix structure by using a novel **Kronecker *sum* (KS)** structure for the precision matrix, $(\mathbf{\Psi} \otimes \mathbf{I}) + (\mathbf{I} \otimes \mathbf{\Theta})$ instead of the KP. This structure enjoys *enhanced sparsity* in comparison to the conventional KP structure of matrix normals.

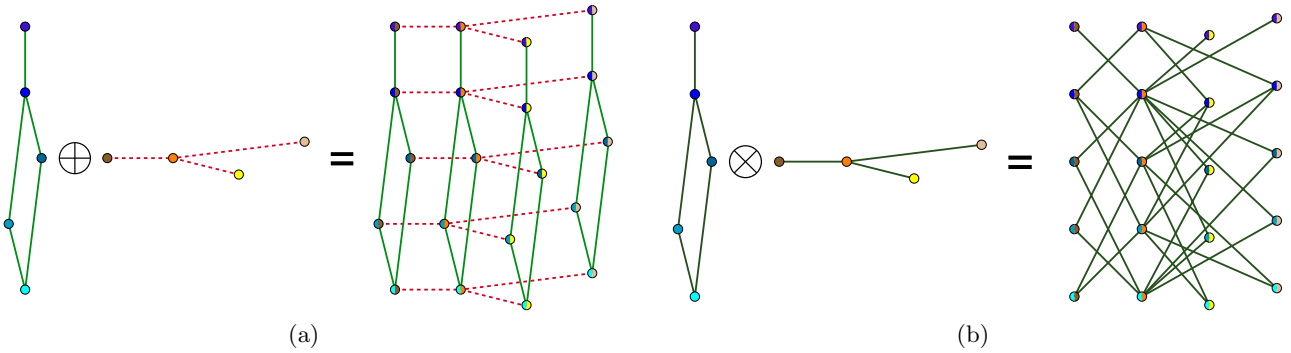
Graph Cartesian Product When operating on adjacency matrices, the KS is also known in algebraic graph theory as the *Cartesian product of graphs* and is arguably the most prominent of graph products (Sabidussi, 1959; Chung, 1996; Imrich et al., 2008). This endows the output of the BiGLasso with a more intuitive and interpretable graph decomposition of the induced GMRF, see figure 1 for an example.

Enhanced Sparsity For a matrix density $\lambda \in [0, 1]$ of both precision matrices the KS has $\mathcal{O}(\lambda np(n + p))$ non-zeros, whereas the KP has $\mathcal{O}(\lambda n^2 p^2)$ non-zeros.

Better Information Transfer in GPs Kronecker product (KP) forms have a known weakness, referred to in the Gaussian process (GP) literature as the *cancellation of inter-task transfer*. Zellner (1962), Binkley & Nelson (1988) pointed out how the consideration of correlations between regression equations leads to a gain in efficiency. A KP covariance function can compromise potentially useful dependencies between the responses of a multi-output GP. In particular, Bonilla et al. (2008, §2.3) showed the following regarding multi-output GPs: if a noise-free covariance is KP-structured[2] and the same inputs are conditioned for all outputs[3], then the predictive mean *uncouples* the different tasks, that is, the posterior (GP conditioned on inputs) factorizes across the outputs such that they are independently computable. The property does not apply under the presence of additive noise, hence the outputs remain coupled. This result first arose in geostatistics under the name of *autokrigeability* (Wackernagel, 2003) and is also discussed for covariance functions in (O'Hagan, 1998). On the contrary, due to its additive form a KS-structured noise-free covariance *enables* inter-task transfer.

[2] One factor for covariances between outputs and one for covariances between points.

[3] A conditioning structure referred to as a *block design*.



(a) (b)

Figure 1. When acting on adjacency matrices of graphs, the Kronecker-sum acts as the Cartesian-product (a) and the Kronecker-product as the tensor-product (b). The lattice-like structure of the Cartesian-product is ideal for modeling dependencies between features as well as samples. More generally, since the Cartesian-product is associative, it can be generalized to model GMRFs of higher-dimensional arrays. Note that here we do not include self-edges (zeros on the diagonals). Based on figures created by David Eppstein, `http://en.wikipedia.org/wiki/Graph_product`.

The autokrigeability result relies on the factorisable property of the inverse of a KP, $(\mathbf{\Psi} \otimes \mathbf{\Theta})^{-1} = \mathbf{\Psi}^{-1} \otimes \mathbf{\Theta}^{-1}$. This property enables a simple flip-flop approach to fitting but also forbids the exploitation of correlation between different outputs. On the other hand, by coupling the outputs with additive noise on the KP, flip-flop is no longer straightforward. Stegle et al. (2011) addressed this issue by adding i.i.d. noise to a KP covariance — a *low-rank* factor for confounding effects and a *sparse-inverse* factor for inter-sample dependencies — and exploiting identities of the vec(.) notation for efficient computation within the matrix normal model.

To summarize our contributions, in contrast to existing approaches that use the KP structure, the KS *preserves* the inter-task transfer. Our algorithm maintains the simplicity of the flip-flop with a simple trick of transposing the matrix-variate (samples become features and vice versa). At the same time, the induced Cartesian factorization of graphs provides a more parsimonious interpretation of the induced Markov network.

The rest of this paper is structured as follows. We describe the matrix normal model with the KS-structured inverse-covariance in §2. In §3, we present the BiGLasso algorithm for learning the parameters of the KS inverse-covariance. We present some simulations in comparison to a recent KP-structured matrix normal model of Leng & Tang (SMGM, 2012) in §4 and an application to an example from the COIL dataset in §5. We conclude in §6.

2. Matrix-normal model with the Kronecker-sum structure

To motivate our KS-structured model, first we consider the standard case where matrix-variate data $\mathbf{Y}$ are sampled i.i.d. from a matrix-normal distribution (or matrix Gaussian). This is a generalization of the Gaussian distribution towards higher-order tensor support[4]. The matrix normal can be reparametrized such that the support is now over vectorised representations of random matrices and it naturally follows a KP-structured covariance,

$$\mathrm{vec}(\mathbf{Y}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Psi}_n^{-1} \otimes \mathbf{\Theta}_p^{-1}\right). \tag{1}$$

2.1. The KP-based SMGM

Under the sparsity assumption of the KP-structured precision matrix $\mathbf{\Psi}_n \otimes \mathbf{\Theta}_p$, the SMGM estimator (Sparse Matrix Graphical Model) of Leng & Tang (2012) is an extension of GLasso that iteratively minimizing the ℓ_1-penalized negative likelihood function of $(\mathbf{\Psi}_n, \mathbf{\Theta}_p)$ for the KP-structured matrix normal:

$$\min_{\mathbf{\Theta}_p, \mathbf{\Psi}_n} \Big\{ \tfrac{1}{Nnp} \sum_{i=1}^{N} \mathrm{tr}\left(\mathbf{Y}_i \mathbf{\Theta}_p \mathbf{Y}_i^\top \mathbf{\Psi}_n\right) - \tfrac{1}{n}\log|\mathbf{\Psi}_n| - \tfrac{1}{p}\log|\mathbf{\Theta}_p| + \lambda_1||\mathbf{\Psi}_n||_1 + \lambda_2||\mathbf{\Theta}_p||_1 \Big\}, \tag{2}$$

where $\mathbf{Y}_i$ is the i-th matrix sample, N is the sample size and λ_1, λ_2 the regularization parameters.

Complexity and Biconvexity The SMGM objective is a *non-convex* problem as the trace produces

[4] A vector is an order-1 tensor, a matrix is an order-2 tensor and so on.

interaction terms between the two precision matrices. However, it is *biconvex* as convexity holds only with respect to either precision matrix individually but not jointly. The flip-flop approach to minimization proceeds by fixing one of the precision matrices (say, the columns-precision matrix $\mathbf{\Theta}_p$), thus reducing the above to a convex GLasso problem on $\mathbf{\Psi}_n$ with a projected covariance ($\sum_i \mathbf{Y}_i \mathbf{\Theta}_p \mathbf{Y}_i^\top$). Similarly, another GLasso step fits the the columns-precision matrix $\mathbf{\Theta}_p$ with a fixed rows-precision $\mathbf{\Psi}_n$. As such its complexity on each iteration is that of two GLasso problems or $\mathcal{O}(n+p)$ lasso regressions. Note that each GLasso step involves an additive $\mathcal{O}(N)$ time-complexity term as the summation depends on a new estimate of the precision matrix, so in total the complexity is $\mathcal{O}(n+p+N)$.

2.2. The KS-based BiGLasso

Let $\mathbf{Y} \in \mathbb{R}^{n \times p}$ be a random matrix. If its rows are generated as i.i.d. samples from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_p)$, then the sampling distribution of the sufficient statistic $\mathbf{Y}^\top \mathbf{Y}$ is $Wishart(n, \mathbf{\Sigma}_p)$ with n degrees of freedom and scale matrix $\mathbf{\Sigma}_p$. Similarly, if the columns are generated as i.i.d. samples from $\mathcal{N}(\mathbf{0}, \mathbf{\Gamma}_n)$, then the sampling distribution is $Wishart(p, \mathbf{\Gamma}_n)$.

From a *maximum entropy* viewpoint we can constraint these second-order moments in a model both for the features and the datapoints of a design matrix.

To do so, we combine these sufficient statistics in a model for the entire matrix $\mathbf{Y}$ as

$$p(\mathbf{Y}) \;\propto\; \exp\left\{-\mathrm{tr}\left(\mathbf{\Psi}_n \mathbf{Y} \mathbf{Y}^\top\right) - \mathrm{tr}\left(\mathbf{\Theta}_p \mathbf{Y}^\top \mathbf{Y}\right)\right\} \;, \quad (3)$$

where $\mathbf{\Psi}_n \in \mathbb{R}^{n \times n}$ and $\mathbf{\Theta}_p \in \mathbb{R}^{p \times p}$ are positive definite matrices. This is equivalent to a joint factorized Gaussian distribution (see §3 in supplementary material) for the $n \times p$ entries of $\mathbf{Y}$, with a precision matrix of the form

$$\mathbf{\Omega} \;\triangleq\; \mathbf{\Psi}_n \oplus \mathbf{\Theta}_p \;=\; \mathbf{\Psi}_n \otimes \mathbf{I}_p + \mathbf{I}_n \otimes \mathbf{\Theta}_p \;, \quad (4)$$

where $\otimes$ is the *Kronecker-product* and $\oplus$ the *Kronecker-sum* operator. Thus,

$$\omega_{ij,kl} \;=\; \psi_{i,k}\delta_{j,l} + \delta_{i,k}\theta_{j,l} \;, \quad (5)$$

for $i,k \in \{1,\ldots,n\}$ and $j,l \in \{1,\ldots,p\}$. As an immediate benefit of this parameterization, while the full covariance matrix has $\mathcal{O}(n^2p^2)$ entries, these are governed in our model by only $\mathcal{O}(n^2+p^2)$ parameters.

Given data in the form of some design matrix $\mathbf{Y}$, the BiGLasso estimates the sparse KS-structured inverse-covariance of a matrix normal through the ℓ_1-penalized negative likelihood function of $(\mathbf{\Psi}_n, \mathbf{\Theta}_p)$:

$$\min_{\mathbf{\Theta}_p, \mathbf{\Psi}_n} \Big\{ n\ \mathrm{tr}\left(\mathbf{\Theta}_p \mathbf{S}\right) + p\ \mathrm{tr}\left(\mathbf{\Psi}_n \mathbf{T}\right) - \ln|\mathbf{\Psi}_n \oplus \mathbf{\Theta}_p| + \lambda\|\mathbf{\Theta}_p\|_1 + \gamma\|\mathbf{\Psi}_n\|_1 \Big\} \;, \quad (6)$$

$$\text{where} \quad \mathbf{S} \triangleq \tfrac{1}{n}\mathbf{Y}^\top\mathbf{Y} \quad \text{and} \quad \mathbf{T} \triangleq \tfrac{1}{p}\mathbf{Y}\mathbf{Y}^\top \quad (7)$$

are empirical covariances across the datapoints (rows) and features (columns) respectively.

Complexity and Convexity The BiGLasso objective is a *convex* problem because the negative log-determinant is convex and the KS is an affine operation. Therefore any coordinate descent algorithm for (6) converges to the global minimum. A solution simultaneously estimates *two graphs* – one over the columns of $\mathbf{Y}$, corresponding to the sparsity pattern of $\mathbf{\Theta}_p$, and another over the rows of $\mathbf{Y}$, corresponding to the sparsity pattern of $\mathbf{\Psi}_n$. Note that (6) does not require a summation over the datapoints in each step as was the case in (2). Also note that since $\omega_{ii,jj} = \psi_{ii} + \theta_{jj}$, the diagonals of $\mathbf{\Theta}_p$ and $\mathbf{\Psi}_n$ are not identifiable (though we could restrict the inverses to correlation matrices). However, this does not affect the estimation of the graph *structure* (locations of zeros). Similarly, the BiGLasso time complexity is that of two GLasso problems or $\mathcal{O}(n+p)$ lasso regressions. Note that a naive GLasso approach on matrix data would be $\mathcal{O}(np)$ but both the BiGLasso and SMGM exploit the special KS or KP precision structures that they respectively assume.

3. A penalized likelihood algorithm for the BiGLasso

A note on notation If $\mathbf{M}$ is an $np \times np$ matrix written in terms of $p \times p$ blocks, as

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \ldots & \mathbf{M}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{M}_{n1} & \ldots & \mathbf{M}_{nn} \end{bmatrix} \;,$$

then $\mathrm{tr}_p(\mathbf{M})$ is the $n \times n$ matrix of traces of such blocks[5]:

$$\mathrm{tr}_p(\mathbf{M}) = \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}_{11}\right) & \ldots & \mathrm{tr}\left(\mathbf{M}_{1n}\right) \\ \vdots & \ddots & \vdots \\ \mathrm{tr}\left(\mathbf{M}_{n1}\right) & \ldots & \mathrm{tr}\left(\mathbf{M}_{nn}\right) \end{bmatrix} \;.$$

We alternate between optimizing over $\mathbf{\Psi}_n$ while holding $\mathbf{\Theta}_p$ fixed and optimizing over $\mathbf{\Theta}_p$ while holding $\mathbf{\Psi}_n$ fixed.

[5] In a sense, this generalizes the conventional trace operator as $\mathrm{tr}_{np}(\mathbf{M}) = \mathrm{tr}\left(\mathbf{M}\right)$.

First we consider the case where there is no regularization. From (6), the first step of the optimization problem is reduced to

$$\min_{\mathbf{\Psi}_n}\Big\{p\ \mathrm{tr}\left(\mathbf{\Psi}_n\mathbf{T}\right)\ -\ \ln|\mathbf{\Psi}_n\oplus\mathbf{\Theta}_p|\Big\}\ . \tag{8}$$

Section 2 in the supplementary material shows how to take the gradient of (8) with respect to $\mathbf{\Psi}_n$. Combining (5) and (6) of the supplementary material we obtain the stationary point:

$$\mathbf{T}-\tfrac{1}{2p}\mathbf{T}\circ\mathbf{I}=\tfrac{1}{p}\mathrm{tr}_p(\mathbf{W})-\tfrac{1}{2p}\mathrm{tr}_p(\mathbf{W})\circ\mathbf{I}\ , \tag{9}$$

where we define $\mathbf{W}\triangleq(\mathbf{\Psi}_n\oplus\mathbf{\Theta}_p)^{-1}$. We partition $\mathbf{V}\triangleq\frac{1}{p}\ \mathrm{tr}_p(\mathbf{W})$ as

$$\mathbf{V}=\begin{bmatrix}v_{11} & \mathbf{v}_{1\backslash1}^\top\\ \mathbf{v}_{1\backslash1} & \mathbf{V}_{\backslash1\backslash1}\end{bmatrix}\ , \tag{10}$$

where $\mathbf{v}_{1\backslash1}$ is a vector of size $n-1$ and $\mathbf{V}_{\backslash1\backslash1}$ is a $(n-1)\times(n-1)$ matrix. Despite the complex form of the stationarity condition, only the lower-left block of its partition will be of use:

$$\begin{aligned}&\mathbf{t}_{1\backslash1}=\tfrac{1}{p}\mathrm{tr}_p(\mathbf{W}_{1\backslash1})=\mathbf{v}_{1\backslash1},\ \text{and also from (7)},\\ &\mathbf{t}_{1\backslash1}=(t_{21},\ldots,t_{n1})^\top=\tfrac{1}{p}(\mathbf{y}_2^\top\mathbf{y}_1,\ \ldots,\mathbf{y}_n^\top\mathbf{y}_1)^\top.\end{aligned} \tag{11}$$

Similarly, we partition $\mathbf{W}$ into blocks:

$$\mathbf{W}=\begin{bmatrix}\mathbf{W}_{11} & \mathbf{W}_{1\backslash1}^\top\\ \mathbf{W}_{1\backslash1} & \mathbf{W}_{\backslash1\backslash1}\end{bmatrix}\ ,$$

where $\mathbf{W}_{11}$ is a $p\times p$ matrix and $\mathbf{W}_{1\backslash1}$ is a $p(n-1)\times p$ matrix. Then from the bottom-left block of

$$\begin{aligned}&\mathbf{W}\mathbf{\Omega}=\\ &\begin{bmatrix}\mathbf{W}_{11} & \mathbf{W}_{1\backslash1}^\top\\ \mathbf{W}_{1\backslash1} & \mathbf{W}_{\backslash1\backslash1}\end{bmatrix}\begin{bmatrix}\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p & \ldots & \psi_{in}\mathbf{I}_p\\ \vdots & \ddots & \vdots\\ \psi_{n1}\mathbf{I}_p & \ldots & \psi_{nn}\mathbf{I}_p+\mathbf{\Theta}_p\end{bmatrix}\\ &=\mathbf{I}_n\otimes\mathbf{I}_p\ ,\end{aligned} \tag{12}$$

we get

$$\begin{aligned}&\mathbf{W}_{1\backslash1}(\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)+\mathbf{W}_{\backslash1\backslash1}(\boldsymbol{\psi}_{1\backslash1}\otimes\mathbf{I}_p)=\mathbf{0}_{n-1}\otimes\mathbf{I}_p\\ &\mathbf{W}_{1\backslash1}+\mathbf{W}_{\backslash1\backslash1}\begin{bmatrix}(\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)^{-1}\psi_{21}\\ \vdots\\ (\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)^{-1}\psi_{n1}\end{bmatrix}=\mathbf{0}_{n-1}\otimes\mathbf{I}_p\\ &\mathbf{W}_{1\backslash1}\ +\ \mathbf{W}_{2\backslash1}(\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)^{-1}\psi_{21}\ +\ldots\\ &\quad\cdots+\ \mathbf{W}_{n\backslash1}(\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)^{-1}\psi_{n1}=\mathbf{0}_{n-1}\otimes\mathbf{I}_p\ ,\end{aligned} \tag{13}$$

with $\mathbf{0}_{n-1}$ as the vector of $n-1$ zeros. According to the stationary point in (11), taking the blockwise trace $\mathrm{tr}_p(.)$ of both sides, gives the equation:

$$p\ \mathbf{t}_{1\backslash1}+\mathbf{A}_{\backslash1\backslash1}\boldsymbol{\psi}_{1\backslash1}=\mathbf{0}_{n-1},\quad\text{where} \tag{14}$$

$$\mathbf{A}_{\backslash1\backslash1}^\top\triangleq\begin{bmatrix}\mathrm{tr}_p\left\{\mathbf{W}_{2\backslash1}(\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)^{-1}\right\}^\top\\ \vdots\\ \mathrm{tr}_p\left\{\mathbf{W}_{n\backslash1}(\psi_{11}\mathbf{I}_p+\mathbf{\Theta}_p)^{-1}\right\}^\top\end{bmatrix}.$$

By imposing an ℓ_1 penalty on $\boldsymbol{\psi}_{1\backslash1}$, this problem reduces to a Lasso regression:

$$\min_{\boldsymbol{\psi}_{1\backslash1}}\Big\{\|-p\ \mathbf{t}_{1\backslash1}-\mathbf{A}_{\backslash1\backslash1}\boldsymbol{\psi}_{1\backslash1}\|_2^2\ +\ \lambda\|\boldsymbol{\psi}_{1\backslash1}\|_1\Big\}\ . \tag{15}$$

After estimating $\psi_{1\backslash1}$, we compute $\mathbf{W}_{1\backslash1}$ by substituting into (13). It remains to compute $\mathbf{W}_{11}$. This follows from (12), which gives

$$\mathbf{W}_{11}=(\mathbf{I}-\mathbf{W}_{1\backslash1}^\top(\boldsymbol{\psi}_{1\backslash1}\otimes\mathbf{I}))(\psi_{11}\mathbf{I}+\mathbf{\Theta}_p)^{-1}\ . \tag{16}$$

This algorithm iteratively estimates columns of $\mathbf{\Psi}_n$ and $\mathbf{W}$ in this manner. The procedure for estimating $\mathbf{\Theta}_p$, for fixed $\mathbf{\Psi}_n$, becomes directly parallel to the above simply by *transposing* the design matrix (samples become features and vice-versa) and applying the algorithm. Algorithm 1 outlines the BiGLasso. In our experiments we treat λ and γ as the same parameter and the precision matrices $\mathbf{\Psi}_n$ and $\mathbf{\Theta}_p$ are initialized as identity matrices. The empirical mean matrix is subtracted from each dataset. Although cross-validation can be used to tune the regularization parameters, it typically overselects.

Algorithm 1 BiGLasso

Input: $\mathbf{Y},\lambda,\gamma$ and initial estimates of $\mathbf{\Psi}_n$ and $\mathbf{\Theta}_p$
$\mathbf{T}\leftarrow p^{-1}\mathbf{Y}\mathbf{Y}^\top$
repeat
  # *Estimate* $\mathbf{\Psi}_n$:
  for $i=1\ldots n$ **do**
    Partition $\mathbf{\Psi}_n$ into $\psi_{ii},\boldsymbol{\psi}_{i\backslash i}$ and $\mathbf{\Psi}_{\backslash i\backslash i}$.
    Find a sparse solution of (15).
    Substitute $\boldsymbol{\psi}_{i\backslash i}$ into (13) to compute $\mathbf{W}_{i\backslash i}$.
    Compute $\mathbf{W}_{ii}$ with (16).
  end for
  # *Estimate* $\mathbf{\Theta}_p$:
  Proceed as if estimating $\mathbf{\Psi}_n$ with input $\mathbf{Y}^\top,\lambda,\gamma$.
until (6) converges or maximum iterations reached.

4. Simulations

To empirically assess the efficiency of BiGLasso, we generate the datasets described below from centered Gaussians with Kronecker-product **(KP)** and

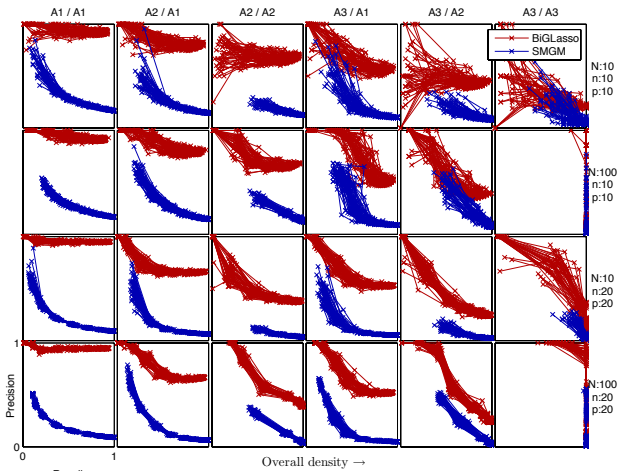


Figure 2. Simulation results on data generated from Kronecker-sum structures. Each box shows a recall-precision plot for a particular setup (shown along the top and right margin). Structure recovery can be exact, as the sample size increases for the A3/A3 combination (most right column).

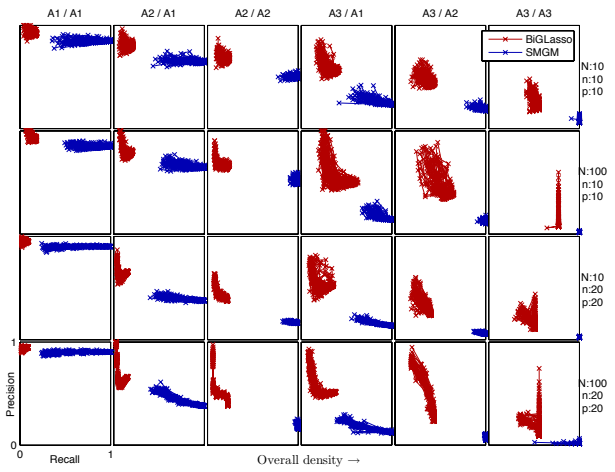


Figure 3. Simulation results on data generated from Kronecker-product structures.

Kronecker-sum **(KS)** precision matrices. We run the BiGLasso and SMGM using the ℓ_1 penalty. The $\mathbf{\Theta}_p$ and $\mathbf{\Psi}_n$ precision matrices in both cases are generated in accordance to (§4, Leng & Tang, 2012); namely, as either of the following $d \times d$ blocks (d being either p or n) of increasing density:

1. $\mathbf{A}_1$: Inverse AR(1) (auto-regressive process) such that $\mathbf{A}_1 = \mathbf{B}^{-1}$ with $B_{ij} = 0.7^{|i-j|}$.
2. $\mathbf{A}_2$: AR(4) with $A_{ij} = I(|i-j| = 0) + 0.4I(|i-j| = 1) + 0.2I(|i-j| = 2) + 0.2I(|i-j| = 3) + 0.1I(|i-j| = 4)$, $I(.)$ being the indicator function.
3. $\mathbf{A}_3 = \mathbf{B} + \delta\mathbf{I}$, where for each $B_{ij} = B_{ji}, i \neq j$, $P(B_{ij} = 0.5) = 0.9$ and $P(B_{ij} = 0) = 0.1$. The diagonal is zero and δ is chosen such that the condition number of $\mathbf{A}_3$ is d. Since the condition number is $k(\mathbf{A}_3) = d = \frac{\lambda_1+\delta}{\lambda_d+\delta}$, the ratio of largest-to-smallest eigenvalue, then $\delta = \frac{d\lambda_d - \lambda_1}{1-d}$.

Figures 2 and 3 show the *recall* $= \frac{\#\{\widehat{\Omega}_{ij}\neq 0 \ \&\ \Omega_{ij}\neq 0\}}{\#\{\Omega_{ij}\neq 0\}}$ (or *true-positive rate*) and *precision* $= \frac{\#\{\widehat{\Omega}_{ij}=0 \ \&\ \Omega_{ij}=0\}}{\#\{\widehat{\Omega}_{ij}=0 \ \&\ \Omega_{ij}=0\}+\#\{\widehat{\Omega}_{ij}=0 \ \&\ \Omega_{ij}=1\}}$ across 50 replications to assess the $\widehat{\mathbf{\Omega}}$ estimates under various setups.

Each box shows a particular setup that varies in block combination $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3)$, in block sizes (n, p), in sample size N generated from the matrix-normal and by the structure used (KS or KP) to generate the sample. Each curve in a box is the solution-path of a replication in precision-recall space for a range of regularization settings $\lambda = 5^x$, for $x \in [-6, -2]$ interpolated 10 times. The blocks are arranged such that the overall density of the structured precision matrices increases from left to right.

We note that since blocks A1,A2 have a fixed form, for such combinations each curve is a different sample from the same graph structure. Only A3 is random so in combinations involving A3, each box has a different random A3 and consequently generates a set of 50 replicates from a different graph. At a glance this has little effect.

Figures 2 and 3 also compare against the results of SMGM (using the Lasso penalty) on data simulated from the matrix-normal with KS structures. Leng & Tang (2012) had also ran comparisons against the MLE method of Dutilleul (1999) (an unpenalized variant of SMGM), ridge-SMGM (SMGM with an ℓ_2 penalty instead of ℓ_1) and the GLasso of Friedman et al. (2008) (on vectorized samples from $\mathcal{N}\left(\mathbf{0}, \mathbf{\Psi}_n^{-1} \otimes \mathbf{\Theta}_p^{-1}\right)$, i.e. ignoring the matrix structure). They consistently outperformed all of these methods, so for brevity we compare only against the SMGM. Similarly, figure 3 visualizes the simulations under KP structures.

By the empirical distributions of these solution-paths (50 for each model in each box), it is no surprise that the intrinsically denser SMGM tends to have low precision (many false-positives) for smaller values of λ. On the contrary, BiGLasso tends to have low recall (many false-negatives) due to its intrinsically sparser

structure.

Block A3 is the only randomized sparse structure whereas A1 and A2 are more "artificial" as they respectively model an inverse-AR(1) and AR(4) and they yield banded precision matrices. Of interest is the observation that the largest effect of the increase in sample size ($10 \to 100$) seems to occur on the A3/A3 combination (right end column of boxes). More precisely in Figure 2, we note the difference from box (1,6) to (2,6) and from (3,6) to (4,6). The sample size is very effective: with sufficiently large sample size N, BiGLasso starts to recover exactly and SMGM occupies lower regions in general.

In Figure 3, since the data generation process uses Kronecker-product structures, the SMGM is expected to outperform our method. Indeed for lower-density structure, the recovery rate of the SMGM seems consistently better than BiGLasso. and recovery can be almost exact for the SMGM for combination A1/A1. However, as the overall density increases, the performance of BiGLasso is balanced. Again, for combinations involving A3, larger sample sizes benefit BiGLasso more.

In summary, KP-simulated data proved harder to tackle for both methods than KS-generated data. These simulations have shown that the BigLasso consistently outperforms the SMGM on KS-simulations, with the possibility of exact recovery on large sample sizes. On KP-simulations the comparison is less clear, but the BiGLasso proves more practical for denser Kronecker-product structures and the SMGM more practical for sparser structures.

5. An Example from the COIL Dataset

In this section we carry out a simple video analysis of a rotating rubber duck from the COIL dataset[6]. The video consists of gray-scaled images, see Figure 4.

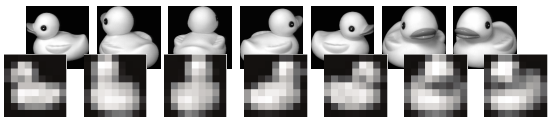


Figure 4. Video of a rotating rubber duck. Original resolution of 128×128 pixels (back row) and reduced resolution of 9×9 pixels (front row).

The goal is on two fronts: to recover the conditional dependency structure over the frames and the structure over the pixels. For simplicity, we reduced the resolution of each frame and sub-sampled the frames (at a ratio 1:2). After vectorizing the frames (stacking their columns into 81×1 vectors) and arranging them into a design matrix $\mathbf{Y}$, the resulting single "datapoint" that BiGLasso has to learn from is 36×81 (#frames $\times$ vectorized frame length). Unlike our previous simulations where we had many matrix-samples, here the challenge is to learn from this single matrix ($N = 1$).

Despite the big loss in resolution, the principal component (PCA) subspace of the rotating duck seems to remain smooth, see Figure 5. Being a time-series, the video is expected to resemble a 1D manifold, "homeomorphic" to the one recovered by PCA shown in figure 5, so we applied the BiGLasso on the reduced images.

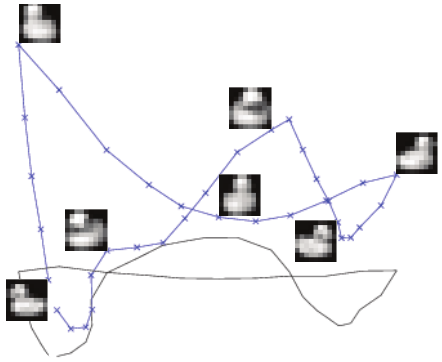


Figure 5. 1D manifold of the rotating duck in 3D space, recovered by PCA and projecting onto the 3 principal eigenvectors of $\mathbf{Y}^\top\mathbf{Y}$. The black curve serves as a shadow to aid perspective. Note that the blue line is drawn only by knowledge of the frame ordering and PCA is responsible solely for the reduced embedding.

Indeed, the left panel of figure 6 shows the row-precision parameter of BiGLasso capturing a *manifold-like* structure where the first and last frames join, as expected of a 360° rotation. The model recovered the temporal manifold structure, or in other words, we could use it to *connect the dots* in Figure 5 in case the true order of the frames was unknown (or randomly given to us).

The right panel of Figure 6 shows the conditional dependency structure over the pixels. This figure shows strong dependencies at intervals of 9 — that is, roughly in line with the size of a frame (due to the column-wise ordering of the pixels). This is expected, as neighboring pixels are more likely to be conditionally dependent.

[6] `http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php`

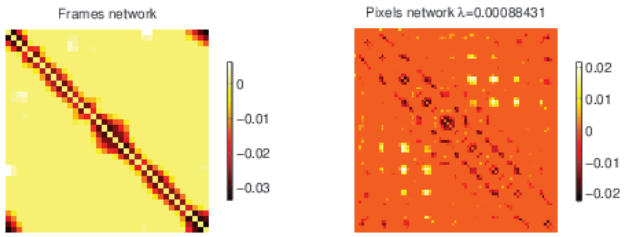


Figure 6. The estimated row and column-precision of BiGLasso with $\lambda = \gamma \approx 10^{-3}$.

A more intuitive picture of the induced Markov network is shown in Figure 7. A Gaussian graphical model can be naturally interpreted as a system of springs, where the off-diagonal entries of the inverse-covariance represent the *negative stiffness* of the springs. Therefore by the colorbar, a negative-color represents an "attracting" spring between those two pixels and a positive-colour represents a "repulsing" spring. Naturally, in the frames network almost all non-zero elements are negative.

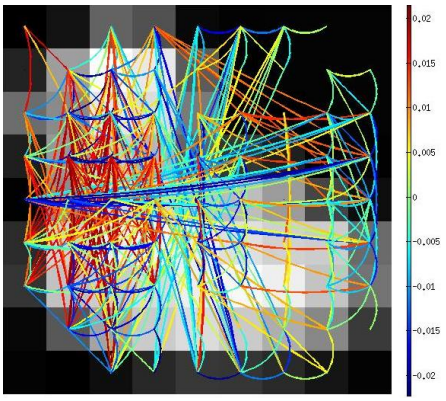


Figure 7. The Markov network induced by the column-precision over the pixels (superimposed over the first frame for reference of the pixel locations).

6. Conclusion

In this paper we proposed new techniques for modeling *conditional* dependencies, as encoded by the inverse-covariance of a matrix normal density. In high-dimensional cases the Markov network structures induced by a graph could be approximated by factorisations such as the tensor-product (Kronecker-product of precision matrices). In this work, we motivated a novel application of the Cartesian factorization of graphs (Kronecker-sum of precision matrices), as a more parsimonious and interpretable structure for inter-sample and inter-feature conditional dependencies. We introduced the bigraphical Lasso, an algorithm for the simultaneous point-estimation of the structures of two Gaussian graphical models: one over the rows of a matrix-sample and the other over its columns. This was demonstrated to good effect through simulations as well as a small example from the COIL dataset.

For future research, the Kronecker sum structure may be of interest in both Gaussian processes and modeling higher order tensors. In multi-output GPs, a Kronecker-product noise-free covariance decouples the outputs when a block design is used. The additive form is an appealing feature of the Kronecker-sum for the *preservation* of inter-task transfer, thereby leading to potential applications on Kronecker-sums of kernels for multi-output Gaussian processes. The associativity of the Kronecker sum may also yield an approach to the modeling of datasets organized into 3 or higher-dimensional arrays (amounting to GMRFs over higher-order tensors) with dependencies across any subset of the array dimensions.

Software and Data

Related source code for reproducing the experiments will appear on `github.com/alkalait/biglasso`.

Acknowledgments

AK would like to thank Amos Storkey, Eleni Vasilaki, Ricardo Silva and Martin Wainwright for useful discussions. The authors would like to thank the reviewers for their useful comments. Research supported in part by NSF Grant IIS-1116730 and ONR grant N000141210762.

References

Banerjee, O., El Ghaoui, L., and d'Aspremont, A. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, June 2008.

Binkley, J. K. and Nelson, C. H. A note on the efficiency of seemingly unrelated regression. *The American Statistician*, 42(2):137–139, 1988.

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. I. Multi-task Gaussian process prediction. In Platt, J.C., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 153–160, Cambridge, MA, 2008. MIT Press, Cambridge, MA.

Chung, F. R. K. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92).*

American Mathematical Society, December 1996. ISBN 0821803158. URL http://www.worldcat.org/isbn/0821803158.

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

Dutilleul, P. The mle algorithm for the matrix normal distribution. *Journal of statistical computation and simulation*, 64(2):105–123, 1999.

Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456): 1348–1360, 2001.

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

Gupta, A. K. and Nagar, D. K. *Matrix variate distributions*. Chapman Hill, 1999.

Imrich, W., Klavzar, S., and Rall, D. F. *Topics in Graph Theory: Graphs and Their Cartesian Product*. AK Peters Ltd, 2008. ISBN 1568814291, 9781568814292.

Lauritzen, S. L. *Graphical models*, volume 17. Oxford University Press, USA, 1996.

Lawrence, N. D. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, November 2005.

Lawrence, N. D. A unifying probabilistic perspective for spectral dimensionality reduction: Insights and new models. *Journal of Machine Learning Research*, 13:1609–1638, 2012.

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

O'Hagan, A. A Markov property for covariance structures. *Statistics Research Report 98-13, Nottingham University*, 1998.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323.

Sabidussi, G. Graph multiplication. *Mathematische Zeitschrift*, 72:446–457, 1959. ISSN 0025-5874. URL http://dx.doi.org/10.1007/BF01162967. 10.1007/BF01162967.

Stegle, O., Lippert, C., Mooij, J., Lawrence, N. D., and Borgwardt, K. Efficient inference in matrix-variate Gaussian models with iid observation noise. *Advances in Neural Information Processing Systems*, 24:443, 2011.

Tipping, Michael E. and Bishop, Christopher M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, B*, 6(3):611–622, 1999. doi: doi:10.1111/1467-9868.00196.

Tsiligkaridis, T., Hero, A., and Zhou, S. On convergence of Kronecker graphical lasso algorithms. *Signal Processing, IEEE Transactions on*, PP(99): 1, 2013. ISSN 1053-587X. doi: 10.1109/TSP.2013.2240157.

Wackernagel, H. *Multivariate geostatistics*. Springer, 2003.

Zellner, A. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American statistical Association*, 57(298):348–368, 1962.

Zhang, Y. and Schneider, J. Learning multiple tasks with a sparse matrix-normal penalty. *Advances in Neural Information Processing Systems*, 23:2550–2558, 2010.