Ludwig-Maximilians-Universität München
MSc Data Science
Modeling Under Dependence

Winter Semester 22/23

Graphical lasso and Mixed Graphical Model

March 1, 2023

Group members:

Urosevic, Dusan
<dusan.urosevic@campus.lmu.de>

Massaro, Patricio
<p.massaro@campus.lmu.de>

Contents

1 Introduction

This report focuses on the problem of estimating sparse undirected graphical models using Lasso regularization, which is particularly useful when the number of features exceeds the number of records in the dataset. Our objective is to replicate and understand the results of two influential publications in this field. In 2007, Friedman [1] developed a substantially faster method for estimating the inverse covariance matrix using Lasso regularization, which is now commonly referred to as the graphical Lasso. In 2015, Lee and Hastie [2] extended the graphical Lasso to accommodate not only continuous variables but also discrete variables. The report provides an overview of the basic definitions and estimation procedures for both methods, as well as practical applications and use cases. Finally, extensions to the methods discussed in this report are presented.

The structure of the report is as follows:

- Section 1 introduces the problem of structure learning and outlines the setting in which we can apply the graphical Lasso method.
- Chapter 2 covers the graphical Lasso method, which was originally developed for continuous variables and is described in [1].
- Chapter 3 extends the graphical Lasso to mixed variable types, as discussed in [2].

1.1 Structure learning

The general setting of structure learning is as follows. We have a dataset $D = (X_1, X_2, \dots X_n)$ which represents n *i.i.d.* samples of a multivariate variable X. We wish to extract the dependence structure that exists between the p individual components of $X = (x_1, x_2, \dots x_p)$. Establishing which (conditional) dependencies are present in the underlying distribution is the main goal of structure learning. Depending on the domain, these dependence structures can be the final goal of our analysis themselves. Examples of this would be, data on the level of activity present in different regions of the brain (figure 1.1) or the presence of different genes in gene networks (figure 1.2). In these cases, individual predictions about the level of brain activity or the presence of a gene are not very informative. If instead we ask the question, which brain regions/genes show a high level of dependence, we can gain insight into which brain regions/genes likely have a shared function. In other cases, the learned structure might just be another input for creating a predictive model. An example of this would be a network model for the amount car traffic present in different regions of a large city (figure 1.3). Here the focus isn't on understanding the dependence between different streets/neighbourhoods but trying to predict traffic jams before they occur in order to make take corrective measures.

1.2 Graphical Lasso - Assumptions

The graphical Lasso method should be used when the following criteria are met. Firstly, the graphical models that we want to estimate are Markov random fields. This means that the resulting graphs are undirected and that they only capture the existence of

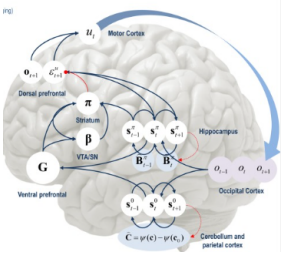


Figure 1.1: Brain activity

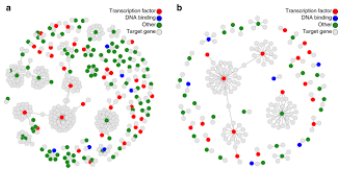


Figure 1.2: Gene networks

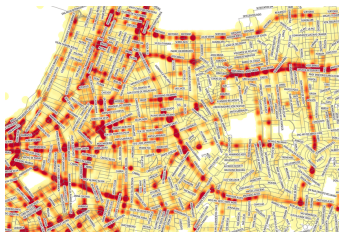


Figure 1.3: Car traffic

(conditional) independence between variables, and not causal relationships (which would be the case for Bayesian networks). Next, we are assuming a parametric model of our multivariate distribution $P(X)$. Specifically we assume a Multivariate Gaussian distribution. This allows the extraction of the entire dependence structure from the precision matrix $\Theta = \Sigma^{-1}$. Note that this assumption implies that the standard graphical Lasso method is used only when all of the p components of our multivariate distribution are continuous. Rule 1.1

$$\Theta_{i,j} = 0 \Leftrightarrow X_i \perp X_j \mid X \setminus \{X_i, X_j\} \tag{1.1}$$

means that if the element in the i-th row and j-th column of the precision matrix is equal to 0, then the variables X_i, X_j are independent conditioned on all other variables. In Markov models this independence is represented through the absence of an undirected edge between X_i and X_j.In figure 1.4, for example, we can see that the variables Raf and Jnk are independent conditioned on all other 9 variables. For variables such as $PIP3$ and Erk that don't have a single neighbour, the Markov property simplifies to regular independence with all other variables. These, and similar observations, show that structure learning using the graphical Lasso can also function as an initial step in estimating high-dimensional distributions by turning it into a problem of estimating several lower dimensional distributions.

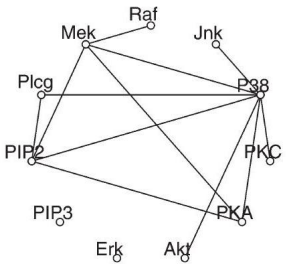


Figure 1.4: MRF example

Lastly, the graphical Lasso employs a type of L_1 regularisation meaning that we assume some level of sparsity in the structure. Ideally, it should be used in cases where estimating the precision matrix using regular Max Likelihood is unstable, for example in the case where $p >> n$. The previously mentioned gene networks are a good example of this type of setting. Here, it is likely that the number of different genes recorded is much larger than the number of participants whose genetic data we have access to.

2 Graphical Lasso

We have already introduced the main criteria that should be met for graphical Lasso to be a viable method of structure learning:

- The joint distribution is a p-dimensional MV Gaussian (implying continuous data)
- We are interested in the dependence structure and not causal relationships
- Standard ML estimates are unstable due to high dimensional setting

In this chapter we present the graphical Lasso method in detail. We start with the derivation of the objective function. Next, we present an approximate method for solving it, the pair-wise regression [3]. The rest of the chapter is dedicated to the Covariance Lasso method introduced by Friedman [1].

2.1 Definition

We start from the assumed MV Gaussian distribution of X.

$$X \overset{d}{\sim} \mathcal{N}(x|\mu,\Sigma) \Leftrightarrow p(X) = \frac{1}{(2\pi)^{(p/2)}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)\right\} \tag{2.1}$$

By using the fact that $\Theta = \Sigma^{-1}$ and that $\frac{1}{det(A)} = det(A^{-1})$ we can rewrite the density in eq.2.1 as

$$p(X) = \frac{1}{(2\pi)^{(p/2)}}|\Theta|^{1/2} \exp\left\{-\frac{1}{2}(x-\mu)^\intercal \Theta (x-\mu)\right\} \tag{2.2}$$

We calculate the **log-likelihood** for this density, while ignoring constant factors, and get

$$\ell(x;\mu,\Sigma) \propto \frac{n}{2}\log\det(\Theta) - \frac{1}{2}\sum_{i=1}^{n}(x^{(i)}-\mu)^\intercal\Theta(x^{(i)}-\mu) \tag{2.3}$$

This, when using matrix notation, can further be simplified to

$$\ell(x;\mu,\Sigma) \propto \frac{n}{2}\log\det(\Theta) - \frac{n}{2}\operatorname{tr}(S\Theta) \tag{2.4}$$

where S is the empirical covariance matrix and $\operatorname{tr}(A) = \sum_{i=1}^{n} a_{ii}$ is the trace of a matrix. As already mentioned, in high-dimensional cases estimating the structure using this log-likelihood objective produces estimates with a large variance. To tackle this, we introduce a **preference for sparse models** using a penalisation term. Inducing sparsity in the structure, for Gaussian models, is equivalent to minimising the number number of non-zero elements in the precision matrix. Because of this we make use of the following penalisation term

$$\|\Theta\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}|\Theta_{i,j}| \tag{2.5}$$

We add (2.5) to the previous log-likelihood (2.4), ignoring once again constant factors, and we get the **objective function of the graphical Lasso**:

$$\log\det\Theta - \mathrm{tr}(S\Theta) - \lambda\|\Theta\|_1 \tag{2.6}$$

where λ is the penalisation parameter that determines the level of sparsity in our model. The rest of this chapter focuses on different ways of estimating the solution for this objective. The first approach, pair-wise regression [3], is an approximate method that ignores the constraints on Θ. The second method, introduced in Friedman et al (2007) [1], solves the constrained problem and is presented in more detail.

2.2 Estimation

Here we present the two main procedures for estimating Θ in the graphical Lasso method.

2.2.1 Pair-wise regression

Our first attempt at solving for the objective function was presented in Meinshausen & Bühlmann (2006) [3]. The idea is to estimate the results of the original joint lasso problem by performing p **independent** lasso regressions. This is done by individually fitting each of the p components using the others as predictors.

$$\begin{aligned} x_1 &= c_{12}x_2 + c_{13}x_3 + \ldots + c_{1p}x_p \\ x_2 &= c_{21}x_1 + c_{23}x_3 + \ldots + c_{2p}x_p \\ &\ldots \\ x_p &= c_{p1}x_1 + c_{p2}x_2 + \ldots + c_{p(p-1)}x_{(p-1)} \end{aligned} \tag{2.7}$$

Using the regression coefficients obtained from the p Lasso regressions (2.7), structure learning is performed according to the following rule

$$c_{ij} = 0 \text{ and } c_{ji} = 0 \implies X_i \perp X_j \mid X \setminus \{X_i, X_j\} \tag{2.8}$$

In other words, if X_i is not present in X_j-s regression and vice-versa than we say that the two variables are conditionally independent. This approximation makes sense in the case of a Gaussian model, since correlation and dependence are equivalent.

The main issue with extracting the structure using this approximate method is that its estimates are only consistent when $n \to \infty$. There are **no guarantees** on convergence speed nor on the quality of the estimates when $p >> n$. However, we still present this simple method, as its main idea of using component-wise lasso regressions was adopted in other algorithms that solve the graphical Lasso objective.

2.2.2 Covariance Lasso

The *Covariance Lasso* [1] is a numeric algorithm that solves the objective $\log\det\Theta - \mathrm{tr}(S\Theta) - \lambda\|\Theta\|_1$ for Θ, indirectly by estimating Σ through iterative use of constrained Lasso regression. We assume that we have n observations of a p-dimensional Gaussian distribution. This means that it is fully specified by the parameter set (Σ, μ) whose dimensions are $p \times p$ and $p \times 1$ respectfully.

In order to go into more detail, we first introduce some additional notation. With $\hat{\Sigma}$ we denote the current estimate of the covariance matrix. Further we perform the following partitioning on $\hat{\Sigma}$ and the empirical estimate S.

$$\hat{\Sigma} = \begin{pmatrix} \hat{\Sigma}_{11} & \hat{\sigma}_{12} \\ \hat{\sigma}_{12}^T & \hat{\sigma}_{22} \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix} \tag{2.9}$$

where the top-left matrices $\hat{\Sigma}_{11}$ and S_{11} are both of dimension $(p-1) \times (p-1)$. This means that we are isolating the last column and row in both matrices. This partition can be used for arbitrary variables by rotating the rows and columns in such a way such that the coefficients associated with the variable of interest are always last. The $(p-1)$-dimensional column vector $\hat{\sigma}_{12}$ can be seen as an analogue to the regression coefficients in the pair-wise regression method. It reflects the level of dependence between the current variable of interest and the rest of the variables. Banerjee et al. (2007) [4] show that a solution for the original objective satisfies

$$\hat{\sigma}_{12} = \text{argmin}_y \left\{ y^T \hat{\Sigma}_{11}^{-1} y : \|y - s_{12}\|_\infty \leqslant \lambda \right\} \tag{2.10}$$

This is a box-constrained quadratic problem, meaning that it can be solved by using standard numerical procedures. Using convex-duality Banerjee further shows that eq. 2.10 is equivalent to solving the dual problem

$$\min_\beta \left\{ \frac{1}{2} \left\| \hat{\Sigma}_{11}^{1/2} \beta - b \right\|^2 + \lambda \|\beta\|_1 \right\} \tag{2.11}$$

where $b = \hat{\Sigma}_{11}^{-1/2} s_{12}$. So if $\hat{\beta}$ solves the dual problem then $\hat{\sigma}_{12} = \hat{\Sigma}_{11} \hat{\beta}$ solves the primal problem.

The main idea of the Covariance Lasso is to cycle through all of the variables and in each iteration update the respective $\hat{\sigma}_{12}$. For this we use a block coordinate descent approach. This means that in each iteration, all other components of $\hat{\Sigma}$ are held constant, as we perform a greedy optimisation on the coefficients associated with the current variable. This is done by solving the dual problem (2.11) for β. Note that this dual problem is just a regular lasso regression. We repeat this until a convergence criteria is met (usually the element-wise change in $\hat{\Sigma}$ being below a certain threshold). Once the algorithm has converged we can derive the precision matrix Θ by making use of

$$\theta_{12} = -\theta_{22} \beta, \text{ where } \Theta = \begin{pmatrix} \Theta_{11} & \theta_{12} \\ \theta_{12}^T & \theta_{22} \end{pmatrix} \tag{2.12}$$

It should be noted that the objective function is differentiable w.r.t the diagonal elements of Σ. This is a consequence of the fact that Σ has to be a positive definite matrix which means that it will always have positive diagonal elements. Using the partial derivative w.r.t the diagonal elements we arrive at an analytical solution $\hat{\sigma}_{ii} = s_{ii} + \lambda$ for $i = 1, 2, \ldots p$.

2.2.3 Algorithm

The full *Covariance Lasso* algorithm is as follows:

1. Start with $\hat{\Sigma} = S + \lambda I$. The diagonal of $\hat{\Sigma}$ remains unchanged in what follows.
2. For each $j = 1, 2, \dots p$, solve the lasso problem $\min_\beta \left\{ \frac{1}{2} \left\| \hat{\Sigma}_{11}^{1/2} \beta - b \right\|^2 + \lambda \|\beta\|_1 \right\}$, which takes as input the inner products $\hat{\Sigma}_{11}$ and s_{12}. This gives a $(p-1)$ vector solution $\hat{\beta}$. Update the corresponding row and column of $\hat{\Sigma}$ using $\hat{\sigma}_{12} = \hat{\Sigma}_{11}\hat{\beta}$
   - If $avg(\Delta\hat{\sigma}_{ij}) < t$ stop algorithm
3. Repeat step **2.**

The main difference of this algorithm, when compared to the pair-wise regression approach, is that the individual lasso regressions are not performed independent of each other. Information is passed on between iterations through the updated $\hat{\Sigma}$ which allows for joint-optimisation while respecting the constraints needed for $\hat{\Sigma}$ to represent a valid covariance matrix.

2.2.4 Update step

In each iteration of the Covariance lasso we permute the rows and columns of $\hat{\Sigma}$ and S so that the current variables coefficients are last. In step 2. we solve a single lasso problem. Let $V = \hat{\Sigma}_{11}$ be the upper left block of the current estimate, and $u = s_{12}$. Then, we have that for $j = 1, 2, \dots p$:

$$\hat{\beta}_j \leftarrow \frac{S\left(u_j - \sum_{k \neq j} V_{kj}\hat{\beta}_k, \lambda\right)}{V_{jj}} \tag{2.13}$$

where $S(x, \lambda)$ is the **soft-threshold** operator, defined as:

$$S(x, \lambda) = \text{sign}(x)(|x| - \lambda)_+ \tag{2.14}$$

The soft-threshold function does the following. For coefficients whose absolute value is greater than λ it shrinks them towards 0 by a factor of λ. Smaller coefficients will be shrunken directly to 0. This update step is similar to that of the regular Lasso regression. In fact, Banarjee [4] shows that if $\Sigma_{11} = S_{11}$ then the structural model obtained by the Covariance Lasso algorithm will be identical to that of the pair-wise regression approach. If instead of $V = \hat{\Sigma}_{11}$ we use $V = S_{11}$ we will arrive at pair-wise regression.

2.2.5 Precision Matrix Recovery

Lastly, we show how the precision matrix, and thus dependence structure, can be extracted once the Covariance lasso has converged. After estimating $\hat{\Sigma}$ we can easily derive $\hat{\Theta}$. Using $\Sigma\Theta = \mathbb{I}$ we get

$$\begin{aligned} \hat{\Sigma}_{11}\theta_{12} + \hat{\sigma}_{12}\theta_{22} &= 0 \\ \hat{\sigma}_{12}^T\theta_{12} + \hat{\sigma}_{22}\theta_{22} &= 1 \end{aligned} \tag{2.15}$$

Combining this with the fact that $\hat{\beta} = \hat{\Sigma}_{11}^{-1}\hat{\sigma}_{12}$ we get

$$\begin{aligned}\hat{\theta}_{22} &= 1/\left(\hat{\sigma}_{22} - \hat{\sigma}_{12}^T \hat{\Sigma}_{11}^{-1} \hat{\sigma}_{12}\right) = 1/\left(\hat{\sigma}_{22} - \hat{\sigma}_{12}^T \hat{\beta}\right) \\ \hat{\theta}_{12} &= -\hat{\Sigma}_{11}^{-1} \hat{\sigma}_{12} \hat{\theta_{22}} = -\hat{\beta}\hat{\theta}_{22}\end{aligned} \tag{2.16}$$

This means that there are no extra steps/variables needed in order to be able to derive $\hat{\Theta}$ from $\hat{\Sigma}$. We can get a slight improvement in performance if we memorise the $\hat{\beta}$ for each variable.

2.3 Application

We use the graphical Lasso algorithm to extract the dependence structure in cell signaling data. The dataset contains information on the amount of $p = 11$ different proteins detected in individual cells. We have $n = 7466$ samples in total from 9 files associated with different experimental scenarios. The different proteins are used as a form of communication between cells for what processes need to be performed (fig. 2.1). Here, determining the dependence structure could help in finding candidate proteins that share a biological function. Sachs et al. (2003) [5] provides, along with the dataset itself, a Bayesian network derived using several methods for causal inference. We moralise this directed graph in order to obtain a Markov network. We then compare this newtwork with results gained by applying the graphical Lasso and other methods to the data. Here, for the sake of comparison, we use the PC algorithm as a representative of non-parametric structure learning algorithms.

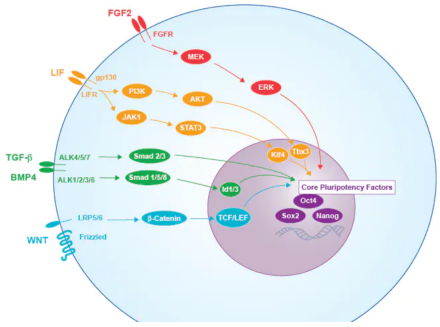


Figure 2.1: Cell signaling using proteins

2.3.1 Implementation

To analyse the cell signaling data we use the `glasso` package in R provided by Friedman [6]. This is a direct implementation of the Covariance lasso algorithm presented in the previous section. Before combining the contents of the 9 files, and analysing the results, some variables had to be renamed/reordered for consistency. For operations on graphs and their visualisation we use the `igraph` package in R [7]. The implementations for this chapter are available at the **following Github repository**.

2.3.2 Results

As an initial sanity check we compare results obtained by varying the penalisation parameter λ. We vary between $\lambda = 0$, meaning no penalisation up to $\lambda = 90000$. Beyond

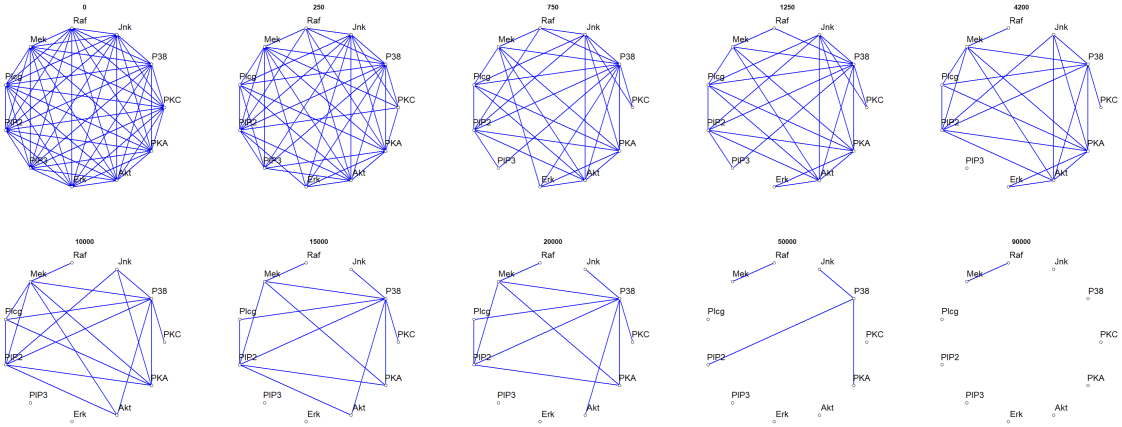


Figure 2.2: Covariance Lasso with varying lambda

this point the penalisation term is so large that no edges are left in the Markov graph. In figure 2.2 we can see that the Covariance Lasso behaves as expected, eliminating individual edges as λ is increased. It should be noted, that in the general setting, this behaviour doesn't have to occur. Unlike regular lasso regression, Covariance lasso has the added constraint of Σ having to be a valid covariance matrix. This in turn can cause individual edges to reappear for larger λ. The overall sparsity of the graph, however, will always increase with larger λ.

Next, in order to demonstrate that there is no single "sweet-spot" value for the penalisation parameter we visualise the difference between the Covariance lasso derived graphs (2.2) and the Ground truth model. As we can see in figure 2.3 increasing λ quickly leads to losing many of the dependencies existing in the ground truth model. Figure 2.4 shows us that if we try to circumvent this by using a very small λ, we end up creating a large number of false edges. In this case the structural model is too dense to be of any practical use. This means that when using the Covariance lasso on arbitrary data, the λ has to be chosen in a way that achieves a balance between the desired sparsity and recovery of the true dependencies.

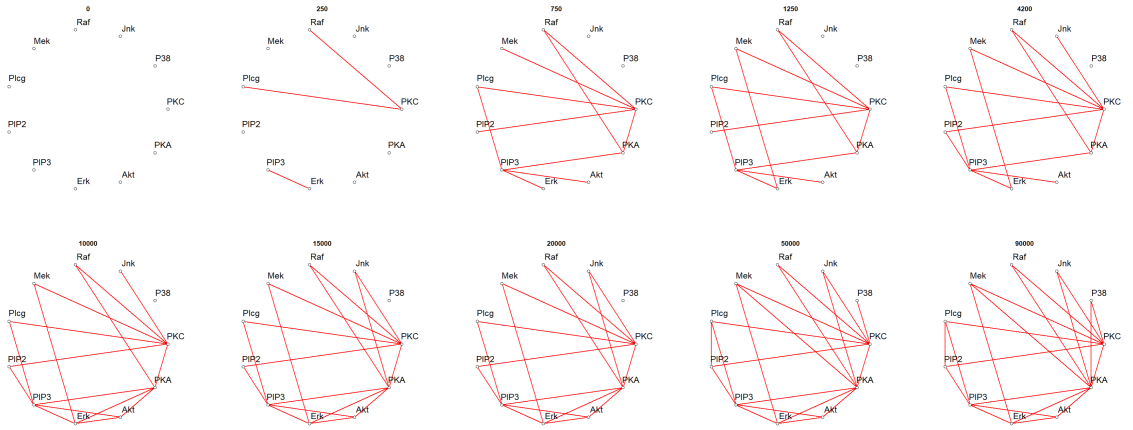


Figure 2.3: True edges that are lost in our results

Lastly, we compare the Covariance Lasso results with those of the PC algorithm.

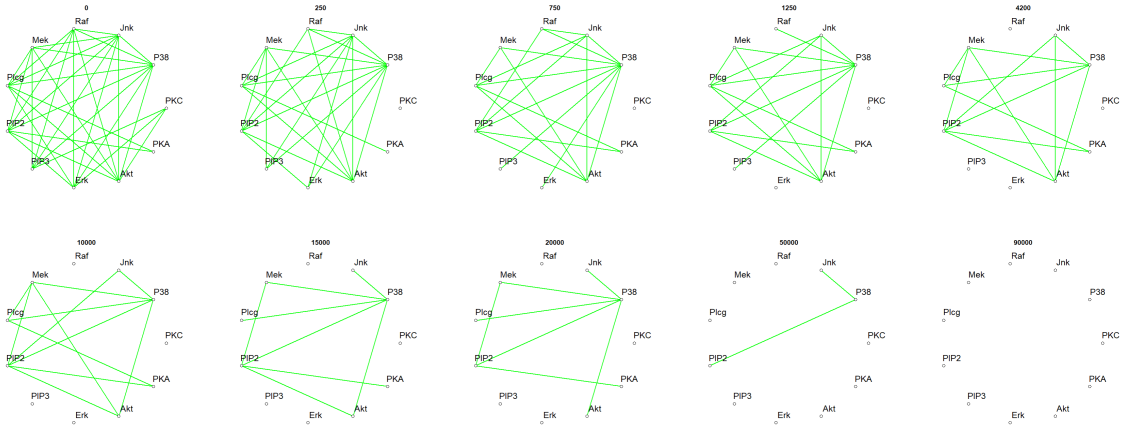


Figure 2.4: False edges that appear in our results

The PC algorithm, in short, is an non-parametric algorithm for structure learning. It is based on performing statistical tests (with a predefined significance level α) for determining conditional independence between components of a multivariate distribution. The complete PC algorithm actually produces a Bayesian network. However, the first step of the algorithm is determining the networks Markov equivalence class which is a valid Markov random field. We extract these models, also called skeleton graphs, for different values of α using the `pcalg` package in R [8].

We compare the ground truth model (fig. 2.5) with the Covariance lasso, where $\lambda = 4200$ (fig. 2.6) , and the PC skeleton (fig. 2.7), where $\alpha = 0.0005$. These graphs were chosen as they have the same number of overall edges as the ground truth model. On first glance, it seems that the PC algorithm performs slightly better. In order to make the results more clear we also visualise the intersects of the two models with the ground truth graph (2.9 and 2.10).

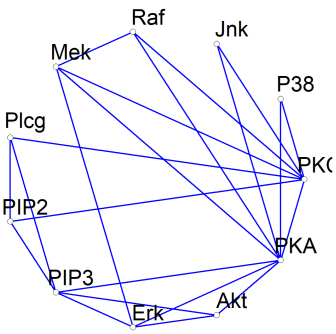


Figure 2.5: Ground truth

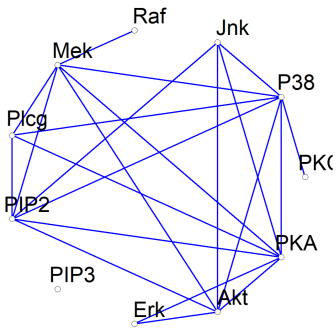


Figure 2.6: GLasso $\lambda = 4200$

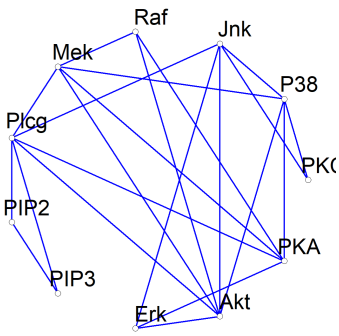


Figure 2.7: PC - $\alpha = 0.0005$

Here, we can see that in the case of the cell signaling data, both the PC and the Covariance lasso pick up on roughly half of the local dependencies of the true model with neither showing a clear advantage. For instance, the PC algorithm detects edges (Raf, PKA) and (JNK, PKC) that the Covariance lasso loses. However, the Covariance lasso detects other edges such as (Jnk, PKA) and (Akt, PKA) that are missing in the PC results. The shared characteristic of both algorithms is that they both have a crucial

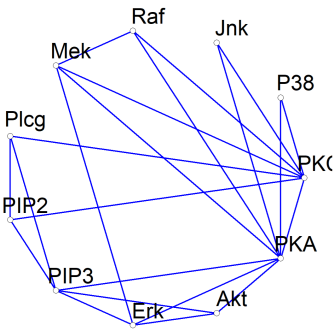


Figure 2.8: Ground truth

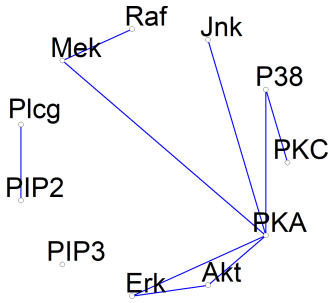


Figure 2.9: intersect with GLasso result

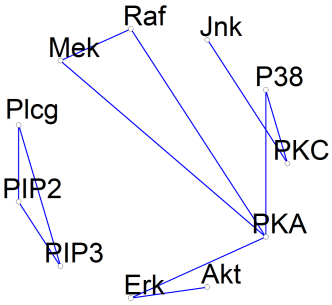


Figure 2.10: interesect with PC result

hyperparameter for determining the overall sparsity of the resulting model.

Finally, we note that our models, for varying λ, are identical to those presented by Friedman. However, in certain versions of the paper the graphs extracted from the cell signaling wildly differ from ours. As the R package that implements Covariance lasso [6] allows for additional parameters such as initial values of $\hat{\Sigma}$ and $\hat{\Theta}$ we assume that the authors later changed the starting parameters without specifying the concrete values that they were using.

2.3.3 Synthetic data example

We test the `glasso` function provided by Friedman [6] in different scenarios. We run the function on random samples of MV Gaussians that differ in the sample size n, the dimensionality of the data d, and for cases where the true model is either dense or sparse. In the case of the dense model, $\Theta_{ij} = 1$ for $i \neq j$ and 2 for $i = j$, meaning that the corresponding graph is fully connected. For the sparse scenario, the variables form a ring like structure. Specifically, $\Theta_{ii} = 1, \Theta_{i,(i-1)} = \Theta_{(i-1),i} = 0.5$ and the rest are set to 0. For the dense model, including high dimensional scenarios, we were able to recover the true sparsity structure. For the sparse model, around half of the true edges could be recovered while retaining the true level of sparsity.

2.4 Extensions

We started this chapter with the simple pair-wise regression [3] model which gives us an approximate solution. Then, we went into detail on the Covariance Lasso introduced by Friedman [1]. This algorithm, when first introduced, achieved significant improvements in run time (up to a factor of 2000) compared to the previous best exact method for solving the graphical Lasso objective. We now give a brief overview of some improvements to the algorithm that have been introduced since then.

2.4.1 Regularisation parameter (λ) selection

The standard graphical Lasso algorithm leaves the choice of the regularisation parameter completely up to the user, which is not ideal as it has a significant influence on both runtime and the final model. Although the original algorithm had Bayesian implementation, which allowed for a prior to be defined over λ, these solutions were not scalable

to high-dimensional settings.

MCPeSe (Monte Carlo penalty selection) for graphical Lasso [9], introduced in 2020, solves this issue by implementing a Monte Carlo based approach for selecting the value of λ. In short, the algorithm is based on running the standard frequentest algorithm for a few different values of the regularisation parameter. The complete solution paths from those runs are then used to simulate the posterior distribution of λ. This can be done using either rejection sampling or the Metropolis-Hastings algorithm. The main contribution here is the option of automatic penalty selection that is both computationally efficient and scalable to high dimensional settings thanks to its internal use of the frequentest algorithm.

Figure 2.11, from the original MCPeSe paper, shows the runtime comparison between different penalty selection criteria based on the dimensionality of the data.

- MCPeSe - black line
- Rotation Information Criterion (RIC) - purple line
  Similar performance and benefits to MCPeSe can be used as an alternative.
- Stability approach to regularisation selection (StARS) - red line
  Based on creating several subsamples of the data for each value of λ and then selecting the value for which the extracted graphs are the most stable across different subsamples. Significant increase in runtime after $P = 100$.
- Bayes Glasso - blue line.
  Fully Bayesian solution to the GLasso. Worst in terms of sacalability.

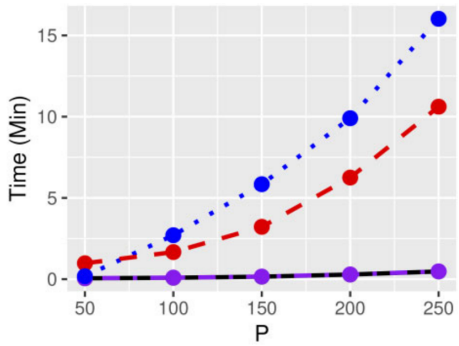


Figure 2.11: Penalty selection criteria - runtime comparison

2.4.2 Thresholding and Analytical solutions

Although the graphical Lasso has been a very popular algorithm for structure learning, particularly in fields like genealogy, it's mathematical properties as an optimisation problem were not well known until relatively recently. Fattahi and Sojoudi [10] provide proofs of properties of the GLasso. Some of their most important contributions are an analytical solution for acyclic graphs and an additional approximate solution for the general case. These solutions, which are meant to be used for relatively sparse graphs,

are based on the assumption of equivalence between the GLasso and thresholding of the sample covariance.

Figure 2.12 shows the time comparison between their closed-form solution and the two previous most efficient algorithms. Here we see that this solution makes it possible to work with $P > 20.000$ for which the previous algorithms would fail to finish in less than 2 hours. This allows the method to be applied to extremely large networks that can be seen in domains such as transportation, large energy grids and intelligent systems with many individual sensors.

d	m	Closed-Form	QUIC-C	QUIC-W	GLASSO-C	GLASSO-W	Elem.
2000	9894	0.1	2.0	1.4	42.8	13.5	0.2
2000	20022	0.1	3.0	2.1	43.8	15.3	0.2
4000	20094	0.5	13.9	7.5	460.8	135.1	2.1
4000	40382	0.5	21.5	12.0	467.6	156.2	2.9
8000	40218	2.5	78.7	49.3	3675.1	1011.2	11.3
8000	79890	2.5	111.7	88.4	3784.3	1278.8	22.2
12000	60192	7.8	243.8	153.1	$\star$	3233.0	31.8
12000	119676	7.4	333.6	251.0	$\star$	3437.2	70.2
16000	80064	17.1	570.0	322.8	$\star$	6545.0	67.2
16000	160094	18.5	787.4	616.4	$\star$	9960.8	174.8
20000	99954	39.4	1266.5	539.4	$\star$	$\star$	107.8
20000	200018	37.4	1683.8	1392.5	$\star$	$\star$	211.5
40000	200290	495.4	$\star$	$\star$	$\star$	$\star$	$\star$
80000	401798	1450.4	$\star$	$\star$	$\star$	$\star$	$\star$

Figure 2.12: Runtime comparison of Closed-form solution and previous methods

Without going into detail, Sojoudi [10], introduces the concept of sign and inverse-consistent matrices and further defines a set of conditions based on these concepts. When these conditions are met, such as in the case of acyclic graphs, an exact solution for the GLasso problem can be given in eq. 2.17.

$$S_{ij}^{opt} = \begin{cases} \frac{1}{\Sigma_{ii}} \left(1 + \sum_{(i,m) \in \mathcal{E}^{opt}} \frac{\left(\Sigma_{im}^{\text{res}}\right)^2}{\Sigma_{ii}\Sigma_{mm} - \left(\Sigma_{im}^{\text{res}}\right)^2} \right) & \text{if} \quad i = j, \\ \dfrac{-\Sigma_{ij}^{\text{es}}}{\Sigma_{ii}\Sigma_{jj} - \left(\Sigma_{ij}^{\text{res}}\right)^2} & \text{if} \quad (i,j) \in \mathcal{E}^{opt}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.17}$$

Note that here Σ is used to denote the **sample** covariance matrix, while the Σ^{res} denotes the result of performing soft thresholding element-wise on the sample covariance. S^{opt} is our final estimate for the precision matrix Θ. $\mathcal{E}^{opt}$ denotes the edge set that is achieved by using simple thresholding on the sample covariance.

3 Mixed Graphical Model

The graphical Lasso algorithm presented in 2 considers the problem of learning the structure of a pairwise graphical model over continuous variables. For discrete models, a pairwise Markov random field can be estimated. In real-world applications, however, typical data sources contain both continuous and discrete variables. The model proposed in this section is able to learn the structure of models with a mix of continuous and discrete attributes, using an approach that generalizes the graphical Lasso.

3.1 Definition

Consider a dataset of n records, p continuous attributes and q discrete attributes, to which one-hot encoding is applied. The discrete variables don't have a limit on the amount of levels that they may have. x_s will denote the sth of p continuous variables, while y_j will denote the jth of q discrete variables. The discrete y_r takes on L_r states. The model proposed by the authors consists of a pairwise Markov random field with density:

$$p(x, y; \Omega) \propto \exp\left(\sum_{s=1}^{p}\sum_{t=1}^{p} -\frac{1}{2}\beta_{st}x_s x_t + \sum_{s=1}^{p}\alpha_s x_s \; + \sum_{s=1}^{p}\sum_{j=1}^{q}\rho_{sj}\left(y_j\right)x_s + \sum_{j=1}^{q}\sum_{r=1}^{q}\phi_{rj}\left(y_r, y_j\right)\right) \tag{3.1}$$

The model is parametrized by $\Omega = [\{\beta_{st}\}, \{\alpha_s\}, \{\rho_{sj}\}, \{\phi_{rj}\}]$. β_{st} represent the continuous-continuous edge potential, α_s represent the continuous node potential, $\rho_s j(y_j)$ represent the continuous-discrete edge potential and $\phi_{rj}(y_r, y_j)$ represent the discrete-discrete edge potential.

The most important features of the model are:

- If the data has only continuous variables, the model simplifies to a multivariate Gaussian. On the other side, if the data has only discrete variables, the model is the usual discrete pairwise Markov random field.
- The conditional distributions of each variable given the others are Gaussian linear regression and multiclass logistic regression if the variables are continuous or discrete, respectively.

The conditional distributions will be used during the estimation procedure explained in 3.2. In the case of the continuous variables, the likelihood and the negative log-likelihood for the model can be written as :

$$p\left(x_s \mid x_{\setminus s}, y; \Omega\right) = \frac{\sqrt{\beta_{ss}}}{\sqrt{2\pi}}\exp\left(\frac{-\beta_{ss}}{2}\left(\frac{\alpha_s + \sum_j \rho_{sj}\left(y_j\right) - \sum_{t\neq s}\beta_{st}x_t}{\beta_{ss}} - x_s\right)^2\right) \tag{3.2}$$

$$-log\; p\left(x_s \mid x_{\setminus s}, y; \Omega\right) = -\frac{log\;\beta_{ss}}{2} + \frac{\beta_{ss}}{2}\left(\frac{\alpha_s}{\beta_{ss}} + \sum_j \frac{\rho_{sj}(y_j}{\beta_{ss}} - \sum_t \frac{\beta_{st}}{\beta_{ss}}x_t - x_s\right)^2 \tag{3.3}$$

In the case of discrete variables, the likelihood and negative log-likelihood result in:

$$p\left(y_r \mid y_{\setminus r,}, x; \Omega\right) = \frac{\exp\left(\sum_s \rho_{sr}\left(y_r\right) x_s + \phi_{rr}\left(y_r, y_r\right) + \sum_{j \neq r} \phi_{rj}\left(y_r, y_j\right)\right)}{\sum_{l=1}^{L_r} \exp\left(\sum_s \rho_{sr}(l) x_s + \phi_{rr}(l,l) + \sum_{j \neq r} \phi_{rj}\left(l, y_j\right)\right)} \tag{3.4}$$

$$-log\ p\left(y_r \mid y_{\setminus r,}, x; \Omega\right) = -log\ \frac{\exp\left(\sum_s \rho_{sr}\left(y_r\right) x_s + \phi_{rr}\left(y_r, y_r\right) + \sum_{j \neq r} \phi_{rj}\left(y_r, y_j\right)\right)}{\sum_{l=1}^{L_r} \exp\left(\sum_s \rho_{sr}(l) x_s + \phi_{rr}(l,l) + \sum_{j \neq r} \phi_{rj}\left(l, y_j\right)\right)} \tag{3.5}$$

3.1.1 Conditional Independence

The conditional distributions of any graphical model are of critical importance. The absence of an edge in a graph corresponds to the nodes (variables) being conditionally independent. conditional independence can be read off from the conditional distribution of a variable given all the others. Moreover, it is captured via the regression coefficient.

The parameters of the pairwise mixed graphical model need special considerations because of their dimensions. In this case, 3 types of edges exist:

- β_{st} is a scalar, corresponding to the edge between two continuous variables. If this coefficient is 0, it implies that x_s and x_t are conditionally independent given all the other variables.
- ρ_{sj} is a vector with length equal to the number of levels of the discrete variable involved. If $\rho_{sj}(y_j) = 0$ for all values of y_j, then y_j and x_s are conditionally independent given all the other variables. The absence of an edge requires that the entire vector is 0.
- ϕ_{rj} is a matrix with size equal to the number of levels in the first discrete variable times the number of levels in the second discrete variable involved. If $\phi_{rj}(y_r, y_j) = 0$ for all values of y_r and y_j, the variables are conditionally independent given all other variables. The absence of an edge requires all the components of the matrix to be 0.

A graphical description of the process is depicted in fig 3.1. It is important to note that the matrix is symmetric, because each parameter blocks appears in two of the conditional probability regressions.

3.2 Estimation

3.2.1 Pseudolikelihood

The most intuitive approach to find a good estimate of Ω is to minimize the negative log-likelihood of the samples, as shown in eq. 3.6. The function is convex, making gradient-descent algorithms suitable for the task.

$$\log p(x, y; \Omega) = \sum_{s=1}^{p} \sum_{t=1}^{p} -\frac{1}{2} \beta_{st} x_s x_t + \sum_{s=1}^{p} \alpha_s x_s + \sum_{s=1}^{p} \sum_{j=1}^{q} \rho_{sj}\left(y_j\right) x_s + \sum_{j=1}^{q} \sum_{r=1}^{j} \phi_{rj}\left(y_r, y_j\right) - \log Z(\Omega) \tag{3.6}$$

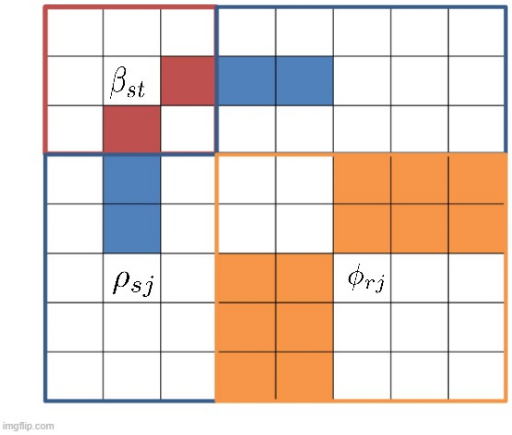


Figure 3.1: Representation of the parameters Ω using a symmetric matrix. This example has 3 continuous and two discrete variables with 2 and 3 levels respectively.

However, the last term involves a high-dimensional integrals. Maximum likelihood estimation in continuous models is known as a computationally intractable problem, and the mixed model is at least as difficult this case, even without considering the discrete part.

An efficient and consistent estimator is the pseudolikelihood [11], that consists on the products of all the conditional distributions. If the negative log is applied, the pseudo-likelihood is the sum of the conditional distributions.

$$\tilde{\ell}(\Omega \mid x, y) = -\sum_{s=1}^{p} \log p\left(x_s \mid x_{|s}, y; \Omega\right) - \sum_{r=1}^{q} \log p\left(y_r \mid x, y_r; \Omega\right) \tag{3.7}$$

Where the first term represent the negative log-likelihood of the continuous variables, represented in 3.3, and the second term depicts the negative log-likelihood of the discrete variables, shown in 3.5.

3.2.2 Penalisation

As in the graphical Lasso, The mixed graphical model applies a penalisation on the number of edges to force sparsity in the parameters. The pairwise nature of the model motivates the following regularization:

$$\lambda \left(\sum_{s<t} \mathbb{I}\left[\beta_{st} \neq 0\right] + \sum_{sj} \mathbb{I}\left[\rho_{sj} \not\equiv 0\right] + \sum_{r<j} \mathbb{I}\left[\phi_{rj} \not\equiv 0\right] \right) \tag{3.8}$$

All parameters that correspond to the same edge are grouped in the same indicator function. That means, if any component of the vector ρ_{sj}, that represents the relation between x_s and y_j, is not 0, the indicator function will activate.

As the penalisation specified in 3.8 is non-convex, the l_0 sparsity is changed by appropiate convex relaxations. For Scalars (β), the l_1 norm can be used. For vectors (ρ), the l_2 norm is used and in the case of matrices (ϕ), the choice is the Frobenius norm.

$$\lambda \left(\sum_{s=1}^{p} \sum_{t=1}^{s-1} |\beta_{st}| + \sum_{s=1}^{p} \sum_{j=1}^{q} \|\rho_{sj}\|_2 + \sum_{j=1}^{q} \sum_{r=1}^{j-1} \|\phi_{rj}\|_F \right) \tag{3.9}$$

Combining the pseudolikelihood and the penalisation, the objective function to be minimized can be written in the following way:

$$\min_{\Omega} \ell_\lambda(\Omega) = \ell(\Omega) + \lambda \left(\sum_{s=1}^{p} \sum_{t=1}^{s-1} |\beta_{st}| + \sum_{s=1}^{p} \sum_{j=1}^{q} \|\rho_{sj}\|_2 + \sum_{j=1}^{q} \sum_{r=1}^{j-1} \|\phi_{rj}\|_F \right) \tag{3.10}$$

3.2.3 Calibrated regularizers

The objective function described in 3.10 considers each of the group penalties as equals, without taking the size of each parameter into account. To calibrate the regularizer, weights are introduced for each group:

$$\min_{\Omega} \ell_\lambda(\Omega) = \ell(\Omega) + \lambda \left(\sum_{s=1}^{p} \sum_{t=1}^{s-1} w_{st} |\beta_{st}| + \sum_{s=1}^{p} \sum_{j=1}^{q} w_{sj} \|\rho_{sj}\|_2 + \sum_{j=1}^{q} \sum_{r=1}^{j-1} w_{rj} \|\phi_{rj}\|_F \right) \tag{3.11}$$

The weights should be chosen so that each parameter is treated equally under the fully-factorized independence model (p_F). Based on KKT conditions, any of the parameter groups Ω_g is non-zero if

$$\left\| \frac{\partial \ell}{\partial \Omega_g} \right\| > constant \cdot w_g \tag{3.12}$$

The left term of the equation has different dimensions for each parameter group, it could be a scalar, a vector or a matrix. Logically, in bigger dimensions there is a better random chance that at least one of the elements being non-zero. Because of that, the weights should be chosen such that:

$$\begin{aligned} \mathbb{E}_{pf} \left\| \frac{\partial \ell}{\partial \Omega_g} \right\| &= constant \cdot w_g \\ w_g &\propto \mathbb{E}_{pf} \left\| \frac{\partial \ell}{\partial \Omega_g} \right\| \end{aligned} \tag{3.13}$$

As the expectation is simpler to compute in closed form when it is squared, we apply the squared operator to both sides.

$$w_g \propto \sqrt{\mathbb{E}_{pf} \left\| \frac{\partial \ell}{\partial \Omega_g} \right\|^2} \tag{3.14}$$

Using 3.14, the weights take the following form

$$\begin{aligned} w_{st} &= \sigma_s \sigma_t, \quad w_{sj} = \sigma_s \sqrt{\sum_a p_a (1 - p_a)}, \quad p_a = \Pr(y_r = a) \\ w_{rj} &= \sqrt{\sum_a p_a (1 - p_a) \sum_b q_b (1 - q_b)}, \quad q_b = \Pr(y_j = b) \end{aligned} \tag{3.15}$$

where σ_s is the standard deviation of the continuous variable x_s, $p_a = P(y_r = a)$ and $q_b = P(y_j = b)$.

3.2.4 Optimisation procedure

Finding the minimum of the objective function described in 3.10 represents a convex optimization problem that can be decomposed into the sum of two functions. One of them, the negative log-pseudolikelihood, is convex and smooth. The penalisation term is also convex but possibly non-smooth.

Problems of this type are well-suited for the proximal gradient algorithm, if it is possible to compute the proximal operator of the non-smooth function. The proximal operator for a convex function g can be defined as :

$$\text{prox}_{gt}(x) = \underset{u}{\text{argmin}} \frac{1}{2t} \|x - u\|^2 + g(u) \tag{3.16}$$

If the negative log-pseudolikelihood is defined as f and the penalisation is defined as g, minimising it's sum locally can be written as :

$$\begin{aligned} &\underset{u}{\text{argmin}} f(x_k) + \nabla f(x_k)^T (u - x_k) + \frac{1}{2t} \|u - x_k\|^2 + g(u) \\ &\quad = \underset{u}{\text{argmin}} \frac{1}{2t} \|u - (x_k - t\nabla f(x_k))\|^2 + g(u) \\ &\quad = \text{prox}_{gt}(x_k - t\nabla f(x_k)) \end{aligned} \tag{3.17}$$

The proximal operator takes the form of soft-thresholding for the continuous-continuous edge potentials β_{st} and group soft-thresholding for the continuous-discrete edge potential ρ_{sj} and discrete-discrete edge potential ϕ_{rj}. The equations for both operators is displayed in eq. 3.18 respectively, an example of soft-thresholding is displayed in fig. 3.18

$$\begin{aligned} S(x,t) &= \left(1 - \frac{t}{|x|}\right)_+ x \\ S(\boldsymbol{w},t) &= \left(1 - \frac{t}{||\boldsymbol{w}||}\right)_+ \boldsymbol{w} \end{aligned} \tag{3.18}$$

The proximal gradient algorithm is an iterative process, where a tolerance τ is defined. If the difference between the objective functions of two consecutive iterations is less than the tolerance, the algorithm is consideredas converged and the results are reported. Additionally, a maximum number of iterations I is defined. It's full pseudo-code is given in alg. 1.

3.3 Application

The mixed graphical model is applied to a census survey dataset, consisting of 8 variables. Two of them are continuous: age and log-wage. The other six are discrete: marital status (5 levels), race (4 levels), education (5 levels), job class (2 levels), health (2 levels) and health insurance (2 levels). One-hot encoding is applied to the discrete variable. The objective of the use case is to understand conditional independence between these variables for different levels of penalisation.

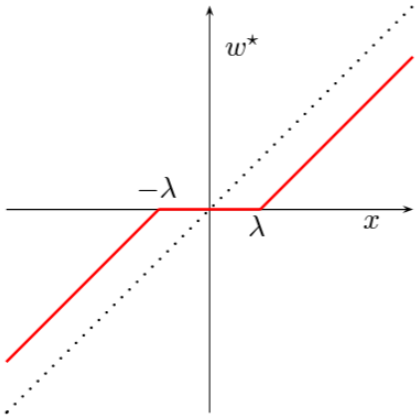


Figure 3.2: Soft threshold operator

Algorithm 1 Proximal gradient

Initialize Ω^0, λ, w_g
Define max iterations I, tolerance τ and step size t
Calculate objective obj_0 at iteration $i = 0$
while $i < I$ or $\delta_{obj} < \tau$ **do**
    $i+ = 1$
    Calculate gradient step $\Omega^i = \Omega^{i-1} - t\Delta f$
    Calculate proximal operator $\Omega^i = prox_g(\Omega^i)$
    Calculate objective obj_i
    if $\frac{obj_i - obj_{i-1}}{obj_i} < \tau$ or $i = I$ **then**
        break **while**
    Repeat

3.3.1 Implementation

The implementation of the mixed graphical model can be found in the **following github repository**. Before running the optimization, auxiliary tasks are done:

- Clean Survey data : Remove identifiers, divide dataset in continuous and discrete variables.
- Apply one-hot encoding to the discrete variables.
- Define weights as recommended in subsec. 3.2.3
- Initialize parameters.

To keep track of the parameters over the optimization procedure iterations, all the parameters are saved into a vector. The size of the vector can be calculated as :

$$T = p + p^2 + p \cdot Y + Y^2 \tag{3.19}$$

Where p and Y represent the total number of continuous variables and total number of levels in the discrete variables, respectively.

For each iteration, the vector is saved into a matrix, forming a matrix of T times the number iterations required for convergence or the Iteration limit I. The last column of the matrix represent the reported results by the proximal gradient algorithm. An example of the matrix after I iterations can be found below:

$$\begin{bmatrix} \alpha_1^0 & \alpha_1^1 & \dots & \alpha_1^I \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_p^0 & \alpha_p^1 & \dots & \alpha_p^I \\ \beta_{11}^0 & \beta_{11}^1 & \dots & \beta_{11}^I \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{pp}^0 & \beta_{pp}^1 & \dots & \beta_{pp}^I \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{pY}^0 & \rho_{pY}^1 & \dots & \rho_{pY}^I \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{YY}^0 & \phi_{YY}^1 & \dots & \phi_{YY}^I \end{bmatrix} \tag{3.20}$$

For the experiment, the regularization parameter is varied over the interval $[10^{-6}, 1]$, with a finer sampling in the zone $[10^{-2}, 10^{-1}]$. As the authors recommend using $\lambda = \sqrt{\frac{log(p+q)}{n}}$, the value was included in the analysis. The maximum number of iterations I is 5000, the tolerance τ has a value of 10^{-5}. The step size t used in the proximal gradient descent is 0.0005.

3.3.2 Results

The algorithm converged before the maximum number of iterations for all values of λ. The impact of the regularization can be seen in fig. 3.3. At low values of λ, all the parameters in Ω_g are non-zero. As the regularization term gets more relevant, more and more parameters go to 0.

Another way to visualize the impact of regularization is to look at how one particular parameter changes with λ. The continuous-continuous edge potential β_{12} , representing the relation between age and log-wage, is displayed in fig. 3.4. As expected, the magnitude of the parameter decreases as λ increases.

To provide a more comprehensive illustration of the impact of regularization, we present a heatmap of the continuous-discrete edge potentials ρ in Figure 3.5. With a low value of lambda, the majority of the parameters retain non-zero values, whereas a high value results in log-wage parameters being forced to zero. Notably, the discrete variable age exhibits non-zero values in four out of six instances.

The results in the next figures come from fitting the model using the recommended value of $\lambda = \sqrt{\frac{log(p+q)}{n}} = 0.026$. The continuous-discrete edge potentials are displayed in fig. 3.7. At that level of regularization, the *age* variable has non-zero values for all the discrete variables.

The discrete-discrete edge potentials ϕ for the recommended level of regularization are displayed in fig. 3.7. To analyse the relationship between discrete variables, the parameters for all of their levels must be reviewed. At this level of regularization, the marital status and education are conditionally independent, as long as race and health.

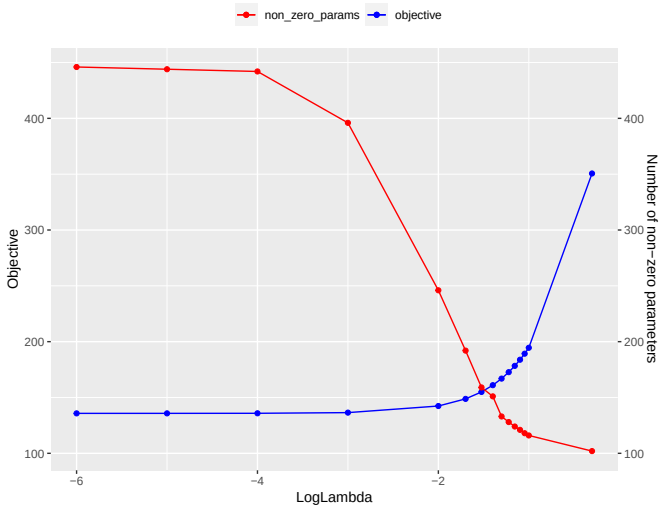


Figure 3.3: Objective function value (blue) and number of non-zero parameters (red) over different values of λ. Each dot represents an evaluation, linear interpolation was used.

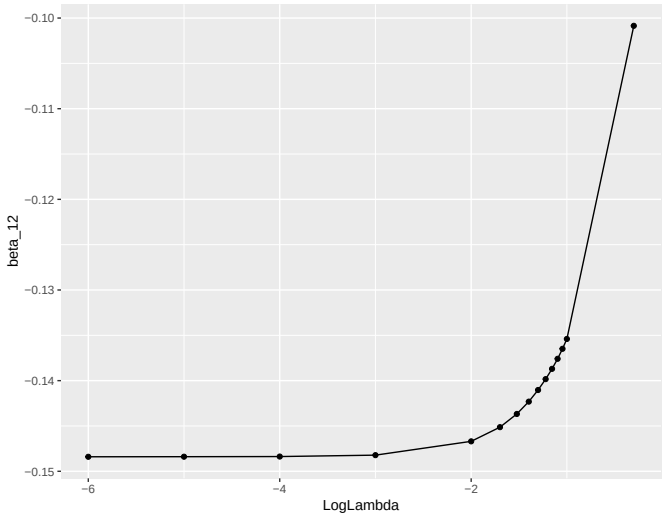
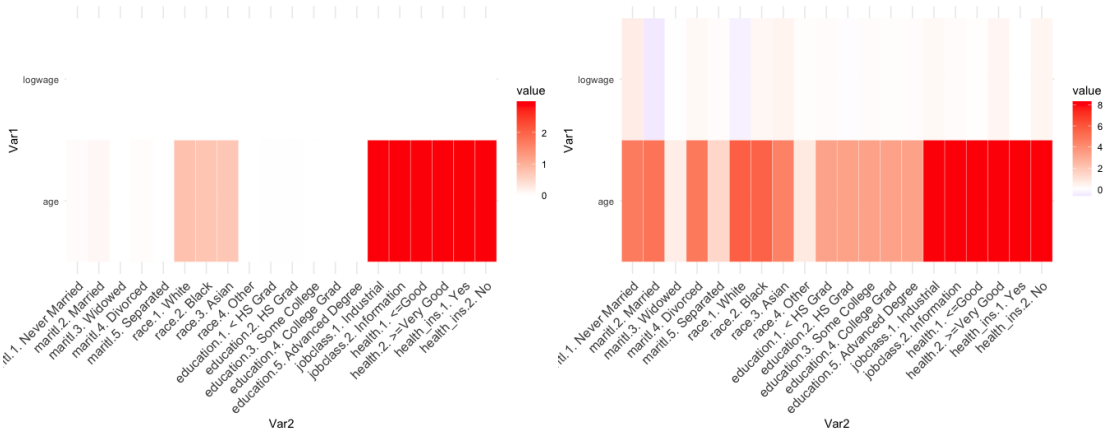


Figure 3.4: Value of β_{12} over different values of λ. Each dot represents an evaluation, linear interpolation was used.

In the case of education and job class, it can be seen that two out of eight parameters are non-zero, so conditionally independence cannot be stated. A similar thing happens for health and health insurance, with 2 out of 4 parameterers being non-zero.



(a) Heatmap of ρ for $\lambda = 1$

(b) Heatmap of ρ for $\lambda = 10^{-5}$

Figure 3.5: Heatmap of ρ for a high and low value of λ

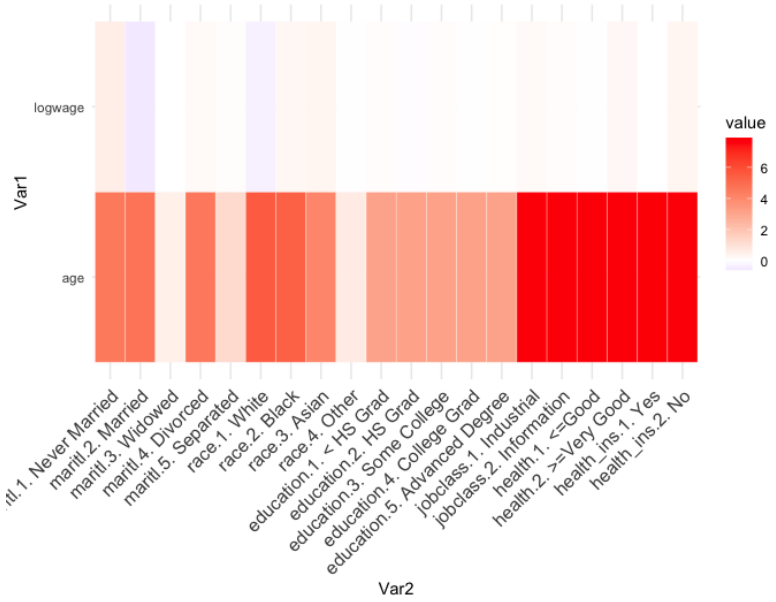


Figure 3.6: Continuous-discrete edge potentials ρ using the recommended value of $\lambda = 0.026$

3.4 Extensions

3.4.1 High Dimensional Mixed Graphical Model

While Lee and Hastie [2] were developing the mixed graphical model, the authors of this extension independently created a model that can be viewed as a more generalized version of the former. The proposed model is a simplified conditional Gaussian model for mixed data that is both flexible enough to capture any dependence structure and simple enough to be applied to high-dimensional data. One important difference is that this model is fitted using a node-based regression approach, which means regressing each variable on the others iteratively, while Lee and Hastie maximize a joint pseudo-likelihood objective function. This results in much faster convergence speed.

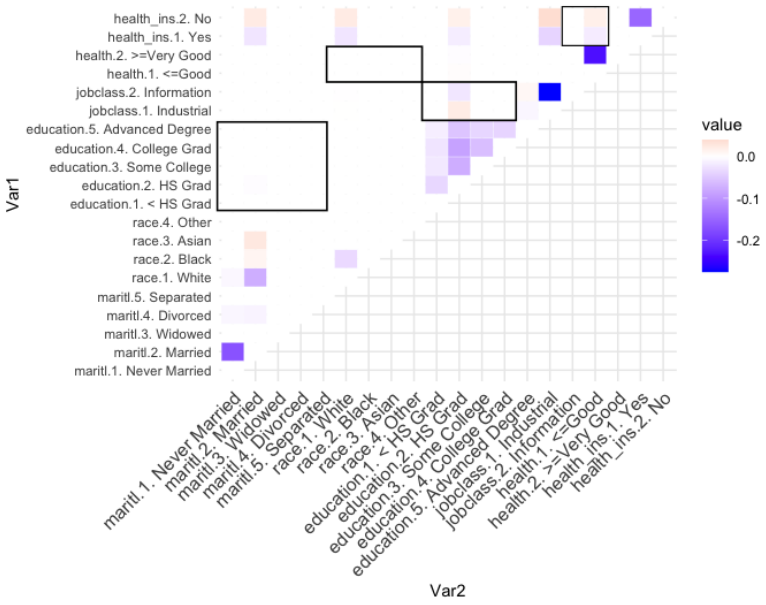


Figure 3.7: Discrete-discrete edge potential ϕ for $\lambda = 0.026$. Black squares show the parameters necessary to assess conditional independence between discrete variables.

To measure the effectiveness of the model, ROC curves (False positive rate vs True positive rates) of the continuous-continous (ZZ), continuous-discrete (ZY) and discrete-discrete (YY) edges are plotted. In figs. 3.8 and 3.9, the performance of different approaches are observed in cases where the true model only contains main effects or when it also contains interaction effects. One very special feature of the proposed model is that it can capture interactions between the variables. This results in the performance improvement seen in 3.9.

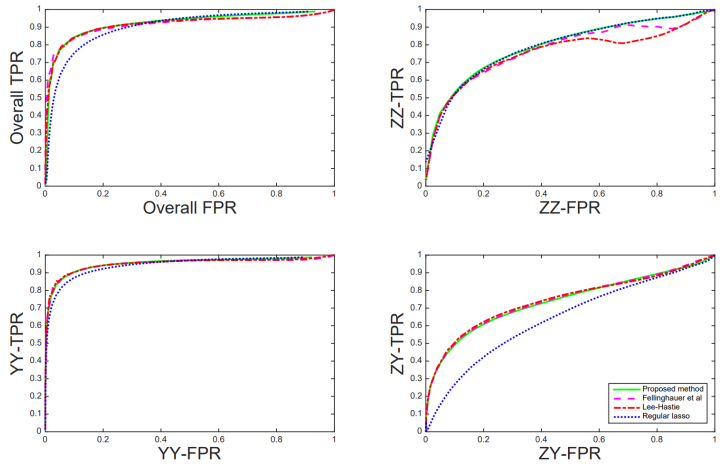


Figure 3.8: Edge-based ROC curves for three graphical model methods when there are only main effects in the true model.

By not assuming constant conditional covariance for all the continuous variables and by fitting separate regressions, the proposed model captures more complex relationships between the nodes and is suitable for applications that imply high-dimensional data.

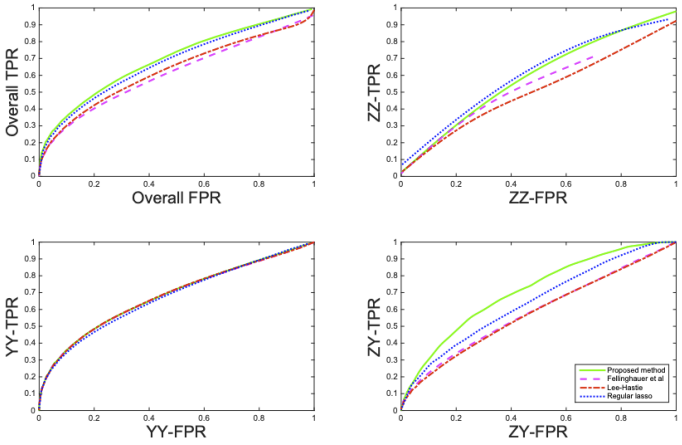


Figure 3.9: Edge-based ROC curves for three graphical model methods when there are both main effects and interactions in the true model.

3.4.2 Functional Graphical Models

Mixed graphical models are effective tools for modeling relationships among scalar variables, but they cannot be directly applied to systems that involve functional variables. In the case of manufacturing technologies, such as the plasma spray process depicted in Figure 3.10, the physical processes involved are often complex and dependent on a multitude of factors. For instance, the quality of the coating surface is directly influenced by *in-situ* process variables like particle velocity and size. These processes are typically measured as functional variables over time, providing valuable information that can be used to model the plasma spray.

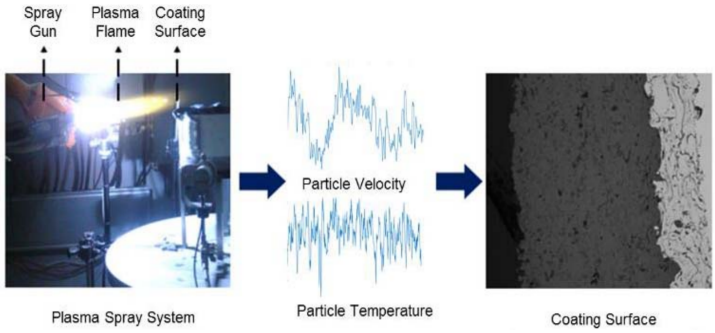


Figure 3.10: Plasma Spray process diagram.

In their investigation [12], the authors utilized Bayesian networks composed of acyclic directed arcs between nodes to model the plasma spray process. Specifically, they demonstrated how the final coating properties can be affected by various process setting variables, such as power supply and carrier gas speed, as well as in-situ process variables reflecting the particle conditions, such as particle velocity. It is worth noting that the in-situ process variables themselves are also impacted by the process setting variables.

One approach to incorporate functional representation into the mixed graphical model is to calculate summary statistics of the functional variables and treat them as

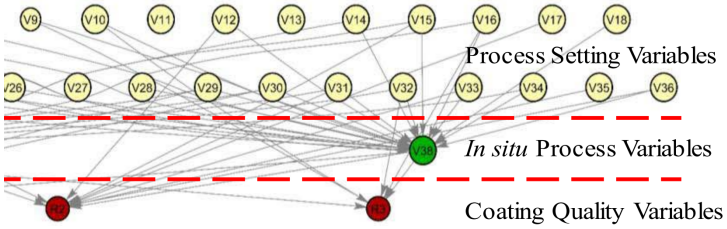


Figure 3.11: Bayesian network related to the plasma spray process.

scalar variables. However, this approach oversimplifies the problem by ignoring their intrinsic structure. An alternative approach is to represent the functional variables using basis expansion, such as Fourier transforms. Each functional variable is then transformed into a vector of basis expansion coefficients, which can be used as inputs to the mixed graphical model. This approach preserves the underlying structure of the functional variables and allows for more accurate modeling of complex processes such as the plasma spray process.

Following the approach of Lee and Hastie, an $l1$ penalty is applied to the scalar variables to encourage sparsity. For the functional variables, the difference from the mean is used as a penalty, as shown in Equation 3.21. This is based on the assumption that in a stationary manufacturing process, the relationships between the basis expansion coefficients in a node and its parent node will be similar. Therefore, the basis expansion coefficients of the same functional node should have similar coefficients in the functional response regression model.

This penalty promotes the similarity of model coefficients for the basis expansion coefficients belonging to the same functional variable. As a result, the model coefficients can be estimated by minimizing a penalized loss function that has two different forms, depending on whether the current node is functional or scalar.

$$\Omega_2(C) = \sum_{t=1}^{m} \|\boldsymbol{c}_t - \boldsymbol{c}_{\text{mean}}\|_2^2 = \|CR\|_F^2, R = \begin{bmatrix} 1-\frac{1}{m} & -\frac{1}{m} & \cdots & -\frac{1}{m} \\ -\frac{1}{m} & 1-\frac{1}{m} & \cdots & -\frac{1}{m} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{m} & -\frac{1}{m} & \cdots & 1-\frac{1}{m} \end{bmatrix} \tag{3.21}$$

The authors have developed a computational procedure for learning the proposed model, utilizing variable ordering based on domain knowledge. Their results have shown that the method outperformed other benchmark algorithms as of 2017, with the requirement of prior knowledge of variable orderings. However, the correlation between variables belonging to the same category, such as process setting or in-situ variables, are not considered. An extension of the graphical model can be made by incorporating undirected edges that represent correlations among variables within the same category. This additional step would enhance the model's ability to capture relationships among variables and lead to improved accuracy in modeling complex manufacturing processes like plasma spraying.

4 Conclusions

Throughout our work, we successfully reproduced the results of two publications by implementing their respective algorithms. This played a crucial role in our understanding of their working principles and optimization procedures.

We utilized the `glasso` library [6] to apply the graphical lasso algorithm in analyzing cell signaling data. We were able to observe the change in overall sparsity and the recovered true edges for varying values of λ. After this, we compared the outputs of the graphical Lasso to the first step of the PC-algorithm, where the Markov equivalence class (a valid Markov random field) is determined. We were able to draw an analogy between the sparsity parameters, λ in graphical lasso and α in the PC-algorithm, for achieving an overall sparse model. We also saw that neither method had a clear advantage in recovering the edges of the ground truth model at the correct sparsity. Finally we introduced the MCPeSe method for selecting λ in cases where the true sparsity is unknown.

For the mixed graphical model, we had to implement it from scratch as the authors' published MATLAB code was not executable at present time. Our implementation in R is now publicly available in a Github repository. We tested the algorithm on a survey dataset and successfully achieved varying levels of sparsity in the model based on the parameter λ. We performed several checks to ensure that the algorithm behaved as expected and used a recommended λ value for interpreting the results.

Since these methods are relatively old, multiple extensions and improvements have been developed in this active field. These include methods for selecting the best regularization parameter for graphical lasso based on a Monte Carlo approach, and deriving analytical solutions for graphical lasso that require significantly less computational effort. For the mixed graphical model, a natural extension is a model that captures more complex relationships like interaction terms, with the additional feature of shorter convergence times. Additionally, the model has been adapted to functional variables where some attributes are functions of others over time, with applications in industrial processes like plasma spray.

References

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 12 2007.

[2] Jason D. Lee and Trevor J. Hastie. Learning the structure of mixed graphical models. *Journal of Computational and Graphical Statistics*, 24(1):230–253, 2015.

[3] Nicolai Meinshausen and Peter Buhlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

[4] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach. Learn. Res.*, 9:485–516, jun 2008.

[5] Karen Sachs, Omar Perez, Dana Pe'er, Douglas Lauffenburger, and Garry Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science (New York, N.Y.)*, 308:523–9, 05 2005.

[6] Friedman. https://cran.r-project.org/web/packages/glasso/glasso.pdf, 2017.

[7] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[8] Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H. Maathuis, and Peter Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.

[9] Markku Kuismin and Mikko J Sillanpää. MCPeSe: Monte Carlo penalty selection for graphical lasso. *Bioinformatics*, 37(5):726–727, 08 2020.

[10] Salar Fattahi and Somayeh Sojoudi. Graphical lasso and thresholding: Equivalence and closed-form solutions, 2017.

[11] Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):179–195, 1975.

[12] Hongyue Sun, Shuai Huang, and Ran Jin. Functional graphical models for manufacturing process modeling. *IEEE Transactions on Automation Science and Engineering*, 14(4):1612–1621, 2017.