ELENG 227BT - Project Report
Joint Graphical Lasso

Gonzalo Benegas (ID: 3033485263) gbenegas@berkeley.edu

Shijie Gu (ID: 3035297760) shijiegu@berkeley.edu

Sebastian Prillo (ID: 3035135585) sprillo@berkeley.edu

Contents

Chapter 1

The Graphical Lasso

1.1 Introduction and the Canonical Objective

Suppose we are given n i.i.d. observations $X_1, \ldots, X_n$ of a p-dimensional random vector and we want to estimate its joint distribution. If we assume that the X_i follow a multivariate Normal distribution, $X_i \overset{\text{i.i.d.}}{\sim} N(\mu, \Sigma)$, then the density for a single example is:

$$p(x) = ((2\pi)^p \det(\Sigma))^{-1/2} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)) \tag{1.1}$$

The Maximum Likelihood Estimator (MLE) of the data is readily seen to be $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n X_i := \bar{X}_n$ the empirical mean, and $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (X_i - \bar{X}_n)(X_i - \bar{X}_n)^T := S \in \mathbb{R}^{p\times p}$ the empirical covariance matrix. However, this is not necessarily a *robust* estimate of the true underlying covariance matrix. In the high dimensional setting where $p > n$, $\hat{\Sigma}$ is not even invertible, leading to a degenerate distribution. We also usually have some strong prior knowledge about the data distribution, for example we know that most pairs of variables are conditionally independent given the values of all the other variables.

High dimensionality and sparse interactions between variables are not uncommon in many applications, for example when working with bulk gene expression data: the number of genes sequenced is around 20000, with only a few tens or hundreds of observations (Imagine a rare genetic disease that affects only a few individuals), and most genes only interact directly (i.e. are *co-regulated*) with few other genes. It turns out that under the multivariate Normal assumption, variables i and j are conditionally independent given all the other variables if and only if $\Theta_{i,j} = 0$, where $\Theta = \Sigma^{-1}$ is the inverse covariance matrix, also known as the precision matrix. Therefore, one way to encode our prior assumption of sparse interactions and robustly estimate Σ is to instead perform maximum likelihood estimation of Θ *under a lasso penalty*. This also solves the degeneracy issue of $\hat{\Sigma}$. In turn, after the model is fit, we can explore the conditional dependencies found by the model to extract valuable insights from the data, for example, pairs of genes that are involved in the same pathway. In many cases, extracting these networks is the only actual goal, and the probabilistic multivariate Gaussian framework is just one means to this end. In this report we will focus on the probabilistic approach, acknowledging that there is also a vast literature on non-likelihood based methods for estimating sparse graphical models (see section 3.2 for a brief overview of some of these methods).

The method of finding the underlying sparse graphical structure of the data through optimizing an L_1 penalized multivariate Gaussian maximum log-likelihood objective is called the *graphical lasso*. The objective is:

$$\boxed{\max_{\Theta \succ 0} \log\det(\Theta) - \text{trace}(\Theta S) - \lambda\|\Theta\|_1} \tag{1.2}$$

where $\lambda \geq 0$ is the regularization strength. Note that this is a convex optimization problem. From a robust optimization perspective, 1.2 is the same as maximizing the data log-likelihood under box uncertainty on the empirical covariance matrix S. Indeed:

$$\max_{\Theta \succ 0} \min_{\|U\|_\infty \leq \lambda} \log\det(\Theta) - \text{trace}(\Theta(S+U)) \tag{1.3}$$

$$= \max_{\Theta \succ 0} \log\det(\Theta) - \text{trace}(\Theta S) - \max_{\|U\|_\infty \leq \lambda} \text{trace}(\Theta U) \tag{1.4}$$

and since by dual norms we have $\max_{\|U\|_\infty \leq \lambda} \text{trace}(\Theta U) = \lambda \|\Theta\|_1$, we recover equation 1.2.

In the next sections we discuss different methods used to solve the graphical lasso problem, as proposed chronologically.

1.2 Coordinate-Descent Based Methods

The first methods proposed to solve the graphical lasso problem were based on coordinate descent. They rely on the use of duality to transform the original problem into a more tractable one. We first review the algorithm of [1] and then the algorithm of [2].

1.2.1 Dual Problem

Following [1], we can derive the dual problem using dual norms as follows:

$$\begin{aligned}
&\max_{\Theta \succ 0} \log\det(\Theta) - \text{trace}(\Theta S) - \lambda\|\Theta\|_1 \\
=&\max_{\Theta \succ 0} \min_{\|U\|_\infty \leq \lambda} \quad \log\det(\Theta) - \text{trace}(\Theta(S+U)) \\
=&\min_{\|U\|_\infty \leq \lambda} \max_{\Theta \succ 0} \quad \log\det(\Theta) - \text{trace}(\Theta(S+U)) \\
&\text{with the inner maximum obtained at } \Theta^{-1} = S+U \\
=&\min_{\|U\|_\infty \leq \lambda} \quad -\log\det(S+U) - p \qquad (1.5) \\
&\text{which letting } W = S+U (= \Theta^{-1}), \text{ and ignoring the constant } p \text{ term, is equivalent to} \\
\equiv&\boxed{\max_{W} \log\det(W) \quad \text{s.t.} \|W-S\|_\infty \leq \lambda} \qquad (1.6)
\end{aligned}$$

The condition $W \succ 0$ is omitted since $\log\det W = -\infty$ if W is not invertible. By dualizing, we are now optimizing over the covariance matrix $W = \Theta^{-1}$ instead of the precision matrix.

1.2.2 Dual's Subproblem and the First Algorithm

A block coordinate descent algorithm to find the off-diagonal elements of 1.6 - and thus solve problem 1.6, since it is convex - can be derived using Schur complements. To do so, let's first introduce some notation from [1]:

Definition. For any symmetric matrix A, let $A_{\backslash k \backslash j}$ denote the matrix produced by removing column k and row j. Also, let a_j denote column j with the diagonal element A_{jj} removed.

With these definitions, if we move the j-th column and row of the matrix W to the last row and column, which can be expressed as: $W = \begin{bmatrix} W_{\backslash j \backslash j} & w_j \\ w_j^T & W_{jj} \end{bmatrix}$, by Schur complements, we have:

$$\det(W) = (W_{jj} - w_j^T (W_{\backslash j \backslash j})^{-1} w_j) \det(W_{\backslash j \backslash j})$$

The first obvious observation is that to maximize $\det(W)$, we should always make W_{jj} as large as possible. Therefore, the optimal W^* satisfies $W^*_{kk} = S_{kk} + \lambda$ necessarily, hence we can fix the diagonal terms of W and

only focus on the off-diagonal entries. Second, given the current value of W, we can improve the objective in 1.6 by updating the j-th row/column as follows:

$$\boxed{w_j \leftarrow \arg\min_{w_j} \frac{1}{2} w_j^T W_{\setminus j \setminus j}^{-1} w_j \quad \text{s.t.} \|w_j - s_j\|_\infty \leq \lambda} \tag{1.7}$$

Note that equality holds in equation 1.7 for all j (i.e. we reach a fixed point) iff W is a global minima of 1.6 (because 1.6 is a convex problem). Moreover, note that 1.7 is a box-constrained quadratic program. This way, we arrive at the first algorithm for solving the graphical lasso: [1] propose using an interior point procedure, where the matrix W is initialized as $W_0 = S + \lambda I$ (which guarantees that $W_0 \succ 0$ and that the diagonal elements are correct) and the above update rule is performed until convergence. Convergence is assessed by analyzing the duality gap, which is the difference between the objective of equation 1.5 and the objective of equation 1.2, that is:

$$\boxed{\text{Duality Gap} = \text{trace}(\Theta S) - p + \lambda \|\Theta\|_1} \tag{1.8}$$

We call equation 1.7 the *dual's subproblem*, since it is the coordinate descent step which is applied repeatedly to solve the dual problem 1.6.

1.2.3 Dual of the Dual's Subproblem and the Second Algorithm

The Dual

What if instead of solving the box-constrained QP of equation 1.7, we solve its dual? The work [2] proposes to do this, because as we will see, the dual can be solved efficiently.

The dual of the dual's subproblem 1.7 can be easily derived using, once again, dual norms (via conic duality):

$$\begin{aligned}
&\min_{w_j} \frac{1}{2} w_j^T W_{\setminus j \setminus j}^{-1} w_j \quad \text{s.t.} \|w_j - s_j\|_\infty \leq \lambda \\
=&\min_{w_j} \max_{\|\beta_j\|_1 \leq t} \frac{1}{2} w_j^T W_{\setminus j \setminus j}^{-1} w_j - \beta_j^T (w_j - s) - \lambda t \\
=&\max_{\|\beta_j\|_1 \leq t} \min_{w_j} \frac{1}{2} w_j^T W_{\setminus j \setminus j}^{-1} w_j - \beta_j^T (w_j - s) - \lambda t
\end{aligned}$$

The interior optimization problem is an unconstrained quadratic program which can be readily solved in close form as $w_j = W_{\setminus j \setminus j} \beta_j$. This way, the dual of problem 1.7 is in the end:

$$\boxed{\min_{\beta_j} \frac{1}{2} {\beta_j}^T W_{\setminus j \setminus j} \beta_j - s_j^T \beta_j + \lambda \|\beta_j\|_1} \tag{1.9}$$

This optimization problem resembles lasso regression. To see this, consider the design matrix $X \in \mathbb{R}^{n \times p}$ and lasso regression of $X_{\setminus j}$ onto x_j with weights β_j, the objective is thus:

$$\begin{aligned}
&\min_{\beta_j} \frac{1}{2} \|x_j - X_{\setminus j} \beta_j\|_2^2 + \lambda \|\beta_j\|_1 \\
\equiv&\boxed{\min_{\beta_j} \frac{1}{2} \beta_j^T S_{\setminus j \setminus j} \beta_j - s_j^T \beta_j + \lambda \|\beta_j\|_1}
\end{aligned} \tag{1.10}$$

The similarity between 1.9 and 1.10 is striking. It reveals that the graphical lasso can be viewed as p coupled lasso regression problems, where the solution to one lasso regression problem is linked to the coefficients of the other lasso regression problems (via $w_j = W_{\setminus j \setminus j} \beta_j$), unlike equation 1.10 which are p decoupled lasso regression problems.

The whole point of solving the dual 1.9 instead of the primal 1.7 is that the dual is a lasso regression type problem whereas the primal is a constrained QP, and there exist fast solvers for lasso regression type problems, as developed by concurrent work of the authors of [2] in [3]. Despite the fact that solving the primal 1.7 and the dual 1.9 is equivalent, their computational time can differ dramatically: the fast coordinate descent method of [2] can be 30 to 4000 times faster than the interior point procedure of [1] for solving 1.7, according to [2]. This algorithm is plainly called the *Graphical Lasso* algorithm. In the next section we review the last step in the actual computation of 1.9.

Reduction to Univariate Lasso Regression

As we saw, the dual of the dual's subproblem is the lasso regression type problem (equation 1.9):

$$\min_{\beta_j} \frac{1}{2}{\beta_j}^T W_{\backslash j \backslash j} \beta_j - s_j^T \beta_j + \lambda \|\beta_j\|_1 \tag{1.11}$$

The work of [3] shows that coordinate descent is an effective method for solving this lasso regression problem. In the particular case of equation 1.11, we look to optimize the coordinate β_{ji} while all others are fixed. For notational simplicity, let $V = W_{\backslash j \backslash j}$ (which is positive definite). The objective for β_{ji} is then:

$$\min_{\beta_{ji}} \frac{1}{2}\beta_{ji}^2 V_{ii} - \beta_{ji}(s_{ji} - \sum_{k \neq i} V_{ki}\beta_{jk}) + \lambda|\beta_{ji}|$$

which can be rewritten in univariate lasso form:

$$\boxed{\min_{\beta_{ji}} \frac{1}{2}\Big(\beta_{ji} - \frac{s_{ji} - \sum_{k \neq i} V_{ki}\beta_{jk}}{V_{ii}}\Big)^2 + \frac{\lambda}{V_{ii}}|\beta_{ji}|}$$

The solution to this problem is well known to be (this was, in fact, a problem in our homeworks):

$$\boxed{\beta_{ji} \leftarrow \text{soft\_threshold}(s_{ji} - \sum_{k \neq i} V_{ki}\beta_{jk}, \lambda)/V_{ii}}$$

where soft_threshold is the soft-threshold operator:

$$\text{soft\_threshold}(x, t) = \text{sign}(x)(|x| - t)_+ \tag{1.12}$$

This way, we have reduced the original problem of optimization over Θ to repeated application of univariate lasso regression - which is thus the workhorse behind the Graphical Lasso algorithm as proposed by [2].

Proof of Equivalence: the dual of the dual's subproblem and the primal

The relationship between the dual of the dual's subproblem (1.9) and the original problem (1.2) can be established directly as shown in [2], which we review here. Formally, we claim that Θ is optimal for 1.2 if and only if $\beta_j = W_{\backslash j \backslash j}^{-1} w_j$ minimizes 1.9 for each $1 \leq j \leq p$ (recall that $W = \Theta^{-1}$) and also $W_{kk} = S_{kk} + \lambda$ $(1 \leq k \leq p)$. To show this, taking the subgradient of the original objective (1.2), we get Θ is optimal iff:

$$W - S - \lambda \text{sign}(\Theta) = 0 \tag{1.13}$$

where:

$$\{\text{sign}(X)\}_{ij} = \text{sign}(x_{ij}) = \begin{cases} 1, & \text{if } x_{ij} > 0 \\ \gamma \in [-1, 1], & \text{if } x_{ij} = 0 \\ -1, & \text{if } x_{ij} < 0 \end{cases} \tag{1.14}$$

On the other hand, taking the subgradient of 1.9, we get that Θ is optimal iff $W_{kk} = S_{kk} + \lambda$ $(1 \leq k \leq p)$ and:

$$W_{\backslash j \backslash j}\beta_j - s_j + \lambda \text{sign}(\beta_j) = 0 \ (1 \leq j \leq p)$$

where $\beta_j = W^{-1}_{\backslash j \backslash j} w_j$, i.e. iff $W_{kk} = S_{kk} + \lambda$ $(1 \leq k \leq p)$ and:

$$w_j - s_j + \lambda \text{sign}(W^{-1}_{\backslash j \backslash j} w_j) = 0 \ (1 \leq j \leq p)$$

We claim that this last condition, paired with $W_{kk} = S_{kk} + \lambda$ $(1 \leq k \leq p)$, is equivalent to 1.13. The condition $W_{kk} = S_{kk} + \lambda$ $(1 \leq k \leq p)$ is equivalent to 1.13 for the diagonal elements, since this choice of diagonal elements of W is optimal no matter what the other elements of W are. As for the off-diagonal elements in 1.13, all we have to show is that $\text{sign}(\theta_j) = \text{sign}(W^{-1}_{\backslash j \backslash j} w_j)$. This is true because from $\Theta W = I$ we get that, in particular, $\theta_j = -\theta_{jj} W^{-1}_{\backslash j \backslash j} w_j$. Because $\theta_{jj} > 0$ (since $\Theta \succ 0$), the sign reversal between $W^{-1}_{\backslash j \backslash j} w_j$ and θ_j is obtained, and equivalence is proved.

1.2.4 Implementation Details

In the implementation of [2], convergence is declared if the average absolute change in W is less than $t \cdot \text{ave}(S^{-\text{diag}})$, where $S^{-\text{diag}}$ are the off-diagonal elements of the empirical covariance matrix S, and t is a fixed threshold, set by default at 0.001. Once we find β_j, we update w_j via $w_j = V\beta_j$ and continue with β_{j+1}, and so on. Note that since β_j is a sparse vector, the product $w_j = V\beta_j$ can be computed in rp operations, where r is the number of non-zero elements of β_j.

Once we converge to a solution W, we can recover Θ without inverting W by leveraging our β variables as follows: From the identity $W\Theta = I$, we get:

$$\begin{aligned} W_{\backslash j \backslash j}\theta_j + w_j \Theta_{jj} &= 0 \\ w_j^T \theta_j + W_{jj}\Theta_{jj} &= 1 \end{aligned}$$

We thus deduce:

$$\begin{aligned} \Theta_{jj} &= 1/(W_{jj} - w_j^T W^{-1}_{\backslash j \backslash j} w_j) \\ \theta_j &= -W^{-1}_{\backslash j \backslash j} w_j \Theta_{jj} \end{aligned}$$

It appears like we would need to invert $W_{\backslash j \backslash j}$ to recover Θ, but this is actually non needed. Indeed, we know that $W^{-1}_{\backslash j \backslash j} w_j = \beta_j$, hence:

$$\begin{aligned} \Theta_{jj} &= 1/(W_{jj} - w_j^T \beta_j) \\ \theta_j &= -\beta_j \Theta_{jj} \end{aligned}$$

which gets rid of the matrix inversion. This has a simple interpretation too:

$$\theta_j \textit{ is just a scaling of } \beta_j \textit{ by } -\Theta_{jj}$$

where Θ_{jj} is computed with the first equation based on W and β_j. This should not be very surprising, as β is the variable of the *dual* of the *dual*'s subproblem, and dualizing twice in many cases produces the primal problem again. Of course, here we are not literally dualizing twice: we are dualizing a *subproblem* of the dual.

Putting it all together, the Graphical Lasso algorithm as proposed by [2] is:

Algorithm 1: The Graphical Lasso

Input: $S \in \mathbb{R}^{p \times p}$, the sample covariance matrix
$\lambda > 0$, the regularization strength
$t > 0$, the threshold to determine convergence
Result: $\Theta \in \mathbb{R}^{p \times p}$, the estimated sparse precision matrix.
$W = \Theta^{-1}$ the estimated covariance matrix
$W = S + \lambda I$;
$\beta = 0 \in \mathbb{R}^{p \times (p-1)}$;
do
  for *j = 1,2,...,p* **do**
    $V = W_{\setminus j \setminus j}$;
    for *i = 1,2,...,p-1* **do**
      $\beta_{ji} \leftarrow \text{soft\_threshold}(s_{ji} - \sum_{k \neq i} V_{ki}\beta_{jk}, \lambda)/V_{ii}$;
    end
    $w_j \leftarrow V\beta_j$ (replacing both row and column j of W);
  end
while *Average absolute change in W is more than* $t \cdot ave(S^{-diag})$;
$\Theta = 0 \in \mathbb{R}^{p \times p}$;
for *j = 1,2,...,p* **do**
  $\Theta_{jj} = 1/(W_{jj} - w_j^T \beta_j)$;
  $\theta_j = -\beta_j \Theta_{jj}$;
end
Return Θ, W;

In fact, the algorithm is so simple that it is easy to write a bare-bones, working implementation in python:

```
import numpy as np

def graphical_lasso(S, l, threshold=1e-3):
    def soft_threshold(x, l):
        return np.sign(x) * np.max([np.abs(x) - l, 0])
    p = S.shape[0]
    W = S + l * np.identity(p)
    beta = np.zeros(shape=(p, p - 1))
    while True:
        W_old = np.copy(W)
        for j in range(p):
            minus_j = [idx for idx in range(p) if idx != j]
            V = W[minus_j, :][:, minus_j]
            s_j = S[minus_j, j]
            for i in range(p - 1):
                beta[j, i] = soft_threshold(s_j[i] - np.dot(V[:, i], beta[j]) \
                                            + V[i, i] * beta[j, i], l) / V[i, i]
            w_j = V @ beta[j]
            W[minus_j, j] = w_j
            W[j, minus_j] = w_j
        if np.mean(np.abs(W - W_old)) < threshold * np.mean(np.abs(S - np.diag(np.diag(S)))):
            break
    theta = np.zeros_like(W)
    for j in range(p):
        minus_j = [idx for idx in range(p) if idx != j]
        w_j = W[j, minus_j]
        theta[j, j] = 1.0 / (W[j, j] - np.dot(w_j, beta[j]))
        theta[minus_j, j] = -beta[j] * theta[j, j]
    return theta, W
```

1.3 Speeding up the Graphical Lasso

Although much insight for how to solve the problem has been derived from duality, one additional insight can be obtained directly from the subgradient (KKT condition) of the original objective (equation 1.13, which is $W - S - \lambda \text{sign}(\Theta)$). In particular, we see that if $|S_{ij}| \leq \lambda$, then $\Theta_{ij} = 0$ and $W_{ij} = 0$ is a solution. We also know that the inverse of a block diagonal matrix is also block diagonal with the same block structure. With these two useful properties, we could arrange rows and columns of the soft-thresholded matrix soft_threshold(S, λ) into block diagonal form $\begin{bmatrix} S'_1 & & \\ & \ddots & \\ & & S'_N \end{bmatrix}$. The nodes which appear in each of the blocks of this soft-thresholded matrix can be identified, and going back to our original problem, we could just consider the problem of estimating W_i and Θ_i *separately* for each of these N components using their empirical covariance matrices S_i, $i = 1, \ldots, N$. This produces the exact same solution as the original problem (Because off-block entries satisfy the condition $|S_{ij}| \leq \lambda$ and $W_{ij} = 0$ (consider equation 1.7), therefore $\Theta_{ij} = 0$ with diagonal matrix inverse property.); for a full proof, see [4]. This method of first finding connected components from the empirical covariance matrix and then solving each graphical lasso subproblem separately provides great speed improvement of the algorithm, and is currently used in the implementation of the `glasso` function in R [4]. We separately implemented this version of graphical lasso in `python`.

1.4 Other Variants of the Graphical Lasso

One popular variant of the Graphical Lasso involves not penalizing the diagonal terms of the precision matrix. In this case, the optimization objective is instead:

$$\boxed{\max_{\Theta \succ 0} \log\det(\Theta) - \text{trace}(\Theta S) - \lambda \sum_{i \neq j} |\Theta_{ij}|} \tag{1.15}$$

More generally, one can apply a different penalization weight to different entries in the precision matrix, thus obtaining:

$$\boxed{\max_{\Theta \succ 0} \log\det(\Theta) - \text{trace}(\Theta S) - \|P \odot \Theta\|_1} \tag{1.16}$$

The methods, proofs, and algorithms mentioned previously remain almost unchanged when the objective is generalized in this way. For example, in the Graphical Lasso Algorithm 20, it suffices to replace λ with P_{ij} in the soft-thresholding operator, and use $W = S + P \odot I$ upon initialization.

Chapter 2

The Joint Graphical Lasso

2.1 Introduction and the Generalized Objective

Imagine data coming from two related, but different, distributions. For example, imagine RNA sequencing data of healthy cells and tumor cells. It is expected that healthy and tumor cells share many common pathways and thus underlying sparse gene co-regulation network. However, tumor cells have some disrupted networks, and we would like to find these from data as they would give us insights into the mechanisms behind cancer. The goal of the Joint Graphical Lasso is to find the sparse graphical structure of two or more distributions that are known to share a lot of their structure, and in turn reveal their key differences.

The literature on joint estimation of sparse graphical models via the probabilistic framework is usually framed as solving the optimization problem:

$$\max_{\Theta} \sum_{k=1}^{K} n_k(\log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)})) - P(\Theta)$$

where n_k is the size of dataset k (or 1, if we want to all give datasets equal weight), $S^{(k)}$ is the empirical covariance matrix of dataset k, and $\Theta^{(k)}$ its estimated precision matrix. For simplicity we also write $\Theta = (\Theta^{(1)}, \ldots, \Theta^{(K)})$ which causes no confusion. Finally–and most importantly–P is a regularizer that encourages the precision matrices $\Theta^{(k)}$ to be sparse and have similar structure. We will refer to the objective cast in the form above as **generic joint graphical lasso form** in this chapter. In the next sections we review two proposals in this direction, again in chronological order.

2.2 Guo *et al.*'s Proposal–decomposing same and different

The first approach–to the best of our knowledge–on modeling joint sparse graphical structure was introduced in [5]. In this work, the author proposed to decompose each entry of the precision matrix into two components: $\Theta_{ij}^{(k)} = \alpha_{ij} \cdot \beta_{ij}^{(k)}$, where α is the common entry shared across all $\Theta^{(1)}, ..., \Theta^{(K)}$, whereas $\beta_{ij}^{(k)}$ allows kth precision matrix's i, j entry $\Theta_{i,j}^{(k)}$ to differ from that of α_{ij}'s if α_{ij} is not zero. With this decomposition, the objective is:

$$\max_{\Theta} \sum_{k=1}^{K} n_k(\log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)})) - \eta_1 \sum_{i\neq j} |\theta_{i,j}| - \eta_2 \sum_{i\neq j} \sum_{k=1}^{K} |\gamma_{i,j}^{(k)}| \tag{2.1}$$

where η_1 serves as a normal graphical lasso parameter that tunes the sparsity of the precision matrix and η_2 is the joint graphical lasso parameter that tunes the similarity across precision matrices across categories. Overall, this objective encourages *shared zeros* in the precision matrices.

To massage the objective into the generic form, where only Θ_{ij} is involved, we can make use of the AM-GM inequality on the last two terms. We can get the following form (a more detailed proof is available

in citation [6]):

$$\max_{\Theta} \sum_{k=1}^{K} n_k(\log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)})) - \lambda \sum_{i\neq j}(\sum_{k=1}^{K} |\Theta_{i,j}^{(k)}|)^{1/2}$$

Therefore, we have arrived at the generic joint graphical lasso form, with the regularizer taken to be:

$$\boxed{P(\Theta) = \lambda \sum_{i\neq j}(\sum_{k=1}^{K} |\Theta_{ij}^{(k)}|)^{\frac{1}{2}}} \tag{2.2}$$

And the overall objective is:

$$\max_{\Theta} \sum_{k=1}^{K} \log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)}) - \lambda \sum_{i\neq j}(\sum_{k=1}^{K} |\Theta_{ij}^{(k)}|)^{\frac{1}{2}} \tag{2.3}$$

2.2.1 Solving the problem via linear approximation–decouples the problem into graphical lasso subproblems

The function P is non-convex and tricky to deal with. To optimize the joint objective, [5] thus proposes an iterative approach where we repeatedly approximate P at each step with a tractable function and solve the tractable problem. The function $g(x) = \sqrt{x}$ is approximated at point $x_0 > 0$ as follows:

$$\sqrt{x} = \frac{x}{\sqrt{x}} \cong \frac{x}{\sqrt{x_0}}$$

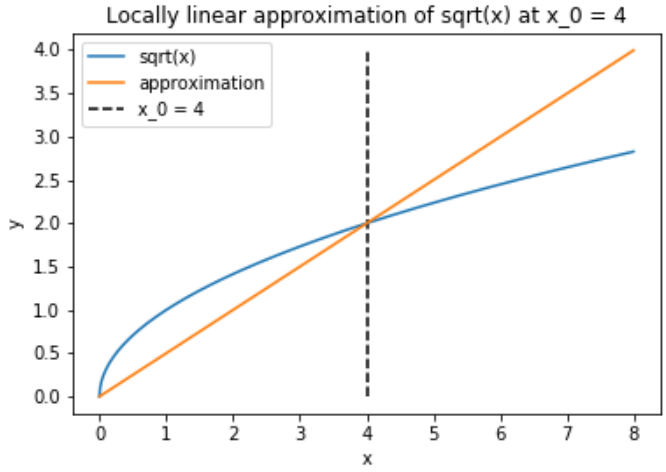

Figure 2.1: Locally Linear Approximation

This way, at each step, if $\tilde{\Theta}^{(1)}, \ldots, \tilde{\Theta}^{(K)}$ are the current values of the precision matrices, we consider the following approximate optimization objective instead:

$$\max_{\Theta} \sum_{k=1}^{K} \log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)}) - \lambda \sum_{i\neq j}(\sum_{k=1}^{K} |\Theta_{ij}^{(k)}|)/(\sum_{k=1}^{K} |\tilde{\Theta}_{ij}^{(k)}|)^{\frac{1}{2}}$$

This objective is separable in the $\Theta^{(k)}$; indeed, it can be written in the following form which evidences its separability:

$$\sum_{k=1}^{K} \max_{\Theta^{(k)}} \Big(\log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)}) - \|\Omega \odot \Theta^{(k)}\|_1 \Big)$$

where:

$$\Omega_{ij} = (\sum_{k=1}^{K} |\tilde{\Theta}_{ij}^{(k)}|)^{-\frac{1}{2}}$$

and the diagonal of Ω is zero. That is, the problem reduces to separate estimation of K Graphical Lasso problems with a generalized L_1 weight matrix as we saw in section 1.4. This is elegant, as the estimation of joint sparse structure is reduced to repeated Graphical Lasso subproblems. The downside is that the Graphical Lasso has to solved multiple times until convergence, making the joint approach of Guo *et al.* [5] computationally expensive.

2.3 Danaher *et al.*'s Proposal

The second paper we will review regarding joint estimation of sparse graphical models is [7]. In this paper, two different regularizers are proposed.

2.3.1 Group Graphical Lasso (GGL)

One of the two objectives proposed resembles the proposal by Guo *et al.* closely; it involves the regularizer:

$$\boxed{P(\Theta) = \lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} |\Theta_{ij}^{(k)}| + \lambda_2 \sum_{i \neq j} (\sum_{k=1}^{K} (\Theta_{ij}^{(k)})^2)^{\frac{1}{2}}} \tag{2.4}$$

This form has straightforward interpretation upon looking: the Graphical Lasso's L_1 penalty is used to induce sparsity on the precision matrices, but a group lasso penalty is incorporated that encourages similar sparsity structure among the precision matrices, albeit not enforcing them to have the *exact same* non-zero values. This is called the *Group Graphical Lasso*.

Penalty 2.4 can be connected back to Guo's proposal. Specifically, modifying equation 2.1 into

$$\max_{\Theta} \sum_{k=1}^{K} n_k (\log\det(\Theta^{(k)}) - \text{trace}(S^{(k)}\Theta^{(k)})) - \eta_1 \sum_{i \neq j} (\theta_{i,j})^2 - \eta_2 \sum_{i \neq j} \sum_{k=1}^{K} (\gamma_{i,j}^{(k)})^2$$

We can then arrive at a penalty form of:

$$P(\Theta) = \lambda \sum_{i \neq j} (\sum_{k=1}^{K} (\Theta_{ij}^{(k)})^2)^{\frac{1}{2}}$$

Therefore, GGL's penalty could be understood as a modification of Guo's proposal *plus* an additional L1-norm penalty on all precision matrices. We could then conclude that this proposal, as Guo's, only encourages shared *zeros* (pattern of sparsity).

2.3.2 Fused Graphical Lasso (FGL)

The other objective considered in [7] is the *Fused Graphical Lasso*, which considers the regularizer:

$$\boxed{P(\Theta) = \lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} |\Theta_{ij}^{(k)}| + \lambda_2 \sum_{k,k'} \sum_{i,j} |\Theta_{ij}^{(k)} - \Theta_{ij}^{(k')}|} \tag{2.5}$$

In addition to the L_1 regularizer for all the precision matrices, it adds an extra penalty to encourage the precision matrix weights to be the same at each position. This is different from Guo *et al.*'s proposal 2.2 and GGL, which only encourages shared *zeros* (pattern of sparsity).

2.3.3 Solving GGL and FGL via ADMM

For both the Fused Graphical Lasso and the Group Graphical Lasso, [7] proposes optimization based on an alternating directions method of multipliers (ADMM) [8]. The generalized objective 2.1 is first rewritten as follows, introducing variables $Z^{(k)}$ tied to the $\Theta^{(k)}$:

$$\min_{\Theta} \underbrace{-\sum_{k=1}^{K} n_k(\log\det(\Theta^{(k)}) - \operatorname{trace}(S^{(k)}\Theta^{(k)}))}_{f(\Theta)} + \underbrace{P(\Theta)}_{g(\Theta)} = \min f(\Theta) + g(\Theta) = \boxed{\min_{\Theta=Z} f(\Theta) + g(Z)} \tag{2.6}$$

which is trivially equivalent to the augmented objective:

$$\min_{\Theta=Z} f(\Theta) + g(Z) + \frac{\rho}{2}\sum_{k=1}^{K} \|\Theta^{(k)} - Z^{(k)}\|_F^2$$

for any $\rho > 0$ (since the penalty we added is identically zero on the feasible set). The Lagrangian of this augmented objective is:

$$L_\rho(\Theta, Z, U) = f(\Theta) + g(Z) + \frac{\rho}{2}\sum_{k=1}^{K} \|\Theta^{(k)} - Z^{(k)}\|_F^2 + \sum_{k=1}^{K} \operatorname{trace}(U^{(k)}(\Theta^{(k)} - Z^{(k)}))$$

Because the Lagrangian is now augmented with an extra quadratic term compared to the bare Lagrangian of equation 2.6, it is called the *Augmented Lagrangian*.

By reparameterizing U as ρU, we obtain the *Scaled Augmented Lagrangian*:

$$L_\rho(\Theta, Z, U) = f(\Theta) + g(Z) + \frac{\rho}{2}\sum_{k=1}^{K} \|\Theta^{(k)} - Z^{(k)} + U^{(k)}\|_F^2 - \frac{\rho}{2}\sum_{k=1}^{K} \|\operatorname{trace}(U^{(k)})\|_F^2$$

Dual ascent based on the Scaled Augmented Lagrangian can now be performed. To do this, the dual function is:

$$g(U) = \min_{\Theta, Z} L_\rho(\Theta, Z, U) \tag{2.7}$$

The gradient of g with respect to U is:

$$\nabla_{U^{(k)}} g|_U = \rho(\tilde{\Theta}^{(k)} - \tilde{Z}^{(k)})$$

where $\tilde{\Theta}^{(k)}, \tilde{Z}^{(k)}$ are the minimizers in equation 2.7. Therefore, to perform dual ascent we can repeatedly solve, starting from some valid initialization:

1. $\Theta, Z \leftarrow \arg\min L_\rho(\Theta, Z, U)$
2. $U \leftarrow U + (\Theta - Z)$

In a typical first order gradient ascent based approach, we would consider the update $U \leftarrow U + \alpha\rho(\Theta - Z)$ for some *learning rate* α. Why is it then that we just settle for $U \leftarrow U + (\Theta - Z)$? It turns out that this update enjoys good theoretical properties; the details can be found in [8].

We are still not done, because the first step $\Theta, Z \leftarrow \arg\min L_\rho(\Theta, Z, U)$ requiring *joint* minimization over Θ, Z is still quite hard. However, we can perform one step of coordinate descent in each direction, obtaining:

1. $\Theta \leftarrow \arg\min L_\rho(\Theta, Z, U)$
2. $Z \leftarrow \arg\min L_\rho(\Theta, Z, U)$
3. $U \leftarrow U + (\Theta - Z)$

This is known as the *alternating directions method of multipliers* (ADMM) algorithm. In general, the ADMM method can be applied to any problem of the form:

$$\min_{Ax+Bz=c} f(x) + g(z)$$

Instantiating the algorithm for our particular problem, we get:

Algorithm 2: The Joint Graphical Lasso

Input: $S^{(1)}, \ldots, S^{(K)} \in \mathbb{R}^{p \times p}$, the sample covariance matrices
$n_1, \ldots, n_K > 0$, the dataset sizes
$\lambda_1, \lambda_2 > 0$, the regularization strengths
$t > 0$, the threshold to determine convergence
Result: $\Theta^{(1)}, \ldots \Theta^{(K)} \in \mathbb{R}^{p \times p}$, the estimated sparse precision matrices.
do
  for k = *1,...,* K **do**
    $VDV^T \leftarrow \text{SVD}(S^{(k)} - \rho Z^{(k)}/n_k + \rho U^{(k)}/n_K)$;
    $\tilde{D} \leftarrow \frac{n_k}{2\rho}(-D + (D^2 + 4\rho/n_k)^{1/2})$;
    $\Theta^{(k)} \leftarrow V\tilde{D}V^T$;
    $Z^{(k)} = \arg\min_{Z^{(k)}} \frac{\rho}{2} \sum_{k=1}^{K} \|Z^{(k)} - (\Theta^{(k)} - U^{(k)})\|_F^2 + P(Z)$;
    $U^{(k)} \leftarrow U^{(k)} + \Theta^{(k)} - Z^{(k)}$;
  end
while *not converged*;
Return $\Theta^{(1)}, \ldots, \Theta^{(K)}$;

The first three steps in the inner loop compute the update for $\Theta^{(k)}$; the update to $Z^{(k)}$ is specific to the penalty and is derived separately; finally, $U^{(k)}$ is updated. The bottleneck is computing the SVD decomposition of a $p \times p$ matrix, which has complexity $\mathcal{O}(p^3)$. The details are in [7].

2.4 Speeding up the Joint Graphical Lasso

Unlike the simple Graphical Lasso, the objective for the Joint Graphical Lasso depends on the choice of the regularizer P, so Theorems like that seen in section 1.3 must to be tailored to each choice of P. Guo *et al.*'s approach is very particular because it leverages the Graphical Lasso, so any speedup made to the Graphical Lasso benefits this approach. However, no Theorem is provided by [5] that decouples the optimization problem (1) from the very beginning, and (2) based on the *original* objective. The work of [7], in contrast, provides Theorems for both the Fused Graphical Lasso (FGL) and the Group Graphical Lasso (GGL), based on the original objective, which allow us to decouple the problem once at the beginning of optimization. They prove the following two Theorems:

Theorem (FGL separability). Consider the FGL optimization problem with $K = 2$ classes. Let C_1 and C_2 be a non-overlapping partition of the p variables, such that $C_1 \cap C_2 = \phi, C_1 \cup C_2 = \{1, 2, \ldots, p\}$. The following conditions are necessary and sufficient for the variables in C_1 to be completely disconnected from those in C_2 in each of the resulting network estimates:

1. $|n_1 S_{ij}^{(1)}| \leq \lambda_1 + \lambda_2$ for all $i \in C_1$ and $j \in C_2$,
2. $|n_2 S_{ij}^{(2)}| \leq \lambda_1 + \lambda_2$ for all $i \in C_1$ and $j \in C_2$, and
3. $|n_1 S_{ij}^{(1)} + n_2 S_{ij}^{(2)}| \leq 2\lambda_1$ for all $i \in C_1$ and $j \in C_2$.

Furthermore, if $K > 2$, then

$$|n_k S_{ij}^{(k)}| \leq \lambda_1 \text{ for all } i \in C_1,\ j \in C_2,\ k = 1, \ldots, K$$

is a sufficient condition for the variables in C_1 to be completely disconnected from those in C_2.

Theorem (GGL separability). Consider the GGL optimization problem with $K \geq 2$ classes. Let C_1 and C_2 be a non-overlapping partition of the p variables, such that $C_1 \cap C_2 = \phi, C_1 \cup C_2 = \{1, 2, \ldots, p\}$. The following conditions are necessary and sufficient for the variables in C_1 to be completely disconnected from those in C_2 in each of the resulting network estimates:

$$\sum_{k=1}^{K} (|n_k S_{ij}^{(k)}| - \lambda_1)_+^2 \leq \lambda_2^2 \text{ for all } i \in C_1, j \in C_2.$$

With these two Theorems, we are able to decompose the original Joint Graphical Lasso problem into smaller subproblems, and run our algorithm on those instead. This reduces the complexity of each inner iteration for a given k from $\mathcal{O}(p^3)$ to $\sum \mathcal{O}(p_i^3)$ where p_i is the dimension of each subproblem, which is considerably faster if the p_i are smaller than p.

Chapter 3

Other Research Directions

The Graphical Lasso is an active area of research, and although we have focused on probabilistic joint learning of sparse graphical models, here we mention other important research directions.

3.1 Closed-form Solutions

The Theorem in section 1.3 reveals a tight connection between the empirical covariance matrix and the solution to the Graphical Lasso: the thresholded empirical covariance matrix and the solution to the Graphical Lasso have the same block diagonal structure. In particular, these two matrices have a similar pattern of non-zero elements. Under what conditions do these non-zero entries *match exactly*? Can this lead us to a better understanding of the Graphical Lasso, and perhaps to a closed-form solution, like in the case of lasso regression? Recent work [9] [10] shows some surprising results in this direction. Here we summarize one of their main results, which roughly states that the Graphical Lasso has a closed form solution if the soft-thresholded empirical covariance matrix induces an acyclic graph.

Definition. For a matrix M, let $\|M\|_{\max} = \max_{i \neq j} |M_{ij}|$ be the largest absolute value of an off-diagonal element.

Definition. Given a symmetric matrix $M \in \mathbb{S}^p$, the **support graph or sparsity graph** of M is defined as a graph with the vertex set $\mathcal{V} := \{1, 2, \ldots, p\}$ and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ such that $(i, j) \in \mathcal{V}$ if and only if $M_{ij} \neq 0$, for every two different vertices $i, j \in \mathcal{V}$. The support graph of M captures the sparsity pattern of the matrix M and is denoted as $\text{supp}(M)$.

Definition. Define the **residue of S relative to λ** as a matrix $S^{\text{res}}(\lambda) \in \mathbb{S}^p$ such that the (i, j) entry of $S^{\text{res}}(\lambda)$ is equal to $S_{ij} - \lambda \text{sign}(S_{ij})$ if $i \neq j$ and $|S_{ij}| > \lambda$, and is equal to 0 otherwise. Furthermore, define the **normalized residue of S relative to λ** as:

$$\tilde{S}^{\text{res}}(\lambda) = D^{-1/2} S^{\text{res}}(\lambda) D^{-1/2}$$

where D is the diagonal matrix with $D_{ii} = S_{ii}$ for every $i \in \{1, 2, \ldots, p\}$.

In other words, $S^{\text{res}}(\lambda)$ and $\tilde{S}^{\text{res}}(\lambda)$ are soft-thresholded, and soft-thresholded-then-normalized versions of S respectively. Note that they both have zeros on the diagonal. Theorem in section 1.3 precisely states that $S^{\text{res}}(\lambda)$ and Θ have the same block diagonal structure.

We can now state the following closed-form solution Theorem for the Graphical Lasso:

Theorem. Assume that the following conditions are satisfied:

- The graph $\text{supp}(S^{\text{res}}(\lambda))$ is acyclic.
- $I + \tilde{S}^{\text{res}}(\lambda)$ is positive-definite.

- $\|\tilde{S}^{\text{res}}(\lambda)\|^2_{\max} \leq \min_{\substack{i\neq j \\ |S_{ij}|\leq\lambda}} \frac{\lambda - |S_{ij}|}{\sqrt{S_{ii}S_{jj}}}$

Then the sparsity pattern of the optimal solution Θ corresponds to the spasity pattern of $S^{\text{res}}(\lambda)$ and, in addition, Θ can be obtained via the following formula:

$$\Theta_{ij} = \begin{cases} \frac{1}{S_{ii}}\left(1 + \sum_{(i,m)\in\mathcal{E}} \frac{(S^{\text{res}}_{im}(\lambda))^2}{S_{ii}S_{mm} - (S^{\text{res}}_{im}(\lambda))^2}\right), & \text{if } i = j \\ \frac{-S^{\text{res}}_{ij}(\lambda)}{S_{ii}S_{jj} - (S^{\text{res}}_{ij}(\lambda))^2} & \text{if } (i,j) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where $\mathcal{E}$ is the edge set of $\text{supp}(S^{\text{res}}(\lambda))$.

Proofs are given in [10].

3.2 Non-Likelihood Based Estimators

More recently, non-likelihood based estimators have been proposed for the Graphical Lasso and the Joint Graphical Lasso, driven by the need for more scalable algorithms. Here we briefly mention a few of these proposals to provide a broader context on work being done on the Graphical Lasso; they are not the focus of this project.

3.2.1 CLIME

'CLIME' stands for *constrained l_1-minimization for inverse covariance estimation* and was introduced in [11]. This method proposes to optimize the objective:

$$\begin{aligned} &\min_{\Theta} \|\Theta\|_1 \\ &\text{subject to } |S\Theta - I|_\infty \leq \lambda \end{aligned} \tag{3.2}$$

where λ is a hyperparameter. The intuition is clear, for we want:

1. Θ to be close to an inverse of S, which is achieved by the constraint $|S\Theta - I|_\infty \leq \lambda$.
2. Θ to be sparse, which is achieved by choosing to minimize $\|\Theta\|_1$.

Since the solution Θ of this problem is not necessarily symmetric, the authors propose to symmetrize it by taking the pointwise maximum of Θ with its transpose; they show that this matrix is positive definite with high probability. To solve the CLIME objective 3.2, note that the problem is separable in the columns of Θ. If Θ_i is the i-th column of Θ and e_i is the i-th standard basis vector, we want:

$$\begin{aligned} &\min \|\Theta_i\|_1 \\ &\text{subject to } |S\Theta_i - e_i|_\infty \leq \lambda \end{aligned} \tag{3.3}$$

for all $1 \leq i \leq n$. These are just p linear programs.

3.2.2 Elementary Estimator (EE)

Similar in spirit to CLIME [11], the Elementary Estimator (EE) introduced in [12] proposes the following estimator, which can be solved in closed form: Let $\rho_\nu(x)$ be any thresholding operator, such as the soft thresholding operator 1.12. Let $T_\nu(S)$ be the result of pointwise thresholding the elements of the matrix S. Then, if $T_\nu(S)$ is invertible, the Elementary Estimator is given by the solution to the problem:

$$\min_{\Theta} \|\Theta\|_{1,\text{off}} \tag{3.4}$$

$$\text{subject to } \|\Theta - T_\nu(S)^{-1}\|_{\infty,\text{off}} \leq \lambda \tag{3.5}$$

($\|\cdot\|_{\cdot,\text{off}}$ stands for the norm applied to the off-diagonal elements only). Here ν is a hyperparameter that makes $T_\nu(S)$ invertible. λ controls sparsity as usual. The solution is closed form, as we just have to soft-threshold $T_\nu(S)^{-1}$ by λ.

3.2.3 FASJEM

'FASJEM' stands for a *fast and scalable joint estimator for multi-sGGM* and was introduced in [13]. FASJEM adapts the Elementary Estimator to the multi-class setting, considering the following optimization objective:

$$\begin{aligned}\min_{\Theta=(\Theta^{(1)},\dots,\Theta^{(K)})} &\sum_{k=1}^{K}\|\Theta^{(k)}\|_1 + \epsilon P(\Theta^{(1)},\dots,\Theta^{(K)}) \\ \text{subject to } &\|\Theta^{(k)} - T_\nu(S^{(k)})^{-1}\|_\infty \le \lambda \quad \forall k \\ &P^*(\Theta^{(1)} - T_\nu(S^{(1)})^{-1},\dots,\Theta^{(K)} - T_\nu(S^{(K)})^{-1}) \le \epsilon\lambda\end{aligned}$$

Here P is a norm encouraging the Θ matrices to be similar (as the penalty P in the JGL equation 2.1), and P^* is its dual norm. The authors consider the cases where P the group lasso in GGL [7], leading to the FASJEM-G model, and the group infinity norm, leading to the FASJEM-I model. The use a proximal algorithm to optimize the objective, and introduce an efficient GPU implementation. The authors show [13] that each iteration of their optimization procedure has cost $\mathcal{O}(Kp^2)$, in constrast to $\mathcal{O}(Kp^3)$ of the GGL [7].

3.2.4 D-Trace Estimator

In [14] the Gaussian log-likelihood is swapped out for a simple quadratic function in Θ:

$$\min_{\Theta\succ 0} \frac{1}{2}\left\langle \Theta^2, S\right\rangle - \text{trace}(\Theta) + \lambda\|\Theta\|_1 \tag{3.6}$$

note that when $\lambda = 0$ and S is invertible, the minimum occurs precisely at $\Theta = S^{-1}$; hence this objective has similar properties to the Graphical Lasso objective, yet does not have a log determinant term in it. This makes the above estimator mathematically convenient. The authors go on to show an alternating directions method to solve this problem. The estimator is coined 'D-Trace' because $\frac{1}{2}\left\langle \Theta^2, S\right\rangle - \text{trace}(\Theta)$ is the difference of two traces.

A D-Trace-inspired estimator is proposed later in [15] to learn the difference Δ between two networks; let the empirical covariance matrices of the two classes be S_1 and S_2, then we optimize:

$$\min_{\Delta} \frac{1}{4}(\langle S_1\Delta, \Delta S_2\rangle + \langle S_2\Delta, \Delta S_1\rangle) - \langle \Delta, S_1 - S_2\rangle + \lambda\|\Delta\|_1$$

When $\lambda = 0$, the minimum is attained at $\Delta = S_2^{-1} - S_1^{-1}$.

Chapter 4

Experiments

4.1 Synthetic Data Experiment

We reproduce Figure 2 and Table 1 in the Joint Graphical Lasso (JGL) paper [7]. The details follow, which are cited from [7], with some remarks of our own.

4.1.1 Data Generation

Network Generation

A three-class problem ($K = 3$) is considered. There are $p = 500$ features and $n = 150$ observations (so, $p >> n$). The features belong to 10 equally sized unconnected subnetworks of 50 features each. In [7] each subnetwork is generated using a power law degree distributions, but details on how to generate the network are not given, so instead we generate similar-looking networks with the following recursive algorithm: we choose the degree of the root node of the network uniformly at random in the set $[3, 4, 5]$, then create equally-sized subtrees hanging from the root recursively.

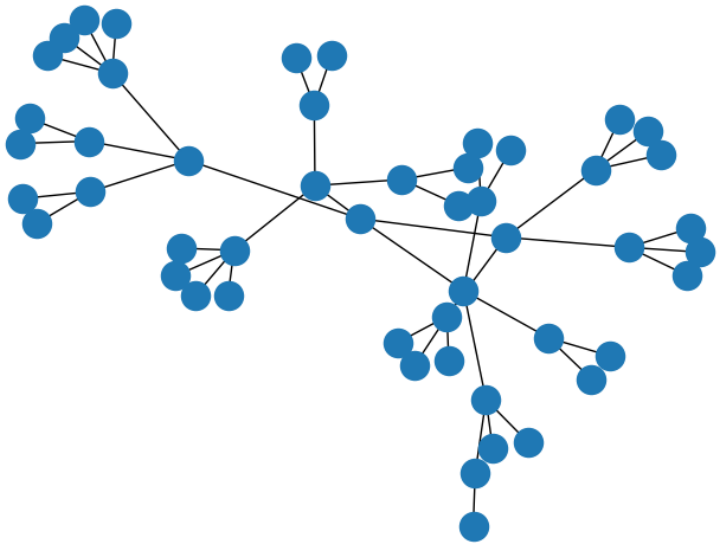

Figure 4.1: One of 10 random subnetworks on 50 nodes

To create the network for each class, we start by creating a common network composed of 10 subnetworks of size 50 nodes each. Then, to make the networks for the classes differ, in network 3 we remove all edges in one of the subnetworks, and then in networks 2 and 3 we remove all the edges in another subnetwork. The

following figure, borrowed from [7] shows the overall structure of the three networks underlying the three classes:

Figure 4.2: Network used to generate simulated data sets: full edges are common to all three classes; broken edges are present only in classes 1 and 2, and dotted edges are present only in class 1

Covariance Matrix Generation

Now that we have the network structure for the 3 classes, we must generate covariance matrices whose inverses obey these network structures. We first generate the covariance matrix for class 1. We start with a 500×500 matrix Θ with 1s on the diagonal, 0s on elements not corresponding to network edges, and values from a uniform distribution with support on $\{[-0.4, -0.1] \cup [0.1, 0.4]\}$ on elements corresponding to edges from the first class. So far the matrix Θ is neither symmetric not positive definite. To ensure positive definiteness we divide each off-diagonal element of Θ by 1.5 times the sum of the absolute values of off-diagonal elements in its row. Now we make Θ symmetric by averaging it with its own transpose. This results in our symmetric, positive-definite Θ. We could obtain Σ by just inverting Θ, however [7] instead define:

$$\Sigma_{ij} = d_{ij} \times \frac{(\Theta^{-1})_{ij}}{\sqrt{(\Theta^{-1})_{ii}(\Theta^{-1})_{jj}}}$$

where $d_{ij} = 0.6$ if $i = j$ and $d_{ij} = 1$ if $i = j$. Finally, we create the covariance matrices for the classes 2 and 3 by resetting one shared block and a class 3 specific block to the identity (thus matching the networks generated above).

Having defined the covariance matrices, we finally draw iid Multivariate Normal samples from each class, centered at 0.

4.1.2 Models

The following models are considered:

- Fused Graphical Lasso (FGL): As per 2.5. We use the R `JGL` package by [7].
- Group Graphical Lasso (GGL): As per 2.4. We use the R `JGL` package by [7].

- Graphical Lasso (GL): We train the Graphical Lasso 1.2 on each class separately. We use the python package `skggm` which implements the QUIC Graphical Lasso.
- Guo *et al.*: As per 2.2. Software is not available for this model, so we implement it ourselves leveraging the QUIC Graphical Lasso solver from `skggm` for inner loop optimization.

4.1.3 Metrics

The following metrics are considered:

- Test Negative Log-Likelihood (KL-Divergence): The KL divergence from the estimated distribution $N(0, (\hat{\Theta}^{(k)})^{-1})$ to the real distribution $N(0, \Sigma^{(k)})$, taken over the three distributions:

$$\text{dKL} = \frac{1}{2}\sum_{k=1}^{K}[-\log\det(\hat{\Theta}^{(k)}\Sigma^{(k)}) + \text{trace}(\hat{\Theta}^{(k)}\Sigma^{(k)})]$$

  Smaller KL-Divergence is better.

- True Positive Edges (TPE): Of the edges predicted by the estimated precision matrix, how many are true edges, i.e. $\#\{(k,i,j)|i<j, 1\le k\le 3, \hat{\Theta}^{(k)}_{ij}\neq 0, ((\Sigma^{(k)})^{-1})_{ij}\neq 0\}$. (Note that they are aggregated over the three networks.)
- False Positive Edges (FPE): Of the edges predicted by the estimated precision matrix, how many are *not* true edges, i.e. $\#\{(k,i,j)|i<j, 1\le k\le 3, \hat{\Theta}^{(k)}_{ij}\neq 0, ((\Sigma^{(k)})^{-1})_{ij}= 0\}$. (Note that they are aggregated over the three networks.)
- True Positive Differential Edges (TPDE): An edge (i,j) is said to be *differential* between two classes k,k' if $((\Sigma^{(k)})^{-1})_{ij}\neq((\Sigma^{(k')})^{-1})_{ij}$. Ideally, an edge (i,j) is predicted to be differential between two classes k,k' if $\hat{\Theta}^{(k)}_{ij}\neq\hat{\Theta}^{(k')}_{ij}$. However, $\hat{\Theta}^{(k)}_{ij}, \hat{\Theta}^{(k')}_{ij}$ might be close but not identical. This is likely the case for all models except the Fused Graphical Lasso. Because of this, the notion of predicted differential edges is relaxed in [7] to be $|\hat{\Theta}^{(k)}_{ij}-\hat{\Theta}^{(k')}_{ij}|>10^{-2}$. With this definition, TPDEs are those predicted to be differential which are actually differential:

$$\text{TPDE} = \#\{(k,k',i,j)|1\le k<k'\le 3, i<j, |\hat{\Theta}^{(k)}_{ij}-\hat{\Theta}^{(k')}_{ij}|>10^{-2}, ((\Sigma^{(k)})^{-1})_{ij}\neq((\Sigma^{(k')})^{-1})_{ij}\}$$

  The condition $|\hat{\Theta}^{(k)}_{ij}-\hat{\Theta}^{(k')}_{ij}|>10^{-2}$ is replaced by $\hat{\Theta}^{(k)}_{ij}\neq\hat{\Theta}^{(k')}_{ij}$ for FGL.

- False Positive Differential Edges (FPDE): Of those edges predicted to be differential, how many are actually not differential:

$$\text{FPDE} = \#\{(k,k',i,j)|1\le k<k'\le 3, i<j, |\hat{\Theta}^{(k)}_{ij}-\hat{\Theta}^{(k')}_{ij}|>10^{-2}, ((\Sigma^{(k)})^{-1})_{ij}=((\Sigma^{(k')})^{-1})_{ij}\}$$

  The condition $|\hat{\Theta}^{(k)}_{ij}-\hat{\Theta}^{(k')}_{ij}|>10^{-2}$ is replaced by $\hat{\Theta}^{(k)}_{ij}\neq\hat{\Theta}^{(k')}_{ij}$ for FGL.

4.1.4 Results: Reproducing Figure 2

Table 2 in [7] compares the performance on the above metrics of the four models across several settings of their hyperparameters.

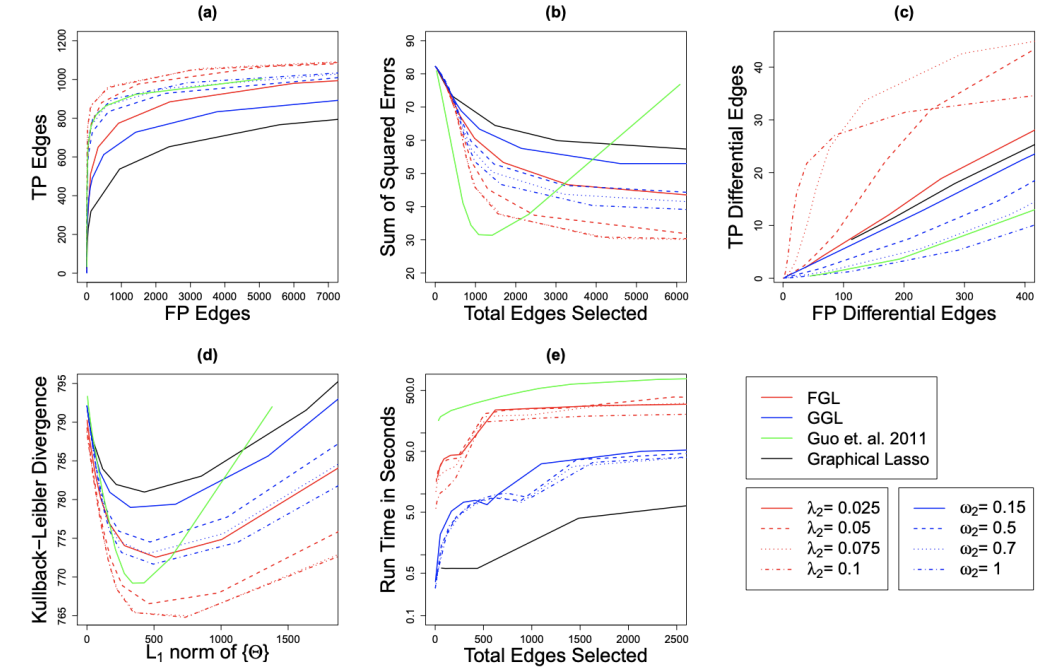

Figure 4.3: Reported

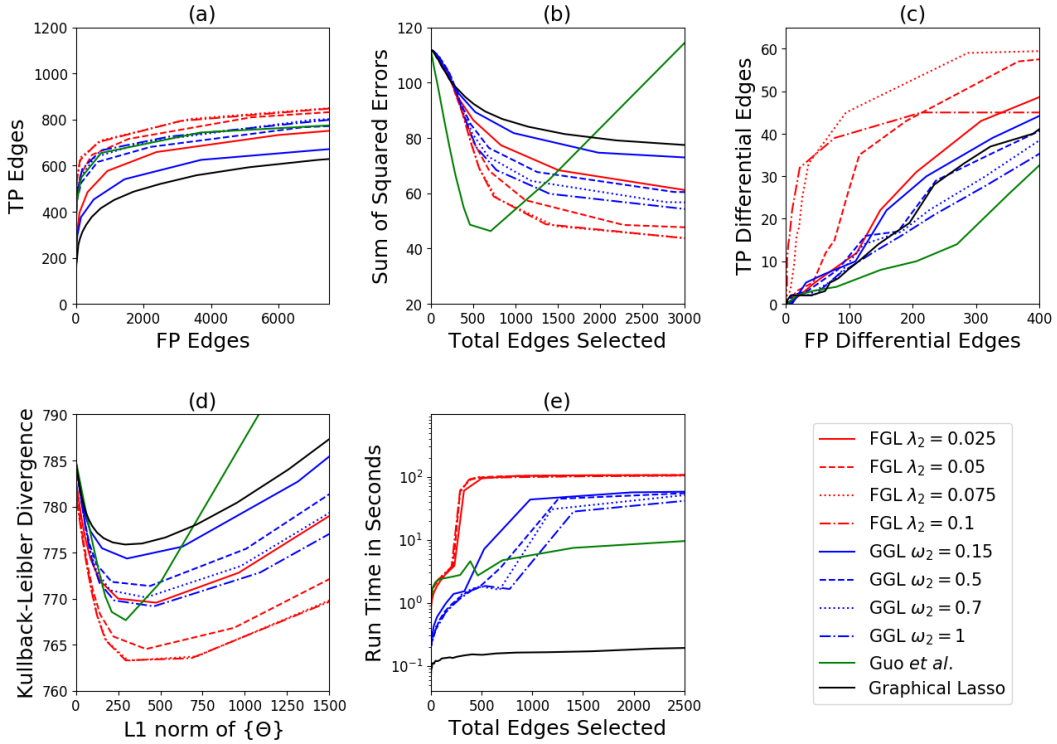

Figure 4.4: Reproduced

Analysis

We were able to reproduce the results reported in Figure 2 in the paper [7] quite closely. The axes were scaled slightly to match each reported plot. Possible differences between our plots and those reported might be because:

- The graph we generated does not exactly follow the protocol of [7] (since they do not provide the exact protocol in the paper),
- The results are obtained from only *one* simulation - and are not an average - hence there is sampling noise.

One plot where our axis differs substantially from the reported one is plot (b), where our x-range is half as long as the one reported. We conjecture that in [7] edges might have been double counted by considering the total number of non-zero off-diagonal edges in the precision matrices, rather than half this quantity. This is, however, only one hypothesis, and we cannot know because no code is provided by [7] to reproduce their results.

From the plots we can see that the Fused Graphical Lasso has the best modeling power in terms of KL-Divergence and edge discovery; however, it is also the slowest method. The approach of Guo *et al.* is competitive in the sparse regime (high λ). There is one remarkable difference between our results and those reported, which is the execution time of the Guo *et al.* model: our implementation is two orders of magnitude faster than that used in [7]. It turns out that Joint Graphical Lasso paper [7] used software provided by the authors of the Guo *et al.* paper [5] to run their model. However, the Guo *et al.* model [5] was published in February 2011, just months before the publication of the second-order QUIC Graphical Lasso method [16] in NeurIPS in December 2011, which greatly improved the state of the art in solving the Graphical Lasso problem. Because the Guo *et al.* model builds directly on top of the Graphical Lasso solver, it benefits from this improved solver (unlike the Joint Graphical Lasso proposal from [7], which relies on ADMM). Therefore, by using the QUIC Graphical Lasso solver as the underlying solver in the Guo *et al.* model, we are able to obtain a two fold increase in speed. Finally, the Joint Graphical Lasso paper [7] was submitted on October 2011, just one month before QUIC was published. By the time the Joint Graphical Lasso paper [7] was published in 2014, QUIC was already popular, but the experiments of [7] did not use the QUIC implementation in their experiments. Because of this unfortunate timing, the Guo *et al.* model does not look very competitive in the Joint Graphical Lasso paper, but as our results show, it is actually very competitive, particularly in the sparse regimes where it has good metrics.

4.1.5 Results: Reproducing Table 1

For each model hyperparameter selection was performed in [7] using an approximate Akaike Information Criterion (AIC). The approximate AIC is defined as[1]:

$$\text{AIC}(\lambda) = \sum_{k=1}^{K} [n_k \, \text{trace}(S^{(k)} \hat{\Theta}_{\lambda}^{(k)}) - n_k \log \det(\hat{\Theta}_{\lambda}^{(k)}) + 2E_k] \tag{4.1}$$

where E_k is the number of non-zero edges in $\hat{\Theta}_{\lambda}^{(k)}$. The values of hyperparameters λ minimizing the AIC were chosen as the final model.

Reported

Method	Tuning Parameters	AIC	dKL	TPE	FPE	TPDE	FPDE
FGL	$\lambda_1 = 0.175, \lambda_2 = 0.025$	1465	774	884	2406	77	4977
GGL	$\omega_1 = 0.225, \omega_2 = 1$	1470	776	898	736	53	1456
Graphical Lasso	$\lambda = 0.2$	1471	781	766	5578	85	11609
Guo *et al.* (2011)	$\lambda = 0.4$	1338	791	1003	5080	89	8992

Figure 4.5: Results reported in [7]; average over 100 repetitions.

Reproduced

Method	Tuning Parameters	AIC	dKL	TPE	FPE	TPDE	FPDE
FGL	$\lambda_1 = 0.175, \lambda_2 = 0.025$	1481	769	617	2353	74	5027
GGL	$\omega_1 = 0.225, \omega_2 = 1$	1484	771	615	645	48	722
Graphical Lasso	$\lambda = 0.2$	1484	775	558	5115	79	8008
Guo *et al.* (2011)	$\lambda = 0.04$	1364	794	712	3643	82	6237

Figure 4.6: Results reproduced by us; average over 100 repetitions.

Remark. Note that for the GGL model, the hyperparameters $\omega_1 = 0.225, \omega_2 = 1$ correspond to values of $\lambda_1 = 0, \lambda_2 = 0.318$ respectively. This is not mentioned explicitly in [7] but follows from their equations. This is interesting because is means that the L_1 Graphical Lasso regularizer is dropped from the GGL penalty 2.4, and all that remains is the group lasso penalty.

Analysis

We found that the value of $\lambda = 0.4$ reported for the Guo *et al.* model was too high and produced an empty graph for us. Since no open source implementation is available for the Guo *et al.* model, nor any code is provided for reproducing the results in [7] we cannot pinpoint the reason. However, using a value of $\lambda = 0.04$ yielded very similar results to those reported, so we used this value of λ instead. This aside, all metrics align quite well to those reported. Possible reasons for differences include those mentioned in the previous section 4.1.4. Another caveat we found is that the AIC reported seems to have been normalized by the sample size (150), which is not the same as the equation 4.1 reported in [7].

We can see that the joint modeling approaches FGL and GGL proposed in [7] have the highest training log-likelihood (i.e. lowest KL-Divergence), indicating that they are better probabilistic models for the true underlying data distribution. In terms of network discovery, the selected GGL model (based on AIC) is the most conservative, but yields the lowest false discovery rate for edges and differential edges. We note that in an exploratory data analysis application, selection based on AIC is not neccesarily a good idea, as values of the hyperparameters that produce sparser networks might be preferred to reduce the false discovery rate.

[1] It is approximate because constant summands involving π have been dropped

Finally, we also found that AIC might not be an ideal metric to perform model selection for the Guo et al. model: the Guo *et al.* model selected has a very poor KL-Divergence. However, when trained with a value of $\lambda = 0.06$. the Guo *et al.* model achieves a KL-Divergence as low as 767, which is very competitive; this corresponds to the lowest point of the green curve in figure 4.4 (d). We found that lower values of λ, such as $\lambda = 0.02$, produce an *even lower* AIC for the Guo *et al.* model, as low as 1107, yet a KL-Divergence of 1079. We believe the authors of JGL [7] did not find this model during their search because for this value of λ, the model takes a lot more time to train: around 10 minutes with our very fast implementation leveraging QUIC. Therefore, it would have taken hours for the the authors of JGL [7] to train the model, as they did not have access to the QUIC solver. We conjecture that the value of λ they found is the lowest they tried. The fact that this Guo *et al.* model achieves such a low AIC yet poor generalization means that the model overfits the training data easily without using too many parameters (i.e. non-zero edges). This makes sense, since the Guo *et al.* penalty encourages shared zeros across the inferred precision matrices, *while otherwise not penalizing large values in the precision matrices.* It would be interesting to investigate this in more detail and confirm (or debunk) our suspicions.

4.1.6 Conclusion

The main finding of our reproducibility efforts is that the method of Guo *et al.* is a lot more competitive than reported in the JGL paper [7]. Due to the unfortunate timing of events, the JGL authors did not have knowledge of the QUIC Graphical Lasso algorithm at the time of submitting their manuscript (although when it was finally published 2 years later, QUIC was widely available).

A natural question stemming from our reproducibility efforts is whether there is some way to get the best of both worlds:

- The great modeling power of the Fused Graphical Lasso [7]
- Fast learning algorithms by reduction to the simple weighted Graphical Lasso as in Guo *et al.* [5]

Reduction of the Joint Graphical Lasso to the simple weighted Graphical Lasso seems to be a neat idea, as it allows the Joint Graphical Lasso solver to leverage any algorithmic improvements developed for solving the simple weighted Graphical Lasso.

4.1.7 Code

The code to reproduce Figure 2 and Table 1 in the paper [7] are provided under the `src` directory. The main entry point is `src/python/main.py`. Run it with `$ python main.py`. The resulting figures will be found under `src/python/reproduce/report`. It takes us about 7 hours to run these experiments. `R` must be installed as well as its `JGL` package. The required python packages are found in `src/python/requirements.txt`. You can make sure everything is set up by first doing a mock run with `$ python main.py --test`, which takes only about 10 minutes.

4.2 Application to biological data

The central dogma of molecular biology describes how genetic information flows from DNA segments called genes which are transcribed to RNA and then translated to protein. Proteins do most of the work inside of the cell; for example, hemoglobin transports oxygen and keratin has a structural role in hair. Interestingly, some proteins known as transcription factors regulate the production of other proteins by modulating the transcription of their respective genes. The field of gene regulatory network inference (see [17] for an introduction) aims to uncover regulatory relationships from the covariance of gene expression, and the Graphical Lasso has been one of the main methods in the field.

Recently, a new single-cell RNA-seq dataset was published [18] consisting of gene expression measurements of hundreds of thousands of cells from mice across different stages of aging (see Fig. 4.7). This dataset presents various opportunities for joint modeling of regulatory networks with the Joint Graphical Lasso. For example, learning joint regulatory networks across different tissues or cell types. Here, we decided to apply the Joint Graphical Lasso to jointly model regulatory networks across ages, for specific cell types, which is a novel opportunity offered by this dataset.

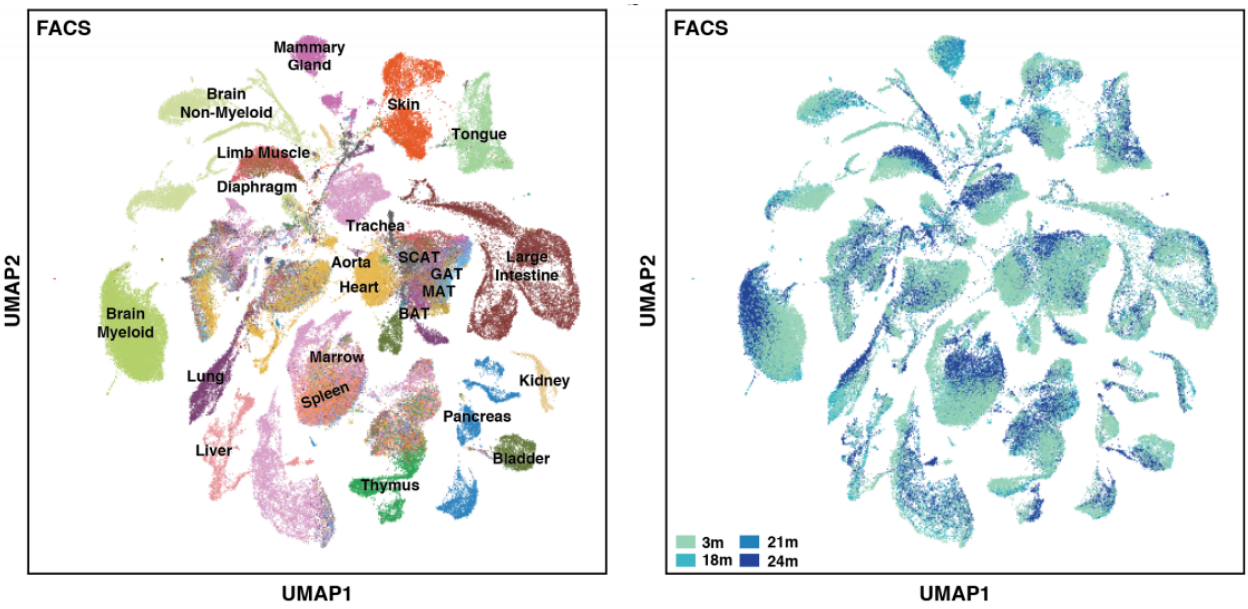

Figure 4.7: The Tabula Muris Senis dataset. Dimensionality reduction of gene expression in hundreds of thousands of cells with UMAP. On the left, cells are colored by tissue of origin. On the right, cell are colored by mouse age in months. (Fig. from [18])

4.2.1 Modeling gene expression in a single class of cells with the Graphical Lasso

We decided to start by applying the simple Graphical Lasso to infer a sparse precision matrix from gene expression in a homogeneous group of cells and compare the results to the empirical, maximum-likelihood precision matrix. We focused on 2253 cells from the young mice, in particular microglial cells, which play a role in immunity in the brain. For this first experiment we focused on the top thousand genes with the highest variance in this group of cells. Gene expression was standardized, as described in the original Joint Graphical Lasso paper [7].

To study the importance of the regularization hyperparameter of the Graphical Lasso, we split the data into training (70%) and test (30%) sets and evaluated the log-likelihood over a range of hyperparameter values (Figure 4.8). As we lower the sparsity penalty the training set log-likelihood goes up. This is expected since as the hyperparameter goes to zero the optimization objective approaches the log-likelihood.

However, the test set log-likelihood is maximized with the hyperparameter set around 0.08. This means that this particular sparsity penalty yields a model that generalizes better than both the maximum likelihood estimator (hyperparameter equal to 0) and a model in which all genes are independent (hyperparameter set very high).

We then compared the empirical inverse covariance to the inverse covariance inferred by the Graphical Lasso with optimal sparsity level (chosen with cross-validation) (Fig. 4.9). We see the empirical inverse covariance is dense, while the Graphical lasso inverse covariance - which we have seen is a better model of gene expression - is in fact very sparse.

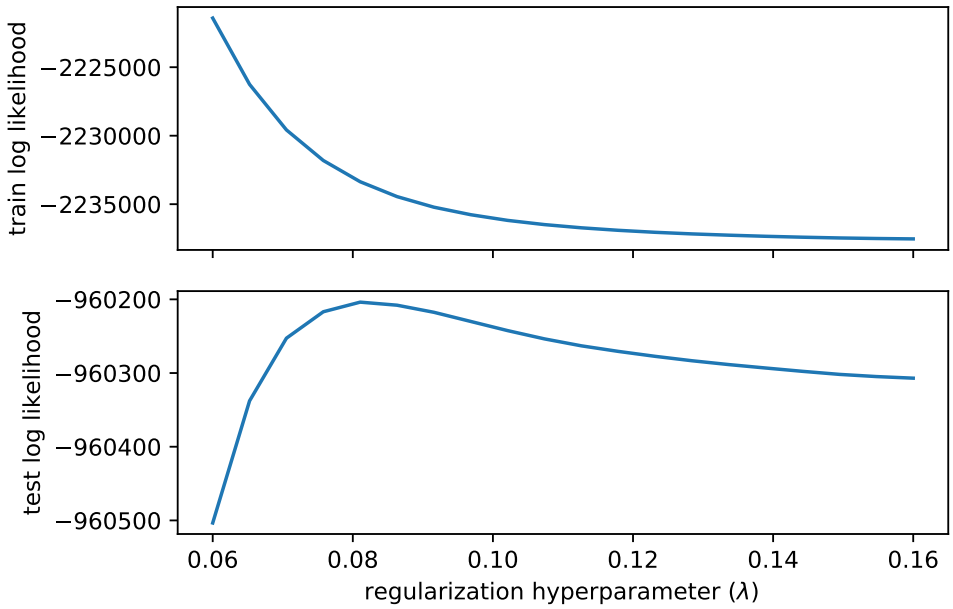

Figure 4.8: Log likelihood evaluated on train (top) and test (bottom) gene expression data from microglial cells, as a function of the regularization hyperparameter λ

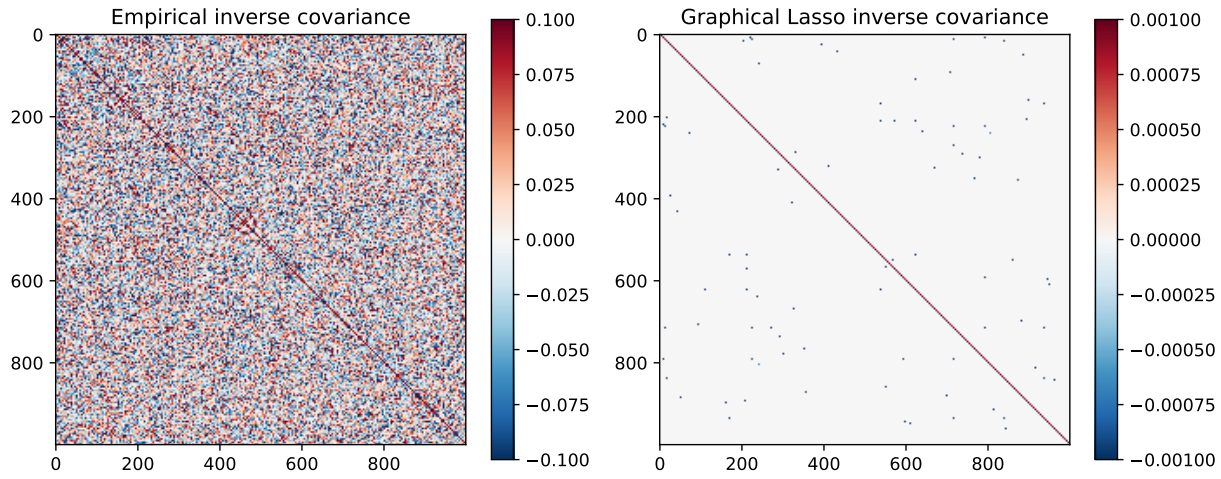

Figure 4.9: Inverse covariances for expression of 1000 genes in 2253 microglial cells from young mice. On the left, empirical inverse covariance. On the right, inverse covariance inferred with the Graphical Lasso. Genes on the axis.

4.2.2 Modeling gene expression in two distinct classes of cells with the Joint Graphical Lasso

When we visualize the gene expression of microglial cells, we see that there is a shift in distribution between the young and old cells (Fig 4.10). We expect young and old cells to share a lot of biochemical pathways, such as those involved in metabolism and cell division, while having specific differences due to aging, such as more activity related to DNA repair and tumor control. We decided to use the Joint Graphical Lasso to jointly model the expression of 2000 genes in two classes of cells: 2243 young cells and 4268 old cells.

We did not find a good implementation for tuning the two hyperparameters for the Joint Graphical Lasso, which is not trivial, and we did not explore much on that front. For the current analysis we chose a sparsity penalty of 0.08 which was optimal for the previous simple Graphical Lasso analysis, and a between-classes fused lasso penalty of 0.05. Gene expression in each class was standardized, as performed in the original Joint Graphical Lasso paper [7]. In order to obtain confidence intervals around the estimates of the precision matrix, 50 bootstrap resampling steps were performed.

The general structure of the learned networks is the following:

- 19197 $\pm$ 777 edges are shared by young and old cells
- 623 $\pm$ 100 edges are unique to young cells
- 1086 $\pm$ 93 edges are unique to old cells

which suggests a majority of shared regulatory structure with non-negligible age-specific interactions.

We then decided to look at the top ten pairs of genes with the largest change in partial correlation across aging (Fig. 4.11). The partial correlation ρ, the degree of association between two variables when all other variables have been controlled for, can be obtained from a precision matrix Θ as follows:

$$\rho_{ij} = -\frac{\Theta_{ij}}{\sqrt{\Theta_i \Theta_j}}$$

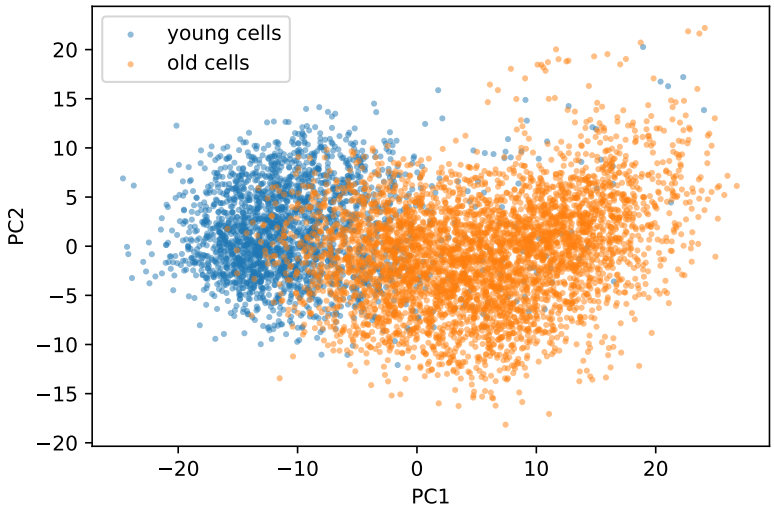

Figure 4.10: Principal Component Analysis of gene expression in young and old microglial cells from the brain

We specifically ranked pairs of genes by the strictly standardized mean difference, which is defined as the difference in means divided by the square root of the sum of the variances. We found significant differences in partial correlation between these pairs of genes, and many combinations: partial correlations that decrease or increase with age, and partial correlations that reduce in absolute value or just go to exactly zero. Two genes are over represented in our list of top 10 interactions most affected by aging: Jun and Lars2. Jun is an oncogene, that is, a gene with the potential to cause cancer. In fact, it is the first oncogene to be discovered to be a transcription factor [19], that is, a gene which regulates the expression of other genes. The general pattern we see in Fig. 4.11 is Jun going from having positive partial correlation with other genes to having close to zero partial correlation in old age. This could mean a loss of regulatory power of the Jun gene, potentially because of accumulated mutations through the aging process. Lars2 encodes for an enzyme which catalyzes a chemical modification of transport RNA, a key molecule for protein synthesis. It has recently been found to promote mammary tumor formation [20], but it's not clear what this change in its interactions would mean in the context of the brain tissue.

4.2.3 Future work

An important step of practical importance would be to make available a scalable implementation of the Joint Graphical Lasso, including hyperparameter optimization. It will be interesting to analyze changes in regulatory networks in other cell types relevant to aging, such as fat tissue and heart. We hope that covariance analysis can help with hypothesis generation, but it is evident that a lot of in-depth study of the relationship between specific genes is necessary to make progress understanding complex biological processes such as aging.

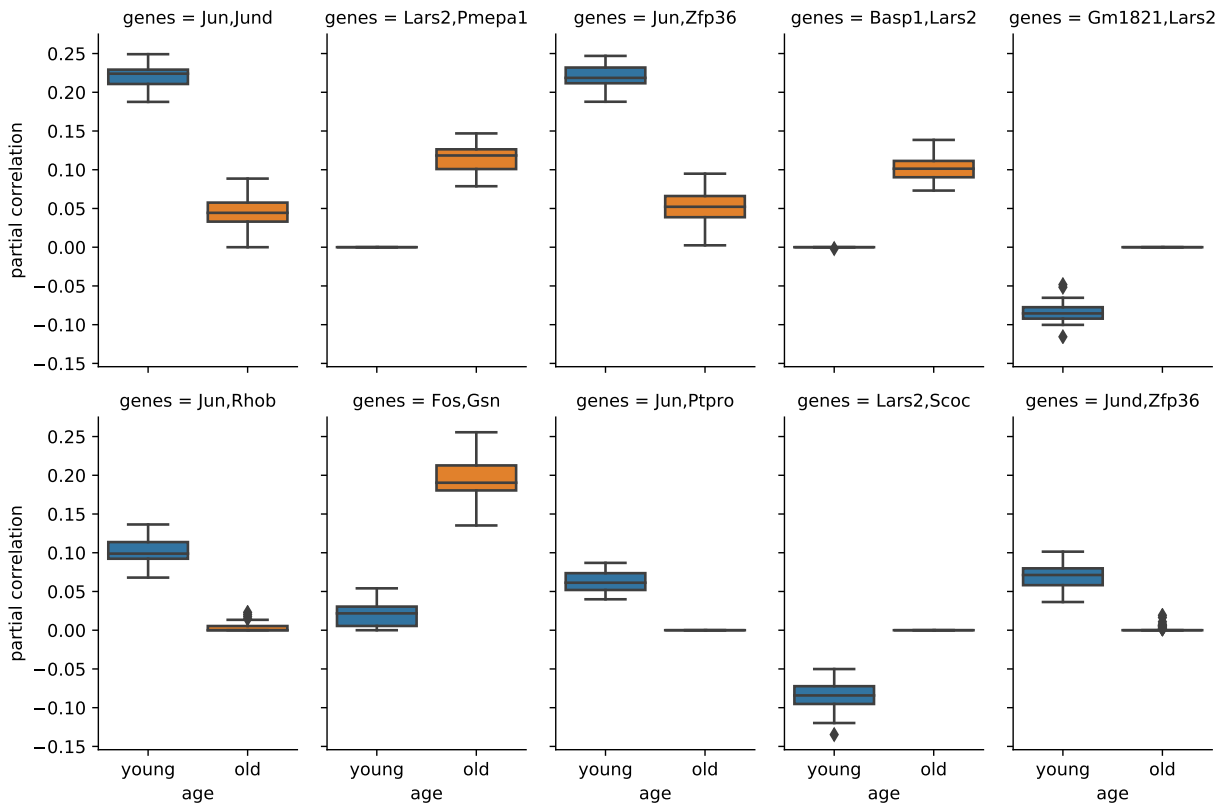

Figure 4.11: Partial correlation of 10 pairs of genes in young and old microglial cells from the brain

Bibliography

[1] 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, June 2008.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics (Oxford, England)*, 9:432–41, 08 2008.

[3] Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1, 09 2007.

[4] Daniela M. Witten, Jerome H. Friedman, and Noah Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.

[5] J. Guo, E. Levina, G. Michailidis, and J. Zhu. Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15, Mar 2011.

[6] Group Variable Selection via a Hierarchical Lasso and Its Oracle Property. Weighted fused pathway graphical lasso for joint estimation of multiple gene networks. *arXiv*, 2010.

[7] Patrick Danaher, Pei Wang, and Daniela Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 76:373–397, 03 2014.

[8] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[9] Somayeh Sojoudi. Equivalence of graphical lasso and thresholding for sparse graphs. *Journal of Machine Learning Research*, 17(115):1–21, 2016.

[10] Salar Fattahi and Somayeh Sojoudi. Graphical lasso and thresholding: Equivalence and closed-form solutions. *Journal of Machine Learning Research*, 20(10):1–44, 2019.

[11] T. Cai, Weidong Liu, and Xi Luo. A constrained l1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106, 02 2011.

[12] Eunho Yang, Aurelie C. Lozano, and Pradeep Ravikumar. Elementary estimators for graphical models. In *NIPS*, pages 2159–2167, 2014.

[13] Beilun Wang, Ji Gao, and Yanjun Qi. A Fast and Scalable Joint Estimator for Learning Multiple Related Sparse Gaussian Graphical Models. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1168–1177, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[14] Teng Zhang and Hui Zou. Sparse precision matrix estimation via lasso penalized D-trace loss. *Biometrika*, 101(1):103–120, 02 2014.

[15] Huili Yuan, Ruibin Xi, Chong Chen, and Minghua Deng. Differential network analysis via lasso penalized D-trace loss. *Biometrika*, 104(4):755–770, 10 2017.

[16] Cho jui Hsieh, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Mátyás A. Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2330–2338. Curran Associates, Inc., 2011.

[17] Vân Anh Huynh-Thu and Guido Sanguinetti. *Gene Regulatory Network Inference: An Introductory Survey*, pages 1–23. Springer New York, New York, NY, 2019.

[18] Angela Oliveira Pisco, Nicholas Schaum, Aaron McGeever, Jim Karkanias, Norma F Neff, Spyros Darmanis, Tony Wyss-Coray, Stephen R Quake, et al. A single cell transcriptomic atlas characterizes aging tissues in the mouse. *bioRxiv*, page 661728, 2019.

[19] Peter K Vogt. Fortuitous convergences: the beginnings of jun. *Nature Reviews Cancer*, 2(6):465, 2002.

[20] Yongliang Huo, Timothy Su, Qiuyin Cai, and Ian G Macara. An in vivo gain-of-function screen identifies the williams-beuren syndrome gene gtf2ird1 as a mammary tumor promoter. *Cell reports*, 15(10):2089–2096, 2016.