UNIVERSITY OF CALIFORNIA
Los Angeles

Distributed Joint Inference of Graphical Models

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Gilbert Paul Neuner

2024

ABSTRACT OF THE THESIS

Distributed Joint Inference of Graphical Models

by

Gilbert Paul Neuner
Master of Science in Statistics
University of California, Los Angeles, 2024
Professor George Michailidis, Chair

This thesis introduces an algorithm for estimating Gaussian graphical models from multiple subpopulations sharing some dependence structure. The algorithm uses proximal gradient descent to estimate solutions to a neighborhood regression problem along with L1 and graph Laplacian penalty terms. The graph Laplacian penalty term induces similarity amongst neighborhood regression coefficients belonging to subpopulations known to share much dependence structure. Further, the algorithm can be distributed amongst agents and a server to ensure that agents do not share their datasets with each other. The distributed algorithm is equivalent to the non-distributed algorithm from an input-output perspective. The algorithm is compared with the graphical lasso on multiple synthetic datasets.

The thesis of Gilbert Paul Neuner is approved.

Qing Zhou

Arash Ali Amini

George Michailidis, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

1 Introduction **1**

1.1 Main Contribution 1

1.2 Roadmap 1

1.3 Notation 2

2 Background **7**

2.1 Graphical Models 7

2.2 Gaussian Graphical Models 8

2.3 Inference of Precision Matrix 9

2.4 Multiple Subpopulations 10

2.5 Directed Graphical Models 13

3 Non-Distributed Algorithm **16**

3.1 Optimization Problem 16

3.2 Application of Proximal Gradient Descent 18

3.3 Block Matrices 19

3.4 Statement of Non-Distributed Algorithm 19

3.5 High-Level Overview of Non-Distributed Algorithm 21

3.6 Step Size Scheme and Stopping Criteria 22

3.7 Post-Processing 23

4 Distributed Algorithm **24**

4.1 The Distributed Setting 24

4.2 Separating the Objective Function 25
4.3 Statement of Distributed Algorithm 26
4.4 Algorithmic Details 30

5 Numerical Results 31
5.1 Experimental Setups: 1-6 31
5.2 Experimental Setup: 7 35
5.3 Data Generation 37
5.4 Choice of Tuning Parameters 38
5.5 Results: Experiments 1-6 39
5.6 Results: Experiment 7 42

A Derivation of Equation 3.1 44

References 47

LIST OF FIGURES

5.1 Subpopulation graphs: Experiments 1-6 32

5.2 Venn diagrams: Experiments 1-6 34

5.3 Subpopulation graph: Experiment 7 35

5.4 Support of Ω, experiment 5 38

LIST OF TABLES

5.1 Summary of experiments 1-6 33

5.2 Results, experiments 1-6 40

5.3 F1 score by subpopulation, experiments 1-6 41

5.4 Normalized RMSE, experiments 1-6 42

5.5 Results, experiment 7 42

5.6 F1 score by subpopulation, experiment 7 43

CHAPTER 1

Introduction

1.1 Main Contribution

The main contribution of this thesis is developing a joint estimation method for neighborhood regression coefficients which takes into account shared dependence structure between subpopulations and which can be distributed so that agents need not share datasets.

1.2 Roadmap

- Chapter 1 provides an introduction for the thesis.
- Chapter 2 provides background on graphical models, particularly graphical models. It also reviews methods for inference of graphical models in both the single population and multiple subpopulation cases.
- Chapter 3 states the main optimization problem, derives the non-distributed algorithm as an instance of proximal gradient descent, and breaks down pseudocode for the non-distributed algorithm.
- Chapter 4 explains the distributed setting and formulates the distributed algorithm, which is equivalent to the non-distributed algorithm from an input-output perspective.
- Chapter 5 provides numerical results for the distributed algorithm and the graphical lasso in 7 different experimental settings.

- Appendix A computes the gradient of the differentiable part of the objective function, which is needed for proximal gradient descent.

1.3 Notation

Matrices

B_{ij}

If B is a matrix, then B_{ij} denotes the ij-th element of B.

$B_{\cdot i}$

If B is a matrix, then $B_{\cdot i}$ denotes the i-th column of B.

Lists of Matrices

$C^{(k)}$

Lists of matrices will be indexed by superscript (k). If C is a list of K matrices, then its elements would be written $C^{(1)}, \ldots, C^{(k)}, \ldots, C^{(K)}$.

C_{ij}

If C is a list of K matrices, then C_{ij} denotes the vector of length K consisting of $(C_{ij}^{(1)}, \ldots, C_{ij}^{(K)})$.

$(C)^{(s)}$

Iteration number of an algorithm will be indexed by superscript (s), outside of parentheses. If the iterations of the algorithm are divided into subparts, then s may be fractional.

combining notations

These notations are compatible. For example, $(C_{ij}^{(k)})^{(s)}$ denotes the ij-th element of the k-th element of the list of matrices C, having been updated by the s-th iteration

of an algorithm.

Block Matrices

block matrix

Let $D \in \mathbb{R}^{Kp \times Kp}$. D can be visualized as divided into K^2 blocks, where the ij-th block $D_{ij} \in \mathbb{R}^{p \times p}$ consists of the $[(i-1)p+1]$-th to ip-th rows and $[(j-1)p+1]$-th to jp-th columns of D. Observe that this overloads the notation for an element of a matrix.

block column of a block matrix

The notation for a column of a matrix is overloaded for block matrices. If $D \in \mathbb{R}^{Kp \times Kp}$ is a block matrix, then $D_{\cdot k}$ denotes the $[(k-1)p+1]$-th to kp-th columns of D.

block Hadamard product

If $B \in \mathbb{R}^{p \times p}$ is a matrix, and $D \in \mathbb{R}^{Kp \times Kp}$ is a block matrix, then $B \square D \in \mathbb{R}^{Kp \times Kp}$ is the block matrix where $(B \square D)_{ij} = B_{ij} D_{ij}$. Note that $B_{ij} \in \mathbb{R}$ but $D_{ij} \in \mathbb{R}^{k \times k}$.

Other Notation

In order of appearance, in section 2.4 and chapters 3, 4 and 5:

K

Number of subpopulations

k

Subpopulation index, $k \in \{1, \ldots, K\}$

n_k

Sample sizes

p

Dimension

Σ

Covariance matrices $\Sigma = \{\Sigma^{(1)}, \ldots, \Sigma^{(K)}\}$, $\Sigma^{(k)} \in \mathbb{R}^{p \times p}$

X

Data matrices $X = \{X^{(1)}, \ldots, X^{(K)}\}$, $X^{(k)} \in \mathbb{R}^{n_k \times p}$, rows drawn from $N(0, \Sigma^{(k)})$

W

Adjacency matrix of subpopulation graph $W \in \mathbb{R}^{K \times K}$

Θ

Correlation matrices $\Theta = \{\Theta^{(1)}, \ldots, \Theta^{(K)}\}$, $\Theta^{(k)} \in \mathbb{R}^{p \times p}$

$\hat{\Theta}$

Estimate of correlation matrices $\hat{\Theta} = \{\hat{\Theta}^{(1)}, \ldots, \hat{\Theta}^{(K)}\}$, $\hat{\Theta}^{(k)} \in \mathbb{R}^{p \times p}$

Ψ

Sample correlation matrices $\Psi = \{\Psi^{(1)}, \ldots, \Psi^{(K)}\}$, $\Psi^{(k)} \in \mathbb{R}^{p \times p}$

$\|\cdot\|_L$

graph Laplacian penalty $\|\cdot\|_L \colon \mathbb{R}^K \to \mathbb{R}$

β

Neighborhood regression coefficients $\beta = \{\beta^{(1)}, \ldots, \beta^{(K)}\}$, $\beta^{(k)} \in \mathbb{R}^{p \times p}$, $\beta_{ij}^{(k)} = -\Omega_{ij}^{(k)} / \Omega_{jj}^{(k)}$

$\hat{\beta}$

Estimate of neighborhood regression coefficients $\hat{\beta} = \{\hat{\beta}^{(1)}, \ldots, \hat{\beta}^{(K)}\}$, $\hat{\beta}^{(k)} \in \mathbb{R}^{p \times p}$

$\|\cdot\|_2$

Euclidean norm

λ

L1 parameter $\lambda > 0$

ρ

Laplacian parameter $\rho > 0$

$f(\beta; W, X, \lambda, \rho)$

Objective function

$g(\beta; W, X, \rho)$

Differentiable part of objective function

$h(\beta; \lambda)$

Proximable part of objective function

$S_\kappa(\cdot)$

Soft-thresholding operator

a, b

Step size parameters $a > 0$, $b \in (0, 1)$

$s_{\max}$, tol, $t_{\min}$

Parameters for checking stopping criteria

s

Iteration counter

t_s

Step size

$Y^{(k)}$

Matrix $Y^{(k)} \in \mathbb{R}^{p \times p}$, $Y_{qr} = \langle X_{\cdot q}^{(k)}, X_{\cdot r}^{(k)} \rangle$

$\| \cdot \|_F$

Frobenius norm

$g_0(W, \beta, \rho)$

Laplacian part of objective function

$g_k(X^{(k)}, \beta^{(k)})$

Neighborhood regression parts of objective function

$h_k(\beta^{(k)}, \lambda)$

Soft-thresholding parts of objective function

f_k

Part of objective function computed by agent

$e_\bullet$

Number of edges expected to be shared by the subpopulations in $\bullet$

$A^\bullet$

An Erdos-Renyi graph expected to have $e_\bullet$ edges

CHAPTER 2

Background

2.1 Graphical Models

The PhD thesis [Pla18] gives an excellent background on graphical models. Much of that information is applicable for this thesis, so the background given here borrows heavily from the background of [Pla18].

In the following definition, f is an arbitrary density function.

Let $X = \begin{bmatrix} X_1 & \dots & X_p \end{bmatrix} \in \mathbb{R}^{1\times p}$ be a random (row) vector with positive density. Let A, B, C be disjoint subsets of $\{1, \dots, p\}$. Then X_A and X_B are **conditionally independent** given X_C, written $X_A \perp\!\!\!\perp X_B \mid X_C$, if

$$f(X_A, X_B \mid X_C) = f(X_A \mid X_C) f(X_B \mid X_C).$$

Let $G = (V, E)$ be a graph, where $V = \{1, \dots, p\}$. Then we can associate each random variable X_i with the i-th vertex of G.

A graph $G = (V, E)$ is a **undirected graphical model** with respect to the random vector X if it satisfies one of the following **Markov properties**:

Global Markov Property (G)

$$\forall A, B, C \subseteq V : C \text{ separates } A \text{ and } B \implies X_A \perp\!\!\!\perp X_B \mid X_C.$$

Local Markov Property (L)

$$\forall i : X_i \perp\!\!\!\perp X_{V \setminus (\mathrm{ne}(i) \cup \{i\})} \mid X_{\mathrm{ne}(i)}$$

Pairwise Markov property (P)

$$\forall i, j : (i, j) \notin E, i \neq j \implies X_i \perp\!\!\!\perp X_j \mid X_{l, l \neq i, j}.$$

The Markov properties are subject to the following containments [Lau96]:

Proposition 1 (Markov property containments)**.** *For any probability distribution over an undirected graph G,*

$$(G) \implies (L) \implies (P).$$

Moreover, if the probability measure is strictly positive, then

$$(G) \iff (L) \iff (P).$$

In particular, if X is multivariate Gaussian, then $(G) \iff (L) \iff (P)$.

2.2 Gaussian Graphical Models

Let $X = \begin{bmatrix} X_1 & \dots & X_p \end{bmatrix} \in \mathbb{R}^p$ be multivariate normal, with covariance matrix Σ. Without loss of generality, it may be assumed that the mean is 0. Let $\Omega = \Sigma^{-1}$ be the **precision matrix** of X, which is symmetric.

Proposition 2 (adjacency and precision have same support)**.** *If $X = \begin{bmatrix} X_1 & \dots & X_p \end{bmatrix} \sim N(0, \Sigma)$ and $G = (V, E)$ is a graphical model with respect to X, then*

$$(i, j) \in E \iff \Omega_{ij} \neq 0.$$

This result is useful because it shifts the inferential target from the adjacency matrix of the graphical model of X to the precision matrix of X.

2.3 Inference of Precision Matrix

Methods for inferring the precision matrix of a Gaussian random vector may be categorized as **global** or **local**. While global methods recover Ω itself, local methods only recover the support of Ω. Local methods involve recovering the neighborhood set of each vertex.

All methods presented here assume that Ω is sparse. Without a sparsity assumption, Ω would not be recoverable in the high-dimensional setting. Also, the sparser Ω is, the more interpretable the corresponding graphical model is. The sparsity assumption is reflected by L1 penalization terms.

One global method is the **graphical lasso** [FHT07]. The graphical lasso is an algorithm for solving the optimization problem

$$\hat{\Omega} = \underset{\Omega \succeq 0}{\operatorname{argmax}} \left\{ \log\det\Omega - \operatorname{tr}(S\Omega) - \lambda \sum_{i \neq j} |\Omega_{ij}| \right\}.$$

In the above, S is the empirical covariance matrix, $\log\det\Omega - \operatorname{tr}(S\Omega)$ is the Gaussian log-likelihood of Ω, and $-\lambda \sum_{i \neq j} |\Omega_{ij}|$ is an L1 regularization term.

An alternative global method is given in [RBL08], which is based on the optimization problem

$$\hat{\Theta} = \underset{\Theta = \Theta^\top, \Theta \succ 0}{\operatorname{argmax}} \left\{ \log\det\Theta - \operatorname{tr}(\Psi\Theta) - \lambda \sum_{i \neq j} |\Theta_{ij}| \right\}.$$

In the above, Ψ is the sample correlation matrix and $\log\det\Theta - \operatorname{tr}(\Psi\Theta)$ is the Gaussian log-likelihood of the correlation matrix Θ. The **SPICE** estimator $\hat{\Theta}$ can be transformed to an estimate for Ω via $\hat{\Omega} = \hat{\Xi}^{-1}\hat{\Theta}\hat{\Xi}^{-1}$, where $\hat{\Xi}^2 = \operatorname{diag}(S)$.

By contrast, **neighborhood regression** [MB06] is a local method. Neighborhood re-

gression involves inferring $\beta \in \mathbb{R}^{p\times p}$ where $\beta_{ij} = -\Omega_{ij}/\Omega_{jj}$ and $\beta_{ii} = 0$. This new inferential target is useful because it has the same support as Ω and by extension the adjacency matrix of the graphical model.

Neighborhood regression is possible because of the following observation:

Proposition 3 (support of precision as regression coefficients)**.** *If* $x = \begin{bmatrix} X_1 & \dots & X_p \end{bmatrix} \sim N(0, \Sigma)$*, then any* X_j *given* $X_{l,l\neq j}$ *is Gaussian with*

$$\mathbb{E}[X_j \mid X_{l,l\neq j}] = -\sum_{i\neq j} \beta_{ij} X_i$$
$$\mathrm{Var}(X_j \mid X_{l,l\neq j}) = \Omega_{jj}^{-1}.$$

This, along with the sparsity assumption, is the setup for p lasso regression problems. Let $X \in \mathbb{R}^{n\times p}$ be the data matrix, where the rows are draws of $x = \begin{bmatrix} X_1 & \dots & X_p \end{bmatrix} \sim N(0, \Sigma)$. The optimization problem is

$$\hat{\beta} = \underset{\beta\in\mathbb{R}^{p\times p}, \mathrm{diag}(\beta)=0}{\mathrm{argmin}} \left\{ \frac{1}{2n} \sum_{j=1}^{p} \|X_{\cdot j} - \sum_{i\neq j} X_{\cdot i}\beta_{ij}\|_2^2 + \lambda \sum_{i\neq j} |\beta_{ij}| \right\}.$$

Neighborhood regression is an algorithm for solving the above optimization problem. However, there is no guarantee that $\hat{\beta}$ is symmetric. One solution recommended by [MB06] is to apply a post-processing step whereby the entries $\hat{A}_{ij}$ of the adjacency matrix $\hat{A} \in \{0,1\}^{p\times p}$ are set to 1 if $\hat{\beta} \neq 0$ and $\hat{\beta} \neq 0$.

2.4 Multiple Subpopulations

In the previous sections, the goal was to infer a graphical model from a single data matrix X. By contrast, suppose that there are multiple data matrices $X^{(k)} \in \mathbb{R}^{n_k\times p}$, $k = 1, \dots, K$, and the goal is to infer a graphical model for each of them. Moreover, it is believed that the

K graphical models share many common edges. A naive approach would be to apply one of the previous methods separately on each of the K data matrices. Better would be a joint method which could leverage the shared structure between the $X^{(k)}$.

For example, the K data matrices may correspond to different cancer subtypes. Being different kinds of cancer, it is expected that the K graphical models would each have some idiosyncratic edges; all being kinds of cancer, it is expected that the K graphical models would share some common edges.

A common strategy for dealing with multiple subpopulations is to extend one of the methods from the previous section, and add a penalty term inducing similarity between subpopulations. For example, [GLM11] and [DWW14] extend the graphical lasso by considering the likelihood

$$\frac{1}{n}\sum_{k=1}^{K} n_k \left(\log\det\Omega^{(k)} - \operatorname{tr}\left(S^{(k)}\Omega^{(k)}\right)\right) \tag{2.1}$$

where $n = \sum_{k=1}^{K} n_k$. To induce similarity between subpopulations, [GLM11] parametrizes

$$\Omega_{ij}^{(k)} = \delta_{ij}\gamma_{ij}^{(k)}$$

where $\delta_{ij} \geq 0$ is a common factor and $\gamma_{ij}^{(k)}$ is an idiosyncratic factor. Then the optimization problem involves the likelihood 2.1 and L1 penalties $\sum_{i\neq k}\delta_{ij}$ and $\sum_{i\neq j}\sum_{k=1}^{K}|\gamma_{ij}^{(k)}|$.

The optimization problem of [DWW14] involves the likelihood 2.1 and penalties $\sum_{k=1}^{K}\sum_{i\neq j}|\Omega_{ij}^{(k)}|$ and $\sum_{i\neq j}\sqrt{\sum_{k=1}^{K}\Omega_{ij}^{(k)^2}}$.

The penalty terms of [GLM11] and [DWW14] encourage $\Omega_{ij}^{(k)}$ and $\Omega_{ij}^{(k')}$ and $\Omega_{ij}^{(k)}$ and $\Omega_{ij}^{(k'')}$ to be equally similar. However, there might be a situation where it is known that some subpopulations are more similar than others. To address this possibility, [SS16] place the K subpopulations onto a **subpopulation graph**, a weighted, undirected graph where the weight of the edge between subpopulations i and j, denoted W_{ij}, represents their degree of similarity. The weights W_{ij} are collected into an adjacency matrix $W \in \mathbb{R}^{K\times K}$.

As an example of a case where the subpopulation graph is known externally, [SS16] give genetic networks of strains of a virus, which should correspond to evolutionary lineages of their phylogenetic trees. On the other hand, the genetic network of cancer cells might be unknown. [SS16] give a method for estimating the subpopulation graph in the case that it is unknown, which involves hierarchical clustering.

The method presented in this thesis will also make use of a subpopulation graph. For simplicity, it is assumed that the subpopulation graph is known, and that all weights W_{ij} are 0 or 1 (i.e., the subpopulation graph is unweighted). This corresponds to the case where subpopulation similarities are captured only by the shape of the subpopulation graph.

The **LASICH** estimator of [SS16] is the solution to the following optimization problem:

$$\hat{\Theta} = \underset{\Theta=\Theta^\top, \Theta \succ 0}{\operatorname{argmin}} \left\{ - \left[\frac{1}{n} \sum_{k=1}^{K} n_k \left(\log\det \Theta^{(k)} - \operatorname{tr}\left(\Psi^{(k)} \Theta^{(k)} \right) \right) \right] + \rho \sum_{k=1}^{K} \sum_{i \neq j} \left| \Theta_{ij}^{(k)} \right| + \rho \rho_2 \sum_{i \neq j} \| \Theta_{ij} \|_L \right\}.$$

Like the optimization problem of [RBL08], this optimization problem involves the Gaussian correlation-based log-likelihood and L1 regularization. Given $\hat{\Theta}$, $\hat{\Omega}$ is recovered analogously to [RBL08].

To account for subpopulation structure, the optimization problem includes a Laplacian term, where the **graph Laplacian penalty** is given by

$$\| \Theta_{ij} \|_L = \sqrt{\sum_{k,k'=1}^{K} W_{kk'} \left(\theta_{ij}^{(k)} - \theta_{ij}^{(k')} \right)^2}.$$

The effect of this term is to exploit the information in W to encourage similarity among values of $\theta_{ij}^{(k)}$ and $\theta_{ij}^{(k')}$.

In contrast to the previous joint approaches, which incorporate a penalty term into global methods, [MM16] incorporates a penalty term into neighborhood regression, a local method.

The **joint structural estimation method (JSEM)** consists of two steps. Suppose that for every $\beta_{ij} = (\beta_{ij}^{(1)}, \ldots, \beta_{ij}^{(K)})$, it is known how to best partition β_{ij} into subsets whose elements are induced to be similar. For every $i \neq j$, this knowledge is encoded by $\mathcal{G}_{ij}$, a partition of $\{1, \ldots, K\}$. For every $g \in \mathcal{G}_{ij}$ and $k \in g$, $\beta_{ij}^{(k)}$ are known to be similar. Then, the first step is to solve

$$\min \left\{ \frac{1}{n} \sum_{k=1}^{K} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)}\|_2^2 + 2 \sum_{i \neq j} \sum_{g \in \mathcal{G}_{ij}} \lambda_{ij}^{[g]} \|\beta_{ij}^{[g]}\|_2 \right\}$$

to obtain $\hat{E}^{(k)} = \left\{(i,j) : \beta_{ij}^{(k)} \neq 0\right\}$. Then, this edge set is used as a constraint for K optimization problems

$$\min_{\substack{\Omega^{(k)} \succeq 0 \\ (i,j) \notin \hat{E}^{(k)} \implies \Omega_{ij}^{(k)} = 0}} \left\{ \operatorname{tr}(S^{(k)} \Omega^{(k)}) - \log\det \Omega^{(k)} \right\}.$$

JSEM is a combination of local and global methods, the penalty term for inducing similarity across subpopulations being added to the local step.

2.5 Directed Graphical Models

This main algorithm of this thesis and the aforementioned methods are for inference of undirected graphical models. Also worth mentioning is **distributed annealing on regularized likelihood score (DARLS)** by [YAZ24]. Like the main algorithm of this thesis, DARLS is a distributed algorithm for inference of graphical models (see chapter 4 for more on distributed algorithms). However, the graphical models inferred by DARLS are directed acyclic graphical models. A directed acyclic graph is a **directed acyclic graphical model (DAG)** if it corresponds to a set of random variables $\{X_1, \ldots, X_p\}$ with probability density factorizing as

$$p(x_1, \ldots, x_p) = \prod_{j=1}^{p} p(x_j \mid \mathrm{PA}_j = pa_j),$$

where $\mathrm{PA}_j \subset \{X_1, \ldots, X_p\} \setminus \{X_j\}$ is the parent set of X_j with pa_j being its value. DAGs are useful for inferring causal relations among variables.

The inferential target of DARLS is β_{ij}, associated with the edge $X_i \to X_j$ and $\beta_{ij} = 0$ if $X_i \notin \mathrm{PA}_j$. [YAZ24] considers the case of **generalized linear DAG (GLDAG)**, where the conditional density is given by

$$p(x_j \mid pa_j, \beta_j) = c_j(x_j) \exp\left(\langle \beta_j^\top x, x_j \rangle - b_j(\beta_j^\top x)\right).$$

Like the multiple subpopulations setting, [YAZ24] considers a case where the data $\{X^{(1)}, \ldots, X^{(K)}\}$ is split amongst K agents. Unlike the multiple subpopulations setting, the $\{X^{(1)}, \ldots, X^{(K)}\}$ of [YAZ24] are iid.

In both this thesis and [YAZ24], the data $\{X^{(1)}, \ldots, X^{(K)}\}$ is split amongst K computational agents. Both this thesis and [YAZ24] make use of distributed algorithms to address privacy concerns as well as concerns about the cost of communicating $X^{(\cdot)}$.

A challenge unique to inference of DAGs is respecting the acyclicity constraint. To address this challenge, it is useful to observe that every directed acyclic graph has at least one topological sort. A permutation π is a **topological sort** of a directed acyclic graph if $a \in \mathrm{PA}_b$ implies that a precedes b in the ordered defined by π. Therefore, the objective function is given by $\min_{\pi \in \mathcal{P}} f(\pi)$, where

$$f(\pi) = \min_{\beta \in \mathcal{D}(\pi)} \sum_{k=1}^{K} \frac{n_k}{n} \ell_{\mathcal{I}_k(\beta)} + \lambda \sum_{i,j} \|\beta_{ij}\|_F.$$

In the above,

- $\ell_{\mathcal{I}_k(\beta)}$ is the normalized negative log-likelihood of the subsample belonging to the k-th computational agent.

- $\mathcal{P}$ is the set of all permutations on $\{1, \ldots, p\}$.
- $\mathcal{D}(\pi)$ is the set of DAGs whose topological sorts are compatible with a permutation $\pi \in \mathcal{P}$.

The basic iteration of DARLS is as follows:

1. Based on the current permutation $\hat{\pi}$, central server proposes "neighboring" permutation π^+.
2. Central server and agents compute $(\beta^+, f(\pi^+))$ via distributed optimization. This step involves proximal gradient descent.
3. Central server sets $(\hat{\pi}, \hat{\beta}, f(\hat{\pi})) \leftarrow (\pi^+, \beta^+, f(\pi^+))$ with probability determined via simulating annealing heuristic.

CHAPTER 3

Non-Distributed Algorithm

3.1 Optimization Problem

Whereas the approach of [SS16] is to add a Laplacian term to the global method of [RBL08], the approach of this thesis is to add a Laplacian term to neighborhood regression, a local approach. The optimization problem is

$$\begin{aligned}\hat{\beta} &= \underset{\beta \in S}{\operatorname{argmin}} \left\{ \sum_{k=1}^{K} \left[\frac{1}{2n_k} \sum_{j=1}^{p} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)}\|_2^2 + \lambda \sum_{i \neq j} |\beta_{ij}^{(k)}| \right] + \rho \sum_{i \neq j} \|\beta_{ij}\|_L^2 \right\} \\ &= \underset{\beta \in S}{\operatorname{argmin}} \Bigg\{ \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{j=1}^{p} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)}\|_2^2 \\ &\qquad + \rho \sum_{i \neq j} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')} \right)^2 + \lambda \sum_{k=1}^{K} \sum_{i \neq j} |\beta_{ij}^{(k)}| \Bigg\}\end{aligned}$$

where

- The optimization variable, β, is a list of K matrices. The k-th element of β is written $\beta^{(k)}$ and has dimension $p \times p$.
- The optimization variable β is constrained to be within the set

$$S = \{\{\beta^{(1)}, \ldots, \beta^{(K)}\} \colon \beta^{(1)}, \ldots, \beta^{(K)} \in \mathbb{R}^{p \times p}, \operatorname{diag}(\beta^{(1)}) = \cdots = \operatorname{diag}(\beta^{(K)}) = 0\}.$$

In other words, the diagonal elements of $\beta^{(k)}$ must all be 0, for $k = 1, \ldots, K$. This constraint ensures that the optimization problem, which does not depend on the diagonal elements of each $\beta^{(k)}$, has a unique solution.

- The data, X, is a list of K matrices. The k-th element of X is written $X^{(k)}$ and has dimension $n_k \times p$.

- The adjacency matrix W, is a matrix with dimension $K \times K$ and

$$\|\beta_{ij}\|_L^2 = \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')}\right)^2.$$

- The Laplacian parameter ρ and L1 parameter λ are positive numbers.

Let

$$\begin{aligned} f(\beta; W, X, \lambda, \rho) = &\sum_{k=1}^{K} \frac{1}{2n_k} \sum_{j=1}^{p} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)}\|_2^2 \\ &+ \rho \sum_{i \neq j} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')}\right)^2 + \lambda \sum_{k=1}^{K} \sum_{i \neq j} |\beta_{ij}^{(k)}| \end{aligned}$$

denote the objective function, and write

$$g(\beta; W, X, \rho) = \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{j=1}^{p} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)}\|_2^2 + \rho \sum_{i \neq j} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')}\right)^2$$

and

$$h(\beta; \lambda) = \lambda \sum_{k=1}^{K} \sum_{i \neq j} |\beta_{ij}^{(k)}|$$

so that

$$f(\beta; W, X, \lambda, \rho) = g(\beta; W, X, \rho) + h(\beta; \lambda).$$

The function $g(\beta; W, X, \rho)$ is called the **differentiable part** of the objective function and $h(\beta; \lambda)$ is called the **proximable part** of the objective function.

3.2 Application of Proximal Gradient Descent

Because the objective function can be separated into a differentiable part and a part which is easily proximable, it can be optimized using proximal gradient descent. Noting that the proximal operator of $h(\beta; \lambda)$ is given by the **soft-thresholding operator**

$$S_\kappa(x) = \begin{cases} x - \kappa & x > \kappa \\ 0 & |x| \leq \kappa \\ x + \kappa & x < -\kappa \end{cases},$$

it is easy to see that the iteration of proximal gradient descent for $f(\beta; W, X, \lambda, \rho)$ is given by

$$(\beta)^{(s)} = S_{\lambda t_s}\left\{(\beta)^{(s-1)} - t_s \nabla g((\beta)^{(s-1)})\right\},$$

where s indexes iteration, and t_s is the step-size at the s-th iteration. The gradient ($i^* \neq j^*$) is given by

$$\frac{\partial g}{\partial \beta_{i^*j^*}^{(k^*)}} = \frac{1}{n_{k^*}} \sum_{i=1}^{p} \beta_{ij^*}^{(k^*)} \left\langle X_{\cdot i^*}^{(k^*)}, X_{\cdot i}^{(k^*)} \right\rangle + \rho \sum_{k=1}^{K} W_{kk^*} (\beta_{i^*j^*}^{(k^*)} - \beta_{i^*j^*}^{(k)}). \tag{3.1}$$

Equation 3.1 holds if

- W is symmetric, i.e. $W_{kk^*} = W_{k^*k}$.

- For $k = 1, \ldots, K$, the diagonal elements of $\beta^{(k)}$ are set to -1. In other words, for $k = 1, \ldots, K$ and $i = 1, \ldots, p$, $\beta_{ii}^{(k)} = -1$.

The values of the diagonal elements of each $\beta^{(k)}$ can be assigned any value without affecting the value of the objective function. Therefore, they are set to -1 in order to simplify writing the gradient and implementing the algorithm.

For the derivation of equation 3.1, see Appendix A.

3.3 Block Matrices

The following notation regarding block matrices is used to write down the Laplacian step succinctly:

block matrix

Let $D \in \mathbb{R}^{Kp \times Kp}$. D can be visualized as divided into K^2 blocks, where the ij-th block $D_{ij} \in \mathbb{R}^{p \times p}$ consists of the $[(i-1)p+1]$-th to ip-th rows and $[(j-1)p+1]$-th to jp-th columns of D. Observe that this overloads the notation for an element of a matrix.

block column of a block matrix

The notation for a column of a matrix is overloaded for block matrices. If $D \in \mathbb{R}^{Kp \times Kp}$ is a block matrix, then $D_{\cdot k}$ denotes the $[(k-1)p+1]$-th to kp-th columns of D.

block Hadamard product

If $B \in \mathbb{R}^{p \times p}$ is a matrix, and $D \in \mathbb{R}^{Kp \times Kp}$ is a block matrix, then $B \square D \in \mathbb{R}^{Kp \times Kp}$ is the block matrix where $(B \square D)_{ij} = B_{ij} D_{ij}$. Note that $B_{ij} \in \mathbb{R}$ but $D_{ij} \in \mathbb{R}^{k \times k}$.

3.4 Statement of Non-Distributed Algorithm

The following algorithm implements proximal gradient descent for $f(\beta; W, X, \lambda, \rho)$.

Algorithm 1 Non-Distributed Algorithm

INPUT: Step size parameters $a > 0$, $b \in (0, 1)$; Stopping criteria $s_{\max}$, tol, and $t_{\min}$; adjacency matrix $W \in \mathbb{R}^{K \times K}$; Data $X^{(k)} \in \mathbb{R}^{n_k \times p}$, $k = 1, \ldots, K$; Initial guess $(\beta^{(k)})^{(0)} \in \mathbb{R}^{p \times p}$, $k = 1, \ldots, K$; Laplacian parameter ρ; L1 parameter λ.

INITIALIZE: Initial iteration $s \leftarrow 1$; Initial step size $t_1 = a$; $Y^{(k)} \in \mathbb{R}^{p \times p}$, $k = 1, \ldots, K$, a symmetric matrix collecting the pairwise inner products of the columns of $X^{(k)}$, i.e., $Y_{qr} \leftarrow \left\langle X^{(k)}_{\cdot q}, X^{(k)}_{\cdot r} \right\rangle$.

REPEAT UNTIL CONVERGENCE:

1: $(\Delta\beta)^{(s-1)} \leftarrow \begin{bmatrix} (\beta^{(1)})^{(s-1)} - (\beta^{(1)})^{(s-1)} & \ldots & (\beta^{(1)})^{(s-1)} - (\beta^{(K)})^{(s-1)} \\ \ldots & \ldots & \ldots \\ (\beta^{(K)})^{(s-1)} - (\beta^{(1)})^{(s-1)} & \ldots & (\beta^{(K)})^{(s-1)} - (\beta^{(K)})^{(s-1)} \end{bmatrix}$, a block matrix.

2: $(\beta)^{(s-2/3)} \leftarrow (\beta)^{(s-1)} - t_s \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(s-1)})_{\cdot k}$

3: **for** $k = 1, \ldots, K$ **do**

4: **for** $i = 1, \ldots, p$ **do**

5: $(\beta^{(k)}_{ii})^{(s-1)} \leftarrow -1$

6: **end for**

7: $(\beta^{(k)})^{(s-1/3)} \leftarrow (\beta^{(k)})^{(s-2/3)} - \frac{t_s}{n_k} Y^{(k)} (\beta^{(k)})^{(s-1)}$

8: **end for**

9: $(\beta)^{(s)} \leftarrow S_{\lambda t_s}((\beta)^{(s-1/3)})$

10: **if** $s > s_{\max}$ or $\|(\beta)^{(s)} - (\beta)^{(s-1)}\|_F <$ tol or $t_s < t_{\min}$ **then**

11: **return** $\hat{\beta} \leftarrow (\beta)^{(s)}$

12: **end if**

13: **if** $f((\beta)^{(s)}; W, X, \lambda, \rho) < f((\beta)^{(s-1)}; W, X, \lambda, \rho)$ **then**

14: $s \leftarrow s + 1$

15: $t_s \leftarrow a$

16: **else**

17: $t_s \leftarrow b t_s$

18: **end if**

3.5 High-Level Overview of Non-Distributed Algorithm

The iteration of the non-distributed algorithm can be broken down into 5 main steps.

- Lines 1 and 2 constitute the **Laplacian step**, by which $(\beta)^{(s-2/3)}$ is obtained from $\beta^{(s-1)}$.
- Lines 3 through 8 constitute the **neighborhood regression step**, by which $(\beta)^{(s-1/3)}$ is obtained from $(\beta)^{(s-2/3)}$.
- Line 9 is the **soft-thresholding step** by which $(\beta)^{(s)}$ is obtained from $(\beta)^{(s-1/3)}$.
- Stopping criteria are checked in lines 10 through 12.
- Step size for the next iteration is determined in lines 13 through 18.

The Laplacian step and neighborhood regression step comprise the gradient descent part of proximal gradient descent, i.e.,

$$(\beta)^{(s-1/3)} = (\beta)^{(s-1)} - t_s \nabla g((\beta)^{(s-1)}).$$

Then, the soft-thresholding step completes an iteration of proximal gradient descent.

Using block matrices, the Laplacian step can be written down in just 2 lines.

One detail omitted from the pseudocode is the following: in line 2, $(\beta)^{(s-1)}$ is a list of length K of $p \times p$ matrices, but $t_s \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(s-1)})_{\cdot k}$ is a $Kp \times p$ matrix, so the subtraction $(\beta)^{(s-1)} - t_s \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(s-1)})_{\cdot k}$ does not make grammatical sense. To make the dimensions match, $t_s \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(s-1)})_{\cdot k}$ is coerced to a list of length K by having its k-th element be the matrix consisting of the $[(k-1)p+1]$-th to $[kp]$-th rows of $t_s \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(s-1)})_{\cdot k}$.

3.6 Step Size Scheme and Stopping Criteria

The non-distributed algorithm uses an unsophisticated step size scheme involving two parameters, $a > 0$ and $b \in (0, 1)$. In iteration s, the algorithm obtains $\beta^{(s)}$ as a function of $\beta^{(s-1)}$ and step size t_s. Initially, the algorithm tries $t_s = a$. After the s-th iteration is complete, the algorithm checks if the objective function has been lowered, i.e. $f((\beta)^{(s)}; W, X, \lambda, \rho) < f((\beta)^{(s-1)}; W, X, \lambda, \rho)$. If yes, the iteration counter is advanced ($s \leftarrow s+1$) and the algorithm tries $t_{s+1} = a$. If no, the algorithm retries the s-th iteration, this time with a smaller step size $t_s \leftarrow bt_s$.

The algorithm checks three stopping criteria. First, it checks whether the number of iterations s has exceeded a prespecified maximum number of iterations $s_{\max}$. Second, it checks that $\|(\beta)^{(s)} - (\beta)^{(s-1)}\|_F$ is above a prespecified tolerance. Here, the Frobenius norm $\|\cdot\|_F$ is the square root of the sum of squares of the off-diagonal entries of each layer $(\beta^{(k)})^{(s)} - (\beta^{(k)})^{(s-1)}$. Lastly, it checks that the step size t_s prescribed by the above step size scheme is not smaller than some prespecified minimum step size $t_{\min}$.

Another detail omitted from the pseudocode is the following: for some initial guesses $\beta^{(0)}$, the algorithm would get "stuck" in the sense that $f((\beta)^{(0)}; W, X, \lambda, \rho) < f((\beta)^{(1)}; W, X, \lambda, \rho)$ even after searching many values of t_1. To alleviate this issue, the if statement on line 13 should actually read

$$\textbf{if } f((\beta)^{(s)}; W, X, \lambda, \rho) < f((\beta)^{(s-1)}; W, X, \lambda, \rho) \textbf{ or } (bt_s < t_{\min} \textbf{ and } s < 10).$$

Forcing the algorithm to take steps when it is "stuck" in early iterations allowed it to find more optimal β in the later iterations.

3.7 Post-Processing

To ensure that each $\hat{\beta}^{(k)}$ is symmetric, the following post-processing rule is applied to the off-diagonal elements of $\hat{\beta}^{(k)}$, $k = 1, \ldots, K$:

- If one of $\hat{\beta}_{ij}^{(k)}$ and $\hat{\beta}_{ji}^{(k)}$ is zero, set them both to zero.
- If both of $\hat{\beta}_{ij}^{(k)}$ and $\hat{\beta}_{ji}^{(k)}$ are nonzero and $\hat{\beta}_{ij}^{(k)}$ and $\hat{\beta}_{ji}^{(k)}$ have the same sign, set them both to their mean.
- If both of $\hat{\beta}_{ij}^{(k)}$ and $\hat{\beta}_{ji}^{(k)}$ are nonzero and $\hat{\beta}_{ij}^{(k)}$ and $\hat{\beta}_{ji}^{(k)}$ have opposite signs, set them both to whichever has the larger absolute value.

The first bullet point, which determines the support of $\hat{\beta}$, is the same post-processing rule as equation 7 of [MB06].

CHAPTER 4

Distributed Algorithm

4.1 The Distributed Setting

Suppose that K agents each possess one of the $X^{(k)}$, and that they wish not to communicate the $X^{(k)}$ to each other. This may be due to cost of communication or privacy concerns. Despite this, the agents still desire to leverage the shared structure between the $X^{(k)}$ when inferring $\beta^{(k)}$.

In a **non-distributed algorithm**, a single computational agent solves an optimization problem by itself. For instance, algorithm 1 is non-distributed. By contrast, a **distributed algorithm** is one in which computational agents communicate in order to solve an optimization problem.

To solve the above problem, the algorithm will have to be distributed amongst at least K computational agents, the k-th computational agent handling at least the computations involving $X^{(k)}$. These K computational agents will simply be called the **agents**.

In addition, the distributed algorithm will involve a $[K+1]$-th computational agent, called the **server**. The agents will communicate only with the server, and the server will communicate with each of the K agents.

In the non-distributed algorithm, X occurs in two places: in the neighborhood regression step, and in the computation of the objective function (which is used to determine step size). These computations will have to be distributed amongst the K agents.

Despite not being necessary to distribute, the soft-thresholding step will also be dis-

tributed amongst the K agents. The Laplacian step is not easy to distribute, and thus will be handled by the server.

4.2 Separating the Objective Function

Write

$$g_0(W, \beta, \rho) = \rho \sum_{i \neq j} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')} \right)^2,$$
$$g_k(X^{(k)}, \beta^{(k)}) = \frac{1}{2n_k} \sum_{j=1}^{p} \| X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)} \|_2^2,$$
$$h_k(\beta^{(k)}, \lambda) = \lambda \sum_{i \neq j} |\beta_{ij}^{(k)}|$$

so that

$$f(\beta; W, X, \lambda, \rho) = g_0(W, \beta, \rho) + \sum_{k=1}^{K} g_k(X^{(k)}, \beta^{(k)}) + \sum_{k=1}^{K} h_k(\beta^{(k)}, \lambda).$$

The function $g_0(W, \beta, \rho)$ is called the **Laplacian part** of the objective function, functions $g_k(X^{(k)}, \beta^{(k)})$ are called the **neighborhood regression parts** of the objective function, and functions $h_k(\beta^{(k)}, \lambda)$ are called the **soft-thresholding parts** of the objective function.

The server will handle computing the gradient of g_0, and the K agents will handle computing the gradient of g_k and soft-thresholding $\beta^{(k)}$.

Writing $f_k(X^{(k)}, \beta^{(k)}, \lambda) = g_k(X^{(k)}, \beta^{(k)}) + h_k(\beta^{(k)}, \lambda)$ yields

$$f(\beta; W, X, \lambda, \rho) = g_0(W, \beta, \rho) + \sum_{k=1}^{K} f_k(X^{(k)}, \beta^{(k)}, \lambda).$$

When computing the objective function, the K agents will compute f_k, which will be sent to the server. The server will compute g_0 and sum to obtain the value of the objective function.

4.3 Statement of Distributed Algorithm

Algorithm 4 implements distributed proximal gradient descent for $f(\beta; W, X, \lambda, \rho)$. Its iteration contains two subalgorithms, called the **agent step** and the **server step**.

The agent step contains the neighborhood regression step (lines 1 through 4), the soft-thresholding step (line 5), and computes the neighborhood regression part and soft-thresholding part of the objective function (line 6). The agent step returns $(\beta^{(k)})^{(s)}$ and $(f_k)^{(s)}$ (line 7).

Algorithm 2 Agent Step

INPUT: s, t_s, $X^{(k)} \in \mathbb{R}^{n_k \times p}$, $Y^{(k)}$, $(\beta^{(k)})^{(s-1)}$, $(\beta^{(k)})^{(s-2/3)}$, λ.

1: **for** $i = 1, \ldots, p$ **do**

2: $\quad (\beta_{ii}^{(k)})^{(s-1)} \leftarrow -1$

3: **end for**

4: $(\beta^{(k)})^{(s-1/3)} \leftarrow (\beta^{(k)})^{(s-2/3)} - \frac{t_s}{n_k} Y^{(k)} (\beta^{(k)})^{(s-1)}$

5: $(\beta^{(k)})^{(s)} \leftarrow S_{\lambda t_s}((\beta^{(k)})^{(s-1/3)})$

6: $(f_k)^{(s)} \leftarrow \frac{1}{2n_k} \sum_{j=1}^{p} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} (\beta_{ij}^{(k)})^{(s)}\|_2^2 + \lambda \sum_{i \neq j} |(\beta_{ij}^{(k)})^{(s)}|$

7: **return** $(\beta^{(k)})^{(s)}$, $(f_k)^{(s)}$.

The server step checks stopping criteria (lines 1 through 3), determines step size (lines 4 through 10), and contains the Laplacian step (lines 11 and 12). If a stopping criterion is satisfied, the server step returns $(\beta)^{(s)}$ (line 2); otherwise, it returns $\beta^{(s-1)}$, $\beta^{(s-2/3)}$, s and t_s (line 13).

Algorithm 3 Server Step

INPUT: a, b, $(f_k)^{(s)}$ $(k = 1, \ldots, K)$, s, t_s, W, $(\beta)^{(s-1)}$, $(\beta)^{(s)}$, ρ

1: **if** $s > s_{\max}$ or $\|(\beta)^{(s)} - (\beta)^{(s-1)}\|_F < \text{tol}$ or $t_s < t_{\min}$ **then**

2: **return** $(\beta)^{(s)}$

3: **end if**

4: $f((\beta)^{(s)}) \leftarrow \sum_{k=1}^{K}(f_k)^{(s)} + \rho \sum_{i \neq j} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')}\right)^2$

5: **if** $f((\beta)^{(s)}) < f((\beta)^{(s-1)})$ **then**

6: $s \leftarrow s + 1$

7: $t_s \leftarrow a$

8: **else**

9: $t_s \leftarrow bt_s$

10: **end if**

11: $(\Delta\beta)^{(s-1)} \leftarrow \begin{bmatrix} (\beta^{(1)})^{(s-1)} - (\beta^{(1)})^{(s-1)} & \ldots & (\beta^{(1)})^{(s-1)} - (\beta^{(K)})^{(s-1)} \\ \ldots & \ldots & \ldots \\ (\beta^{(K)})^{(s-1)} - (\beta^{(1)})^{(s-1)} & \ldots & (\beta^{(K)})^{(s-1)} - (\beta^{(K)})^{(s-1)} \end{bmatrix}$, a block matrix.

12: $(\beta)^{(s-2/3)} = (\beta)^{(s-1)} - t_s \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(s-1)})_{\cdot k}$

13: **return** $\beta^{(s-1)}$, $\beta^{(s-2/3)}$, s, t_s

The iteration of the distributed algorithm consists of the agent step and the server step. All K agents complete the agent step, and the k-th agent sends $(\beta^{(k)})^{(s)}$ and $(f_k)^{(s)}$ to the server (lines 1 through 3). Then the server completes the server step. If the server determines that a stopping criterion was satisfied, then the distributed algorithm terminates and returns $\beta^{(s)}$; otherwise, the server sends $(\beta^{(k)})^{(s-1)}$, $(\beta^{(k)})^{(s-2/3)}$, s, t_s to the k-th agent, $k = 1, \ldots, K$.

Algorithm 4 Distributed Algorithm

INPUT: Step size parameters $a > 0$, $b \in (0, 1)$; Stopping criteria $s_{\max}$, tol, and $t_{\min}$; adjacency matrix $W \in \mathbb{R}^{K \times K}$; Data $X^{(k)} \in \mathbb{R}^{n_k \times p}$, $k = 1, \ldots, K$; Initial guess $(\beta^{(k)})^{(0)} \in \mathbb{R}^{p \times p}$, $k = 1, \ldots, K$; Laplacian parameter ρ; L1 parameter λ.

AGENTS INITIALIZE: $Y^{(k)} \in \mathbb{R}^{p \times p}$, $k = 1, \ldots, K$, a symmetric matrix collecting the pairwise inner products of the columns of $X^{(k)}$, i.e., $Y_{qr} \leftarrow \left\langle X_{\cdot q}^{(k)}, X_{\cdot r}^{(k)} \right\rangle$.

SERVER INITIALIZES: Initial iteration $s \leftarrow 1$; Initial step size $t_1 = a$; $(\Delta\beta)^{(0)} \leftarrow \begin{bmatrix} (\beta^{(1)})^{(0)} - (\beta^{(1)})^{(0)} & \ldots & (\beta^{(1)})^{(0)} - (\beta^{(K)})^{(0)} \\ \ldots & \ldots & \ldots \\ (\beta^{(K)})^{(0)} - (\beta^{(1)})^{(0)} & \ldots & (\beta^{(K)})^{(0)} - (\beta^{(K)})^{(0)} \end{bmatrix}$, a block matrix; $(\beta)^{(1/3)} \leftarrow (\beta)^{(0)} - t_1 \rho \sum_{k=1}^{K} (W \square (\Delta\beta)^{(0)})_{\cdot k}$.

SERVER SENDS: $(\beta^{(k)})^{(0)}$, $(\beta^{(k)})^{(1/3)}$, $s = 1$, t_1 to the k-th agent, $k = 1, \ldots, K$.

REPEAT UNTIL CONVERGENCE:

1: **for** $k = 1, \ldots, K$ **do**:
2: k-th agent does agent step; sends $(\beta^{(k)})^{(s)}$ and $(f_k)^{(s)}$ to server.
3: **end for**
4: Server does server step.
5: **if** output of server step is $\beta^{(s)}$ **then**
6: **return** $\hat{\beta} \leftarrow \beta^{(s)}$
7: **else**
8: Server sends $(\beta^{(k)})^{(s-1)}$, $(\beta^{(k)})^{(s-2/3)}$, s, t_s to the k-th agent, $k = 1, \ldots, K$.
9: **end if**

4.4 Algorithmic Details

The non-distributed and distributed algorithms perform the exact same instructions in the exact same order, the only difference being that the distributed algorithm has to handle communication between agents and a server to circumvent communication of $X^{(k)}$. Therefore, the two algorithms are equivalent from an input-output perspective. Moreover, the distributed algorithm inherits all algorithmic details from the non-distributed algorithm, including step size scheme, stopping criteria, and post-processing.

CHAPTER 5

Numerical Results

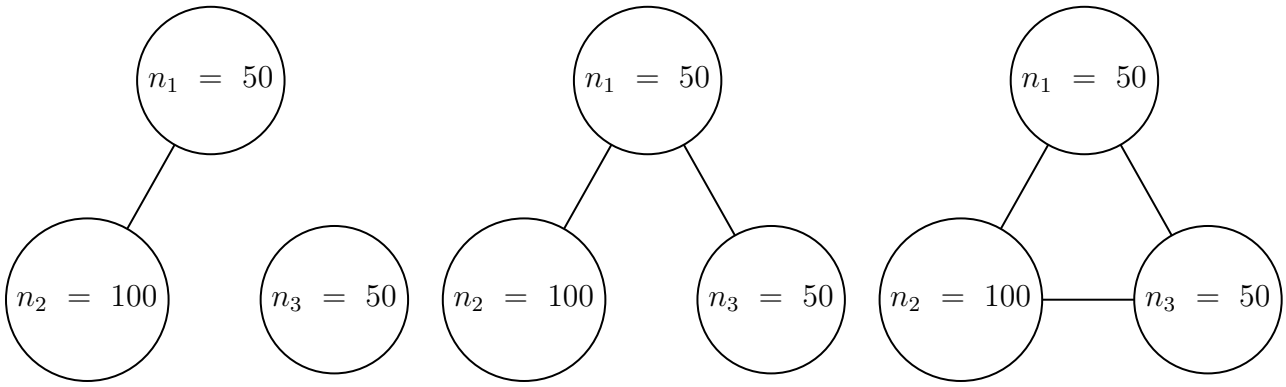
This section details numerical experiments for the algorithm of this thesis - code is available at `https://github.com/gilbert-neuner/graphical-models`.

5.1 Experimental Setups: 1-6

A total of 7 experiments were run, and each experiment was repeated for 10 runs. For each experimental run, the data $X = \{X^{(1)}, \ldots, X^{(K)}\}$ was generated by drawing from $N(0, (\Omega^{(k)})^{-1})$ a total of n_k times. The construction of Ω will be detailed in section 5.3. Then, the distributed algorithm was performed on X. As a benchmark, the graphical lasso was also performed K times, once for each of $X^{(1)}$, ..., $X^{(K)}$. The expectation is that the distributed algorithm should perform better than the graphical lasso since it treats $X^{(1)}$, ..., $X^{(K)}$ jointly.

For experiments 1-6, $K = 3$ subpopulations were considered, with sample sizes $n_1 = 50$, $n_2 = 100$, and $n_3 = 50$, and dimension $p = 100$. Two factors which were varied between experiments were the shape of the subpopulation graph and the sparsity of each $\Omega^{(k)}$.

Three subpopulation graphs were considered, which are given in figure 5.1. Graph 1 was used in experiments 1 and 2, graph 2 was used in experiments 3 and 4, and graph 3 was used in experiments 5 and 6.



(a) Graph 1 (b) Graph 2 (c) Graph 3

Figure 5.1: Subpopulation graphs: Experiments 1-6

The adjacency matrices W were chosen by letting $W_{ij} = 1$ if there is an edge in the corresponding subpopulation graph and $W_{ij} = 0$ otherwise.

- experiments 1, 2: $W = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- experiments 3, 4: $W = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
- experiments 5, 6: $W = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$.

With $p = 100$, each $\Omega^{(k)}$ could contain up to $100(99)/2 = 4950$ edges. In experiments 1, 3, and 5, each $\Omega^{(k)}$ was expected to contain 100 edges, or an expected sparsity of 0.02. In experiments 2, 4, and 6, each $\Omega^{(k)}$ was expected to contain 200 edges, or an expected sparsity of 0.04.

Experiment	Graph	Exp. Edges per Subpop.	Exp. Sparsity per Subpop.
1	1	100	0.02
2	1	200	0.04
3	2	100	0.02
4	2	200	0.04
5	3	100	0.02
6	3	200	0.04

Table 5.1: Summary of experiments 1-6

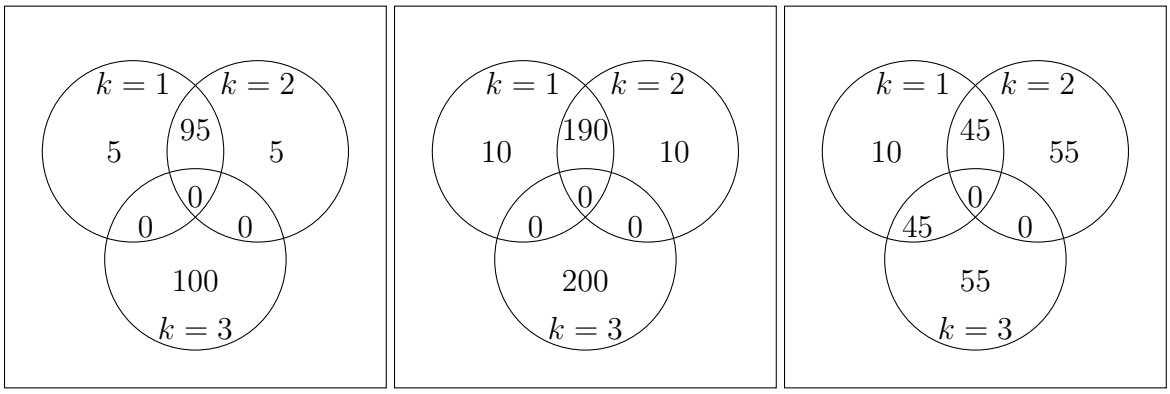
To fully specify the data generating process for Ω, it must be determined how many edges should be expected to occur

- Only in subpopulation 1 (e_1).
- Only in subpopulation 2 (e_2).
- Only in subpopulation 3 (e_3).
- In subpopulations 1 and 2 (e_{12}).
- In subpopulations 1 and 3 (e_{13}).
- In subpopulations 2 and 3 (e_{23}).
- In subpopulations 1, 2, and 3 (e_{123}).

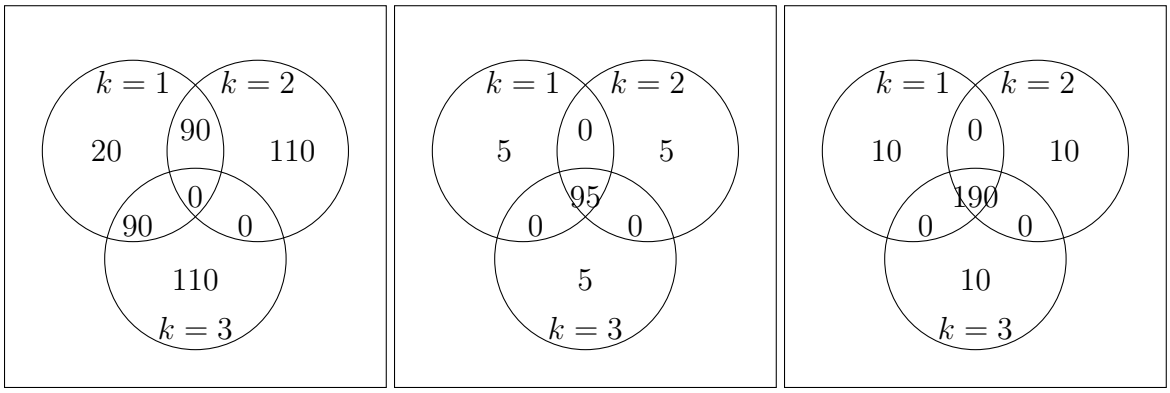
This information is conveniently visualized with a Venn diagram. Of course, each Venn diagram must be subject to the subpopulation graph and sparsity constraints; for example, if the expected edges per subpopulation is 100, then

$$e_1 + e_{12} + e_{13} + e_{123} = e_2 + e_{12} + e_{23} + e_{123} = e_3 + e_{13} + e_{23} + e_{123} = 100.$$

Figure 5.2 gives the Venn diagrams for experiments 1-6.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3



(d) Experiment 4 (e) Experiment 5 (f) Experiment 6

Figure 5.2: Venn diagrams: Experiments 1-6

The expectation is that the distributed algorithm should outperform the graphical lasso since the graphical lasso does not take advantage of shared structure between subpopulations. For example, in subpopulation 1 of experiment 1, there are 95 edges for which the sample size is practically 150 rather than 50, due to the overlap between subpopulations 1 and 2.

By contrast, there is no reason to expect the distributed algorithm to perform better on subpopulation 3 of experiment 1.

5.2 Experimental Setup: 7

For experiment 7, $K = 9$ subpopulations were considered, with sample sizes $n_1 = \cdots = n_9 = 50$, and dimension $p = 150$.

Figure 5.3 gives the subpopulation graph for experiment 7. Note that the subgraphs induced by nodes 1, 2, and 3, nodes 4, 5, and 6, and nodes 7, 8, and 9 are individually the subpopulation graphs for the first six experiments.

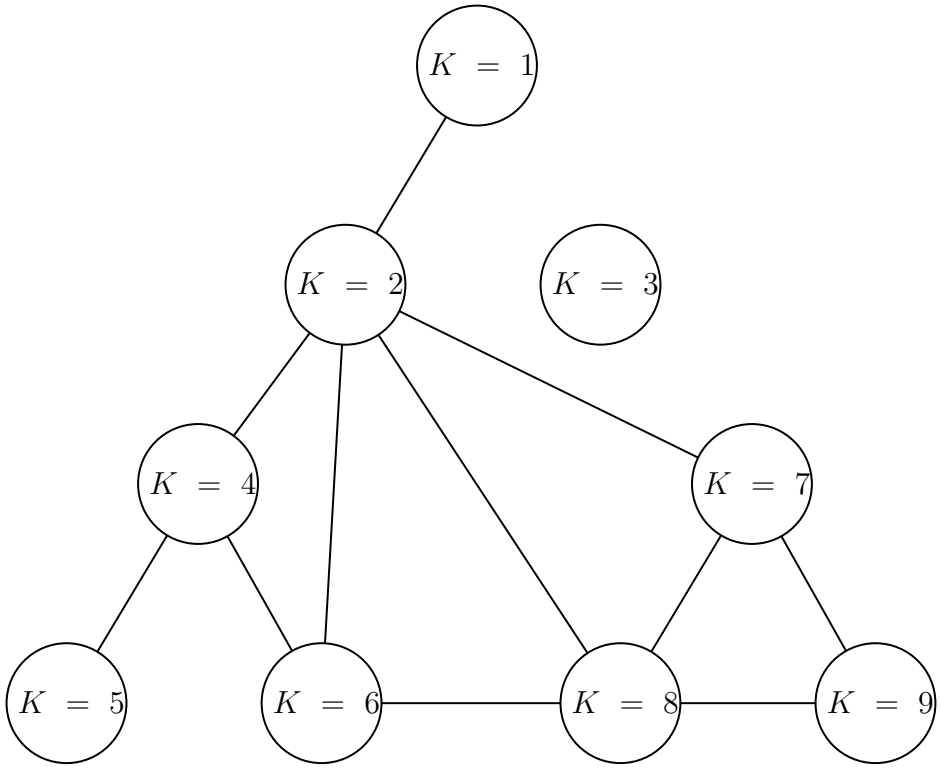


Figure 5.3: Subpopulation graph: Experiment 7

Two adjacency matrices W were considered:

$$W_{\text{correct}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$W_{\text{incorrect}} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The matrix W_{correct} is the adjacency matrix of the subpopulation graph, and the matrix $W_{\text{incorrect}}$ was obtained by replacing zeros with ones and ones with zeros for all off-diagonal elements of W_{correct}. The expectation is that the distributed algorithm should perform better when W is correctly specified.

In accordance with the subpopulation graph, the following "Venn diagram segments" were specified. Each was expected to contain 75 edges.

- Only in subpopulation 1 (e_1).

- Only in subpopulation 3 (e_3).
- Only in subpopulation 5 (e_5).
- Only in subpopulation 9 (e_9).
- In subpopulations 1 and 2 (e_{12}).
- In subpopulations 4 and 5 (e_{45}).
- In subpopulations 2, 4, and 6 (e_{246}).
- In subpopulations 2, 6, and 8 (e_{248}).
- In subpopulations 2, 7, and 8 (e_{278}).
- In subpopulations 7, 8, and 9 (e_{789}).

Consequently, expected number of edges and expected sparsity between subpopulations was not constant. For example, subpopulation 3 was expected to have 75 out of a possible $150(149)/2 = 11175$ edges, or an expected sparsity of 0.007, while subpopulation 2 was expected to have 300 edges, or an expected sparsity of 0.027.

5.3 Data Generation

The method for generating data was similar to that of [SS16]. Given a Venn diagram, Ω was generated in the following way:

1. For each $e_\bullet$, generate the adjacency matrix of an Erdos-Renyi graph expected to have $e_\bullet$ edges, and call it $A^\bullet$.
2. For each $A^\bullet$, for each non-zero above-diagonal element $A^\bullet_{ij}$ and corresponding $A^\bullet_{ji}$, replace with a uniform draw from $(-0.7, -0.5) \cup (0.5, 0.7)$.

3. Form $\Omega^{(k)}$ by creating an empty matrix, putting each $A^{\bullet}$ on its block diagonal in order, and zeroing those block diagonal elements which correspond to $A_{\bullet}$ where $\bullet$ does not contain k.

4. Add 0.1 to the diagonal of $\Omega^{(k)}$ until $\Omega^{(k)}$ is positive definite.

For example, figure 5.4 shows what the support (in black) of $\Omega^{(1)}$, $\Omega^{(2)}$ and $\Omega^{(3)}$ might look like in experiment 5. In accordance with the Venn diagram, $\Omega^{(1)}$ shares edges with $\Omega^{(2)}$ and $\Omega^{(3)}$, and each $\Omega^{(k)}$ has some edges which it doesn't share.

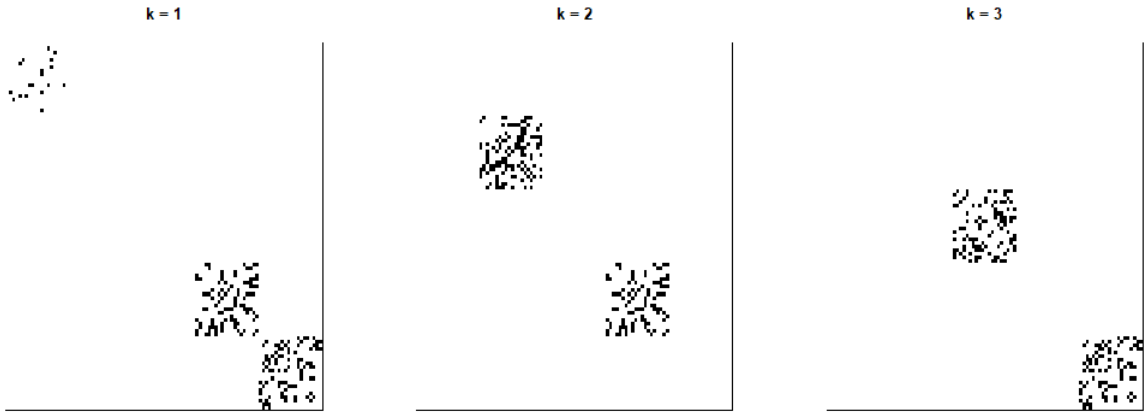


Figure 5.4: Support of Ω, experiment 5

Having generated $\Omega^{(k)}$, the true $\beta^{(k)}$ is calculated by $\beta_{ij}^{(k)} = -\Omega_{ij}^{(k)}/\Omega_{jj}^{(k)}$, and $X^{(k)}$ is generated by drawing from $N(0, (\Omega^{(k)})^{-1})$ a total of n_k times.

5.4 Choice of Tuning Parameters

For all experiments, the parameters of the distributed algorithm were $a = 1$, $b = 0.5$, $s_{\max} = 1000$, $\text{tol} = 10^{-3}$, $t_{\min} = 10^{-6}$, and

$$(\rho, \lambda) \in \{2^{-0.5}, 2^{-1.5}, 2^{-2.5}, 2^{-3.5}, 2^{-4.5}\} \times \{2^{-1.5}, 2^{-2.5}, 2^{-3.5}\}$$

selected to maximize F1 score. For each experiment, the grid search was performed on a dataset distinct from the datasets of the 10 runs, but from the same data generating process.

For experiments 1-6, the initial guess was $(\beta)^{(0)} = \{I_{100\times100}, I_{100\times100}, I_{100\times100}\}$, where $I_{100\times100}$ denotes an identity matrix with dimension 100. For experiment 7, the initial guess was instead an array of 9 identity matrices, each with dimension $p = 150$.

For the graphical lasso, which was used in all experiments, the `glasso` function from the `glasso` package was used, along with default values for all the parameters and s set to the sample covariance matrix, nobs set to 50 or 100 appropriately, and

$$\rho \in \{2^{-0.5}, 2^{-1}, 2^{-1.5}, 2^{-2}, 2^{-2.5}, 2^{-3}, 2^{-3.5}, 2^{-4}, 2^{-4.5}, 2^{-5}\}$$

selected to maximize F1 score.

Note that while ρ is the Laplacian parameter and λ is the L1 parameter of the distributed algorithm, for the `glasso` function ρ is the L1 parameter.

5.5 Results: Experiments 1-6

Since each experiment consisted of 10 runs, all results are reported as their mean plus or minus their standard deviation.

Table 5.2 contains five statistics derived from confusion matrices. These are accuracy, sensitivity, specificity, precision, and F1 score. Actual positives correspond to nonzero off-diagonal entries of Ω, and actual negatives correspond to zero off-diagonal entries of Ω. Predicted positives correspond to nonzero off-diagonal entries of $\hat{\beta}$, and predicted negatives correspond to zero off-diagonal entries of $\hat{\beta}$.

Exp.	Alg.	Accuracy	Sensitivity	Specificity	Precision	F1 Score
1	joint	**0.986**±0.002	0.367±0.078	**0.998**±0.001	**0.828**±0.034	**0.503**±0.069
1	glasso	0.979±0.003	**0.515**±0.075	0.989±0.004	0.497±0.061	0.499±0.033
2	joint	0.966±0.002	0.353±0.043	**0.992**±0.002	**0.644**±0.040	0.454±0.038
2	glasso	**0.967**±0.002	**0.423**±0.035	0.989±0.003	0.633±0.052	**0.505**±0.024
3	joint	**0.981**±0.003	**0.593**±0.071	**0.989**±0.003	**0.541**±0.047	**0.563**±0.037
3	glasso	0.974±0.004	0.523±0.067	0.984±0.005	0.403±0.058	0.452±0.044
4	joint	**0.967**±0.003	**0.491**±0.053	**0.987**±0.004	**0.621**±0.045	**0.545**±0.027
4	glasso	0.959±0.003	0.434±0.058	0.981±0.005	0.496±0.047	0.459±0.031
5	joint	**0.986**±0.002	0.407±0.081	**0.998**±0.001	**0.797**±0.048	**0.533**±0.073
5	glasso	0.979±0.003	**0.441**±0.052	0.990±0.003	0.484±0.065	0.459±0.048
6	joint	0.967±0.002	**0.388**±0.040	0.991±0.002	0.646±0.051	**0.483**±0.036
6	glasso	**0.968**±0.002	0.356±0.046	**0.994**±0.002	**0.716**±0.044	0.474±0.041

Table 5.2: Results, experiments 1-6

Observe that the distributed algorithm performed worst in experiments 1 and 2, when the subpopulation graph contained the least edges.

Table 5.3 contains F1 scores for experiments 1-6, separated by subpopulation.

Exp.	**Alg.**	**k = 1**	**k = 2**	**k = 3**
1	joint	**0.536**±0.082	**0.546**±0.106	0.407±0.093
1	glasso	0.492±0.055	0.538±0.051	**0.463**±0.057
2	joint	0.455±0.040	**0.538**±0.059	0.368±0.039
2	glasso	**0.487**±0.037	0.534±0.037	**0.496**±0.017
3	joint	**0.522**±0.031	**0.652**±0.063	**0.526**±0.040
3	glasso	0.418±0.057	0.524±0.079	0.424±0.033
4	joint	**0.505**±0.037	**0.568**±0.052	**0.550**±0.066
4	glasso	0.425±0.034	0.498±0.086	0.450±0.048
5	joint	**0.524**±0.073	**0.557**±0.084	**0.520**±0.075
5	glasso	0.448±0.065	0.487±0.052	0.445±0.043
6	joint	0.460±0.047	**0.530**±0.032	**0.467**±0.040
6	glasso	**0.471**±0.043	0.487±0.042	0.463±0.046

Table 5.3: F1 score by subpopulation, experiments 1-6

For the distributed algorithm and experiments 1 and 2, F1 scores are highest for subpopulation 2, which has the largest sample size and an edge to subpopulation 1. F1 scores are lowest for subpopulation 3, which has half the sample size of subpopulation 2, and no edges to subpopulations 1 or 2.

For the distributed algorithm and experiments 5 and 6, F1 scores are highest for subpopulation 2, since it has the largest sample size and all subpopulations are connected by an edge.

Table 5.4 contains normalized RMSE for experiments 1-6, computed as $\|\beta - \hat{\beta}\|_F / \|\beta\|_F$.

Exp.	Alg.	Normalized RMSE
1	joint	0.855 ± 0.034
1	glasso	1.403 ± 0.036
2	joint	0.897 ± 0.017
2	glasso	1.557 ± 0.031
3	joint	0.807 ± 0.030
3	glasso	1.350 ± 0.030
4	joint	0.887 ± 0.023
4	glasso	1.512 ± 0.061
5	joint	0.858 ± 0.032
5	glasso	1.269 ± 0.027
6	joint	0.871 ± 0.023
6	glasso	1.375 ± 0.031

Table 5.4: Normalized RMSE, experiments 1-6

5.6 Results: Experiment 7

Table 5.5 contains overall results for experiment 7.

Alg.	Accuracy	Sensitivity	Specificity	Precision	F1 Score	N. RMSE
W_{correct}	**0.988**±0.001	0.561±0.044	0.995±0.002	**0.638**±0.062	**0.593**±0.020	0.889±0.014
glasso	0.987±0.001	0.338±0.031	**0.997**±0.001	0.607±0.051	0.433±0.035	1.234±0.020
$W_{\text{incorrect}}$	0.98±0.003	**0.623**±0.048	0.986±0.004	0.413±0.057	0.492±0.025	0.934±0.010

Table 5.5: Results, experiment 7

Table 5.6 contains F1 scores by subpopulation for experiment 7.

k	W_{correct}	**glasso**	$W_{\text{incorrect}}$
1	**0.594**±0.080	0.424±0.060	0.532±0.113
2	**0.546**±0.029	0.377±0.092	0.461±0.035
3	**0.627**±0.071	0.530±0.078	0.420±0.064
4	**0.620**±0.053	0.462±0.081	0.524±0.082
5	**0.595**±0.057	0.440±0.055	0.481±0.049
6	**0.607**±0.053	0.443±0.056	0.502±0.057
7	**0.614**±0.068	0.452±0.109	0.520±0.079
8	**0.579**±0.060	0.426±0.095	0.497±0.055
9	**0.599**±0.083	0.423±0.095	0.537±0.103

Table 5.6: F1 score by subpopulation, experiment 7

APPENDIX A

Derivation of Equation 3.1

The gradient ($i^* \neq j^*$) is given by

$$\frac{\partial g}{\partial \beta_{i^*j^*}^{(k^*)}} = \frac{1}{n_{k^*}} \sum_{i=1}^{p} \beta_{ij^*}^{(k^*)} \left\langle X_{\cdot i^*}^{(k^*)}, X_{\cdot i}^{(k^*)} \right\rangle + \rho \sum_{k=1}^{K} W_{kk^*} (\beta_{i^*j^*}^{(k^*)} - \beta_{i^*j^*}^{(k)}).$$

Equation 3.1 holds if

- W is symmetric, i.e. $W_{kk^*} = W_{k^*k}$.
- For $k = 1, \ldots, K$, the diagonal elements of $\beta^{(k)}$ are set to -1. In other words, for $k = 1, \ldots, K$ and $i = 1, \ldots, p$, $\beta_{ii}^{(k)} = -1$.

To see this, note that the gradient of the neighborhood regression term is given by

$$
\begin{aligned}
&\frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{j=1}^{p} \|X_{\cdot j}^{(k)} - \sum_{i \neq j} X_{\cdot i}^{(k)} \beta_{ij}^{(k)}\|_2^2 \\
&= \frac{1}{2n_{k^*}} \frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \|X_{\cdot j^*}^{(k^*)} - \sum_{i \neq j^*} X_{\cdot i}^{(k^*)} \beta_{ij^*}^{(k^*)}\|_2^2 \\
&= \frac{1}{2n_{k^*}} \sum_{\ell=1}^{n_{k^*}} \frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \left(X_{\ell j^*}^{(k^*)} - \sum_{i \neq j^*} X_{\ell i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right)^2 \\
&= \frac{1}{2n_{k^*}} \sum_{\ell=1}^{n_{k^*}} \frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \left(X_{\ell j^*}^{(k^*)} - \left(\sum_{i \notin \{i^*, j^*\}} X_{\ell i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right) - X_{\ell i^*}^{(k^*)} \beta_{i^*j^*}^{(k^*)} \right)^2 \\
&= \frac{1}{2n_{k^*}} \sum_{\ell=1}^{n_{k^*}} \frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \left(2X_{\ell i^*}^{(k^*)} \beta_{i^*j^*}^{(k^*)} \left(-X_{\ell j^*}^{(k^*)} + \sum_{i \notin \{i^*, j^*\}} X_{\ell i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right) + \left(X_{\ell i^*}^{(k^*)} \beta_{i^*j^*}^{(k^*)} \right)^2 \right) \\
&= \frac{1}{n_{k^*}} \sum_{\ell=1}^{n_{k^*}} \left(X_{\ell i^*}^{(k^*)} \left(-X_{\ell j^*}^{(k^*)} + \sum_{i \notin \{i^*, j^*\}} X_{\ell i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right) + \left(X_{\ell i^*}^{(k^*)} \right)^2 \beta_{i^*j^*}^{(k^*)} \right) \\
&= \frac{1}{n_{k^*}} \sum_{\ell=1}^{n_{k^*}} X_{\ell i^*}^{(k^*)} \left(X_{\ell i^*}^{(k^*)} \beta_{i^*j^*}^{(k^*)} - X_{\ell j^*}^{(k^*)} + \sum_{i \notin \{i^*, j^*\}} X_{\ell i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right) \\
&= \frac{1}{n_{k^*}} \left\langle X_{\cdot i^*}^{(k^*)}, X_{\cdot i^*}^{(k^*)} \beta_{i^*j^*}^{(k^*)} - X_{\cdot j^*}^{(k^*)} + \sum_{i \notin \{i^*, j^*\}} X_{\cdot i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right\rangle \\
&= \frac{1}{n_{k^*}} \left\langle X_{\cdot i^*}^{(k^*)}, -X_{\cdot j^*}^{(k^*)} + \sum_{i \neq j^*} X_{\cdot i}^{(k^*)} \beta_{ij^*}^{(k^*)} \right\rangle \\
&= \frac{1}{n_{k^*}} \left(- \left\langle X_{\cdot i^*}^{(k^*)}, X_{\cdot j^*}^{(k^*)} \right\rangle + \sum_{i \neq j^*} \beta_{ij^*}^{(k^*)} \left\langle X_{\cdot i^*}^{(k^*)}, X_{\cdot i}^{(k^*)} \right\rangle \right).
\end{aligned}
$$

Adopting the notational convention $\beta_{j^*j^*}^{(k^*)} = -1$, we could write

$$
= \frac{1}{n_{k^*}} \sum_{i=1}^{p} \beta_{ij^*}^{(k^*)} \left\langle X_{\cdot i^*}^{(k^*)}, X_{\cdot i}^{(k^*)} \right\rangle.
$$

For the gradient of the Laplacian term, note that

$$
\begin{aligned}
&\frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \rho \sum_{i \neq j} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{ij}^{(k)} - \beta_{ij}^{(k')} \right)^2 \\
&= \rho \frac{\partial}{\partial \beta_{i^*j^*}^{(k^*)}} \sum_{k,k'=1}^{K} W_{kk'} \left(\beta_{i^*j^*}^{(k)} - \beta_{i^*j^*}^{(k')} \right)^2 \\
&= 2\rho \left(\sum_{k=1}^{K} W_{kk^*} \left(\beta_{i^*j^*}^{(k^*)} - \beta_{i^*j^*}^{(k)} \right) + \sum_{k'=1}^{K} W_{k^*k'} \left(\beta_{i^*j^*}^{(k^*)} - \beta_{i^*j^*}^{(k')} \right) \right) \\
&= 4\rho \sum_{k=1}^{K} W_{kk^*} \left(\beta_{i^*j^*}^{(k^*)} - \beta_{i^*j^*}^{(k)} \right) && \text{(symmetry of } W\text{)} \\
&= \rho \sum_{k=1}^{K} W_{kk^*} \left(\beta_{i^*j^*}^{(k^*)} - \beta_{i^*j^*}^{(k)} \right). && \text{(absorb the constant)}
\end{aligned}
$$

REFERENCES

[DWW14] Patrick Danaher, Pei Wang, and Daniela M Witten. "The joint graphical lasso for inverse covariance estimation across multiple classes." *J. R. Stat. Soc. Series B Stat. Methodol.*, **76**(2):373–397, March 2014.

[FHT07] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Sparse inverse covariance estimation with the lasso.", 2007.

[GLM11] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. "Joint estimation of multiple graphical models." *Biometrika*, **98**(1):1–15, March 2011.

[Lau96] Steffen L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

[MB06] Nicolai Meinshausen and Peter Bühlmann. "High-dimensional graphs and variable selection with the Lasso." *The Annals of Statistics*, **34**(3):1436 – 1462, 2006.

[MM16] Jing Ma and George Michailidis. "Joint Structural Estimation of Multiple Graphical Models." *Journal of Machine Learning Research*, **17**(166):1–48, 2016.

[Pla18] Anna Plaksienko. *Joint estimation of multiple graphical models*. PhD thesis, Gran Sasso Science Institute, 2018.

[RBL08] Adam J. Rothman, Peter J. Bickel, Elizaveta Levina, and Ji Zhu. "Sparse permutation invariant covariance estimation." *Electronic Journal of Statistics*, **2**(none), January 2008.

[SS16] Takumi Saegusa and Ali Shojaie. "Joint estimation of precision matrices in heterogeneous populations." *Electronic Journal of Statistics*, **10**(1):1341 – 1392, 2016.

[YAZ24] Qiaoling Ye, Arash A. Amini, and Qing Zhou. "Federated Learning of Generalized Linear Causal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2024.