Gran Sasso Science Institute
MATHEMATICS IN NATURAL, SOCIAL AND LIFE SCIENCES DOCTORAL PROGRAMME
Cycle XXXIII-AY 2017/2018

Joint estimation of multiple graphical models

PHD CANDIDATE
Anna Plaksienko

ADVISOR
Dr. Claudia Angelini
Istituto per le Applicazioni del Calcolo

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Gran Sasso Science Institute

MATHEMATICS IN NATURAL, SOCIAL AND LIFE SCIENCES DOCTORAL PROGRAMME

Cycle XXXIII-AY 2017/2018

Joint estimation of multiple graphical models

PhD Candidate

Anna Plaksienko

Advisor

Dr. Claudia Angelini

Istituto per le Applicazioni del Calcolo

Thesis submitted for the degree of Doctor of Philosophy

April 20, 2021

Thesis Jury Members
Prof. Dr. Nicoletta Del Buono (Jury President, Università degli Studi di Bari)
Dr. Luisa Cutillo (University of Leeds)
Dr. Paola Maria Rancoita (Università Vita-Salute San Raffaele)
Dr. Francesco Tudisco (Gran Sasso Science Institute)
Prof. Dr. Veronica Vinciotti (Università di Trento)

Thesis Referees
Dr. Luisa Cutillo (University of Leeds)
Dr. Frank Picard (CNRS - ENS Lyon)

Abstract

The fast development of high-throughput technologies such as microarray or next-generation sequencing, and the consequent in-depth investigation of the genome in several international large scale projects, have led to the generation of large amounts of high-dimensional omics datasets. Scientists can use such data to acquire a deep understanding of complex cellular mechanisms, the molecular basis of diseases' development, etc. Among other questions, relationships between different genes or other similar units can reveal regulatory mechanisms whose disruption can be associated with diseases. Network inference methods and, more specifically, graphical models estimation can be used to identify gene relationships and direct interactions not mediated by other factors.

Simply speaking, a graphical model is a graph whose vertices correspond to random variables and edges denote conditional dependence relationships between them. There are plenty of methods for carrying out graphical model inference from a given dataset, even in the high-dimensional setting where the number of variables is much larger than the number of samples (a common situation in omics studies for the enormous number of genes involved and a limited number of samples collected). However, nowadays, it is common to collect and analyze more than one dataset. Multiple datasets can be obtained in different laboratories or with different technologies, arise from various studies, or be of different omics types. Their joint analysis can lead to a more accurate characterization of the underlying biological system, but it also requires specific techniques.

In this thesis, we propose *jewel* – a novel method for the joint analysis of multiple datasets under the assumption that they are drawn from Gaussian distributions that share the same network dependency. In this context, the conditional dependence relationships between variables (genes) are encoded by the inverse covariance matrix. Although we assume that the conditional dependence structure is the same between different conditions, we let the covariance matrices be different to account for different sources of data origin. In this setting, combining the individual datasets into a single one and estimating a sole graphical model would mask the covariance matrices' heterogeneity, while estimating separate models for each case would not take advantage of the common underlying structure. Therefore, a joint analysis of the datasets is preferable, and to this aim in this thesis we present a novel joint estimation method *jewel*. It extends the Meinshausen and Bühlmann regression-based approach to the case of multiple datasets by the mean of a group lasso penalty which guarantees the symmetry of the solution. We design a fast algorithm for the method's implementation, incorporating the smart active shooting approach for a fixed regularization parameter and the warm start approach for an entire grid of regularization parameters. We also state a theorem for *jewel*'s consistency, providing upper and lower bounds for regularization parameter. Moreover, we extend the Bayesian information criterion and cross-validation procedures to the multiple datasets framework to

provide a practical tool for real case applications. We explore the behavior of *jewel* in different simulation settings, analyzing the influence of various input parameters, and comparing the method to other available alternatives for joint estimation, revealing good and competitive performances. Finally, we illustrate the method's performance in real data example regarding transcriptional regulatory networks based on gene expression data. We implement the proposed method in the novel R package `jewel`.

To all struggling PhD students –
don't give up, you got this!

Acknowledgements

First and foremost, I would like to thank my advisor Claudia Angelini for being the best mentor I ever had. I deeply appreciate all the time, effort and patience she has put to not only guide me through my studies but also to help me improve other qualities needed to succeed in academia and generally grow as a researcher. I thank her for always believing in my abilities and constantly motivating me to improve.

I thank Daniela De Canditiis with whom I have collaborated during my PhD studies on the topic, presented in this thesis. It was a great pleasure to work with such a bright, sharp mind, to experience first-hand her teaching talent and to also get to know her cheerful and kind personality.

I would like to thank GSSI PhD coordinators: Pierangelo Marcati for being somewhat of my academic fairy godmother, whose trust in my abilities with accepting me to this PhD program and then taking the time and effort to find me a perfect advisor are still miracles to me that completely changed the course of my career; Nicola Guglielmi for his support and his heartfelt empathy, much needed at those last stages of PhD; and Paolo Antonelli for his help in the final stretch of this journey.

Coming to a young graduate school in a small little-known town may have seemed like a bit of a gamble three years ago, but I am glad to say I have won a jackpot. Not only it was a great pleasure to be a part of GSSI and I have benefitted a lot professionally but I loved the city of L'Aquila which have taught me to see beauty despite the destroyed facade and to never ever give up.

New friendships are another great gain of my PhD journey. I thank Alena for our warm talks and her great advice about academia, Karthik and Sree for being my bureaucracy buddies and always sharing a laugh about yet another paperwork adventure, Kostya for being such a sweet and reliable friend, Monika for her support and unconditional kindness to me, my colleagues in CNR Napoli Eugenio and Valeria for making me feel welcomed during my visits, Silvia for always being someone I can relate to and Luca simply for becoming my friend. I have met many more wonderful people in GSSI, especially Alessia, Antonio, Austin, Cristina, Duyen, Eirini, Giulio, Guiseppe, Mattia, Michele, Mohammad, Reihane, Simone and Trung. I cherish greatly all the happy memories that we have made together. Finally, I thank my friends Nastya – for sharing struggles of an academic career with me – and Sasha – for enthusiastically cheering me on despite choosing a different path in his life – and both of them for always being there for me. It gives me a lot of comfort and strength to have them in my life.

Although it may seem unconventional, I would like to also thank all the subscribers of my Italian blog, which I started along with my doctorate. They have been spectators of my PhD journey from day one and I can not imagine doing this without their kindness and constant support.

I am very blessed to have a family where everyone has always supported my studies but I would like to give special thanks to my husband Vlad and my mom. To Vlad for not hesitating even for a second when I asked him if he would move five thousand km with me so I can pursue my dream of becoming a doctor and for being by my side through all ups and especially all downs. And to my mom for always believing in me without any limit and supporting me at every step of the way, literally every day. She never had any doubt that anything is beyond my reach (even when I had lots) and it is her faith in me that keeps me going.

Contents

List of Symbols and Acronyms

The next list describes several symbols and acronyms that will be later used within the body of the thesis.

Vectors and matrices

$\mathbf{x}$	vectors are denoted by bold lower case letters
$\mathbf{X}$	matrices are denoted by bold capital letters
X, x	variables are denoted by regular letters with no case restrictions
X_i	i-th element of vector $\mathbf{x}$
$\mathbf{x}_{[ij]}$	restriction of vector $\mathbf{x}$ to variables i, j
X_{ij}	ij-th element of matrix $\mathbf{X}$
$X_{\cdot j}$	j-th column of matrix $\mathbf{X}$
$\mathbf{X}_{\cdot -j}$	submatrix of $\mathbf{X}$ without j-th column
$\mathbf{X}^{(k)}$	k-th matrix of the set $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$
$\mathbf{X} \succeq 0$	positive definite matrix
$\det \mathbf{X}$	determinant of matrix $\mathbf{X}$
$\operatorname{tr} \mathbf{X}$	trace of matrix $\mathbf{X}$
$\mathbf{\Sigma}$	covariance matrix of Gaussian distribution
$\mathbf{\Omega}$	precision matrix of Gaussian distribution
$\mathbf{S}$	empirical covariance matrix

Graphs

G	a graph
V	graph's set of vertices
E	graph's set of edges

$\mathbf{A}$ graph's adjacency matrix

(i, j) edge in a graph is denoted with round brackets

$ne(i)$ neighbourhood of a vertex i

$cl(i)$ closure of a vertex i

$d(i)$ degree of a vertex i

Other Symbols

$\|\cdot\|$ l_2-norm

$\|\cdot\|_F$ Frobenius norm

$X \perp\!\!\!\perp Y|Z$ variables X and Y are conditionally independent given variable Z

$(\cdot)_+$ soft-thresholding operator

$\mathbb{P}$ probability measure

$\mathbb{E}$ mathematical expectation

$\mathbb{I}$ indicator function

$\mathbb{R}$ set of real numbers

`jewel` names of R packages are given in the typewriter font

Acronyms

ADMM alternating direction method of multipliers

BIC Bayesian information criterion

CV cross-validation

FPR false positive rate

GGM Gaussian graphical model

GRN gene regulatory network

MLE maximum likelihood estimate

ROC receiver operating characteristic

TPR true positive rate

List of Figures

List of Tables

Chapter 1

Introduction

This thesis will present *jewel*, a novel joint estimation method for Gaussian graphical models under multiple datasets setting. More specifically, this method wants to answer the following question: given several datasets of observations of mostly the same variables under different conditions, how can we infer the connections between these variables? In this chapter, I will first provide a broad motivation to this question. Then, I will give a brief overview of my solution and an outline of the thesis.

1.1 Motivation and aims

This thesis's broad motivation is the interest in developing statistical methods for the analysis of omics data. To explain this, I will first give a simplified idea of "omics" data applications.

Every cell in an organism contains DNA (deoxyribonucleic acid) – a particular long two-strand molecule in the form of a double helix. DNA consists of repeating blocks called nucleotides. The difference between nucleotides is in nitrogenous bases: adenine, thymine, cytosine, and guanine. These bases allow DNA to create a code, a "blueprint" for the growth, development, functioning, and reproduction of cells and organisms. Some parts of DNA are transcribed into mRNA molecules (messenger ribonucleic acid), usually known as transcripts or genes[1]. Then, mRNAs are translated to build proteins (which are the base for all organisms) and other molecules. Using high-throughput instruments such as microarrays or next-generation sequencing, we can measure transcript abundance or gene abundance at the genome-wide scale. We call such information *gene expression*. Gene expression is the heart of transcriptomics, which is one of the so-called omics sciences. It deals with analyzing gene expression data under different conditions to elucidate cell regulatory mechanisms, understand disease onset and progression, investigate response or resistance to drugs, and many others.

One of the "omics" data applications is network inference. Genes can interact with each other to create complex cellular regulatory mechanisms. Subgroups of genes (pathways) work together to develop specific functions whose disruption

[1]Here, for ease of exposition, we use the terms "genes" and "transcripts" interchangeably.

can be associated with diseases. Therefore, one of the questions is building a network describing the gene-to-gene relationships from the gene expression values. Moreover, we can build networks associated with specific conditions and investigate their differences (say, compare regulatory mechanisms in diseased versus healthy tissue or during cell developmental stages). Gene regulatory networks (GRN) have received lots of attention since they can describe different kinds of relationships and provide insights into many biological processes [25]. For example, given an estimated GRN, we can try to identify subgraphs and, therefore, gene pathways that can be used as biomarkers for disease diagnosis (see [9, 14, 15]). This is particularly useful, for example, in cancer studies, where it has been shown that the disease is "activated" by groups of genes rather than an individual one [3, 21]. We can also differentiate the mechanisms of different diseases by comparing their estimated GRNs [43] or use networks in drug design [28, 33]. For more detailed discussion we refer to "Network Medicine: Complex Systems in Human Disease and Therapeutics" [47].

As we can see, GRNs can be extremely useful, and thus they became the initial motivation of my work. Therefore, I chose to focus on network inference and, specifically, on how to infer a graph when several datasets are available. Such a framework is of great interest since nowadays it is widespread to collect several datasets from multiple laboratories and international research projects. Hence, it places this work in the integrative multitask learning framework. While we can face the problem of inferring a network with various mathematical tools from partial differential equations to machine learning techniques, I decided to deal with such a problem using the mean of *graphical models*, since, with this choice, I can encode the conditional dependency structure among the variables with the help of a graph.

Assume we have a dataset of gene expression of p genes measured on N samples. We want to identify the connections between any pair of genes, preferably the direct connections, i.e., those not influenced by other elements or genes in the system. How can we infer this regulatory mechanism? Then, what happens if we have several datasets? And what statistical tools can we use for the inference considering the problem is stated in the high-dimensional setting, and we have to face the curse of dimensionality? In fact, in "omics" data applications, the number of genes (i.e., the number of variables p) is of the order of tens of thousands, while the number of samples N is about few hundreds at best, sometimes even only a few dozens, due to the high experimental costs and the limited amount of certain types of samples. Therefore, almost all the transcriptomics problems are in the high-dimensional ($p >> N$) settings. Network inference is not different.

Specifically, I developed a new method for network estimation in multiple dataset settings. As stated before, I chose to work using graphical models. So, I dealt with graphs where vertices correspond to variables (genes), and edges represent their relations, specifically, conditional dependence – there is an edge between two variables if and only if they are dependent given the rest of the variables. Conditional dependence allows to encode a direct link between two variables, i.e., an influence not explained by the other genes. Moreover, suppose

the variables follow Gaussian distribution (which does hold for many omics applications such as gene expression obtained using microarrays). In that case, it is possible to infer these conditional correlation connections through the non-zero entries of the covariance matrix's inverse. Finally, if more than one dataset is available, it is possible to infer a common network structure with joint estimation techniques. These are the main ingredients of the method developed in this thesis, named *jewel* which is the acronym of **j**oint nod**e-w**ise estimation of multiple Gaussian graphical mod**el**s. *jewel* represents the main result of this work and my answer to the initial question: given several datasets of observations of mostly the same variables under different conditions, how can we infer connections between these variables?

Concluding this section, I would like to stress that although my original motivation was "omics" data application, *jewel* can be applied to other types of data as long as multiple observations of the same variables are collected under different conditions and Gaussian assumption is adequate. Possible examples from the literature can be climate data (temperature, humidity etc.) for different regions [48], share prices for different stocks worldwide, word count on webpages of departments in different universities [36, 53], and many more. Therefore, I will present my results in a very general mathematical environment and, only in the last part, I will specify the application context by examples of gene regulatory network inference.

1.2 Main contributions

The following points summarize the main contributions of this thesis to the literature:

- Development of a novel estimation method, *jewel*: extension of the group lasso penalty to the multiple datasets framework; solution of the corresponding minimization problem; proposal of a smart numerical algorithm that incorporates the active-shooting approach and warm start;
- Proof of the consistency property for *jewel*;
- Extension of the Bayesian information criterion and cross-validation procedures for *jewel*;
- Empirical proof of the advantages of the multiple datasets setting compared to one dataset and of the advantage of joint estimation to naive approaches like concatenation or voting;
- Analysis of the influence of various input parameters on the performance and running time of *jewel*: number of variables, number of samples, network's sparsity, power of preferential attachment for scale-free graph, strength of signal and algorithm's stopping criteria;

- Comparisons with other joint estimation methods, namely JGL [19], and Guo et al. proposal [36], in terms of performance and running time;

- Application of *jewel* to real genomics data;

- Development of the R package `jewel`.

I carried out the work described in this thesis at the Gran Sasso Science Institute, where I spent most of my training. Moreover, I collaborated with Dr. Claudia Angelini and Dr. Daniela De Canditiis at the Institute for Applied Calculus, which I visited several times before the COVID-19 pandemic. Due to the pandemic lockdown limitations, the last year of my activities was from the remote.

I presented part of the results of this thesis as a selected oral talk at the 13th International Conference of the ERCIM WG on Computational and Methodological Statistics (CMStatistics 2020). I also presented the R package and the real data applications as a "poster", i.e., a brief talk, at the Bioinformatics and Computational Biology Conference (BBCC 2020). I am also finalizing a full manuscript describing my work's main results, and I plan to submit it in early 2021.

During my three years of Ph.D. studies, on top of more than six months of interdisciplinary courses, I also attended the 40-hour workshop entitled "Analysis of single-cell RNA-seq data", the 10-hour course "Regularization, penalization techniques and local modeling" and dozens of 4-hour courses on R programming language.

1.3 Outline of the thesis

This thesis consists of seven chapters, including Introduction, Conclusions and future work, and three appendices.

- This chapter 1 provides the motivations which inspired my work and summarizes the main results I have obtained.

- Chapter 2 introduces the basic notions and concepts from graph theory and provides the definitions and properties of conditional independence. We combine the two ideas into the graphical model framework. Then, we specifically focus on the properties of graphical models under the Gaussian assumption.

- Chapter 3 illustrates the network inference in the context of Gaussian graphical models. We start with the classical case of one given dataset and focus on two main approaches: the maximum-likelihood based and regression-based. We then move to the multitask learning framework and give a brief overview of the methods for joint estimation.

- Chapter 4 presents *jewel* – a novel method for joint estimation of Gaussian graphical models from multiple datasets. We describe the minimization problem and its solution and derive a theoretical property of the estimator. Moreover, we propose empirical approaches for estimating the regularization parameter. This chapter constitutes the main methodological achievement of my thesis.

- Chapter 5 investigates the behavior of *jewel* in different simulation settings. We prove the advantages of the joint approach to other naive alternatives and to the case of one dataset. Moreover, we explore the influence of all input parameters on the method's performance. Finally, we compare *jewel* to its competitors. This chapter constitutes the main computational achievement of my thesis.

- Chapter 6 describes an application of *jewel* to the "toy" real data example. Meanwhile, we are working on conducting more extensive experiments.

- Chapter 7 recapitulates my contribution to state of the art and discusses potential improvements and future works.

- Appendix A summarizes the main features of existing joint estimation methods in a condensed table form.

- Appendix B describes the implementation of the method into the R package `jewel` and contains its documentation.

- Appendix C discusses the useful insights we discovered during the process of `jewel` package development.

Chapter 2

Graphical models

Graphical models are a useful theory for encoding the underlying relations between random variables via a graph. They are applied in many different frameworks such as social networks analysis [27], speech recognition [10], modeling brain connectivity [32], gene regulatory networks or protein-protein interactions, and so on (see 8.3.3 in [59] for more references). Two main types of graphical models in the literature are directed graphs (i.e., Bayesian networks) and undirected graphs (i.e., Markov random fields), the latter being the focus of my thesis.

In this chapter, I will first introduce some basic definitions and concepts from graph theory and the concept of conditional independence. Then, I will link the two points and describe undirected graphical models and their properties. Finally, I will focus on Gaussian graphical models.

For a detailed discussion on graphical models and their inference (presented in Chapter 3), I refer the reader to "Graphical Models" by S.L. Lauritzen [46], chapter 7 of "Introduction to High-Dimensional Statistics" by C. Giraud [34], and chapter 8 of "Sparse Modelling: Theory, Algorithms and Applications" by I. Rish and G. Grabarnik [59].

2.1 Basic concepts of graph theory

A graph is a mathematical object used to describe connections between any subjects. For example, it is possible to represent people and their social relationships (e.g., family ties, school, work, or social networks), cities and roads between them, web pages and links from one to another, atoms and their chemical bonds, genes and transcription factors, and many others. Connections or relationships can be physical or not. Moreover, they can be directly observed, such as cities and roads that connect them, or derived from observed data using an inferential procedure. Different types of relationships among the same set of subjects can lead to different graphs.

One of the aims of graph theory is to describe the relations among entities in terms of a graph or a network and incorporate some specific types of relations starting from the observed data. However, graph theory also provides a

mathematical framework to study the topological structure of graphs and their properties and a useful way of representing complex systems using networks. From this point of view, graph theory constitutes a much broader framework than graphical models.

I refer the reader to the seminal book "Spectral Theory" by F. Chung [16] for a mathematical treatment of graphs from the point of view of spectral theory. In brief, the spectral theory allows studying the topological properties of a graph as a function of eigenvalues-eigenvectors decomposition of suitable matrices, associated with the graph. It usually assumes that the graph is known, and the main problem is to investigate if some mathematical properties hold or optimize search or flows.

Another concept, associated with a graph, is the term *"network"*. I suggest the reader the "Network Science" book by A.-L. Barabasi [5] for extensive discussion on networks and their applications. In the rest of the thesis, as suggested in Barabasi's book, I will use terms "graph" and "network" interchangeably, although, as explained in [5], strictly saying there is a difference in the meaning: "graph" refers to a mathematical object, while "network" is used to describe a real life system. However, since a graph is a visual representation of a network, we have that structurally these objects are the same, thus the terms are used as synonyms.

In the following, I will introduce some basic definitions from graph theory and provide simple illustrative examples.

A *graph* $G = (V, E)$ consists of a finite set of *vertices (nodes)* V and a set of *edges* between some (or all) pairs of vertices $E \subseteq V \times V$. The cardinality (or *order*) of a graph G is $p = |V|$, i.e., the number of nodes in G. Figure 2.1 shows an example of a graph where the vertices are cities, and the edges are the high-speed train connections. Note that I depicted the graph nodes according to their geographical position, however, from the mathematical viewpoint, the layout has no importance. When analyzing a graph, we consider only the nodes and the edge lists. Therefore, different graphical visualizations might represent the same graph G if they share the same vertex and edge lists (V, E).

Given a graph G, the edges can be *directed* – going from one node to another but not the other way – or *undirected*, without any direction, simply connecting the vertices. Formally speaking, an edge is undirected if $(i, j) \in E \Leftrightarrow (j, i) \in E$ or if there's no distinction between edges (i, j) and (j, i). An *undirected graph* is a graph with all edges undirected. The *size* $|E|$ of an undirected graph is the total number of its edges. An undirected graph G can be visually depicted by representing the nodes (subjects) V as points or dots and the edges (relationships) E as non-oriented lines joining the corresponding dots. For example, the graph in Figure 2.1 is undirected.

From here on, we will always assume that graph of interest is *simple*, meaning it is undirected, has no *self-loops*, i.e., edges (i, i) going from vertex to itself, and no multiple edges – for any pair of vertices $\{i, j\}$ edge (i, j) can be present in the set E at most once (which is implied by the given definition of the graph).

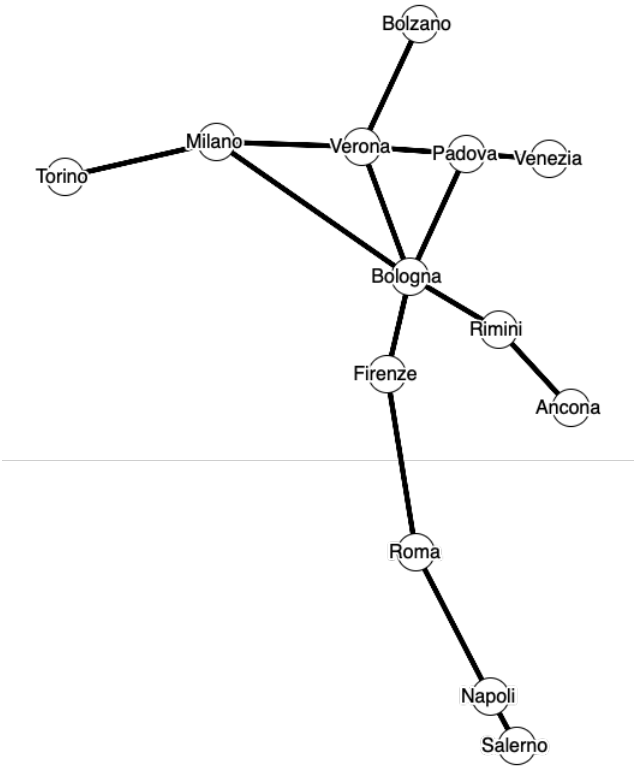


Figure 2.1: Map of some high-speed train routes in Italy as an example of a graph. Vertices correspond to cities, and edges represent routes.

Two vertices are called *adjacent* if they are connected by an edge. The *neighbours* of a node are all its adjacent vertices. Such set, *neighbourhood*, is denoted by $ne(i) = \{j \in V : (i, j) \in E\}$. A *closure* is the neighbourhood and the vertex itself, $cl(i) = ne(i) \cup \{i\}$. The *degree* d_i of the vertex i is the number of its neighbours, i.e., the number of its adjacent nodes.

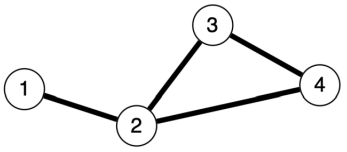
A *path* in a graph is a sequence of distinct vertices where each pair of subsequent nodes is joined by an edge, i.e. $\{j_1, \ldots, j_l\}$ such that $(j_m, j_{m+1}) \in E$ for $m = 1, \ldots, l-1$. A *connectivity component* is a set $V_{con} \subseteq V$ such that there exists a path between any two vertices of the set.

Figure 2.1 shows that the degree of $\{Milano\}$ vertex is equal to $d_M = 3$ and its neighborhood is $ne(Milano) = \{Torino, Verona, Bologna\}$. If we add the node itself, we will get its closure. An example of a path could be the set of edges $\{(Firenze, Bologna), (Bologna, Padova), (Padova, Venezia)\}$. The largest connectivity component here is the graph itself because there are no cities that we can not reach.

An important tool to represent a graph is the *adjacency matrix*: for a graph $G = (V, E)$ with p vertices, the adjacency matrix associated to G is the $p \times p$ matrix $\mathbf{A}$ with $A_{ij} = 1$ if vertices $\{i, j\}$ are connected and $A_{ij} = 0$ otherwise. Since for undirected graphs there is no distinction between edges (i, j) and (j, i), corresponding adjacency matrix is symmetric. In case of a simple graph, i.e. the one with no self-loops, the matrix's diagonal is zero, $A_{ii} = 0\ \forall i$. Since $\mathbf{A}$ is

real and symmetric, it has a complete set of real eigenvalues and an orthogonal eigenvector basis.

The size of the graph G is equivalent to the number of non-zero elements of the adjacency matrix, divided by two (since otherwise we count each edge twice): $|E| = \left(\sum_{i,j}^{p} A_{ij}\right)/2$. Also, the degree of any vertex i can be computed as the sum over the corresponding row or column, $d_i = \sum_{j=1}^{p} A_{ij} = \sum_{k=1}^{p} A_{ki}$. Moreover, we can prove that if G is a connected graph, then its adjacency matrix is irreducible. More in general, the multiplicity of the largest eigenvalue of $\mathbf{A}$ provides the number of connectivity components of G.



$$\iff \qquad \mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Figure 2.2: A graph and its adjacency matrix: entries of the matrix are equal to 1 if there is an edge between corresponding vertices of the graph and equal to 0 otherwise.

Any undirected graph can be encoded using its adjacency matrix. Vice versa, it is possible to reconstruct the graph from the adjacency matrix. Figure 2.2 shows an example of the correspondence between a graph and its adjacency matrix. In this case, nodes are labeled with vertex names, and the adjacency matrix rows/columns follow the natural order. However, when vertex denotes any entities, there is no natural order to give to the adjacency matrix. Therefore, we can permute rows and columns of the adjacency matrix without altering the graphs' topological structure. More precisely, let G_1 and G_2 be two undirected graphs and $\mathbf{A}_1$ and $\mathbf{A}_2$ – their adjacency matrices. We say that G_1 and G_2 are isomorphic if and only if there exists a permutation matrix $\mathbf{P}$ such that $\mathbf{A}_2 = \mathbf{P}\mathbf{A}_1\mathbf{P}^{-1}$. Therefore, without loss of generality, we assume that we can always find an isomorphism that reorders both rows and columns of the adjacency matrix according to our needs.

As mentioned before, the properties of the adjacency matrix (and of other matrices associated with a graph) are studied in a branch of graph theory called spectral graph theory. See "Spectral methods for graph clustering – A survey" by M. Nascimento and A. de Carvalho [55] for a survey on this topic and "Spectral Theory" by F. Chung [16] for in-depth discussion.

A *complete* graph is a graph where each node is connected to any other node. A *subgraph* G_{sub} of G is a graph over the subset of nodes $V_{sub} \subseteq V$ and the subset of edges $E_{sub} \subseteq E$ that connects nodes in V_{sub}. Subgraph can contain only the edges that are present in the original graph but may lack some of them. A *clique* C is a complete subgraph of G. A *maximal clique* is a clique that is not a subgraph of any other larger (in the number of vertices) clique.

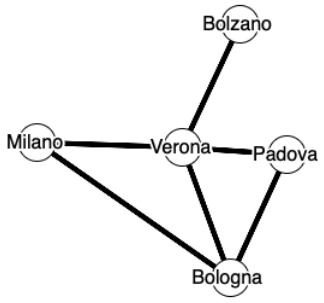


Figure 2.3: A subgraph of the graph in figure 2.1.

Figure 2.3 shows an example of a subgraph of the graph in Figure 2.1. The subgraph is not complete, since there are no edges connecting, for example, the cities $\{Milano\}$ and $\{Padova\}$ or $\{Bolzano\}$ and $\{Bologna\}$. Examples of cliques here are $\{Bolzano, Verona\}$, or $\{Milano, Verona, Bologna\}$, or else $\{Verona, Bologna, Padova\}$. An example of maximal clique here is the subgraph connecting $\{Milano, Verona, Bologna\}$.

Given three sets of vertices $A, B, C \subseteq V$, we say that C *separates* A and B and call set C a *cut set*, if any path from a node in the set A to a node in the set B contains at least one node from the set C. In Fig. 2.3 the set $C = \{Bolzano, Verona, Bologna\}$ is an example of the cut set as it separates sets $A = \{Milano\}$ and $B = \{Padova\}$. Indeed, you can not get from city A to city B without going through one of the cities in C.

In the next chapters of the thesis, we will focus our attention on *sparse* graphs. They are ubiquitous in several real applications where the number of nodes is usually huge, and the number of edges is far from the one of a complete graph. However, there is no precise definition of sparse graphs. Suggestions vary from calling a graph sparse if it has about as many edges as its number of vertices – p – to calling it sparse as long as the number of edges is less than a half of all possible ones – so $\leq p(p-1)/4$. In this thesis, we will assume that a sparse graph has *few* edges, and hence its adjacency matrix is a sparse matrix. In the simulation described in Chapter 5 the highest used sparsity is about 4% of all possible edges.

2.2 Conditional dependence

In many situations, conditional dependence is a more suited concept to represent relationships between variables than marginal dependence. An example from the [34] considers the following experiment: when it snows, it is common to observe huge traffic jams and many snowmen in the parks. If we only observe the number of traffic jams and the number of snowmen, we could conclude that there is a positive correlation between these two variables. However, if we consider both events as a part of a system with more variables, we immediately

realize that correlation occurs because the snowfall causes both events. Conditionally on the snowfall, the traffic jams' size and the number of snowmen are likely to be independent. While correlation fails to represent the relationship in this case accurately, conditional dependency better describes direct links between variables.

I will provide a precise definition of conditional independence and some of its essential properties in the following. The aim is to pave the route to graph theory concepts together with that of conditional independence.

Suppose $\mathbf{x} = (X_1, \ldots, X_p) \in \mathbb{R}^{1\times p}$ is a row random vector and its distribution has a positive density f. Two random variables X_i, X_j are *conditionally independent* given the rest of the variables $X_{\{l, l\neq i, j\}}$, and we denote $X_i \perp\!\!\!\perp X_j | X_{\{l, l\neq i, j\}}$, if and only if

$$f(X_i, X_j | X_{\{l, l\neq i, j\}}) = f(X_i | X_{\{l, l\neq i, j\}}) f(X_j | X_{\{l, l\neq i, j\}})$$

or, in other words, $f(X_i | X_{\{l, l\neq i, j\}})$ does not depend on X_j, given the other variables $X_{\{l, l\neq i, j\}}$. It means that variable X_j is irrelevant for prediction of X_i if all other variables of the vector are known. The concept of conditional independence allows to investigate not only whether variables are related (this can be described by the marginal dependence) but also if there is a direct effect from one to the other.

In the following, I will present some properties of conditional independence that may be of use in this thesis. I will omit the proofs since they follow from the definition of conditional independence.

Proposition 1. *If X, Y, Z are jointly distributed random variables, $X \perp\!\!\!\perp Y | Z$, and $h(\cdot)$ is a measurable function on the sample space of X, then the following properties hold:*

(P1) $Y \perp\!\!\!\perp X | Z$;

(P2) $h(X) \perp\!\!\!\perp Y | Z$;

(P3) $X \perp\!\!\!\perp Y | (Z, h(X))$.

Proposition 2 (P4)**.** *For jointly distributed random variables X, Y, Z and some functions $g(\cdot)$, $h(\cdot)$ it holds*

$$X \perp\!\!\!\perp Y | Z \iff f(x, y, x) = g(x, z) h(y, z).$$

S.L. Lauritzen [46] suggests a nice view on the meaning behind Proposition 1. Let us interpret the random variables as pieces of knowledge obtained from the books and thus understand $X \perp\!\!\!\perp Y | Z$ as "Knowing Z, reading Y is irrelevant for reading X". Then we can rewrite the properties (P1)-(P3) as

(P1) if, knowing Z, reading Y is irrelevant for reading X, then so is reading X for reading Y;

(P2) if, knowing Z, reading Y is irrelevant for reading X, then reading Y is irrelevant for reading any chapter $h(X)$ of the book X;

(P3) if, knowing Z, reading Y is irrelevant for reading X, it remains irrelevant after reading any chapter $h(X)$ of the book X.

2.3 Undirected graphical models

In this section, I will join the concepts of undirected graphs and conditional independence through graphical models.

Assume that we are given a row random vector $\mathbf{x} = (X_1, \ldots, X_p) \in \mathbb{R}^{1\times p}$. Then, we can associate each variable X_i with a vertex i of an undirected graph $G = (V, E)$ with $V = \{1, \ldots, p\}$. There are two approaches to using the structure of the graph G to describe conditional dependence relationships between variables of the random vector $\mathbf{x}$, namely Markov properties and factorization property.

2.3.1 Markov properties

We say that a graph G is an *undirected graphical model or a Markov random field* with respect to the random vector $\mathbf{x}$, if it satisfies one of the following Markov properties:

(G) *Global Markov property*: separation of the sets of vertices implies conditional independence of the sets of variables or

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

Note that Markov chains are a particular illustration of this property. Chain graph, corresponding to a Markov chain, has an edge set $E = \{(1,2),(2,3),\ldots,(p-1,p)\}$. In such graph any single vertex j separates the set $\{1,\ldots,j-1\}$ (past) and the set $\{j+1,\ldots,p\}$ (future). It directly corresponds to a Markov chain's definition: the future is conditionally independent of the past given the present.

(L) *Local Markov property*: any variable X_i is conditionally independent of the rest of the variables given its neighbors or

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

(P) *Pairwise Markov property*: lack of an edge between vertices i and j implies conditional independence between variables X_i and X_j or

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

In general, these three Markov properties defined above do not necessarily coincide (see examples 3.5 and 3.6 in [46]). Nevertheless, they are related as described in the following proposition:

Proposition 3 (Lauritzen [46], p. 33)**.** *For any undirected graph G and any probability distribution on* $\mathbf{x}$ *it holds that*

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

Proof. Assume that for graph G global Markov property (G) holds. Let's denote $A = \{i\}$, $B = V\backslash(ne(i) \cup \{i\})$, $C = ne(i)$ for any vertex i. Then C separates A and B and according to (G), $X_A \perp\!\!\!\perp X_B|X_C$ which is exactly $X_i \perp\!\!\!\perp X_{V\backslash(ne(i)\cup\{i\})}|X_{ne(i)}$, so (L) holds.

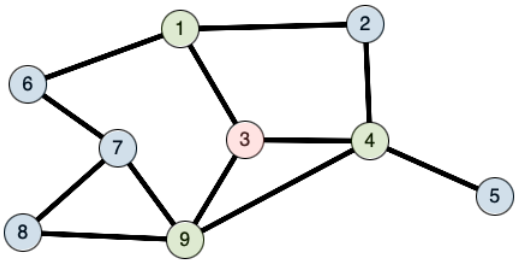


Figure 2.4: Illustration for the proof of $(G) \Rightarrow (L)$. If $i = 3$, then set $A = \{i\} = \{3\}$ (shown in red), set $C = ne(i) = \{1, 4, 9\}$ (green) and set $B = V\backslash(ne(i) \cup \{i\}) = \{2, 5, 6, 7, 8\}$ (blue). Set C separates sets A and B.

Now assume that (L) holds, meaning $X_i \perp\!\!\!\perp X_{V\backslash(ne(i)\cup\{i\})}|X_{ne(i)} = X_i \perp\!\!\!\perp Y|Z$. If there's a vertex j from $V\backslash(ne(i) \cup \{i\})$, i.e. such that there's no edge between i and j, then

$$\Big(\big(V\backslash(ne(i) \cup \{i\})\big)\backslash\{j\}\Big) \cup ne(i) = V\backslash\{i, j\}.$$

Let's define function $h(Y)$ as removing the variable X_j from the set, i.e.

$$h(Y) = h(X_{V\backslash(ne(i)\cup\{i\})}) = X_{\left(V\backslash(ne(i)\cup\{i\})\right)\backslash\{j\}}.$$

Then $W = (Z, h(Y))$ would be the union of Z and $h(Y)$ sets, namely

$$(Z, h(Y)) = (X_{ne(i)}, X_{\left(V\backslash(ne(i)\cup\{i\})\right)\backslash\{j\}}) = X_{V\backslash\{i,j\}}.$$

From the property (P3) we get

$$X_i \perp\!\!\!\perp Y|Z \Rightarrow X_i \perp\!\!\!\perp Y|(Z, h(Y))$$
$$X_i \perp\!\!\!\perp X_{V\backslash(ne(i)\cup\{i\})}|X_{ne(i)} \Rightarrow X_i \perp\!\!\!\perp X_{V\backslash(ne(i)\cup\{i\})}|X_{V\backslash\{i,j\}}$$

Now if we define function $g(Y)$ as selecting the variable X_j from the set, i.e.

$$g(Y) = g(X_{V\backslash(ne(i)\cup\{i\})}) = X_j,$$

we can use property (P2) to get

$$X_i \perp\!\!\!\perp Y|W \Rightarrow X_i \perp\!\!\!\perp h(Y)|W \Rightarrow$$
$$X_i \perp\!\!\!\perp X_{V\backslash(ne(i)\cup\{i\})}|X_{V\backslash\{i,j\}} \Rightarrow X_i \perp\!\!\!\perp X_j|X_{V\backslash\{i,j\}}$$

So for two vertices i and j absence of an edge implied conditional independence of corresponding variables given all the others, which is exactly pairwise Markov property (P). □

Note here that, under some mild conditions, the reverse implications also hold, hence, all three Markov properties become equivalent. A well-known example is when the probability measure is strictly positive (which is the case for Gaussian distribution, for example). This result was originally stated in Pearl and Paz, and the proof can be found in Lauritzen [46] (Theorem 3.7). In the following, I will discuss a stronger result – involving the factorization property.

2.3.2 Factorization property

An alternative approach to connect the probabilistic and graphical structures is based on factorization. If $f(\mathbf{x})$ is the density function of vector's $\mathbf{x}$ probability measure $\mathbb{P}$, we say that it *factorizes* or *satisfies the factorization property (F)* with respect to graph G if

$$f(\mathbf{x}) = \prod_{C \in \mathcal{C}} \phi_C(X_C),$$

where $\mathcal{C}$ is the set of all maximal cliques (complete subgraphs that are not a part of a larger complete subgraph) and $\phi_C(X_C)$ are the non-negative real-valued *potential or compatibility functions* depending only on variables corresponding to each clique.

According to the graph structure, factorization provides a way to compactly represent a distribution over a large number of variables.

2.3.3 Equivalence between Markov properties and factorization property

In general setting, factorization property implies Markov properties:

Proposition 4. *For any undirected graph G and any probability distribution on $\mathbf{x}$ it holds that*

$$(F) \Rightarrow (G) \Rightarrow (L) \Rightarrow (P).$$

Proof. We only have to show that $(F) \Rightarrow (G)$ since other implications have been proved in Proposition 3. So let's assume that (F) holds, meaning that density function factorizes over maximum cliques $f(\mathbf{x}) = \prod_{C \in \mathcal{C}} \phi_C(X_C)$.

Let A, B, S be any triple of disjoint subsets of V such that S separates A and B. Consider $G_{V \setminus S}$ and denote $\bar{A}$ the connectivity components in it which contain A. We denote the rest of the vertices $\bar{B} = V \setminus (\bar{A} \cup S)$.

Since A and B are separated by S, their elements are in different connectivity components of $G_{V \setminus S}$ and any clique of G is either in $\bar{A} \cup S$ or in $\bar{B} \cup S$. It means that by factorization property

$$f(X) = \prod_{C \in \mathcal{C}} \phi_C(X_C) = \prod_{C \in \mathcal{C}_{\bar{A} \cup S}} \phi_C(X_C) \prod_{C \in \mathcal{C}_{\bar{B} \cup S}} \phi_C(X_C) =$$
$$= g(X_{\bar{A} \cup S}) h(X_{\bar{B} \cup S}) = g(X_{\bar{A}}, X_S) h(X_{\bar{B}}, X_S).$$

By property (P4) we get that such factorization implies $X_{\bar{A}} \perp\!\!\!\perp X_{\bar{B}} | X_S$. If we now define function $k(\cdot)$ as getting the complement of the set, then by applying it to both variables and using the (P2) property, we would get $X_A \perp\!\!\!\perp X_B | X_S$ for sets A and B which are separated by set S – which is exactly global Markov property. □

In the case of a strictly positive probability measure, all these properties become equivalent.

Theorem 1 (Hammersley and Clifford)**.** *A probability distribution of a random vector* $\mathbf{x}$ *with positive and continuous density* f *satisfies the pairwise Markov property for undirected graph* G *if and only if it factorizes according to* G*.*

The proof can be found in Lauritzen [46] (Theorem 3.9).

2.4 Gaussian graphical models

A well-known graphical model is the Gaussian graphical model (GGM), i.e., a graph encoding a multivariate Gaussian distribution. This distribution is often used for modeling continuous variables in various applications as it is well-studied and holds a lot of convenient properties.

Let $\mathbf{x} = (X_1, \ldots, X_p) \in \mathbb{R}^{1\times p}$ follow a p-variate Gaussian distribution $\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu} \in \mathbb{R}^{1\times p}$ and non-singular covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$:

$$\mathbb{P}(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\det(\boldsymbol{\Sigma})^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T},$$

where $\det(\cdot)$ is the determinant of the matrix. Without loss of generality we can also assume $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, i.e., the mean $\boldsymbol{\mu} = \mathbf{0}_p$ or, equivalently, that random variables are centered.

2.4.1 A useful property of Gaussian distribution

Let $A = \{1, \ldots, k\}$ and $B = \{1, \ldots, p\}\backslash A$ be two subsets of variables constituting a partition of the random vector $\mathbf{x} = (X_1, \ldots, X_p)$. Without loss of generality we can reorder the variables and denote the random vector as $\mathbf{x} = (X_1, \ldots, X_p) = (X_A, X_B)$ with the same $\mathcal{N}(0, \boldsymbol{\Sigma})$ distribution.

We assume that $\boldsymbol{\Sigma}$ is non-singular and with $\boldsymbol{\Omega}$ denote its inverse, the $p\times p$ *precision* (or *concentration*) matrix. Therefore, we have $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Omega}_{AA} & \boldsymbol{\Omega}_{AB} \\ \boldsymbol{\Omega}_{BA} & \boldsymbol{\Omega}_{BB} \end{pmatrix}$. Note that both $\boldsymbol{\Sigma}$ and $\boldsymbol{\Omega}$ are symmetric, implying that, for example, $\boldsymbol{\Omega}_{AA}^T = \boldsymbol{\Omega}_{AA}$ or $\boldsymbol{\Omega}_{AB}^T = \boldsymbol{\Omega}_{BA}$. Note also that $\boldsymbol{\Omega}_{AA}^{-1}$ denotes the inverse of $\boldsymbol{\Omega}_{AA}$, not the $(\boldsymbol{\Omega}^{-1})_{AA} = \boldsymbol{\Sigma}_{AA}$.

With these notations, we will now introduce a useful property of Gaussian distribution.

Proposition 5 (Giruad [34]). *In the described setting, the conditional distribution of X_A given X_B is Gaussian $\mathcal{N}(-\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}X_B^T, \mathbf{\Omega}_{AA}^{-1})$. In other words, we have the decomposition*

$$X_A = -X_B\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB} + \varepsilon_A, \quad \textit{where} \quad \varepsilon_A \sim \mathcal{N}(0, \mathbf{\Omega}_{AA}^{-1}) \textit{ is independent of } X_B.$$

Proof. Let us denote $f(x_A, x_B)$ – density of the distribution of X, $f(x_B)$ – of X_B, and $f(x_A|x_B)$ – of X_A given X_B. By definition

$$\begin{aligned} f(x_A|x_B) =& \frac{f(x_A, x_B)}{f(x_B)} = \\ =& \frac{\frac{1}{(2\pi)^{p/2}(\det \mathbf{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_A & x_B \end{pmatrix} \mathbf{\Sigma}^{-1} \begin{pmatrix} x_A^T \\ x_B^T \end{pmatrix}\right)}{\frac{1}{(2\pi)^{\frac{p-k}{2}}(\det \mathbf{\Sigma}_{BB})^{1/2}} \exp\left(-\frac{1}{2} x_B \mathbf{\Sigma}_{BB}^{-1} x_B^T\right)} = \\ =& \frac{1}{(2\pi)^{k/2}} \frac{(\det \mathbf{\Sigma}_{BB})^{1/2}}{(\det \mathbf{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_A & x_B \end{pmatrix} \begin{pmatrix} \mathbf{\Omega}_{AA} & \mathbf{\Omega}_{AB} \\ \mathbf{\Omega}_{BA} & \mathbf{\Omega}_{BB} \end{pmatrix} \begin{pmatrix} x_A^T \\ x_B^T \end{pmatrix} + \right. \\ & \left. \frac{1}{2} x_B \mathbf{\Sigma}_{BB}^{-1} x_B^T \right). \end{aligned}$$

Let's first consider the fraction with determinants. By properties of block matrices, for $\mathbf{\Sigma}$ and $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ it is true that

$$\begin{aligned} \det \mathbf{\Sigma} &= \det \begin{pmatrix} \mathbf{\Sigma}_{AA} & \mathbf{\Sigma}_{AB} \\ \mathbf{\Sigma}_{BA} & \mathbf{\Sigma}_{BB} \end{pmatrix} = \det \mathbf{\Sigma}_{BB} \det\left(\mathbf{\Sigma}_{AA} - \mathbf{\Sigma}_{AB}\mathbf{\Sigma}_{BB}^{-1}\mathbf{\Sigma}_{BA}\right) \\ \mathbf{\Omega}_{AA}^{-1} &= \mathbf{\Sigma}_{AA} - \mathbf{\Sigma}_{AB}\mathbf{\Sigma}_{BB}^{-1}\mathbf{\Sigma}_{BA} \\ \mathbf{\Sigma}_{BB}^{-1} &= \mathbf{\Omega}_{BB} - \mathbf{\Omega}_{BA}\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB} \end{aligned}$$

By first two equalities the fraction becomes

$$\frac{(\det \mathbf{\Sigma}_{BB})^{1/2}}{(\det \mathbf{\Sigma})^{1/2}} = \frac{1}{\left(\det\left(\mathbf{\Sigma}_{AA} - \mathbf{\Sigma}_{AB}\mathbf{\Sigma}_{BB}^{-1}\mathbf{\Sigma}_{BA}\right)\right)^{1/2}} = \frac{1}{(\det \mathbf{\Omega}_{AA}^{-1})^{1/2}}.$$

Now let's move to the power of the exponent:

$$\begin{aligned} & -\frac{1}{2}\begin{pmatrix} x_A & x_B \end{pmatrix} \begin{pmatrix} \mathbf{\Omega}_{AA} & \mathbf{\Omega}_{AB} \\ \mathbf{\Omega}_{BA} & \mathbf{\Omega}_{BB} \end{pmatrix} \begin{pmatrix} x_A^T \\ x_B^T \end{pmatrix} + \frac{1}{2} x_B \mathbf{\Sigma}_{BB}^{-1} x_B^T = \\ & -\frac{1}{2}\left(x_A\mathbf{\Omega}_{AA}x_A^T + x_B\mathbf{\Omega}_{BA}x_A^T + x_A\mathbf{\Omega}_{AB}x_B^T + x_B\mathbf{\Omega}_{BB}x_B^T\right) + \frac{1}{2} x_B \mathbf{\Sigma}_{BB}^{-1} x_B^T = \\ & -\frac{1}{2}(x_A\mathbf{\Omega}_{AA}x_A^T + x_B\mathbf{\Omega}_{BA}x_A^T + x_A\mathbf{\Omega}_{AB}x_B^T + x_B(\mathbf{\Omega}_{BB} - \mathbf{\Sigma}_{BB}^{-1})x_B^T) \end{aligned}$$

From the third result for block matrices we obtain $\mathbf{\Omega}_{BB} - \mathbf{\Sigma}_{BB}^{-1} = \mathbf{\Omega}_{BA}\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}$. We can substitute this into the previous equality and then insert the term $\mathbf{\Omega}_{AA}\mathbf{\Omega}_{AA}^{-1}$ into some of the addends:

$$
\begin{aligned}
&-\frac{1}{2}(x_A\mathbf{\Omega}_{AA}x_A^T + x_B\mathbf{\Omega}_{BA}x_A^T + x_A\mathbf{\Omega}_{AB}x_B^T + x_B\mathbf{\Omega}_{BA}\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}x_B^T) = \\
&-\frac{1}{2}\big(\underbrace{x_A}_{u}\underbrace{\mathbf{\Omega}_{AA}}_{\mathbf{W}}x_A^T + \underbrace{x_B\mathbf{\Omega}_{AB}^T\mathbf{\Omega}_{AA}^{-1}}_{v}\mathbf{\Omega}_{AA}x_A^T + x_A\mathbf{\Omega}_{AA}\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}x_B^T + \\
&\qquad\qquad x_B\mathbf{\Omega}_{AB}^T\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AA}\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}x_B^T\big) = \\
&-\frac{1}{2}\left(u\mathbf{W}u^T + v\mathbf{W}u^T + u\mathbf{W}v^T + v\mathbf{W}v^T\right) = -\frac{1}{2}(u+v)\mathbf{W}(u+v)^T = \\
&-\frac{1}{2}\left(x_A + x_B\mathbf{\Omega}_{AB}\mathbf{\Omega}_{AA}^{-1}\right)\mathbf{\Omega}_{AA}\left(x_A + x_B\mathbf{\Omega}_{AB}\mathbf{\Omega}_{AA}^{-1}\right)^T
\end{aligned}
$$

Combining results for the determinant and for the exponent, we see that

$$
\begin{aligned}
f(x_A|x_B) =& \frac{f(x_A, x_B)}{f(x_B)} = \\
=& \frac{1}{(2\pi)^{k/2}}\frac{1}{(\det\mathbf{\Omega}_{AA}^{-1})^{1/2}}\times \\
& \exp\left(-\frac{1}{2}\left(x_A + x_B\mathbf{\Omega}_{AB}\mathbf{\Omega}_{AA}^{-1}\right)\mathbf{\Omega}_{AA}\left(x_A + x_B\mathbf{\Omega}_{AB}\mathbf{\Omega}_{AA}^{-1}\right)\right)
\end{aligned}
$$

which is exactly the density of Gaussian distribution $\mathcal{N}(-X_B\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}, \mathbf{\Omega}_{AA}^{-1})$. □

From this theorem we can derive another useful property which will be strategic for the next chapters of this thesis.

Proposition 6. *If* $\mathbf{x} = (X_1, \ldots, X_p) \sim \mathcal{N}(0, \mathbf{\Sigma})$*, then distribution of any* X_j*,* $j = 1, \ldots, p$*, given the other variables is still Gaussian with mean and variance given by*

$$
\mathbb{E}(X_j|X_{\{l,l\neq j\}}) = -\sum_{i\neq j}\frac{\Omega_{ij}}{\Omega_{jj}}X_i \quad \textit{and} \quad var(X_j|X_{\{l,l\neq j\}}) = \Omega_{jj}^{-1}.
$$

Proof. If we consider the set of one variable $A = \{j\}$ and a set of all the other variables $B = \{1, \ldots, p\}\backslash A$, then by the previous proposition the distribution of $X_A = X_j$ given X_B is $\mathcal{N}(-X_B\mathbf{\Omega}_{AA}^{-1}\mathbf{\Omega}_{AB}, \mathbf{\Omega}_{AA}^{-1}) = \mathcal{N}(-X_B\Omega_{jj}^{-1}\mathbf{\Omega}_{AB}, \Omega_{jj}^{-1})$. By some algebra we have that $-X_B\Omega_{jj}^{-1}\mathbf{\Omega}_{AB}$ is exactly $-\sum_{i\neq j}X_i\frac{\Omega_{ji}}{\Omega_{jj}}$. Since $\mathbf{\Omega}$ is symmetric, we can equivalently write the decomposition in the following form:

$$
X_j = -\sum_{i:i\neq j}\frac{\Omega_{ij}}{\Omega_{jj}}X_i + \varepsilon_j \quad \text{where} \quad \varepsilon_j \sim \mathcal{N}(0, \Omega_{jj}^{-1}) \text{ is independent of } X_B.
$$

□

Proposition 6 is of great importance for the development of this work since it shows how any variable in a Gaussian vector can be interpreted as linear combination of the remaining variables plus some noise. The coefficients of the linear combination $-\Omega_{ij}/\Omega_{jj}$ are proportional to the entries Ω_{ij} of the precision matrix, and the variance of the noise is inversely proportional to the corresponding diagonal element Ω_{jj}.

2.4.2 Connection with the precision matrix

The great advantage of associating Gaussian distribution with the graphical model G is the connection between the precision matrix $\mathbf{\Omega}$ (which contains partial covariances of the variables) and the adjacency matrix $\mathbf{A}$ of the graph G (a matrix with entries equal to 1 if there is an edge between corresponding nodes of the graph and equal to 0 otherwise, see Fig 2.5). Namely, we have $\operatorname{supp} \mathbf{\Omega} = \mathbf{A}$, i.e., the positions of the non-zero elements of the precision matrix $\mathbf{\Omega}$ coincide with those of $\mathbf{A}$. This result, proved in Theorem 2, is essential since it will allow us to move from the graph inference to the equivalent problem of estimating the support of the precision matrix – we do not even need to estimate the precision matrix entries, but it is sufficient to identify only the position of the non-zero elements.

$$\mathbf{\Omega} = \begin{pmatrix} 0 & 0.67 & 0 & 0 \\ 0.67 & 0 & 0.8 & -0.39 \\ 0 & 0.8 & 0 & 0.75 \\ 0 & -0.39 & 0.75 & 0 \end{pmatrix} \iff \mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \iff$$

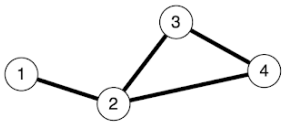


Figure 2.5: Correspondence between precision matrix $\mathbf{\Omega}$, adjacency matrix $\mathbf{A}$ and graph G in the Gaussian graphical model.

Theorem 2 (GGM and precision matrix)**.** *If* $\mathbf{x} = (X_1, \ldots, X_p) \sim \mathcal{N}(0, \mathbf{\Sigma})$ *and* G *is a graphical model with respect to* $\mathbf{x}$*, then any pair of vertices* $\{i, j\}$ *is adjacent* $\iff \Omega_{ij} \neq 0$*, being* $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$*.*

Proof. For a pair of vertices i, j and corresponding variables X_i, X_j let's denote $A = \{i, j\}$ and $B = V \backslash \{i, j\}$. From Proposition 5 we get that the covariance matrix of the conditional distribution of X_i, X_j given $X_{V\backslash\{i,j\}}$ is

$$cov(X_A|X_B) = \mathbf{\Omega}_{AA}^{-1} = \begin{pmatrix} \Omega_{ii} & \Omega_{ij} \\ \Omega_{ij} & \Omega_{jj} \end{pmatrix}^{-1} = \frac{1}{\Omega_{ii}\Omega_{jj} - \Omega_{ij}^2} \begin{pmatrix} \Omega_{jj} & -\Omega_{ij} \\ -\Omega_{ij} & \Omega_{ii} \end{pmatrix} \tag{2.1}$$

Also, by definition of conditional correlation

$$cor(X_i, X_j|X_B) = \frac{cov(X_i, X_j|X_B)}{\sqrt{var(X_i|X_B)var(X_j|X_B)}} \tag{2.2}$$

If vertices i, j are not connected, then by local Markov property $X_i \perp\!\!\!\perp X_j|X_B$. Therefore $cor(X_i, X_j|X_B) = 0$. Then by Eq.(2.2) we get $cov(X_i, X_j|X_B) = 0$

which by Eq.(2.1) implies $\Omega_{ij} = 0$. The same holds for reverse since for Gaussian variables no correlation is equivalent to independence. □

While the spectral graph theory graph allows studying the mathematical properties of graphs and network analysis uses graphs for exploring complex systems, graphical models allow inferring graph structures (i.e., the conditional dependency) from observed data. In the next chapter, I will discuss the inference of Gaussian graphical models – i.e., learning the graph of conditional dependencies between p jointly distributed Gaussian variables from a set of data of such variables. I will specifically focus on the case of sparse graphs with a large number of nodes and only a few edges. For these networks, I will consider the setting where the sample size is smaller than the number of variables. This scenario is typically encountered in several real data applications, such as analyzing omics data and estimating a gene regulatory network from gene expression data. In this case, the high throughput technology's progress made it possible to measure tens of thousands or millions of variables simultaneously (i.e, large p). However, since each experiment's cost is still high, only a limited number of experiments (say some dozens or hundreds at best) are usually collected (i.e., small n).

Chapter 3

Graphical models inference

In the previous chapter, I discussed graph theory and introduced graphical models, showing how a graph can encode the conditional dependency among variables. In this chapter, I will discuss how to infer such graph from observed data.

More specifically, given a dataset $\mathbf{X}$ of N samples over p variables, the problem is to find a graphical model that encodes the conditional dependence structure among the variables. We have to face the following questions: How could we use the data to estimate the underlying graph with a high probability of being correct? How can we deal with the high-dimensionality of the data and the curse of dimensionality? How can we have fast and accurate algorithms when the dimensions of the data increase? How can we extend the methods when multiple datasets are available?

In general, learning a graph from observed data is a very challenging problem. This thesis will focus on datasets under the Gaussian distribution with zero means (without loss of generality since we can always center the data) and unknown covariance matrices. In this case, as explained in Chapter 2, the precision matrix encodes the conditional dependencies. Therefore, inferring the graphical model is equivalent to estimating the non-zero entries of the precision matrix.

Due to the importance of the problem, several classes of methods have been developed to estimate Gaussian graphical models, each with its pro and cons. After a brief introduction, I will discuss two popular approaches widely used in the literature to estimate a sparse graph in high-dimensional settings: the penalized maximum likelihood approach and the approach based on penalized regression. I will first deal with Gaussian graphical models inference in the classical context with a single data matrix $\mathbf{X}$. Then, I will expand the presentation to the case of several data matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$.

3.1 The sparsity assumption

Consider a given data matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$ of N independent and identically distributed samples from a p-variate Gaussian distribution, i.e. each row vector $\mathbf{x}_i = (X_{i1}, \ldots, X_{ip}) \sim \mathcal{N}_p(0, \mathbf{\Sigma})$. How can we now estimate the unknown

underlying graph structure G? Especially in the high-dimensional case, when number of variables p is much larger than number of samples N? Is there any assumptions we can make to derive solvable mathematically but yet adequate and not too simplified models?

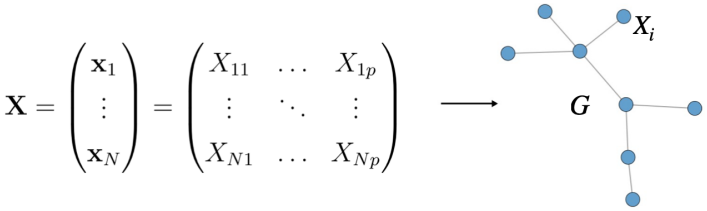


Figure 3.1: How can we infer the underlying graph structure, given a data matrix of N samples from the p-variate Gaussian distribution?

As explained in the book "Statistical Learning with Sparsity: The Lasso and Generalizations" by T. Hastie, R. Tibshirani and M. Wainwright [37], our best "bet" is to assume that the correct model is sparse. If it is not, then the number of samples is too small to estimate the parameters accurately. However, if the correct underlying model is, in fact, sparse, it means that only $m << p$ parameters are non-zero. Thus, the proportion of "samples to variables" is decreased, giving us a better chance for estimating the model.

Moreover, the sparsity assumption helps the overall interpretability of the model. When the number of variables is of the order of hundreds or thousands, inferring too many connections can only decrease the usefulness of the estimator since it seems impossible to derive meaningful conclusions from it. Such considerations are valid for different statistical models, from regression to classification, and were continuously adopted in the high-dimensional statistics in the last few decades. In the context of graphs and conditional dependence, the fewer edges we have – the easier it is for us to analyze and interpret the conditional correlation among variables. Of course, there is a balance of not destroying the possibly important relations when enforcing the sparse estimate. However, as in many statistics applications, it is a trade-off one has to make.

To exploit the sparsity assumption, we need to constrain or regularize the estimation. In the following two sections focusing on two popular estimation approaches, we will first describe the classical problem for estimating a Gaussian graphical model and then introduce its regularization. Meanwhile, we recommend "Covariance selection" paper by A.P. Dempster [22] as a seminal work on sparse estimation.

3.2 Penalized likelihood

This section will first derive the likelihood for the Gaussian graphical model in a general case. It will then introduce the penalized minimization problem under the sparsity assumption and describe the *graphical lasso* algorithm for its solution.

3.2.1 Likelihood for Gaussian graphical models

We consider a given data matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$ of N independent and identically distributed samples from the p-variate Gaussian distribution, i.e. each row vector $\mathbf{x}_i = (X_{i1}, \ldots, X_{ip}) \sim \mathcal{N}_p(0, \mathbf{\Sigma})$. For this data matrix, it is possible to write the likelihood in terms of the precision matrix $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$. In order to do this, we note that for any matrix $\mathbf{A}$ with $\det \mathbf{A} \neq 0$ it holds that $\det(\mathbf{A}^{-1}) = (\det \mathbf{A})^{-1}$. Then, for the Gaussian distribution, we have

$$\mathbb{P}(\mathbf{x}_i) = \frac{\det(\mathbf{\Omega})^{1/2}}{(2\pi)^{p/2}} \exp\left(-\frac{1}{2}\mathbf{x}_i \mathbf{\Omega} \mathbf{x}_i^T\right).$$

Therefore, by definition, the likelihood function for $\mathbf{X}$ is

$$\mathcal{L}(\mathbf{\Omega}) = \prod_{i=1}^{N} \frac{\det(\mathbf{\Omega})^{1/2}}{(2\pi)^{p/2}} \exp\left(-\frac{1}{2}\mathbf{x}_i \mathbf{\Omega} \mathbf{x}_i^T\right) = \frac{\det(\mathbf{\Omega})^{N/2}}{(2\pi)^{pN/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{\Omega} \mathbf{x}_i^T\right). \tag{3.1}$$

Denoting $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i^T \mathbf{x}_i$ – the empirical covariance matrix, the expression in the power of the exponent in Eq.(3.1) can be rewritten as $-\frac{N}{2}\operatorname{tr}(\mathbf{S}\mathbf{\Omega})$. After taking the logarithm of the likelihood function, dropping-off the 2π term (since it does not depend on $\mathbf{\Omega}$) and rescaling the equation by $N/2$, we have

$$\begin{aligned}
\log \mathcal{L}(\mathbf{\Omega}) =& \frac{N}{2}\log\det\mathbf{\Omega} - \underbrace{\frac{pN}{2}\log 2\pi}_{\text{no } \mathbf{\Omega}} - \frac{1}{2}\underbrace{\sum_{i=1}^{N} \mathbf{x}_i \mathbf{\Omega} \mathbf{x}_i^T}_{\operatorname{tr}(\mathbf{X}\mathbf{\Omega}\mathbf{X}^T)} \Rightarrow \\
& \frac{N}{2}\log\det\mathbf{\Omega} - \frac{1}{2}\underbrace{\operatorname{tr}(\mathbf{X}\mathbf{\Omega}\mathbf{X}^T)}_{\operatorname{tr}(\mathbf{X}^T\mathbf{X}\mathbf{\Omega})} \Rightarrow \\
& \frac{N}{2}\log\det\mathbf{\Omega} - \frac{N}{2}\operatorname{tr}\left(\frac{\mathbf{X}^T\mathbf{X}}{N}\mathbf{\Omega}\right) \Rightarrow \\
\log \mathcal{L}(\mathbf{\Omega}) =& \log\det\mathbf{\Omega} - \operatorname{tr}(\mathbf{S}\mathbf{\Omega})
\end{aligned} \tag{3.2}$$

Note that $\log\det\mathbf{\Omega}$ is defined on the space of symmetric matrices as

$$\log\det\mathbf{\Omega} = \begin{cases} -\sum_{j=1}^{p}\log\eta_j(\mathbf{\Omega}) & \text{if } \mathbf{\Omega} \succeq 0 \\ +\infty & \text{otherwise} \end{cases}$$

where $\eta_j(\mathbf{\Omega})$, $j = 1 \ldots p$, are the eigenvalues of matrix $\mathbf{\Omega}$ and we use $\succeq 0$ to denote a positive definite matrix.

The maximum likelihood (or log-likelihood) estimator therefore is defined as

$$\widehat{\mathbf{\Omega}} = \underset{\mathbf{\Omega}\succeq 0}{\arg\max}\left\{\log\det\mathbf{\Omega} - \operatorname{tr}(\mathbf{S}\mathbf{\Omega})\right\}. \tag{3.3}$$

To evaluate the derivative of Eq.(3.2), we denote M_{ij} – the minor associated with ij-th matrix element, i.e., the determinant of the matrix obtained by removing i-th row and j-th column, and $\tilde{\mathbf{\Omega}} = (M_{ij})$ – the $p \times p$ matrix of cofactors. Then the derivative of the first term in Eq. (3.2) can be calculated as follows

$$\begin{aligned}(\log\det\mathbf{\Omega})' = \frac{1}{\det\mathbf{\Omega}}(\det\mathbf{\Omega})' = \frac{1}{\det\mathbf{\Omega}}\frac{\partial}{\partial\Omega_{ij}}\left(\sum_{j=1}^{N}(-1)^{i+j}\Omega_{ij}M_{ij}\right)\\ = \frac{1}{\det\mathbf{\Omega}}\left((-1)^{i+j}M_{ij}\right) = \frac{1}{\det\mathbf{\Omega}}\tilde{\mathbf{\Omega}}^T \underset{by\ def}{=} \mathbf{\Omega}^{-T}\end{aligned}$$

where $\mathbf{\Omega}^{-T} = (\mathbf{\Omega}^{-1})^T$.

In a similar fashion, we can find the derivative of the second term of the log-likelihood function:

$$(\mathrm{tr}(\mathbf{S\Omega}))' \underset{by\ def}{=} \left(\sum_{i,k} S_{ik}\Omega_{ki}\right)' = \frac{\partial}{\partial\Omega_{ij}}\left(\sum_{i,k} S_{ik}\Omega_{ki}\right) = (S_{ji}) = \mathbf{S}^T.$$

Thus, imposing the derivative of Eq.(3.2) equal to zero, we get the equation

$$\mathbf{\Omega}^{-T} - \mathbf{S}^T = 0. \tag{3.4}$$

Hence, we obtain the maximum likelihood estimate (MLE) $\widehat{\mathbf{\Omega}} = \mathbf{S}^{-1}$.

Although the MLE is very effective since it converges to the true precision matrix as sample size N tends to infinity, it loses its potency in high dimensions. When the number of variables (i.e., nodes of the graph) p is comparable or even greater than the number of samples N, the empirical covariance matrix $\mathbf{S}$ is or is close to a rank deficient matrix. Therefore, MLE is either not numerically stable (small changes in the covariance matrix introduce dramatic changes in the precision matrix estimates) or does not exist at all. This fact means that it is necessary to introduce some form of regularization.

Moreover, as explained above, to derive an interpretable estimator in the high-dimensional setting, we usually assume that the true graph is sparse, i.e., it has only a few significant edges, while all the others are zero. Therefore, the precision matrix must be sparse. However, even when MLE exists, the estimated precision matrix does not typically contain many zero elements. Thus, it is impossible to identify the edge locations in an interpretable way. Such a problem occurs because MLE estimates are close to the true elements of $\mathbf{\Omega}$ with high probability, but none of $\widehat{\Omega}_{ij}$ is *exactly* zero – i.e., the estimators of the true zero elements of the precision matrix can be very small, but most likely almost none of them will be zeros.

Therefore, to learn the structure of a Gaussian graphical model in a high-dimensional setting with a sparsity hypothesis, it is necessary to introduce a sparsity constraint term into the likelihood function explicitly.

3.2.2 Penalized likelihood

In high dimensions, sparsity assumption makes the problem mathematically feasible. As said in the previous section, the sparsity requirement is motivated by recovering the support of the precision matrix accurately, and it also enhances the interpretability of a model – the fewer edges we have, the better we can investigate the estimated connections and infer conclusions.

How can we control the sparsity, i.e., the number of edges or the number of non-zero elements in the precision matrix? Let us consider the following l_0-based penalty

$$\rho_0(\mathbf{\Omega}) = \sum_{i \neq j} \mathbb{I}[\Omega_{ij} \neq 0] = 2|E|, \tag{3.5}$$

which explicitly specifies how many edges are present in the graph G, with E being its edge set. We could then modify the optimization problem in Eq.(3.3) by pre-specifying the maximum allowed number of edges e:

$$\widehat{\mathbf{\Omega}} = \underset{\substack{\mathbf{\Omega} \succeq 0, \\ \rho_0(\mathbf{\Omega}) \leq e}}{\arg\max} \left\{ \log\det \mathbf{\Omega} - \operatorname{tr}(\mathbf{S}\mathbf{\Omega}) \right\}.$$

Unfortunately, such approach defines a non-convex constraint set formed as the union over all possible subsets of e out of p edges whose numerical solution can be challenging and for large p not practicable. For this reason, instead of the penalty in Eq.(3.5), we can consider its convex relaxation by the following l_1-based penalty

$$\rho_1(\mathbf{\Omega}) = \sum_{i \neq j} |\Omega_{ij}|. \tag{3.6}$$

Note that Eq.(3.6) does not penalize the diagonal elements since they are not expected to be zero. With such penalization term, we can write the minimization problem in the Lagrangian form as

$$\hat{\mathbf{\Omega}}^{GL} = \underset{\mathbf{\Omega} \succeq 0}{\arg\max} \left\{ \log\det \mathbf{\Omega} - \operatorname{tr}(\mathbf{S}\mathbf{\Omega}) - \lambda \sum_{i \neq j} |\Omega_{ij}| \right\}, \tag{3.7}$$

where $\lambda \geq 0$ is a regularization parameter controlling the sparsity – larger the λ, less edges in the resulting graph. The *graphical lasso* is a well-know algorithm for the solution of this problem and we will discuss it in the next section.

3.2.3 Graphical lasso

First, let us note that Eq.(3.7) is a convex problem. Hence, it is possible to apply the block coordinate-descent algorithm to solve it. The block coordinate-descent algorithm was introduced by Banerjee, El Ghaoui and d'Aspremont [4] and was later refined by Friedman, Hastie and Tibshirani [29] who called such

algorithm *graphical lasso*; other approaches are also suggested in [60, 72]. Here, we present the algorithm described in [29]. We start by considering the pseudo-gradient of the function to maximize, expressed by the following equation:

$$\boldsymbol{\Omega}^{-1} - \mathbf{S} - \lambda\boldsymbol{\Psi} = 0. \tag{3.8}$$

The first two terms were derived in a similar fashion as in Eq.(3.4) for a non-penalized maximum likelihood problem and are indeed classical gradient while the third term, $\boldsymbol{\Psi}$, is a $p \times p$ matrix with elements

$$\Psi_{ij} = \begin{cases} 0 & \text{if } i = j \\ \operatorname{sgn}(\Omega_{ij}) & \text{if } \Omega_{ij} \neq 0 \\ \in [-1, 1] & \text{if } \Omega_{ij} = 0. \end{cases}$$

Denoting $\mathbf{W} \in \mathbb{R}^{p\times p}$ the current working version of $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}$, we can partition all the matrices into one column versus the rest and rewrite Eq.(3.8). For convenience, we pick the last column:

$$\begin{pmatrix} \mathbf{W}_{11} & \mathbf{w}_{12}^T \\ \mathbf{w}_{12} & W_{22} \end{pmatrix} - \begin{pmatrix} \mathbf{S}_{11} & \mathbf{s}_{12}^T \\ \mathbf{s}_{12} & S_{22} \end{pmatrix} - \lambda \begin{pmatrix} \boldsymbol{\Psi}_{11} & \boldsymbol{\psi}_{12}^T \\ \boldsymbol{\psi}_{12} & \Psi_{22} \end{pmatrix} = 0.$$

Here, $\mathbf{W}_{11}$, $\mathbf{S}_{11}$ and $\boldsymbol{\Psi}_{11}$ are $(p-1) \times (p-1)$ matrices, $\mathbf{w}_{12}$, $\mathbf{s}_{12}$ and $\boldsymbol{\psi}_{12}$ are row vectors of dimension $(p-1)$, and W_{22}, S_{22} and Ψ_{22} are scalars.

If we now focus on the last row and last column and consider the definition of $\boldsymbol{\Psi}$, the upper right block of the equation can be written as

$$\mathbf{w}_{12}^T - \mathbf{s}_{12}^T - \lambda \operatorname{sgn}(\boldsymbol{\omega}_{12}^T) = 0. \tag{3.9}$$

where $\boldsymbol{\omega}_{12}$ is a row vector of dimension $(p-1)$. We will return to this equality later.

Now, since $\mathbf{W}$ is the inverse of $\boldsymbol{\Omega}$, the following is true:

$$\begin{pmatrix} \mathbf{W}_{11} & \mathbf{w}_{12}^T \\ \mathbf{w}_{12} & W_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Omega}_{11} & \boldsymbol{\omega}_{12}^T \\ \boldsymbol{\omega}_{12} & \Omega_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.10}$$

Thus, we can derive $\mathbf{w}_{12}^T$ as

$$\begin{aligned} \mathbf{W}_{11}\boldsymbol{\omega}_{12}^T + \mathbf{w}_{12}^T\Omega_{22} &= 0 \\ \mathbf{w}_{12}^T &= -\mathbf{W}_{11}\boldsymbol{\omega}_{12}^T/\Omega_{22}. \end{aligned}$$

If we now define the vertical vector $\boldsymbol{\beta}_{12} = -\boldsymbol{\omega}_{12}^T/\Omega_{22}$ of dimension $(p-1)$, the previous equation becomes:

$$\mathbf{w}_{12}^T = \mathbf{W}_{11}\boldsymbol{\beta}_{12}.$$

Now, considering that $\operatorname{sgn}(\Omega_{22}) = 1$ since the matrix is symmetric positive definite, we have that $\operatorname{sgn}(\boldsymbol{\beta}_{12}) = -\operatorname{sgn}(\boldsymbol{\omega}_{12})$. Returning to Eq.(3.9), it can be rewritten as

$$\mathbf{W}_{11}\boldsymbol{\beta}_{12} - \mathbf{s}_{12}^T + \lambda \operatorname{sgn}(\boldsymbol{\beta}_{12}) = 0. \tag{3.11}$$

Let us now consider the usual linear regression set up with the outcome variables $\mathbf{y}$, predictor matrix $\mathbf{Z}$ and lasso penalty, i.e., the minimization problem $\frac{1}{2N}(\mathbf{y} - \mathbf{Z}\beta)(\mathbf{y} - \mathbf{Z}\beta)^T + \lambda|||\boldsymbol{\beta}||_1$ (see chapter 2 of [37]). Then its subgradient equation is

$$\frac{1}{N}\mathbf{Z}\mathbf{Z}^T\boldsymbol{\beta} - \frac{1}{N}\mathbf{Z}\mathbf{y} + \lambda\,\mathrm{sgn}(\boldsymbol{\beta}) = 0. \tag{3.12}$$

When we compare the subgradient in Eq.(3.12) to the subgradient in Eq.(3.11) we see the correspondence between $\frac{1}{N}\mathbf{Z}\mathbf{Z}^T$ and $\mathbf{W}_{11}$, and $\frac{1}{N}\mathbf{Z}\mathbf{y}$ and $\mathbf{s}_{12}^T$. This fact allows us to solve Eq.(3.11) by using the algorithm developed for lasso regression, treating each variable as the response variable and the other $p-1$ variables as the predictors. The graphical lasso algorithm consists of updating at each step the matrix $\mathbf{W}$ by cyclically updating its columns with lasso regression algorithm. For the sake of clarity, Algorithm 1 shows the pseudo-code of the entire procedure.

Algorithm 1 Graphical lasso

INITIALIZE $\mathbf{W} = \mathbf{S} + \lambda\mathbf{I}$

for $j = 1 \dots p$ **do**
REPEAT UNTIL CONVERGENCE
1. Partition matrix $\mathbf{W}$ into part 1: all but the j-th row and column and part 2: j-th row and column.
2. Solve $\mathbf{W}_{11}\boldsymbol{\beta}_{12} - \mathbf{s}_{12}^T + \lambda\,\mathrm{sgn}(\boldsymbol{\beta}_{12}) = 0$ by algorithm for lasso regression.
3. Update $\mathbf{w}_{12}^T = \mathbf{W}_{11}\hat{\boldsymbol{\beta}}_{12}$.
end for

In the final cycle (for each j) solve $\hat{\boldsymbol{\omega}}_{12} = -\hat{\boldsymbol{\beta}}_{12}^T\hat{\Omega}_{22}$, $\hat{\Omega}_{22} = 1/(W_{22} - \mathbf{w}_{12}\hat{\boldsymbol{\beta}}_{12})$.

The last step of the algorithm comes from the definition of $\boldsymbol{\beta}_{12} = -\boldsymbol{\omega}_{12}^T/\Omega_{22}$ and from the identity $\mathbf{w}_{12}\boldsymbol{\omega}_{12}^T + W_{22}\Omega_{22} = 1$, obtained from the last row and last column of Eq.(3.10). Note that the algorithm provides not only the estimate of the support of precision matrix but also the estimate $\hat{\boldsymbol{\Omega}}$ of $\boldsymbol{\Omega}$ itself. This point constitutes a relevant difference with the regression-based approach, introduced in the next section.

For a more in-depth discussion of the graphical lasso estimator's theoretical properties, we suggest [34, 37]. In contrast, for more insights into the implementation of the graphical lasso, we refer to [76] and the documentation and vignettes of the R packages `glasso` [31] and `huge` [39], the latter being an efficient implementation in C++.

3.3 Regression-based approach

The *graphical lasso* is a global method since it estimates the full precision matrix $\boldsymbol{\Omega}$ and not only its support, which encodes the full graph structure. An

alternative class of procedures, known as neighborhood or regression-based methods, are instead local. They are based on the observation that recovering a full graph is equivalent to recovering each vertex's neighborhood set. This kind of methods concentrates only on recovering matrix support and not the full matrix.

The pairwise Markov property connects the conditional independence of variables and the adjacency matrix corresponding to the same graph G: there is no edge between two nodes if and only if the corresponding variables are conditionally independent given the remaining ones. However, in the Gaussian setting, the pairwise Markov property is equivalent to the local Markov property, which states that any variable is conditionally independent of the rest given its neighbors. This observation means that we can learn the entire graph if we learn the neighborhood of each vertex, i.e., if we locate the non-zeros entries in the i-th row/column of the precision matrix for all i. Hence, we can learn the connections of each vertex if we properly select the significant variables in the linear regression model for Gaussian distributions as anticipated in Proposition 6.

3.3.1 Penalized regression estimation

Proposition 6 of Section 2.4 states that if $\mathbf{x} = (X_1, \dots, X_p) \sim \mathcal{N}_p(0, \mathbf{\Sigma})$ and $\mathbf{\Sigma}^{-1} = \mathbf{\Omega}$, $V = \{1, \dots, p\}$, then for each $j = 1, \dots, p$ one has

$$X_j = -\sum_{i:i\neq j} \frac{\Omega_{ij}}{\Omega_{jj}} X_i + \varepsilon_j \quad \text{where} \quad \varepsilon_j \sim \mathcal{N}(0, \Omega_{jj}^{-1}) \text{ and } \varepsilon_j \perp X_{V\setminus\{j\}}, \tag{3.13}$$

where $X_{V\setminus\{j\}} = (X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_p)$. Eq.(3.13) tells us that since the conditional distribution of X_j given the remaining variables $X_{V\setminus\{j\}}$ is still Gaussian with known mean-covariance structure, we can regard $-\Omega_{ij}/\Omega_{jj}$ as the coefficients of a linear regression problem, where X_j is the response and $X_{V\setminus\{j\}}$ are the predictors. We can derive the regression-based estimation observing that if we properly select variables into the linear regression model of Eq.(3.13), we select the neighborhood of node j in the graph.

To this aim, let us define the matrix of regression coefficients $\mathbf{\Theta} \in R^{p\times p}$ such that $\Theta_{ij} = -\Omega_{ij}/\Omega_{jj}$ for $i \neq j$ and $\Theta_{ii} = 0$. We then rewrite Eq.(3.13) in the following way:

$$X_j = \sum_{i:i\neq j} \Theta_{ij} X_i + \varepsilon_j \quad \text{with} \quad \varepsilon_j \sim \mathcal{N}(0, \Omega_{jj}^{-1}) \text{ and } \varepsilon_j \perp X_{V\setminus\{j\}}. \tag{3.14}$$

By construction, $\operatorname{supp} \mathbf{\Theta} = \operatorname{supp} \mathbf{\Omega}$. Thus, instead of estimating the support of precision matrix, we move to the problem of regression coefficients matrix's support estimation.

If we introduce vector $\boldsymbol{\theta}_j = (\Theta_{1j}, \dots, \Theta_{i(j-1)}, \Theta_{i(j+1)}, \Theta_{pj}) \in \mathbb{R}^{p-1}$, which represents the i-th row of $\mathbf{\Theta}$ without the diagonal element, we can regard Eq.(3.14) as a linear regression model in the unknown $\boldsymbol{\theta}_j$, where we regress each X_j over

all the remaining variables $X_{V\setminus\{j\}}$ and estimate the regression coefficients using ordinary least squares

$$\hat{\boldsymbol{\theta}}_j = \underset{\boldsymbol{\theta}\in\mathbb{R}^{p-1}}{\arg\min}\left\{\frac{1}{2n}||X_j - \sum_{i\neq j} X_{V\setminus\{j\}}\boldsymbol{\theta}_i||^2\right\}.$$

We remember, however, our main sparsity assumption. Therefore, as in the previous section, we introduce l_1-penalty:

$$\hat{\boldsymbol{\theta}}_j = \underset{\boldsymbol{\theta}\in\mathbb{R}^{p-1}}{\arg\min}\left\{\frac{1}{2n}||X_j - \sum_{i\neq j} X_{V\setminus\{j\}}\boldsymbol{\theta}_i||^2 + \lambda||\boldsymbol{\theta}_j||_1\right\} \tag{3.15}$$

and again, achieve the lasso regression problem. If we solve problem in Eq.(3.15) for $j = 1\ldots p$, we can obtain all columns of regression coefficient matrix $\hat{\boldsymbol{\Theta}}$. However, since the entries of $\hat{\boldsymbol{\Theta}}$ are only proportional to the entries of $\boldsymbol{\Omega}$, with this approach, we do not estimate the precision matrix, but only its support by selecting the non-zero regression coefficients.

Now let us rewrite the problem in Eq.(3.15) for a data matrix $\mathbf{X}$. To this purpose, we denote $X_{\cdot j}$ – the j-th column of data matrix and $\mathbf{X}_{\cdot -j}$ – data matrix without the column j. Now, collecting the Eq. (3.15) over all p variables, we get the following equation

$$\begin{aligned}\hat{\boldsymbol{\Theta}} &= \underset{\substack{\boldsymbol{\Theta}\in\mathbb{R}^{p\times p}\\ diag=0}}{\arg\min}\left\{\frac{1}{2n}\sum_{j=1}^{p}||X_{\cdot j} - \sum_{i\neq j} X_{\cdot i}\Theta_{ij}||^2 + \lambda\sum_{i\neq j}|\Theta_{ij}|\right\}\\ &= \underset{\substack{\boldsymbol{\Theta}\in\mathbb{R}^{p\times p}\\ diag=0}}{\arg\min}\left\{\frac{1}{2n}||\mathbf{X} - \mathbf{X}\boldsymbol{\Theta}||_F^2 + \lambda\sum_{i\neq j}|\Theta_{ij}|\right\},\end{aligned} \tag{3.16}$$

which was introduced by Meinshausen and Bühlmann in [51]. Since it is by construction a sum of p independent regressions, we can split it into p minimization subproblems on $\mathbb{R}^{p-1}$ and solve them separately, as described in Algorithm 2.

Algorithm 2 Neighbourhood regression for graph estimation

for $j = 1\ldots p$ **do**

1. Extract the column vector $X_{\cdot j}$ and the submatrix $\mathbf{X}_{\cdot -j}$.
2. Solve lasso problem $\hat{\boldsymbol{\theta}}_j = \underset{\theta\in\mathbb{R}^{p-1}}{\arg\min}\left\{\frac{1}{2n}||X_{\cdot j} - \sum_{i\neq j} X_{\cdot i}\boldsymbol{\theta}_j||^2 + \lambda\rho_1(\boldsymbol{\theta}_j)\right\}$
3. Return the neighbourhood estimate $\hat{ne}(j) = \{i \in V\setminus\{j\} : \hat{\Theta}_{ij} \neq 0\}$

end for

To guarantee symmetry, construct the adjacency matrix: $\hat{A}_{ij} = 1$ if $\hat{\Theta}_{ij} \neq 0$ AND/OR $\hat{\Theta}_{ji} \neq 0$.

For the implementation of this algorithm, we suggest the R package `huge` [39]. For which concerns the theoretical properties we recommend [34] for a complete overview.

We observe that this method does not enforce any symmetry constraints on $\hat{\mathbf{\Theta}}$, i.e., $\hat{\Theta}_{ij}$ and $\hat{\Theta}_{ji}$ are not guaranteed to be zero or non-zero simultaneously. Of course, this fact represents a drawback, since for an undirected graph G, if node i is a neighbour of node j, then also the contrary must be true, i.e., node j must be a neighbour of node i. Therefore, to construct the adjacency matrix of the underlying graphical model G, one has to apply a post-processing rule on the estimated $\hat{\mathbf{\Theta}}$ matrix in order to restore symmetry. The most widely used ones are those proposed by Meinshausen and Bühlmann in [51], namely AND or OR rules which either put an edge between nodes i, j if both $\hat{\Theta}_{ij} \neq 0$ and $\hat{\Theta}_{ij} \neq 0$ or if at least one of them is non-zero.

In the next section, we will discuss a proposal to avoid such post-processing, which will incorporate the symmetry of matrix $\hat{\mathbf{\Theta}}$ directly into the estimation process.

3.3.2 Group penalization

For an undirected graph G, we do not distinguish between edges (i, j) and (j, i), since we expect its adjacency matrix $\mathbf{A}$ to be symmetric. Since matrix $\mathbf{A}$ is encoded by the support of the precision matrix $\mathbf{\Omega}$ which we estimate by the support of the regression coefficient matrix $\mathbf{\Theta}$, it is natural to assume that latter has to be symmetric. However, as we have observed above, the solution of penalized regression problem in Eq.(3.16) might not lead to estimates $\hat{\mathbf{\Theta}}$ with zero/non-zero elements located in symmetric positions to the diagonal, hence a post-processing rule has to be applied to symmetrize the support. In order to avoid such additional step and, above all, to exploit the symmetry information in the learning process of the matrix itself, authors Friedman, Hastie and Tibshirani proposed in [30] the following minimization problem:

$$\hat{\mathbf{\Theta}} = \underset{\substack{\mathbf{\Theta}\in\mathbb{R}^{p\times p}\\ diag=0}}{\arg\min} \left\{ \frac{1}{2n} \sum_{j=1}^{p} ||X_{\cdot j} - \sum_{i\neq j} \Theta_{ij} X_{\cdot i}||^2 + \sqrt{2}\lambda \sum_{i<j} \sqrt{\Theta_{ij}^2 + \Theta_{ji}^2} \right\} = \underset{\substack{\mathbf{\Theta}\in\mathbb{R}^{p\times p}\\ diag=0}}{\arg\min} F(\mathbf{\Theta}). \tag{3.17}$$

In the minimization problem in Eq.(3.17) a group lasso penalty is used instead of the simple lasso penalty of Eq.(3.16). By a "group" here we mean the two regression coefficients $(\Theta_{ij}, \Theta_{ji})$ referring to a couple of variables i, j. The proposed group lasso penalty obtains selection at group level instead of coefficients level, hence either a whole group is zero (meaning that both $(\Theta_{ij}, \Theta_{ji}) = 0$) or all of it is not zero (meaning that both $(\Theta_{ij}, \Theta_{ji}) \neq 0$). Factor $\sqrt{2}$ in Eq.(3.17) accounts for the size of the group as usually done in the setting of the group lasso penalty.

Note that when the group of variables is not zero, its elements do not necessarily coincide. Indeed, they are almost certainly different – the penalty imposes the symmetry only on the support of $\hat{\mathbf{\Theta}}$ and not on its elements.

The minimization problem in Eq.(3.17) is convex, but it can not be split into p independent subproblems as we have done for the minimization problem of Eq.(3.16). Hence, we need a specific algorithm to solve it. In particular, we can adopt the algorithm proposed in [20], which was inspired by the group descent approach of Breheny and Huang [12]. This algorithm is iterative, and its main idea is to update one group at a time considering the rest of the groups fixed to their current values. Let us analyze in detail how one group is updated.

To this aim we compute the gradient of $F(\mathbf{\Theta})$, defined in Eq.(3.17), with respect to the group $(\Theta_{ij}, \Theta_{ji})$. When $\sqrt{\Theta_{ij}^2 + \Theta_{ji}^2} \neq 0$, the gradient is equal to

$$\nabla_{ij} F = -\frac{1}{n} \begin{pmatrix} {X_{.i}}^T (X_{.j} - \mathbf{X}\Theta_{.j}) \\ {X_{.j}}^T (X_{.i} - \mathbf{X}\Theta_{.i}) \end{pmatrix} + \frac{\sqrt{2}\lambda}{\sqrt{\Theta_{ij}^2 + \Theta_{ji}^2}} \begin{pmatrix} \Theta_{ij} \\ \Theta_{ji} \end{pmatrix},$$

and when $\sqrt{\Theta_{ij}^2 + \Theta_{ji}^2} = 0$, the gradient is equal to

$$\nabla_{ij} F = -\frac{1}{n} \begin{pmatrix} {X_{.i}}^T (X_{.j} - \mathbf{X}\Theta_{.j}) \\ {X_{.j}}^T (X_{.i} - \mathbf{X}\Theta_{.i}) \end{pmatrix} + \sqrt{2}\lambda \begin{pmatrix} v_1 \\ v_2 \end{pmatrix},$$

where $(v_1, v_2)^T$ is any vector with the norm less or equal to 1.

Now define vector $\mathbf{z} = (z_{ij}, z_{ji})^T$ with entries

$$\begin{aligned} z_{ij} &= -\frac{1}{n} X_{.i}^T \left(X_{.j} - \sum_{m \neq i,\, j} X_{.m} \Theta_{mj} \right) \\ z_{ji} &= -\frac{1}{n} X_{.j}^T \left(X_{.i} - \sum_{m \neq i,\, j} X_{.m} \Theta_{mi} \right). \end{aligned} \tag{3.18}$$

Then, following [12], we have that the multivariate soft thresholding operator is the minimizer of function $F(\mathbf{\Theta})$ with respect to the variables $(\Theta_{ij}, \Theta_{ji})$, i.e.

$$\begin{pmatrix} \hat{\Theta}_{ij} \\ \hat{\Theta}_{ji} \end{pmatrix} = \left(1 - \frac{\sqrt{2}\lambda}{||\mathbf{z}||} \right)_+ \mathbf{z}, \tag{3.19}$$

where the soft-thresholding operator $(\cdot)_+$ is defined as

$$\left(1 - \frac{\sqrt{2}\lambda}{||\mathbf{z}||} \right)_+ \mathbf{z} = \begin{cases} \left(1 - \dfrac{\sqrt{2}\lambda}{||\mathbf{z}||} \right) \cdot \mathbf{z} & \text{if} \quad ||\mathbf{z}|| > \sqrt{2}\lambda \\ 0 \cdot \mathbf{z} & \text{if} \quad ||\mathbf{z}|| \leq \sqrt{2}\lambda \end{cases} \tag{3.20}$$

Summarizing, we update group $(\Theta_{ij}, \Theta_{ji})$ first by evaluating vector $\mathbf{z}$ in Eq.(3.18) with the current value of all the other groups, second by applying the soft-thresholding operator to $\mathbf{z}$ in Eq.(3.19). We cycle over all groups $i > j$ and repeat these steps until we reach convergence. To establish convergence, we require that either the norm of the difference between the current matrix $\hat{\mathbf{\Theta}}$ and the

one calculated at the previous iteration step is smaller than some user-provided threshold or that the maximum number of iterations is reached.

Minimization problem in Eq.(3.17) and algorithm described for its solution are key ingredients of my future discussion. In fact, in the next section, I will describe the problem of Gaussian graphical model estimation in the multiple datasets framework, and after that, in Chapter 4, I will present a new method for solving such a problem. That method represents the main result of my thesis and can be seen as an extension of the method described in Eq.(3.17) to the case of K datasets.

3.4 Multiple datasets framework

In the previous sections, we dealt with the problem of inferring a graph G from a data matrix $\mathbf{X}$. In this section, instead, we aim to infer that graph from *several* data matrices. Our interest is motivated by the fact that nowadays it is becoming widespread to collect several datasets $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ from different sources or categories (groups) or experimental classes. If these datasets share the same (or at least mostly the same) variables, it is then natural to assume that they also share the same conditional dependency structure. For example, we can collect gene expression data from several studies on a particular disease. The studies might differ on the used high-throughput platforms or might be carried out in different laboratories. Nevertheless, the underlying gene regulation mechanism that we want to investigate is the same. Hence, we can analyze all the data simultaneously. Therefore, we have to consider methods ad-hoc for such a framework.

In this subsection, I will first describe the multiple datasets problem, and explain the need to use ad hoc methods to deal with it. Then, I will provide an overview of the joint estimation methods available in the literature (see also Appendix A for a summary). The presentation of the novel joint estimation method, *jewel*, developed in this thesis, is provided later in Chapter 4.

3.4.1 Problem settings

Suppose that we are given $K \geq 2$ datasets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}$. The k-th dataset is a $n_k \times p_k$ matrix which contains n_k observations $(\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{n_k}^{(k)})^T$ where each $\mathbf{x}_i^{(k)} = (X_{i1}^{(k)}, \ldots, X_{ip_k}^{(k)})$ is a p_k-dimensional row vector.

$$\mathbf{X}^{(k)} = \begin{pmatrix} \mathbf{x}_1^{(k)} \\ \vdots \\ \mathbf{x}_{n_k}^{(k)} \end{pmatrix} = \begin{pmatrix} X_{11}^{(k)} & \ldots & X_{1p_k}^{(k)} \\ \vdots & \ddots & \vdots \\ X_{n_k1}^{(k)} & \ldots & X_{n_kp_k}^{(k)} \end{pmatrix}, \quad k = 1 \ldots K.$$

As in previous sections, we assume that the datasets come from centered multivariate Gaussian distributions. However, since we suppose that data comes from different sources, we encode such a difference in terms of covariance matrices. More specifically, we assume that observations in each class $(\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{n_k}^{(k)})^T$ are independent and identically distributed samples from a p_k-variate Gaussian distribution with zero mean and covariance matrix $\mathbf{\Sigma}^{(k)}$. We denote the true precision matrix for the k-th class as $\mathbf{\Omega}^{(k)} = \left(\mathbf{\Sigma}^{(k)}\right)^{-1}$, $k = 1 \ldots K$.

Our second assumption is that majority of the variables $\{X_1, \ldots, X_{p_k}\}$ are in common in all or most of the datasets and that they share a common dependency structure across the datasets. More specifically, we assume that each distribution $\mathcal{N}_p(0, \mathbf{\Sigma}^{(k)})$ is associated with a graphical model $G^{(k)}$, encoded by the support of its precision matrix, and that these $G^{(k)}$, $k = 1 \ldots K$, share common characteristic specified by the application context. Hence, the aim is to recover the shared underlying graphical model G using all the available information sources.

The easiest and most natural way to deal with such a problem is *concatenation*. This approach simply combines the data matrices $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}$ to form a one long matrix $\mathbf{X}^{long}$ of the size $\sum_K n_k \times p$. Then it applies any of the graph estimation methods for one dataset, described in the previous sections. Figure 3.2 illustrates the concatenation approach.

However, the first drawback is that it is only applicable when $p_k = p \ \forall k$, i.e., when all the variables are the same. Moreover, under the assumption of different covariance matrices for each set, this approach performs poorly since it destroys the correlation heterogeneity when merging the datasets (as we will also demonstrate in simulation in Chapter 5).

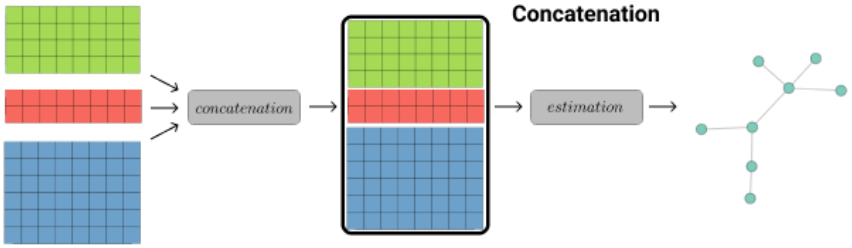


Figure 3.2: Concatenation approach for analysing several datasets: combine all of them in a long matrix and apply method for one dataset.

Another naive approach is *voting*. With this approach, we would analyze each dataset separately, get K graph estimates, and then combine the results in a consensus graph. Figure 3.3 illustrates this idea. For example, we can put an edge in the final graph if it is present in at least m out of K estimated networks, where usually $m = \lceil K/2 \rceil$. Unfortunately, this approach still does not fully exploit the shared structure of the datasets' underlying graph. Moreover, individually estimated graphs might not be accurate when the sample size is small, introducing an error to the final result.

By contrast, a third possible approach is to *jointly* estimate the underlying graphs, taking advantage of multiple datasets, exploiting all available information, and producing results that consider similarities and difference among the

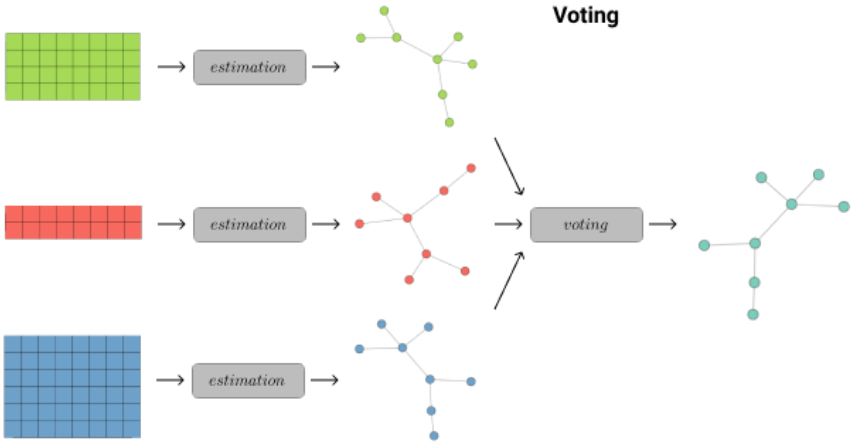


Figure 3.3: Voting approach for analyzing several datasets: estimate the graphs from each dataset independently and then combine the individual estimates into a consensus result.

classes. With the joint approach, all the datasets are analyzed *simultaneously* requiring both goodness of fit and enforcing the graph structure. Figure 3.4 illustrates this idea. Joint strategy requires the development of ad hoc methods. We prove in Chapter 5 that the joint approach provides a significant advantage in performance to naive approaches.

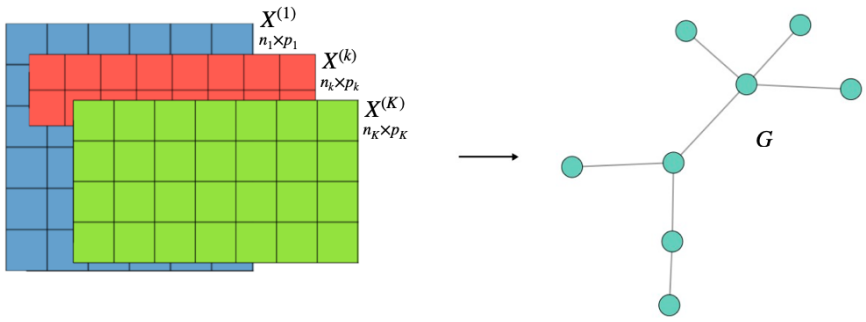


Figure 3.4: When datasets with (mostly) the same variables come from different sources and/or are collected in distinct classes or groups, how can we infer the underline conditional dependency graph?

Since this thesis aims to present a new joint estimation method for Gaussian graphical models in the multiple datasets framework, it is worth to present what has already been developed in the literature. Hence, in the following subsection, we review some of the recent approaches in this framework.

3.4.2 Joint estimation methods in literature

In this subsection, we will review methods for joint estimation available in the literature. Details for each method can be found in Appendix A while in this subsection, we will focus on their main ideas.

The first two methods can be considered an extension of the penalized likelihood estimation for the case $K = 1$ presented in Eq.(3.7) to the case $K \geq 2$. In particular, if we consider the minimization of the negative log-likelihood instead of the maximization of the log-likelihood (to achieve consistent notation with articles in review) and then rewrite the problem in Eq.(3.7) for more than one dataset, we get the following :

$$(\hat{\mathbf{\Omega}}^{(1)}, \ldots, \hat{\mathbf{\Omega}}^{(K)}) = \underset{\substack{\mathbf{\Omega}^{(1)} \succeq 0 \\ \vdots \\ \mathbf{\Omega}^{(K)} \succeq 0}}{\arg\min} \Bigg\{ \sum_{k=1}^{K} \Big(\operatorname{tr}(\mathbf{S}^{(k)}\mathbf{\Omega}^{(k)}) - \log\det \mathbf{\Omega}^{(k)} \Big) + P\left(\mathbf{\Omega}^{(1)}, \ldots, \mathbf{\Omega}^{(K)}\right) \Bigg\} \tag{3.21}$$

where $\mathbf{S}^{(k)}$, $k = 1 \ldots K$, are the empirical covariance matrices of each class and $P\left(\mathbf{\Omega}^{(1)}, \ldots, \mathbf{\Omega}^{(K)}\right)$ is the penalty. As we can see, the extension of the likelihood term is relatively straightforward: in the multiple dataset framework, we add a sum over K to account for all K classes. In this way, we require that all K datasets contribute to the *multitask learning*. However, the penalty extension is not straightforward since it should enforce sparsity and some similarity among the structure classes. The penalty specification will make the difference between the methods.

Guo et al. proposal

Guo et al. [36] proposed to reparametrize each off-diagonal element of the precision matrix $\mathbf{\Omega}^{(k)}$ as $\Omega_{ij}^{(k)} = \beta_{ij}\gamma_{ij}^{(k)}$, $k = 1 \ldots K$. For a pair of variables i, j such decomposition treats $(\Omega_{ij}^{(1)}, \Omega_{ji}^{(1)}, \ldots, \Omega_{ij}^{(K)}, \Omega_{ji}^{(K)})$ as a group. The group's common factor $\beta_{ij} \geq 0$ controls the presence of an edge between nodes i and j in all of the classes and $\gamma_{ij}^{(k)}$ reflects the differences among classes. This parametrization is useful if one aims to infer a common network while preserving some class specific structure. With this decomposition, authors of [36] rewrote the problem in Eq.(3.21) in terms of matrices $\mathbf{B} = (\beta_{ij})_{p\times p}$ and $\mathbf{\Gamma}^{(k)} = (\gamma_{ij}^{(k)})_{p\times p}$ instead of $\mathbf{\Omega}^{(k)}$. Specifically, they introduced the following minimization problem:

$$\min_{\mathbf{B}, \mathbf{\Gamma}^{(1)}, \ldots, \mathbf{\Gamma}^{(K)}} \Bigg\{ \sum_{k=1}^{K} \Big(\operatorname{tr}(\mathbf{S}^{(k)}\mathbf{\Omega}^{(k)}) - \log\det \mathbf{\Omega}^{(k)} \Big) + \lambda_1 \sum_{i\neq j} \beta_{ij} + \lambda_2 \sum_{i\neq j} \sum_{k=1}^{K} |\gamma_{ij}^{(k)}| \Bigg\}. \tag{3.22}$$

Here, the first regularization parameter λ_1 controls the sparsity of the common factors. If β_{ij} is shrunk to zero by the first penalty, there will be no link between variables i, j in any of the K graphs. However, $\beta_{ij} \neq 0$ does not guarantee the presence of the edge (i, j) in all of the graphs since the second penalty with parameter λ_2 can still force $\gamma_{ij}^{(k)} = 0$.

Authors reformulated the problem in Eq.(3.22) in a more convenient form for computational purposes, which resulted in a problem of the type given in Eq.(3.21) with the following hierarchical penalty:

$$P\left(\mathbf{\Omega}^{(1)},\ldots,\mathbf{\Omega}^{(K)}\right)=\lambda\sum_{i\neq j}^{p}\sqrt{\sum_{k=1}^{K}|\Omega_{ij}^{(k)}|},\quad\text{with}\quad\lambda=2\sqrt{\lambda_1\lambda_2}. \tag{3.23}$$

Although this formulation does not allow direct control over the class specific networks as in Eq.(3.22), the supports of resulting estimates $\hat{\mathbf{\Omega}}^{(1)},\ldots,\hat{\mathbf{\Omega}}^{(K)}$ still might be different. Thus, this second formulation still allows to incorporate the difference in the resulting estimates of the graphical models, just not as straightforwardly as presented in Eq.(3.22). For this reason, in Chapter 5 when applying Guo et al. proposal for comparison purposes, we will construct its resulting adjacency matrix as $\hat{\mathbf{A}}^{Guo}=\cup_k \operatorname{supp}\hat{\mathbf{\Omega}}^{(k)}$.

Note that minimization problem in Eq.(3.23) is not convex. For its solution, the authors used an iterative approach, based on local linear approximation [81]. They also proved the consistency and sparistency properties of the estimator.

Joint graphical lasso (JGL)

The proposal of Guo et al. was the precursor of the *joint graphical lasso, JGL*, by Danaher, Wang and Witten [19], as the authors claimed themselves. In fact, in [19] the authors proposed to solve the minimization problem given in Eq. (3.21) with fused graphical lasso or group graphical lasso penalty, resembling the proposal of Guo et al. Here, we concentrate on the group graphical lasso penalty which has the following form:

$$P\left(\hat{\mathbf{\Omega}}^{(1)},\ldots,\hat{\mathbf{\Omega}}^{(K)}\right)=\lambda_1\sum_{k=1}^{K}\sum_{i\neq j}|\Omega_{ij}^{(k)}|+\lambda_2\sum_{i\neq j}\sqrt{\sum_{k=1}^{K}{\Omega_{ij}^{(k)}}^2}. \tag{3.24}$$

The penalty comprises two terms with two independent regularization parameters (similarly to the idea of Guo et al.), forcing a common structure across classes while still allowing for some differences between them. The first term with parameter λ_1 is an extension of lasso penalty to the case of K datasets. It encourages sparsity within each $\hat{\mathbf{\Omega}}^{(k)}$ independently of each other. The second term with λ_2 parameter encourages a similar sparsity pattern across all precision matrices. In particular, when $\lambda_1=0$ and $\lambda_2>0$, pattern of non-zero elements will be identical across all matrices. We will use such settings in methods' comparison in Chapter 5.

The minimization problem given in Eq.(3.21) with penalty given in Eq.(3.24) is convex and its implementation is given in the R package `JGL` [18]. The authors claimed to use ADMM, the alternating directions method of multipliers algorithm, described in [11] for its solution. Moreover, the authors adopted a smart procedure that takes advantage of the separability in covariance matrices with diagonal block structure. The authors stated two theorems on the empirical covariance matrices $\mathbf{S}^{(k)}$, which determine whether it is possible to permute

the variables to achieve disconnected blocks in matrices and then apply JGL to each block separately. In practice, the algorithm first seeks to identify the independent connected components of G and then, for each component, it estimates the corresponding subgraph as a smaller dimension minimization problem. This strategy leads to a great decrease in running time when such a diagonal block structure is present in the input data matrices.

Joint structural estimation method (JSEM)

The *joint structural estimation method (JSEM)* proposed by J. Ma and G. Michailidis in [48] is different from the previous approaches. First of all, it assumes that there exists prior information on how the K networks are structurally related; secondly, JSEM is a two-step procedure with each step being a constrained minimization problem. In particular, in the first step, JSEM infers the sparse network $\hat{G}^{(k)}$ by incorporating the prior knowledge on these networks through a group lasso regression-based approach. In the second step, JSEM maximizes the Gaussian log-likelihood subject to the edge set constraints obtained from the previous step. Specifically, let the prior knowledge on the structural similarity between networks be encoded by a group structure $\mathcal{G} = \cup_{i<j}^{p} \mathcal{G}_{ij}$. Each $\mathcal{G}_{ij}$ is a partition of the set $\{1, \ldots, K\}$ and consists of prior knowledge on the structural similarity for the (i, j)-th pair across classes. For a group $g \in \mathcal{G}_{ij}$, we denote vector $\boldsymbol{\theta}_{ij}^{[g]} = (\Theta_{ij}^{(k)}, \Theta_{ji}^{(k)})_{k \in g}$ – the restriction of vector $(\Theta_{ij}^{(k)})_{k=1,..,K}$ to the group g. With such notations, the first step of JSEM consists of the following group lasso regression-based approach

$$\min \left\{ \frac{1}{n} \sum_{k=1}^{K} ||X_{.j}^{(k)} - \mathbf{X}_{.-j}^{(k)} \Theta_{j.}^{(k)}||^2 + 2 \sum_{i \neq j} \sum_{g \in \mathcal{G}_{ij}} \lambda_{ij}^{[g]} ||\boldsymbol{\theta}_{ij}^{[g]}|| \right\}. \tag{3.25}$$

Authors in [48] proposed to solve problem in Eq.(3.25) by the coordinate descent algorithm and to obtain $\hat{E}^{(k)} = \left\{ (i, j) : \Theta_{ij}^{(k)} \neq 0 \right\}$. Then, in the second step, the authors used these sets $\hat{E}^{(k)}$, $k = 1 \ldots K$, as a constraint for the maximum likelihood problem for each k:

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

As it is, JSEM is a combination of the regression-based and maximum likelihood approaches, and when prior knowledge is available, it can be beneficial. Authors in [48] also proved the JSEM method's consistency results.

For those interested in another example of a joint estimation method that allows incorporating prior information, we also refer to [80], with brief details provided in Appendix A.

Other approaches

There are plenty of other literature proposals, some of them are summarized in Appendix A. Here we will mention only some interesting points. We will start by with method inspired by gene networks.

Let us assume that the datasets $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ contain observations of mostly the same genes coming from different experimental sources. These sources can be of two types (mainly but not only) – same measurements of the same genes but under different conditions or different measurements for the same genes under the same conditions. An example of the first approach could be measuring the gene expressions on different equipment, at different times, or for different disease subtypes. An example of the second approach would be in the multi-omics context when we measure, say, gene expression or methylation levels or genotype for the same set of genes. What happens if we want to explore both possibilities? For this purpose, authors of [74] developed JEGN – *joint estimation of gene networks* – to infer gene networks across multiple subpopulations and data types.

Let us consider K data types and L subpopulations (different subtypes of disease) with KL datasets $\mathbf{X}^{(1,1)}, \ldots, \mathbf{X}^{(1,L)}, \ldots, \mathbf{X}^{(K,1)}, \ldots, \mathbf{X}^{(K,L)}$. As before, we assume that each dataset contains $n_{kl} = n_l \ \forall k$ observations which are independent and identically distributed with zero mean and covariance matrix $\mathbf{\Sigma}^{(k,l)}$. Now, one would expect that different subpopulations share certain common edges, while some edges are unique to subpopulation, although consistent throughout the data types. Therefore, authors proposed to decompose the precision matrices into common and subpopulation-unique components $\mathbf{\Omega}^{(k,l)} = \mathbf{R}^{(k)} + \mathbf{M}^{(k,l)}$ and introduced the following minimization problem:

$$\min_{\substack{\mathbf{\Omega}^{(1)} \succeq 0 \\ \vdots \\ \mathbf{\Omega}^{(K)} \succeq 0}} \Bigg\{ \sum_{k=1}^{K} \sum_{l=1}^{L} n_l \Big(\operatorname{tr}(\mathbf{S}^{(k,l)} \mathbf{\Omega}^{(k,l)}) - \log\det \mathbf{\Omega}^{(k,l)} \Big) + \lambda_1 \lambda_2 L \sum_{i \neq j} \sqrt{\sum_{k=1}^{K} {\mathbf{R}_{ij}^{(k)}}^2} + \lambda_1 (1 - \lambda_2) \sum_{l=1}^{L} \sum_{i \neq j} \sqrt{\sum_{k=1}^{K} {\mathbf{M}_{ij}^{(k,l)}}^2} \Bigg\}.$$

Both penalty terms are group lasso, with the first one enforcing the same sparsity pattern across subpopulations and the second one encouraging the same subpopulation-unique structure for all data types. Regularization parameter λ_1 controls the degree of sparsity, while λ_2 reflects the difference between common and subpopulation specific components. The problem is solved with the ADMM method of [11]. We note here that authors also proposed the extension of JEGN to fit nonparanormal data which could be interesting to explore. Method is implemented in R package `JEGN` [73].

Among the other examples of joint estimation methods, explicitly developed for gene networks, we suggest NETI2 [65], which reconstructs the underlying graph for heterogeneous samples from different cancer subtypes and FSSEM [79], which integrates gene expression and genetic perturbation data.

There is one more interesting point which can be mentioned. While in the methods described above, we assumed that *edges* are (mostly) the same across all the networks, one can instead make this assumption about the connectivity patterns, i.e., search for a path between *nodes* through a specific vertex rather than for direct links. For example, in [53] the authors propose two methods. The

first method is *co-hub node joint graphical lasso* (CNJGL) and it is based on the assumption that the same nodes serve as hubs in each of the K networks. The second method, *perturbed-node joint graphical lasso* (PNJGL), assumes that the differences in the networks are due to the particular nodes that are perturbed across the classes. To define penalties for those methods, authors proposed a novel row-column overlap norm, which can be of interest. As for JGL [19], the problem is solved via the ADMM algorithm of [11], and conditions for the block-diagonal partition of the solution are derived.

In the next chapter, I will discuss the main result of the thesis – *jewel*, a novel method for the joint estimation of graphical models.

Chapter 4

jewel: joint node-wise estimation of multiple Gaussian graphical models

In this chapter, I will present *jewel*, a novel method for **j**oint nod**e-w**ise estimation of multiple Gaussian graphical mod**el**s. This chapter constitutes the main contribution of my thesis.

The idea of *jewel* is inspired by the Meinshausen and Bühlmann method [51] for estimating a Gaussian graphical model from one dataset $\mathbf{X}$. To extend this approach to the case of multiple datasets, $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$, I combined the multi-task learning framework with a suitable group lasso penalty. After defining the problem settings, I will describe the minimization problem where the extension of standard approach to the multiple datasets setting allows for a joint analysis of the data by guaranteeing the goodness of fit. The group lasso penalty is the other key ingredient for the proposed procedure. It will enable both the resulting adjacency matrix's symmetry and enforce the common structure across datasets to emerge. In this chapter, I will also provide a numerical algorithm to solve the minimization problem using the group descent algorithm and establish the estimator's consistency property. Finally, I will discuss how to estimate the regularization parameter from the data using cross-validation and the Bayesian information criterion.

4.1 Problem settings

Assume we have $K \geq 2$ datasets, $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}$, each of them drawn from different Gaussian distribution $\mathcal{N}(0, \mathbf{\Sigma}^{(k)})$, and we want to estimate their underlying Gaussian graphical model under the assumption that K distributions share the same conditional dependency structure represented with the graph G (Fig. 4.1 illustrates the main context). As mentioned in Chapter 3, the K datasets might represent different classes, categories, or data collected under slightly different conditions.

More specifically, for $k = 1 \ldots K$, the k-th dataset $\mathbf{X}^{(k)}$ is a $n_k \times p_k$ data

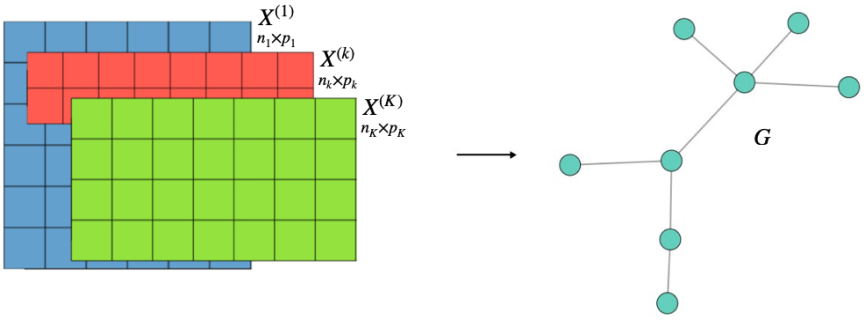


Figure 4.1: Given K datasets, drawn from different Gaussian distribution $\mathcal{N}(0, \mathbf{\Sigma}^{(k)})$, but sharing the same conditional dependency structure G, how can one estimate G?

matrix which contains n_k observations $(\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{n_k}^{(k)})^T$ of p_k variables, so that each $\mathbf{x}_i^{(k)} = (x_{i1}^{(k)}, \ldots, x_{ip_k}^{(k)})$ is a p_k-dimensional row vector. We assume that observations $(\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{n_k}^{(k)})^T$ are independent and identically distributed samples from a p_k-variate Gaussian distribution with zero mean and covariance matrix $\mathbf{\Sigma}^{(k)}$. We denote $\mathbf{\Omega}^{(k)} = (\mathbf{\Sigma}^{(k)})^{-1}$ the true precision matrix for k-th class.

Moreover, we also assume that majority of the variables $\{X_1, \ldots, X_{p_k}\}$, $k = 1 \ldots K$, are in common in all or most of the datasets. In other words, although we allow some variables to be present only in some data classes, we assume that we are mostly observing the same group of variables under different conditions or groups. This assumption serves to model real situations where the variables might not be exactly the same in the k conditions because some of them can be missing variables in particular conditions due to equipment failures or other technological reasons. For example, if we assume to measure genome-wide gene expressions in a disease/condition of interest and collect datasets from different studies, it is expected that few genes might not be observed in one study but are available in others, since their quantification depends on the type of platform used. Nevertheless, the cellular regulatory mechanisms are the same, and we want to investigate them.

Under these assumptions, we have that every distribution $\mathcal{N}(0, \mathbf{\Sigma}^{(k)})$ is associated with a graphical model $G_k = (V_k, E_k)$, where $V_k = \{X_1, \ldots, X_{p_k}\}$ and $E_k \subseteq V_k \times V_k$, i.e. $(i,j) \notin E_k \Leftrightarrow X_i^{(k)} \perp\!\!\!\perp X_j^{(k)} | X_{\{l, l \neq i,j\}}^{(k)}$. Observe that, since we are in the Gaussian case, $E_k \equiv \text{supp}(\mathbf{\Sigma}^{(k)^{-1}})$.

Our main assumption is that there exists a common graph $G = (V, E)$ with $V = V_1 \cup \cdots \cup V_K$ and $E \subseteq V \times V$ such that the following hypothesis is true:

Hypothesis. *$\forall\, i < j \in V_1 \cup \cdots \cup V_K$ it is true that $(i,j) \in E_k$ $\forall k : (i,j) \in V_k$ or $(i,j) \notin E_k$ $\forall k : (i,j) \in V_k$, meaning that for any pair of vertices the edge between them either exists in all graphs where these vertices occur or does not exist in any graphs where they occur.*

Going back to the gene expression example, what does this hypothesis mean? That for a pair of variables $\{i, j\}$ their connecting edge is either present or ab-

sent in all distributions containing such pair. In other words, any two genes are either conditionally independent or conditionally dependent in all of the conditions where they have been measured. Nevertheless, the covariance matrices $\mathbf{\Sigma}^{(k)}$ might be different across the datasets since the gene regulation might be slightly differently tuned in the different conditions. This situation can occur, for example, when we are observing different stages of a given disease or its subtypes.

Since we showed that inferring a Gaussian graphical model is equivalent to estimating the support of the precision matrix (Theorem 2), estimating the graph G translates into the problem of simultaneously inferring the support of all precision matrices $\mathbf{\Omega}^{(k)}$. Then the support of the precision matrix is equivalent to the adjacency matrix associated with the graph G.

4.2 The jewel method

First, we recall the regression based method described in Chapter 3 for a single data matrix. For any fixed k, $k = 1 \dots K$, we defined $\mathbf{\Theta}^{(k)} \in R^{p_k \times p_k}$, with entries $\Theta_{ij}^{(k)} = -\Omega_{ij}^{(k)}/\Omega_{ii}^{(k)}$ and $\Theta_{ii}^{(k)} = 0$. By construction $\operatorname{supp} \mathbf{\Theta}^{(k)} = \operatorname{supp} \mathbf{\Omega}^{(k)}$. Thus, we estimate the support of precision matrix for each individual matrix $\mathbf{X}^{(k)}$ via regression approach by solving the following minimization problem

$$\hat{\mathbf{\Theta}}^{(k)} = \underset{\substack{\mathbf{\Theta} \in \mathbb{R}^{p_k \times p_k} \\ diag=0}}{\arg\min} \left\{ \frac{1}{2n_k} ||\mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Theta}||_F^2 + \lambda\sqrt{2} \sum_{i \neq j} \sqrt{(\Theta_{ij})^2 + (\Theta_{ji})^2} \right\}, \tag{4.1}$$

where λ is a regularization (tuning) parameter. Eq.(4.1) results in a sparse estimate of $\hat{\mathbf{\Theta}}^{(k)}$ which leads us to a symmetric adjacency matrix (i.e., the support of $\mathbf{\Omega}^{(k)}$). Note that although the estimate $\hat{\mathbf{\Theta}}^{(k)}$ is not a symmetric matrix, its support is symmetric by construction since the group penalty $\sqrt{(\Theta_{ij})^2 + (\Theta_{ji})^2}$ guarantees that coefficients associated the group of variables (i, j) and (j, i) are either both zero or different from zero.

We aim to construct a joint estimation method that fully exploits the information of having K datasets. Therefore, we recast the regression-based approach in the framework of multitask learning, and we propose to find the estimated graph from the matrices $\mathbf{\Theta}^{(k)}$ that better fit the corresponding datasets, imposing a group lasso penalty to achieve sparsity and symmetry of the common support across the datasets. In practice,

- Each $\mathbf{\Theta}^{(k)}$ should optimize the regression problem associated to the corresponding dataset $\mathbf{X}^{(k)}$;
- The elements of $\mathbf{\Theta}^{(k)}$ associated with the pair of variables (i, j) should be either zero or different from zero in all the $\mathbf{\Theta}^{(k)}$ containing such pair (i.e. $\hat{\Theta}_{ij}^{(k)} \neq 0$ for all $k : (i, j) \in V_k$ or $\hat{\Theta}_{ij}^{(k)} = 0$ for all $k : (i, j) \in V_k$);

- If the elements associated with the pair (i, j) are equal to zero, then the element associated with the pair (j, i) must be zero to guarantee symmetry.

With these constrains in mind, we propose the *jewel* method, defined by the following joint minimization problem for estimating $\hat{\mathbf{\Theta}}^{(1)}, \ldots, \hat{\mathbf{\Theta}}^{(K)}$:

$$(\hat{\mathbf{\Theta}}^{(1)}, \ldots, \hat{\mathbf{\Theta}}^{(K)}) = \underset{\substack{\mathbf{\Theta}^{(1)} \in \mathbb{R}^{p_1 \times p_1} \\ \vdots \\ \mathbf{\Theta}^{(K)} \in \mathbb{R}^{p_K \times p_K}, \\ diag=0}}{\arg\min} \left\{ \underbrace{\frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} ||\mathbf{X}^{(k)} - \mathbf{X}^{(k)} \mathbf{\Theta}^{(k)}||_F^2}_{multitask\ goodness\ of\ fit} + \underbrace{\lambda \sum_{i<j=1}^{p} \sqrt{g_{ij}} \sqrt{\sum_{k:\{X_i, X_j\} \subseteq V_k} \left(\Theta_{ij}^{(k)}\right)^2 + \left(\Theta_{ji}^{(k)}\right)^2}}_{joint\ penalty} \right\} \tag{4.2}$$

with g_{ij} being the cardinality of the group associated to the pair $\{(i, j\}$, i.e., $g_{ij} = 2|\{k : i, j \in V_k\}|$, and λ – the regularization parameter, controlling the sparsity of the resulting graph.

The joint penalty in Eq.(4.2) enforces the same structure across all datasets, i.e., ensures the fulfillment of the previously stated assumption that for any pair of variables, the edge between them either exists in all graphs where these vertices occur or does not exist in any. It also guarantees symmetry in each class by penalizing simultaneously $\Theta_{ij}^{(k)}$ and $\Theta_{ji}^{(k)}$ elements. As was discussed before, this request is natural considering the underlying graph's structure, but it is rarely met in the methods proposed in the literature, especially those for joint estimation.

Although the minimization problem of Eq.(4.2) can be written and solved when the number of variables p_k is different in each data set (which can occur in some applications), the notations and the formulas simplify a lot when we assume that all variables in the K datasets coincide (i.e., there are no missing variables) – $p_k = p$, $V_k = V$. Hence, from now on, we assume the same number of variables in each dataset, and we denote $G = (V, E)$ the Gaussian graphical model associated with each and all the datasets. In this case, by assumption, all pairs of vertices are present in all graphs, and the cardinality of each group associated with the pair $\{i, j\}$ is equal to $g_{ij} = 2K$.

We use the following notations trough the rest of the chapter:

- λ stands for $\sqrt{2K}\lambda$;
- $\boldsymbol{\theta} = \left(\Theta_{21}^{(1)}, \ldots, \Theta_{p1}^{(1)}, \ldots, \Theta_{1p}^{(K)}, \ldots, \Theta_{(p-1)p}^{(K)}\right)^T$, $\dim(\boldsymbol{\theta}) = p(p-1)K \times 1$, is the vector of the unknown coefficients;
- $\boldsymbol{\theta}_{[ij]} = (\Theta_{ij}^{(1)}, \Theta_{ji}^{(1)}, ..., \Theta_{ij}^{(K)}, \Theta_{ji}^{(K)})$ is the restriction of vector $\boldsymbol{\theta}$ to the variables belonging to the group associated to the pair $\{i, j\}$, hence it has length $g_{ij} = 2K$;

- $X_{.i}^{(k)}$ is the i-th column of matrix $\mathbf{X}^{(k)}$ and $\mathbf{X}_{.-i}^{(k)}$ is the submatrix of $\mathbf{X}^{(k)}$ without the i-th column;

- $\mathbf{y} = \left(X_{.1}^{(1)^T}, \dots, X_{.p}^{(1)^T}, \dots, X_{.1}^{(K)^T}, \dots, X_{.p}^{(K)^T} \right)^T$, is the vector concatenating the columns of all data matrices, $\dim(\mathbf{y}) = Np \times 1$, $N = \sum_{k=1}^{K} n_k$;

-

$$\mathbf{X} = \begin{pmatrix} \begin{pmatrix} \mathbf{X}_{.-1}^{(1)} & 0 & \dots & 0 \\ 0 & \mathbf{X}_{.-2}^{(1)} & \dots & 0 \\ \vdots & & \ddots & \\ 0 & \dots & & \mathbf{X}_{.-p}^{(1)} \end{pmatrix} & & \\ & \ddots & \\ & & \begin{pmatrix} \mathbf{X}_{.-1}^{(K)} & 0 & \dots & 0 \\ 0 & \mathbf{X}_{.-2}^{(K)} & \dots & 0 \\ \vdots & & \ddots & \\ 0 & \dots & & \mathbf{X}_{.-p}^{(K)} \end{pmatrix} \end{pmatrix}$$

  is the block-diagonal matrix made up by the block diagonal matrices $\mathbf{X}_{.-j}^{(k)}$, $k = 1 \dots K$, $j = 1 \dots p$, $\dim(\mathbf{X}) = Np \times p(p-1)K$;

- $\mathbf{D} = blkdiag\left(\frac{1}{\sqrt{n_k}} \mathbf{I}_{n_k p} \right)_{k=1 \dots K}$, $\dim(\mathbf{D}) = Np \times Np$;

- $||\mathbf{u}||_2 = ||\mathbf{u}||$ for all vector $\mathbf{u}$;

- $||\mathbf{u}||_{\mathbf{D}^2}^2 = \mathbf{u}^T \mathbf{D}^2 \mathbf{u} = ||\mathbf{D}\mathbf{u}||_2^2$ for all $\mathbf{u} \in \mathbb{R}^{Np}$.

With these notations, we can rewrite minimization problem of Eq.(4.2) as follows

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^{p(p-1)K}}{\arg\min} \Bigg\{ \underbrace{\frac{1}{2} ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_{\mathbf{D}}^2 + \lambda \sum_{i<j=1}^{p} ||\boldsymbol{\theta}_{[ij]}||}_{F(\boldsymbol{\theta})} \Bigg\}. \tag{4.3}$$

Finally, given the estimated vector $\hat{\boldsymbol{\theta}}$, we can construct the set of edges $\hat{E}$ setting an edge between variables i and j if $\hat{\boldsymbol{\theta}}_{[ij]} \neq 0$, i.e., if the entries in $\hat{\boldsymbol{\theta}}$ corresponding to the group associated to the pair $\{i, j\}$ are different from zero. We stress here that the aim of *jewel* method is not to estimate the precision matrices $\boldsymbol{\Omega}^{(k)}$ but to reconstruct their support. Indeed, matrices $\boldsymbol{\Theta}^{(k)}$ are only a rescaled version of precision matrices with unknown rescaling factor but with the same zero pattern.

4.3 Numerical algorithm

The algorithm for fitting group lasso problems was originally proposed by Yuan and Lin in [71]. The idea is the same as for coordinate descent algorithms but instead of updating individual coordinates we modify the whole group of variables – therefore, we refer to it as *group descent algorithm*. As does the algorithm for the extension of group lasso models to the case of logistic regression by Meier, van de Geer and Bühlmann [50], group descent algorithm requires the orthonormality of the groups. This condition was further explored by Simon and Tibshirani [61] and Breheny and Huang [12], on whose results we base our our findings.

Let us start by noticing that function $F(\boldsymbol{\theta})$ in Eq.(4.3) is convex and separable in terms of the groups. Moreover, we note that matrix $\mathbf{X}$ satisfies the *orthogonal group hypothesis*. This fact means that the restriction of the matrix $\mathbf{X}$ to the columns of each group is orthogonal – in fact, $\mathbf{X}_{\cdot[ij]}$ is orthogonal by construction. Therefore, for a given tuning parameter λ, we can solve the minimization problem by applying the above mentioned iterative group descent algorithm proposed in [12]. The idea is to update each group (pair of variables $\{i, j\}$ and $\{j, i\}$ across all the classes), maintaining the rest of the groups frozen to their current values, then cycling for all $\{i, j\}$ pairs, and repeat until convergence. Let us describe the algorithm in detail.

Given a starting value for vector $\boldsymbol{\theta}$, the *jewel* algorithm updates the group of variables $\{i, j\}$ minimizing function $F(\boldsymbol{\theta})$ for that group and considering the rest of the variables fixed to their current value. Let us fix the group, i.e., the pair $\{i, j\}$ (and $\{j, i\}$) and compute the gradient of $F(\boldsymbol{\theta})$ with respect to the variables $\boldsymbol{\theta}_{[ij]}$. The subgradient is a vector of $g_{ij} = 2K$ components defined as follows

$$\frac{\partial}{\partial \boldsymbol{\theta}_{[ij]}} = \begin{cases} -\left[\mathbf{X}^T\mathbf{D}^2(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right]_{[ij]} + \lambda \dfrac{\boldsymbol{\theta}_{[ij]}}{||\boldsymbol{\theta}_{[ij]}||} & \text{if} \quad ||\boldsymbol{\theta}_{[ij]}|| \neq 0 \\ -\left[\mathbf{X}^T\mathbf{D}^2(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right]_{[ij]} + \lambda \mathbf{v} & \text{if} \quad ||\boldsymbol{\theta}_{[ij]}|| = 0 \end{cases}$$

where $\mathbf{v} \in \mathbb{R}^{2K}$ is any vector with $||\mathbf{v}|| \leq 1$.

From the subgradient we define vector $\mathbf{z} = (z_{ij}^{(1)}, z_{ji}^{(1)}, \ldots, z_{ij}^{(K)}, z_{ji}^{(K)})^T \in \mathbb{R}^{2K}$ with entries

$$\begin{aligned} z_{ij}^{(k)} &= \frac{1}{n_k} X_{\cdot i}^{(k)T} \left(X_{\cdot j}^{(k)} - \sum_{m \neq i,\, j} X_{\cdot m}^{(k)} \Theta_{mj}^{(k)} \right) \\ z_{ji}^{(k)} &= \frac{1}{n_k} X_{\cdot j}^{(k)T} \left(X_{\cdot i}^{(k)} - \sum_{m \neq i,\, j} X_{\cdot m}^{(k)} \Theta_{mi}^{(k)} \right) \end{aligned} \tag{4.4}$$

Vector $\mathbf{z}$ depends on the observed data and the current values of $\boldsymbol{\theta}$. We use it to update regression coefficients by the multivariate soft thresholding operator (see Eq.(3.20)), since it is known from [12] that it is the minimizer of function $F(\boldsymbol{\theta})$ with respect to the variables $\boldsymbol{\theta}_{[ij]}$:

$$\begin{pmatrix} \hat{\Theta}_{ij}^{(1)} \\ \hat{\Theta}_{ji}^{(1)} \\ \vdots \\ \hat{\Theta}_{ij}^{(K)} \\ \hat{\Theta}_{ji}^{(K)} \end{pmatrix} = \left(1 - \frac{\lambda}{||\mathbf{z}||}\right)_+ \mathbf{z}. \tag{4.5}$$

The soft thresholding operator $(\cdot)_+$ acts on vector $\mathbf{z}$ by putting it to zero if its norm is less than regularization parameter λ. In that case, all the regression coefficients of the group associated with the pair $\{i, j\}$ are zeroed too, $\boldsymbol{\theta}_{[ij]} = 0$. On the other hand, if $||\mathbf{z}|| \geq \lambda$, then vector $\mathbf{z}$ is shrunk towards zero and $\boldsymbol{\theta}_{[ij]}$ is updated (see also Algorithm 3).

The update procedure of Eq.(4.4)-(4.5) is repeated cyclically over all the groups with $i < j$ until convergence is achieved. When we proceed in the cycles, we use the latest updated estimate of $\boldsymbol{\theta}$ each time we move to the next group of variables.

Convergence is achieved if either $\sum_k \left|\boldsymbol{\Theta}^{(k,t+1)} - \boldsymbol{\Theta}^{(k,t)}\right| / \sum_k \left|\boldsymbol{\Theta}^{(k,t)}\right| < tol$, where t represents iteration index and *tol* is some user-provided threshold, or if the maximum number of iterations $t_{\max}$ is reached. In Chapter 5 we will investigate the impact of the *tol* stopping criterion on the results. At the end of the iterative procedure, for the pair $\{i, j\}$ the final estimate $\hat{\boldsymbol{\theta}}_{[ij]}$ can be either zero – absence of edge – or different from zero – presence of the edge between variables $\{i, j\}$ in V.

The algorithm can be easily extended to minimize the general model of Eq.(4.2). When the data matrices include different number of variables p_k, vector $\mathbf{z}$ has dimension g_{ij} which can be different for each group of variables $\{i, j\}$. Indeed, $\mathbf{z}$ and $\hat{\boldsymbol{\theta}}_{[ij]}$ incorporate only those datasets for which the variables $\{i, j\}$ were observed. Eq.(4.5) is still valid with $\lambda\sqrt{g_{ij}}$ in place of λ.

Although the *jewel* method's formal description involves large matrices and vectors and several matrix-vector products at each step, its implementation remains computationally feasible even for large datasets. In the following, we will explain why and describe the numerical implementation that we adopted in the R package `jewel` presented in this thesis.

Firstly, let us note that matrix $\mathbf{X}$ and vectors $\mathbf{y}$ and $\boldsymbol{\theta}$, that appear in the minimization problem of Eq.(4.2), do not need to be explicitly built for the implementation of the algorithm. Secondly, the evaluation of vector $\mathbf{z}$, which represents the main computationally demanding part of the updating procedure, can be obtained very efficiently. If we define the residual matrices $\mathbf{R}^{(k)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\boldsymbol{\Theta}^{(k)}$, whose columns are residuals of the linear regressions of the corresponding column j with respect to the others variables $m \neq j$:

$$R_{.j}^{(k)} = X_{.j}^{(k)} - \sum_{m \neq j} X_{.m}^{(k)} \Theta_{mj}^{(k)}, \tag{4.6}$$

and substitute Eq.(4.6) definition into Eq.(4.4), we get

$$
\begin{aligned}
z_{ij}^{(k)} &= \frac{1}{n_k} X_{.i}^{(k)^T} \left(X_{.j}^{(k)} - \sum_{m \neq i,\, j} X_{.m}^{(k)} \Theta_{mj}^{(k)} \right) \\
&= \frac{1}{n_k} X_{.i}^{(k)^T} \Bigg(\underbrace{X_{.j}^{(k)} - \sum_{m \neq j} X_{.m}^{(k)} \Theta_{mj}^{(k)}}_{R_{.j}^{(k)}} + X_{.i}^{(k)} \Theta_{ij}^{(k)} \Bigg) \\
&= \frac{1}{n_k} X_{.i}^{(k)^T} R_{.j}^{(k)} + \underbrace{\frac{1}{n_k} X_{.i}^{(k)^T} X_{.i}^{(k)}}_{1} \Theta_{ij}^{(k)} = \frac{1}{n_k} X_{.i}^{(k)^T} R_{.j}^{(k)} + \Theta_{ij}^{(k)}.
\end{aligned}
$$

This reformulation significantly reduces the complexity of evaluating each element of vector $\mathbf{z}$. Besides, updating the residual matrices $\mathbf{R}^{(k)}$ is also very easy (as described in Algorithm 3) and makes the computational cost affordable.

Moreover, in our implementation of the group descent algorithm, we adopted the active shooting approach, as proposed in [29, 40]. It exploits the problem's sparse nature efficiently, providing an increase in computational speed. The idea is the following: we divide all pairs of variables into "active" – for which corresponding $\boldsymbol{\theta}_{[ij]} \neq 0$ at the current step – and "non-active" – for which corresponding $\boldsymbol{\theta}_{[ij]} = 0$. If after the soft-thresholding evaluation $\boldsymbol{\theta}_{[ij]}$ is zeroed, then "active" status of pair $\{i, j\}$ changes. Once a pair reaches the "non-active" status, then it is never updated again. As we cycle only over the active variables, this strategy reduces the number of groups updated in each iteration, saving computational cost.

From a computational point of view, we define the upper triangular matrix **Active** of dimension $p \times p$, which tracks the current support of $\boldsymbol{\Theta}^{(k)}$, $k = 1 \dots K$ (which are the same for all k by hypothesis). **Active** can be initialized by setting all extra diagonal elements equal to 1, meaning that at the beginning of the procedure, all the groups $\{i, j\}$ are "active". However, other choices can be used if some prior knowledge of the network structure is available. During the iterations, if the soft-thresholding operator zeroes group $\{i, j\}$ coefficient $\boldsymbol{\theta}_{[ij]}$, then $\mathbf{Active}_{ij}$ is set to zero, indicating that that group is no more active. Therefore, its will not be updated anymore. At the end of the algorithm, matrix **Active** contains the estimate of the edge set E, since edge $(i, j) \in \hat{E} \iff \mathbf{Active}_{ij} = 1$.

In Algorithm 3 we provide the pseudo-code for the implementation of *jewel* for a fixed parameter λ. The implementation of the *jewel* method into the R package `jewel` is described in Appendices B and C, where one can also find a discussion on speed-ups provided by the choice of R data structures and functions. Subsection 5.3.3 of Chapter 5 discusses the influence (or, to be exact, its absence) of the *tol* parameter on the estimate of the zero-elements of **Active**. Thus, in the simulation we use $tol = 0.01$ to speed up the calculations. However, this parameter might influence the evaluation of the residual error used to estimate λ as described in Section 4.4. In that case, the default is $tol = 10^{-4}$.

Algorithm 3 The *jewel* algorithm

INPUT: $\mathbf{X}^{(1)}, ..., \mathbf{X}^{(K)}$, λ, *tol* and $t_{\max}$

INITIALIZE:
$\boldsymbol{\Theta}^{(1,0)}, ..., \boldsymbol{\Theta}^{(K,0)}$
$\mathbf{R}^{(k)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\boldsymbol{\Theta}^{(k,0)}$, $k = 1...K$

$$\textbf{Active} = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

REPEAT UNTIL CONVERGENCE
for $j = 1 \dots p$ **do**
  for $i : Active_{ij} \neq 0$ **do**
    evaluate $\mathbf{z} = \left(z_{ij}^{(1)}, z_{ji}^{(1)}, \dots, z_{ij}^{(K)}, z_{ji}^{(K)}\right)$ by

$$z_{ij}^{(k)} = \frac{1}{n_k} X_{\cdot i}^{(k)^T} R_{\cdot j}^{(k)} + \Theta_{ij}^{(k,t)}$$
$$z_{ji}^{(k)} = \frac{1}{n_k} X_{\cdot j}^{(k)^T} R_{\cdot i}^{(k)} + \Theta_{ji}^{(k,t)}$$

    evaluate $thrshld = 1 - \lambda / \|\mathbf{z}\|$
    if $thrshld < 0$ **then**
      $Active_{ij} \leftarrow 0$ and $\mathbf{z} \leftarrow 0$
    else
      $\mathbf{z} \leftarrow \mathbf{z} \cdot thrshld$
    end if
    update residuals

$$R_{\cdot j}^{(k)} = R_{\cdot j}^{(k)} + X_{\cdot i}^{(k)} \left(\Theta_{ij}^{(k,t)} - z_{ij}^{(k)}\right)$$
$$R_{\cdot i}^{(k)} = R_{\cdot i}^{(k)} + X_{\cdot j}^{(k)} \left(\Theta_{ji}^{(k,t)} - z_{ji}^{(k)}\right)$$

    update coefficients $(\Theta_{ij}^{(1,t)}, \Theta_{ji}^{(1,t)}, ..., \Theta_{ij}^{(K,t)}, \Theta_{ij}^{(K,t)}) \leftarrow \mathbf{z}$
  end for
end for
Stop if $\dfrac{\sum_k \left|\boldsymbol{\Theta}^{(k,t+1)} - \boldsymbol{\Theta}^{(k,t)}\right|}{\sum_k \left|\boldsymbol{\Theta}^{(k,t)}\right|} < tol$ or $t > t_{\max}$

OUTPUT: **Active**

In the next section we will discuss the choice of regularization parameter λ.

4.4 Regularization parameter selection

Like any other penalty-based method, *jewel* requires selection of regularization parameter λ which controls the resulting estimator's sparsity. A high value of λ can lead to a more sparse and, therefore, more interpretable estimator. However, it can also result in many false negative edges. By contrast, a small value of λ can lead to estimates with many false positive edges. How to find a balance between these two extremes?

In Section 4.5 we provide both an upper and a lower bound on λ to assure the estimator's sparsistency property. However, such limits are mostly theoretical, and are hard to be estimated in a real case. Many authors resort to using $\lambda = \sqrt{\log p/n}$ or suggest empirical application-driven choice so that resulting model is sufficiently complex to provide novel information and at the same time sufficiently sparse to be interpretable. Others, as well as we did, suggest data-driven methods for estimating the optimal λ. For this purpose we adapt the Bayesian information criterion and the F-fold cross-validation[1] for the case of K datasets. We refer to the Chapter 5 for performance evaluation of both approaches.

4.4.1 Bayesian information criterion

We can adapt the Bayesian information criterion (BIC) to the case of K datasets following the idea proposed in [40] for a single dataset. To define BIC for k-th class, we need to evaluate the logarithm of the residual sum of squares (RSS) and the degree of freedom. In our implementation of *jewel* these quantities come almost for free because the RSSs are the columns of the residual matrices $\mathbf{R}^{(k)}$, and the degree of freedom is the number of non-zero pairs in the **Active** matrix. Specifically, we have

$$BIC^{(k)}(\lambda) = n_k \sum_{i=1}^{p} \log \left|\left| R_{\cdot i}^{(k)} \right|\right|^2 + \log n_k \#\{Active_{ij}(\lambda) \neq 0\}$$

We then define BIC for *jewel* as the weighted sum of the BICs of the individual classes:

$$BIC(\lambda) = \sum_{k=1}^{K} BIC^{(k)}(\lambda) = \sum_{k=1}^{K} n_k \sum_{i=1}^{p} \log \left|\left| R_{\cdot i}^{(k)} \right|\right|^2 + \#\{Active_{ij}(\lambda) \neq 0\} \sum_{k=1}^{K} \log n_k.$$

For the grid of parameters $\lambda_1 < \lambda_2 \cdots < \lambda_L$ the optimal one is obtained as $\lambda_{BIC} = \underset{\lambda_l,\, l=1\ldots L}{\arg\min} BIC(\lambda_l)$. Algorithm 4 shows a sketch of the BIC criterion pseudo-code.

[1] Although the standard term is "k-fold cross-validation", we change it in order to avoid confusion with $k = 1 \ldots K$ being the index of k-th class of the given data.

Algorithm 4 Choosing λ_{BIC}

Fix the grid $\lambda_1 < \lambda_2 \cdots < \lambda_L$.
for $l = 1 \dots L$ **do**
  evaluate $\hat{\mathbf{\Theta}}^{(k)}(\lambda_l)$ and $\mathbf{R}^{(k)}(\lambda_l)$, $k = 1 \dots K$
  evaluate BIC(λ_l)
end for
Estimate $\lambda_{BIC} = \underset{\lambda_l,\, l=1\dots L}{\arg\min} BIC(\lambda_l)$

4.4.2 Cross-validation

The idea of cross-validation (CV) is to split the data into F folds and consequentially use one fold as a testing set and all the others as a training set. In our *jewel* procedure, we divide each data set $\mathbf{X}^{(k)}$ into F folds (with dimension $n_k^f \times p$ each, $n_k^f \approx n_k/F$) and the f-th fold is the union of the f-th fold of each class. As in standard CV procedure, for each parameter λ_l of the grid $\lambda_1 < \cdots < \lambda_L$ and for each fold $(\mathbf{X}_f^{(k)})_{k=1\dots K}$ (training set) we estimate $\hat{\mathbf{\Theta}}_{-f}^{(k)}(\lambda_l)$ and then evaluate its error on the testing set as

$$err(f,l) = \sum_{k=1}^{K} \frac{1}{n_k^f} \left|\left| \mathbf{X}_f^{(k)} - \mathbf{X}_f^{(k)} \hat{\mathbf{\Theta}}_{-f}^{(k)}(\lambda_l) \right|\right|_F^2 .$$

Errors are then averaged over folds $CV(\lambda_l) = \frac{1}{F}\sum_{f=1}^{F} err(f,l)$ and the optimal parameter is chosen as the minimum $\lambda_{CV} = \underset{\lambda_l,\, l=1\dots L}{\arg\min} CV(\lambda_l)$. Algorithm 5 provides CV criterion pseudo-code.

Algorithm 5 Choosing λ_{CV}

Fix the grid $\lambda_1 < \lambda_2 \cdots < \lambda_L$.
Split the data into F-folds: $\mathbf{X}_f^{(k)}$, dim $= n_k^f \times p$, $f = 1 \dots F$, $k = 1 \dots K$.
for $f = 1 \dots F$ **do**
  for $l = 1 \dots L$ **do**
    obtain $\hat{\mathbf{\Theta}}_{-f}^{(k)}(\lambda_l)$ with data $\mathbf{X}_{-f}^{(k)}$, $k = 1 \dots K$, and given λ_l
    evaluate $err(f,l) = \sum_{k=1}^{K} \frac{1}{n_k^f} \left|\left| \mathbf{X}_f^{(k)} - \mathbf{X}_f^{(k)} \hat{\mathbf{\Theta}}_{-f}^{(k)}(\lambda_l) \right|\right|_F^2$
  end for
end for
for $l = 1 \dots L$ **do**
  evaluate $CV(\lambda_l) = \frac{1}{F}\sum_{f=1}^{F} err(f,l)$
end for
Estimate $\lambda_{CV} = \underset{\lambda_l,\, l=1\dots L}{\arg\min} CV(\lambda_l)$.

Note that although cross-validation is often considered a heavy computational procedure since all evaluations have to be carried F times, in our implementation,

the evaluation of $err(f, l)$, $l = 1 \dots L$, $f = 1 \dots F$, can be parallelized over folds. Thus, if the number of folds is smaller than/close to the number of available cores, we can cope with the runtime disadvantage – see Chapter 5.

4.4.3 Warm start

When applying *jewel* over a grid of parameters $\lambda_1 < \lambda_2 \cdots < \lambda_L$, one can use a warm start procedure. The idea is to use the estimator $\hat{\boldsymbol{\Theta}}^{(k)}(\lambda_l)$, $k = 1 \dots K$, obtained with parameter λ_l, for the initialization of the algorithm for parameter λ_{l+1}. Combined with active shooting approach, this allows us to start with matrix $\mathbf{Active} = \operatorname{supp} \hat{\boldsymbol{\Theta}}^{(k)}$ (for any k, since by *jewel* construction all the supports are the same) with plenty of zeros and thus decrease the number of considered active pairs and, therefore, the running time.

In the simulation, we implemented warm start for BIC and cross-validation (inside each fold) – see Chapter 5. However, evaluation of the performance and running time was carried without warm start procedure to eliminate any possibilities of its influence on performance.

4.5 Variable selection consistency

In this section, we establish the variable selection consistency property for the proposed estimator. Its main idea is to provide necessary conditions to ensure that the probability of correctly identifying edges or absence of edges is large. In other words, it derives an upper bound on the parameter λ such that with high probability, we can correctly recover the true edges. At the same time, it derives a lower bound on parameter λ such that with high probability, we can accurately recover the absence of edges. However, since the conditions are given on the true (unknown) parameters $\boldsymbol{\theta}$, it is not possible to use them to derive the optimal λ explicitly – that is why we implemented BIC and CV procedures, described in the previous section.

The findings of this section are largely based on [8].

Let's start with formulation of the minimization problem given in Eq.(4.3) by vector $\boldsymbol{\theta} = \left(\Theta_{21}^{(1)}, \dots, \Theta_{p1}^{(1)}, \dots, \Theta_{1p}^{(K)}, \dots, \Theta_{(p-1)p}^{(K)}\right)^T$. Before presenting the main result in Theorem 3, let us introduce some auxiliary notations and Lemma 1, which will be useful in the proof of the theorem.

Let us denote $\boldsymbol{\theta}^{\mathbf{0}}$ the true parameter vector of dimension $Kp(p-1) \times 1$. It is our unknown, and its non-zero components describe the true edge set E of the graph. $\boldsymbol{\theta}^0$ is naturally divided into $\binom{p}{2}$ groups, each consisting of the true parameters $\boldsymbol{\theta}^0_{[ij]}$ whose row/column index refer to the same pair $\{i, j\}$. Let s denote the true

number of edges in E and define the sets of "active" and "non-active" pairs as

$$S = \{(i,j) : i < j, \boldsymbol{\theta}^0_{[ij]} \not\equiv 0\} = \{(i,j) : i < j, (i,j) \in E\}, \quad |S| = s$$
$$S^c = \{(i,j) : i < j, \boldsymbol{\theta}^0_{[ij]} \equiv 0\} = \{(i,j) : i < j, (i,j) \notin E\}, \quad |S^c| = \frac{p(p-1)}{2} - s = q,$$

respectively. Therefore, S contains all pairs of nodes for which there is an edge in E and S^c contains all pairs of nodes for which there is an absence of edge.

Now let us define analogs of regression problem elements in the case of equivalent formulation of Eq.(4.3). We define matrices $\boldsymbol{\Lambda}^{(k)}$ and vector $\boldsymbol{\varepsilon}$ (similar to Proposition 5):

$$\begin{aligned}
\boldsymbol{\Lambda}^{(k)} &= diag\left(\frac{1}{\Omega^{(k)}_{11}}, \ldots, \frac{1}{\Omega^{(k)}_{pp}}\right) \\
\boldsymbol{\varepsilon} &= (\underbrace{\varepsilon_1^{(1)^T}, \ldots, \varepsilon_p^{(1)^T}}_{\sim \mathcal{N}_{n_1 p}\left(0, \boldsymbol{\Lambda}^{(1)} \otimes \mathbf{I}_{n_1}\right)}, \ldots, \underbrace{\varepsilon_1^{(K)^T}, \ldots, \varepsilon_p^{(K)^T}}_{\sim \mathcal{N}_{n_K p}\left(0, \boldsymbol{\Lambda}^{(K)} \otimes \mathbf{I}_{n_K}\right)})^T, \quad \dim(\boldsymbol{\varepsilon}) = Np \times 1 \\
\boldsymbol{\varepsilon} &\sim \mathcal{N}_{Np}\left(0, blkdiag\left(\boldsymbol{\Lambda}^{(k)} \otimes \mathbf{I}_{n_k}\right)_{k=1\ldots K}\right).
\end{aligned}$$

We then define the analog of empirical covariance matrix $\mathbf{C}$ and some auxiliary stochastic matrix and vectors

$$\begin{aligned}
\mathbf{C} &= \mathbf{X}^T\mathbf{D}^2\mathbf{X}, & \dim(\mathbf{C}) &= p(p-1)K \times p(p-1)K \\
\boldsymbol{\zeta} &= \mathbf{X}^T\mathbf{D}^2\boldsymbol{\varepsilon}, & \dim(\boldsymbol{\zeta}) &= p(p-1)K \times 1 \\
\mathbf{w} &= \boldsymbol{\zeta}_{S^c} - \mathbf{C}_{S^cS}\mathbf{C}^{-1}_{SS}\boldsymbol{\zeta}_S, & \dim(\mathbf{w}) &= 2Kq \times 1 \\
\mathbf{v} &= \mathbf{C}^{-1}_{SS}\boldsymbol{\zeta}_S, & \dim(\mathbf{v}) &= 2Ks \times 1,
\end{aligned}$$

where $\boldsymbol{\zeta}_A$ and $\mathbf{C}_{AA}$ denote the restriction of the vector $\boldsymbol{\zeta}$ and the matrix $\mathbf{C}$ to the rows and columns to the set A.

Given those definitions and by properties of standard multivariate Gaussian distribution, our data can be equivalently modeled as standard linear regression problem $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}^0 + \boldsymbol{\varepsilon}$. Indeed, formulation of Eq. (4.3) is a standard linear regression problem with a group lasso penalty, for which the following lemma holds:

Lemma 1 (Group lasso estimate characterization, cfr. Lemma A.1 in [8])**.** *A vector $\hat{\boldsymbol{\theta}}$ is a solution to convex optimization problem (4.3) if and only if there exists $\boldsymbol{\tau} \in \mathbb{R}^{p(p-1)K}$ such that $[\mathbf{X}^T\mathbf{D}^2(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})] = \lambda\boldsymbol{\tau}$ and*

$$\boldsymbol{\tau}_{[ij]} = \begin{cases} \lambda dir\left(\hat{\boldsymbol{\theta}}_{[ij]}\right), & if \quad \hat{\boldsymbol{\theta}}_{[ij]} \not\equiv 0 \\ \lambda\mathbf{u}, \mathbf{u} \in \mathbb{R}^{2K}, ||\mathbf{u}|| \leq 1 & if \quad \hat{\boldsymbol{\theta}}_{[ij]} \equiv 0, \end{cases}$$

where $dir(\mathbf{u}) = \mathbf{u}/||\mathbf{u}||$ is the directional vector of any non-zero vector $\mathbf{u}$.

We can now state the main result of this section, which has been inspired by Theorem 4.1 of [8].

Theorem 3. *Let $\hat{\boldsymbol{\theta}}$ be the solution of problem in Eq.(4.3), with $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}^0 + \boldsymbol{\varepsilon}$. Suppose that there exists $\delta > 0$ such that, with probability at least $1 - e^{-\delta \log(p)/N}$, one has*

1. *$\mathbf{C}_{SS}$ is invertible.*
2. *(Irrepresentable condition): $\exists\ \alpha \in (0,1) : \forall (i,j) \in S^c$*
   (a) $\left|\left|[\mathbf{C}_{S^cS}\mathbf{C}_{SS}^{-1}\boldsymbol{\tau}]_{ij}\right|\right| \leq \alpha\ \forall \boldsymbol{\tau} \in \mathbb{R}^{2Ks} : \max_{(i,j)\in S} ||\boldsymbol{\tau}_{[ij]}|| \leq 1$
   (b) $\lambda \geq \dfrac{2}{1-\alpha}||\mathbf{w}_{[ij]}||$
3. *(Signal strength): $\forall\ (i,j) \in S$ it holds*

$$\lambda < \left\{||\boldsymbol{\theta}^0_{[ij]}||_2 - ||\mathbf{v}_{[ij]}||\right\} \left|\left|\left[\mathbf{C}_{SS}^{-1}\boldsymbol{\tau}\right]_{[ij]}\right|\right|^{-1} \quad \forall \boldsymbol{\tau} \in \mathbb{R}^{2Ks} : \max_{(i,j)\in S} ||\boldsymbol{\tau}_{[ij]}|| \leq 1$$

then, $\mathbb{P}(\hat{E} = E) \geq 1 - e^{-\delta \log(p)/N}$, where
$E = \{(i,j) : \boldsymbol{\theta}^0_{[ij]} \not\equiv 0\}$ is the true edge set and
$\hat{E} = \{(i,j) : \hat{\boldsymbol{\theta}}_{[ij]} \not\equiv 0\}$ is the estimated edge set.

Proof. To prove set equality $\hat{E} = E$, we verify separately the two inclusions, $\hat{E} \subseteq E$ and $\hat{E} \supseteq E$. Let us first prove inclusion $\hat{E} \subseteq E \iff \hat{\boldsymbol{\theta}}_{[ij]} \equiv 0 \quad \forall (i,j) \in S^c$.

Define $\hat{\boldsymbol{\theta}}^S$ be the solution of the following restricted problem

$$\hat{\boldsymbol{\theta}}^S := \underset{\boldsymbol{\theta}\in\mathbb{R}^{2Ks}}{\arg\min} \left\{ \frac{1}{2}\,||\mathbf{y} - \mathbf{X}_{\cdot S}\boldsymbol{\theta}||^2_{\mathbf{D^2}} + \lambda \sum_{(i,j)\in S} ||\boldsymbol{\theta}_{[ij]}|| \right\}.$$

By Lemma 1, $\exists\ \boldsymbol{\tau}^S \in \mathbb{R}^{2Ks}$ such that $-\mathbf{X}_{\cdot S}^T\mathbf{D}^2(\mathbf{y} - \mathbf{X}_{\cdot S}\hat{\boldsymbol{\theta}}^S) + \lambda\boldsymbol{\tau}^S = 0$ and

$$\boldsymbol{\tau}^S_{[ij]} = \begin{cases} \hat{\boldsymbol{\theta}}^S_{[ij]}/||\hat{\boldsymbol{\theta}}^S_{[ij]}||, & if \quad \hat{\boldsymbol{\theta}}^S_{[ij]} \not\equiv 0 \\ \mathbf{u} \in \mathbb{R}^{2K}, ||\mathbf{u}|| \leq 1, & if \quad \hat{\boldsymbol{\theta}}^S_{[ij]} \equiv 0. \end{cases}$$

Define $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{p(p-1)K}$ such that its restriction to the set of active groups coincides with $\hat{\boldsymbol{\theta}}^S$, while its restriction to the set of non-active groups is zero, i.e.,

$$\hat{\boldsymbol{\theta}}_{[ij]} = \begin{cases} \hat{\boldsymbol{\theta}}^S_{[ij]}, & if \quad (i,j) \in S \\ 0, & if \quad (i,j) \in S^c. \end{cases}$$

To get the first inclusion, we need to prove that $\hat{\boldsymbol{\theta}}$ is a solution of the full problem in Eq.(4.3). By Lemma 1 it is sufficient to prove that $\exists\ \boldsymbol{\tau} \in \mathbb{R}^{p(p-1)K}$: $-\mathbf{X}^T\mathbf{D}^2(Y - \mathbf{X}\hat{\boldsymbol{\theta}}) + \lambda\boldsymbol{\tau} = 0$ and

$$\boldsymbol{\tau}_{[ij]} = \begin{cases} \hat{\boldsymbol{\theta}}_{[ij]}/||\hat{\boldsymbol{\theta}}_{[ij]}||, & if \quad \hat{\boldsymbol{\theta}}_{[ij]} \not\equiv 0 \\ \mathbf{u} \in \mathbb{R}^{2K}, ||\mathbf{u}|| \leq 1, & if \quad \hat{\boldsymbol{\theta}}_{[ij]} \equiv 0. \end{cases} \tag{4.7}$$

When $\hat{\boldsymbol{\theta}}_{[ij]} \not\equiv 0$, conditions in Eq.(4.7) are satisfied by construction of $\hat{\boldsymbol{\theta}}$. When $\hat{\boldsymbol{\theta}}_{[ij]} \equiv 0$, conditions in Eq.(4.7) need to be verified. To this aim, substitute $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}^0 + \boldsymbol{\varepsilon}$ into $-\mathbf{X}^T\mathbf{D}^2(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) + \lambda\boldsymbol{\tau} = 0$ and get

$$\begin{aligned} -\mathbf{X}^T\mathbf{D}^2(\mathbf{X}\boldsymbol{\theta}^0 + \boldsymbol{\varepsilon} - \mathbf{X}\hat{\boldsymbol{\theta}}) + \lambda\boldsymbol{\tau} &= 0 \\ \underbrace{\mathbf{X}^T\mathbf{D}^2\mathbf{X}}_{\mathbf{C}}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0) - \underbrace{\mathbf{X}\mathbf{D}^2\boldsymbol{\varepsilon}}_{\boldsymbol{\zeta}} + \lambda\boldsymbol{\tau} &= 0 \\ \mathbf{C}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0) - \boldsymbol{\zeta} + \lambda\boldsymbol{\tau} &= 0 \end{aligned} \tag{4.8}$$

After properly permuting the indexes of $\mathbf{C}$, $\boldsymbol{\zeta}$ and $\boldsymbol{\tau}$, i.e. placing all the variables belonging to the active groups at the beginning and the non-active ones at the end, Eq.(4.8) becomes

$$\begin{pmatrix} \mathbf{C}_{SS} & \mathbf{C}_{SS^c} \\ \mathbf{C}_{S^cS} & \mathbf{C}_{S^cS^c} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0 \\ 0 \end{pmatrix} - \begin{pmatrix} \boldsymbol{\zeta}_S \\ \boldsymbol{\zeta}_{S^c} \end{pmatrix} + \lambda \begin{pmatrix} \boldsymbol{\tau}_S \\ \boldsymbol{\tau}_{S^c} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

This is equivalent to

$$\begin{cases} \mathbf{C}_{SS}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0) - \boldsymbol{\zeta}_S + \lambda\boldsymbol{\tau}_S & = 0 \\ \mathbf{C}_{S^cS}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0) - \boldsymbol{\zeta}_{S^c} + \lambda\boldsymbol{\tau}_{S^c} & = 0 \end{cases}$$

Solving the first equation for $\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0$, and substituting into the second, we obtain

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0 = \mathbf{C}_{SS}^{-1}(\boldsymbol{\zeta}_S - \lambda\boldsymbol{\tau}_S) \tag{4.9}$$

and then

$$\begin{aligned} \mathbf{C}_{S^cS}\mathbf{C}_{SS}^{-1}(\boldsymbol{\zeta}_S - \lambda\boldsymbol{\tau}_S) - \boldsymbol{\zeta}_{S^c} + \lambda\boldsymbol{\tau}_{S^c} &= 0 \\ \boldsymbol{\tau}_{S^c} = -\frac{1}{\lambda}\mathbf{C}_{S^cS}\mathbf{C}_{SS}^{-1}(\boldsymbol{\zeta}_S - \lambda\boldsymbol{\tau}_S) + \frac{\boldsymbol{\zeta}_{S^c}}{\lambda} \\ \boldsymbol{\tau}_{S^c} = \frac{1}{\lambda}\underbrace{(\boldsymbol{\zeta}_{S^c} - \mathbf{C}_{S^cS}\mathbf{C}_{SS}^{-1}\boldsymbol{\zeta}_S)}_{\mathbf{w}} + \mathbf{C}_{S^cS}\mathbf{C}_{SS}^{-1}\boldsymbol{\tau}_S, \end{aligned}$$

By using hypothesis 2) of the theorem, we get $\forall (i,j) \in S^c$

$$||\boldsymbol{\tau}_{[ij]}^{S^c}|| \leq \frac{1}{\lambda}||\mathbf{w}_{[ij]}|| + ||\left[\mathbf{C}_{S^cS}\mathbf{C}_{SS}^{-1}\boldsymbol{\tau}_S\right]_{[ij]}|| < \frac{\alpha+1}{2} < 1.$$

The second inclusion requires $\hat{E} \supseteq E \iff \hat{\boldsymbol{\theta}}_{[ij]} \not\equiv 0 \ \forall (i,j) \in S$.

We observe that the second inclusion is implied by $||\hat{\boldsymbol{\theta}}_{[ij]} - \boldsymbol{\theta}_{[ij]}^0|| < ||\boldsymbol{\theta}_{[ij]}^0||$ $\forall\ (i,j) \in S$ which is a stronger requirement called *direction consistency* in the original paper [8]. Starting from Eq.(4.9), we get

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^0 = \underbrace{\mathbf{C}_{SS}^{-1}(\boldsymbol{\zeta}_S}_{\mathbf{v}} - \lambda\boldsymbol{\tau}_S) = \mathbf{v} - \lambda\mathbf{C}_{SS}^{-1}\boldsymbol{\tau}_S.$$

Then, by hypothesis 3) of the theorem, we have that $\forall (i,j) \in S$

$$||\hat{\boldsymbol{\theta}}_{[ij]} - \boldsymbol{\theta}_{[ij]}^0|| \leq ||\mathbf{v}_{[ij]}|| + \lambda||\left[\mathbf{C}_{SS}^{-1}\boldsymbol{\tau}_S\right]_{[ij]}|| < ||\boldsymbol{\theta}_{[ij]}^0||.$$

□

Hypotheses of Theorem 3 are weaker than the assumptions of Theorem 4.1 in [8]. In fact, in our setting, the stochastic matrix $\mathbf{C}$ and vector $\boldsymbol{\zeta}$ do not inherit the Gaussian distribution from the data. Hence, our results are based on a probabilistic assumption on these stochastic objects and not on the underlying families of Gaussian distribution, i.e., on their covariance matrices $\boldsymbol{\Sigma}^{(k)}$, $k = 1 \dots K$. However, if, from one hand, this could be a limitation, on the other side, our result provides explicit conditions on the data that, in principle, could be verified given an estimate of vector $\hat{\boldsymbol{\theta}}$. The same would not be possible when the assumptions involve the population matrices $\boldsymbol{\Sigma}^{(k)}$ instead of the population vector $\boldsymbol{\theta}^0$.

4.6 jewel and other joint estimation methods

In this section, we will provide a brief discussion of *jewel* and other joint estimation methods, described in the Section 3.4 of Chapter 3. We will refer to Appendix A with methods' review in a table form for ease of comparison.

While *jewel* is a regression-based method, all the other reviewed joint estimation methods are maximum-likelihood based – even though JSEM is a combination of both approaches and PNJGL/CNJGL use node-based log-likelihood (see the 2nd column of the table in Appendix A). Therefore, unlike other methods, *jewel* does not estimate the entries of precision matrices, only matrices' common support. Some may consider this a disadvantage, however, since we are mostly interested in estimating the graph, focus on its adjacency matrix seems adequate and even more computationally efficient. Indeed, with the proper implementation (like the active-shooting approach that we adopted), *jewel* can be faster than maximum-likelihood based methods (as we will see in the next Chapter 5).

Another noticeable difference between *jewel* and other joint estimation methods is the penalty, namely the fact that *jewel*'s penalty does not yet accommodate the differences between classes. From the 3rd column of the table, we can see that almost all the methods' penalties comprise of two terms: one enforcing the similar structure of the underlying graphs across classes and one allowing for class-specific edges (with exception of Guo et. al proposal that is, however, a reformulation of the problem with two penalty terms). This is certainly something we would like to work on in the future since it can be useful in real data applications. As we will see from the next chapter, under the assumption of the same common graph, *jewel* demonstrated performance comparable to other methods, therefore, we consider it promising and worth further modifications.

Despite the described drawback, *jewel*'s penalty is also its biggest advantage, since it is a group penalty that enforces symmetry in the resulting adjacency matrix. This allows us to avoid any post-processing, proving the final result by the end of the algorithm. As we mentioned before, in the undirected graph setting symmetry of the adjacency matrix is very natural to assume, however, as we can see from the 3rd column of the comparison table, no other methods' penalties provide this property.

Concerning the algorithms for the minimization problem solution (columns 4-5 of the table), many of the reviewed joint estimation methods use the alternating directions method of multipliers (ADMM) and/or the coordinate descent algorithm that has the same idea as the group descent algorithm, used in *jewel*. As do authors of other methods, we prove the consistency of the resulting estimator.

Implementation of the method in an R package (as we did with `jewel`) is not very common, as we can observe from table's 6th column – in fact, only two joint estimation methods have corresponding packages. Therefore, we consider this an advantage of our work. Regarding simulation and real data applications, we conducted the studies for bigger orders of p than most of the reviewed methods, as we will see from Chapters 5 and 6.

Chapter 5

Simulations

This chapter presents several simulation studies' results to demonstrate the empirical performance of *jewel* from different perspectives. Specifically, I conducted four types of experiments. First, I explored the advantages of the joint approach: why using several datasets is beneficial and why the joint approach is better than fitting K independent problems and then using a voting strategy or fitting a single model after concatenating datasets. Second, I explored the influence of different input parameters (like tolerance level) and data parameters (like sparsity or number of variables or samples, i.e., dimensional regime) on the performance of the method. Then, I compared *jewel* with other available methods for joint estimation, such as JGL [19] and Guo et al. proposal [36] in terms of performance (using ROC-curve) and running time. Finally, I evaluated the BIC and CV procedures for selecting the regularization parameter λ for *jewel*.

Before presenting the results, I will describe the generation of the data and the metrics I used in all simulation settings. All results were obtained using the novel R package `jewel`, that I developed, and some auxiliary code (see appendix B for package description and documentation and appendix C for some implementation discussion). The R code for other methods was either publicly available or provided by authors upon request.

5.1 Simulation set-ups

In this work, we implemented a data simulation workflow similar to those used in [19, 36, 48], which is described in the first part of this section. Then, we used it to generate several simulated scenarios. The idea is the following: we first generate a "true" graph G and use it as a support to create K precision matrices $\mathbf{\Omega}^{(k)}$. These matrices are used to construct K covariance matrices $\mathbf{\Sigma}^{(k)}$ from which we can generate K datasets (each of size n_k) sampling from the p-variate normal distribution.

In the second part of this section, we will present the performance evaluation metrics, such as the true positive and false positive rate, the ROC-curve, etc. In the last subsection, we underline minor differences between the algorithm, described in Section 4.3, and the implementation.

5.1.1 Data generation

We start with generating a scale-free graph $G = (V, E)$, i.e., a network whose degree distribution follows a power law. Scale-free networks are believed to describe many natural processes such as protein interactions, social networks, or internet web-graphs. We can generate scale-free graph using the Barabasi-Albert algorithm [6]: new nodes are being connected to the existing ones at each step with the probability proportional to their degree ($p_i = d_i / \sum_j d_j$, where i is one of the existing nodes).

For this purpose we used function `barabasi.game` from the `igraph` package [17]. The input parameters here are the number of variables p, i.e., number of nodes in G, the number of edges added at each step of the algorithm m (controls the sparsity) and the power of the preferential attachment *power*. The resulting graph G is sparse and has $mp - (2m - 1)$ edges. If not stated otherwise, we used $m = 1$ and $power = 1$.

Then, we generate K precision matrices $\mathbf{\Omega}^{(k)}$. To this purpose, for each k, we create a $p \times p$ matrix with 0s for the elements not corresponding to the network edges and symmetric values sampled from the uniform distribution with support on $[-b, -a] \cup [a, b]$ for the elements corresponding to the existing edges. In our work we used $a = 0.2, b = 0.8$, although we noticed that choice of this interval (which we found to be quite diverse in other papers, see Appendix A) has little to no influence on the performance of the method (see Subsection 5.3.2). To ensure positive definiteness, diagonal elements of $\mathbf{\Omega}^{(k)}$ are set equal to $|\eta_{min}(\mathbf{\Omega}^{(k)})| + 0.1$, with $\eta_{min}(\mathbf{A})$ being the minimum eigenvalue of matrix $\mathbf{A}$. Such requirement can be easily verified using the `is.positive.definite` function from `matrixcalc` package [56].

We invert $\mathbf{\Omega}^{(k)}$ with `chol2inv` function and construct $\mathbf{\Sigma}^{(k)}$ with elements

$$\Sigma_{ij}^{(k)} = \frac{\left(\Omega^{(k)}\right)_{ij}^{-1}}{\sqrt{\left(\Omega^{(k)}\right)_{ii}^{-1} \left(\Omega^{(k)}\right)_{jj}^{-1}}}.$$

Finally, for each k, we sample $n_k = n \ \forall k$ independent identically distributed observations from $\mathcal{N}(0, \mathbf{\Sigma}^{(k)})$ with `mvrnorm` function from `MASS` package [66] to get K datasets $\mathbf{X}^{(k)}$.

5.1.2 Performance measures

For each estimated edge set $\hat{E}$ of the true graph structure E, we evaluated

- the *true positives* (TP) – the number of correctly identified edges;
- the *true negatives* (TN) – the number of the correctly identified absence of edges;
- the *false positives* (FP) – the number of estimated edges which were not present in the true graph;

- the *false negatives* (FN) – the number of not identified edges that were present in the true graph.

We can formally define them as

$$TP = |\{(i,j) : (i,j) \in E \cap \hat{E}\}|, \qquad TN = |\{(i,j) : (i,j) \in E^c \cap \hat{E}^c\}|,$$
$$FP = |\{(i,j) : (i,j) \in E^c \cap \hat{E}\}|, \qquad FN = |\{(i,j) : (i,j) \in E \cap \hat{E}^c\}|,$$

where E^c and $\hat{E}^c$ denote sets complements. We then computed the *true positive rate* (TPR) and the *false positive rate* (FPR), defined respectively as

$$TPR = \frac{TP}{TP + FN} \qquad \text{and} \qquad FPR = \frac{FP}{FP + TN}.$$

TPR shows the proportion of edges correctly identified (or probability of identifying the edge correctly), and FPR shows the proportion of edges incorrectly identified (or probability to miss an absence of an edge).

To eliminate the influence of the specific random sample, we generated 20 graphs G for each fixed set of input parameters. Then, for each graph G we generated K datasets $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ by the procedure described in the previous subsection. Finally, we reported TPR and FPR as the average over 20 runs. The other metrics, such as the running time, were also averaged over different realizations.

Since different joint estimation methods of G use different types of normalization of the regularization parameter λ, one can not merely compare their performance for the same fixed value of the parameter. However, *receiver operating characteristic (ROC) curve* allows judging the performance of the method without being influenced by choice of λ. We can obtain the ROC-curve by plotting the average TPR versus the average FPR evaluated for different parameter values. The larger is the *area under the curve (AUC)* – the better is performance. Thus "perfect" ROC-curve resembles the Γ letter – line going from 0 to 1 first vertically and then horizontally, meaning that there is zero probability of false positives and 100% probability to recover all the edges correctly. However, if the curve is close to diagonal, it means that method is almost like random guessing with a 50-50 chance of doing so right. If ROC-curve is below the diagonal, it indicates that the method's results need to be inverted – meaning, recovered edges are actually absence of such and estimated "blanks" are, in fact, edges.

In our experiments to construct the ROC curve, we used a uniform grid of values for λ on the logarithmic scale. The grid consists of 50 values of λ ranging from 0.01 to 1 unless specified otherwise.

Running time is reported as the elapsed time in seconds (unless specified differently) measured by the `system.time` function on the 4-core 3.6GHz processor and 16GB RAM computer.

5.1.3 Implementation vs the algorithm

Implementation of the algorithm in the package `jewel` version 0.1.0 has two minor differences from the pseudo-code of Section 4.3. The first one is that matrix **Active**, encoding the current active pairs of variables, is initialized not as an upper-diagonal matrix with all 1s, but rather as full $p \times p$ matrix. In the update step of the algorithm, not only element $Active_{ij}$ is updated, but also $Active_{ji}$. The further optimization of this implementation to the pseudo-code state will probably give a minimal additional advantage in running time.

The second difference is in the parameter λ – while in Chapter 4 λ was reparametrized as $\sqrt{2K}\lambda$, in the package implementation $\sqrt{2K}$ is a separate term. In this way, it is easier to extend the algorithm to a different number of variables among datasets, i.e., when g_{ij} would not be equal $2K$ anymore. However, as described in the previous subsection, since we judge the performance by the behavior of λ-independent ROC-curve, this difference does not influence any conclusions made about the method.

5.2 Why joint estimation?

In this section, we aim to answer two questions:

1. Why should we use several datasets, when in many applications, data collection is a hard, long, and expensive process? Does the use of K datasets improve the estimates? And how much?

2. Assuming that we could get ourselves several datasets, why should we use a joint estimation method when there are other more straightforward approaches like concatenation or voting? Again, is it worth it? What are the benefits?

To conduct the simulation to answer this questions, we use default parameters described in Subsection 5.1.1: $m = 1$, $power = 1$ and $tol = 0.01$. The rest is described below.

5.2.1 Advantage of using multiple datasets

The following experiment aims to show the gain of performance in estimating the graph when the number of datasets K increases. We simulated 10 datasets $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(10)}$ as described in Subsection 5.1.1 for two different dimensional cases: $p = 100$, $n_k = 50\ \forall k$ $(n/p = 1/2)$ and $p = 500$, $n_k = 100\ \forall k$ $(n/p = 1/5)$. We repeated the datasets generation 20 times. For each case, we first applied *jewel* to each of the 20 runs (containing $K = 10$ datasets at this step) and computed the average TPR and FPR. Then, we sampled $K = 5$ matrices in each run and repeated the procedure. We then sub-sampled $K = 3$ matrices out of the previous 5, then $K = 2$ out of 3 and, finally, $K = 1$ out of 2. In other words,

for each value of K, we applied *jewel* to 20 realizations and evaluated the average TPR and FPR.

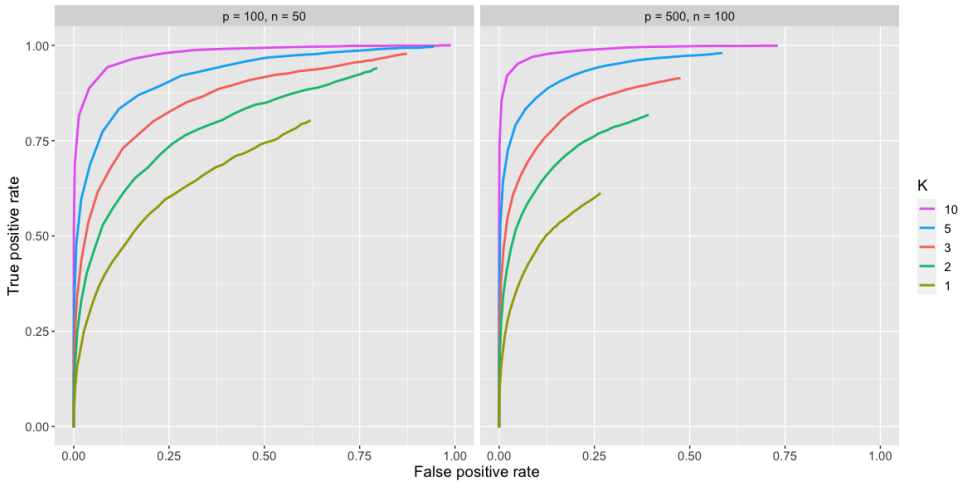


Figure 5.1: ROC-curve for *jewel* method applied to datasets of different K size (denoted by different colors). On the left is the performance for $p = 100, n_k = 50\ \forall k$, on the right – for $p = 500, n_k = 100\ \forall k$.

The average ROC-curve in Fig.5.1 illustrates the performance of *jewel* as a function of K. Results in Fig.5.1 not only answer our first question – yes, usage of multiple datasets improves the estimate – but also demonstrate the trend of improvement as K grows. Although it is expected since when K increases, the overall amount of available data grows, we stress that we *significantly* gain performance even with a limited increase in the number of datasets – see, for example, the improvement going from one dataset to two or three.

Moreover, even though we observe an increase in running time when we increase K, it is not so big, and one can sacrifice it for having a better estimate of the underlying graph. Table 5.1 reports the average running time for the case $p = 500, n_k = 100$ for some chosen values of parameter λ and on the entire grid of values. Although the increase going from $K = 1$ to $K = 3$ is almost double, the overall time is still reasonable. Besides, note that to explore the performance better, we used a λ grid uniform in log-scale, starting from quite a small value. Therefore, half of the values is between 0.01 and 0.1. Table 5.1 shows that the smaller values of λ constitute the heaviest part from the computational point of view since the thresholding operator does not allow to put to zeros many edges. Consequently, the **Active** matrix remains full, and the iterations cycle over almost the entire set of pairs i, j. When λ increases, the thresholding rule starts setting the elements to zero and removing pairs of variables from the **Active** matrix faster. **Active** matrix becomes sparser, the number of pairs to cycle at each iteration reduces and the running time decreases.

This analysis shows that using more than one dataset, even a small number, can improve the performance significantly. This observation is relevant for real data applications since the number of collected datasets is usually small due to the experimental costs.

	$K = 1$	$K = 2$	$K = 3$	$K = 5$	$K = 10$
$\lambda = 0.01$	≈3 min	≈5 min	≈7 min	≈11.5 min	≈26 min
$\lambda = 0.1$	24.8 sec	32.38 sec	41.16 sec	61.5 sec	≈2 min
$\lambda = 0.2$	17.38 sec	21.15 sec	26.28 sec	38.67 sec	80.55 sec
$\lambda = 0.52$	15.97 sec	20.81 sec	26.32 sec	38.56 sec	80.75 sec
$\lambda = 0.83$	16 sec	20.85 sec	26.3 sec	38.76 sec	80.88 sec
$\lambda = 1$	14.02 sec	17.75 sec	22.65 sec	31.09 sec	69.35 sec
grid of 50 λ	≈41.5 min	≈1 hour	≈1.5 hours	≈2.4 hours	≈5.4 hours

Table 5.1: *jewel* running time when applied to a different number of datasets K of size $p = 500$, $n_k = 100$. λ is chosen over the uniform in log-scale grid from 0.01 to 1.

5.2.2 Advantage of the joint approach

This section aims to show that joint analysis is preferable to naive alternatives such as concatenation or voting for the task of combining information from multiple sources. In order to prove that, we apply all three approaches to the same set of multiple datasets and compare the results.

Since we discussed concatenation and voting and their disadvantages in Section 3.4 of Chapter 3, here we only state the idea (see also Fig.5.2 for a schematic representation). *Concatenation* is the most straightforward way to analyze K datasets: combine them into one long matrix and then apply any method for estimation in $K = 1$ case (in our case, *jewel*, which, although targeted to the joint estimation, can also be applied just to one dataset). With *voting* each dataset is first processed independently, obtaining K estimates of the adjacency matrices. Then, we build a consensus matrix by setting an edge if it is present in at least $\lceil K/2 \rceil$ of the estimated matrices.

The simulation setting for this experiment is the following. We generated 20 independent runs each with $K = 3$ datasets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}$ as described in Subsection 5.1.1. We considered two dimensional scenarios, $p = 100$, $n_k = 50$ $\forall k$ and $p = 500$, $n_k = 100$ $\forall k$. For the concatenation approach, as a first step we constructed the long $Kn_k \times p$ matrix and then applied *jewel*. For the voting approach, we applied the method separately to each data matrix $\mathbf{X}^{(k)}$, $k = 1, \ldots, 3$, and then put an edge in the resulting adjacency matrix only if it was present in 2 out of 3 estimated adjacency matrices. Performance for each approach is represented Fig.5.3.

In Fig.5.3 we can observe a significant advantage in processing the datasets jointly to the other two approaches. Indeed, with the same amount of data, *jewel* correctly exploits the commonalities and the differences in K distributions providing a more accurate estimate of E. Instead, the concatenation approach ignores the distributional differences – it creates a single data matrix from not identically distributed datasets, and so it does not respect the standard math-

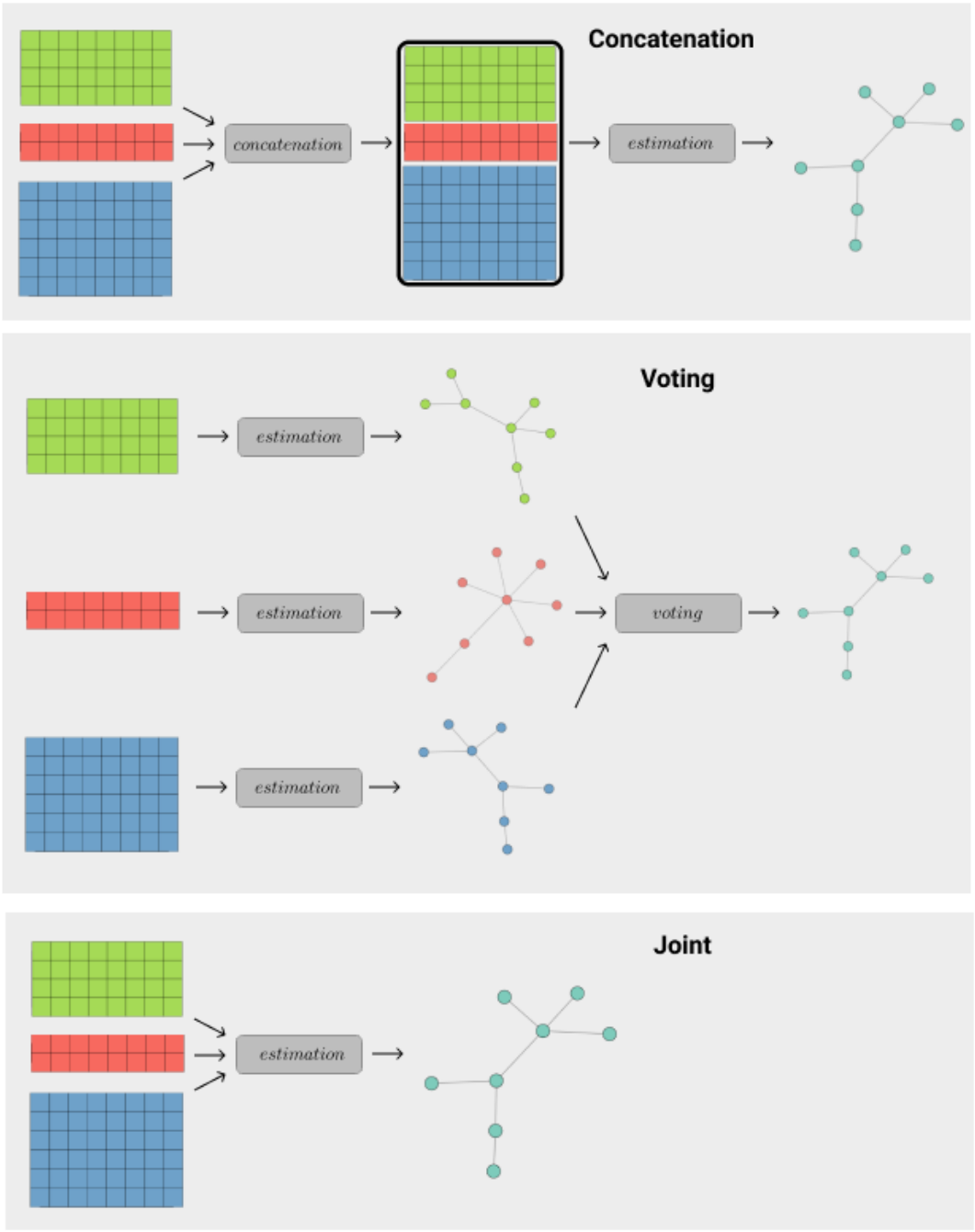


Figure 5.2: Different approaches that can be used for analysing the same set of multiple datasets: concatenation, voting and the joint approach.

ematical hypotheses of any estimators. This results in a significant loss of performance. The only advantage of the concatenation approach is the running time which is ≈ 40 minutes over the λ grid versus ≈ 1.5 hours for the joint approach. However, the loss in performance is so drastic that runtime improvement is hardly worth it.

On the other side, the voting approach does exploit the networks' common structure, but only during the post-processing of the estimator (voting). How-

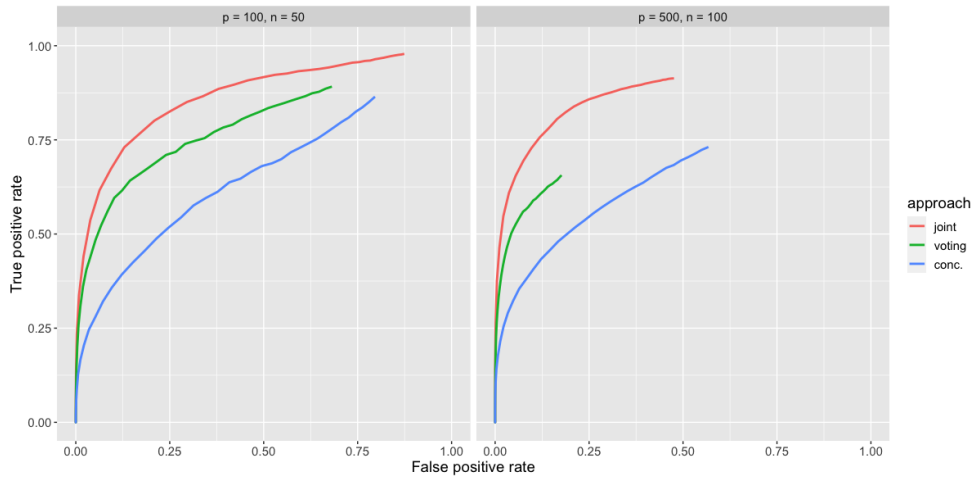


Figure 5.3: ROC-curve for different approaches of inferring the graph from $K = 3$ datasets: joint estimation, voting and concatenation (denoted in different colors). On the left is the performance in case of $p = 100, n_k = 50$, on the right – $p = 500, n_k = 100$.

ever, the inference for each dataset might be noisy due to the limited amount of data available. Therefore, its performance is also not as good as that of the joint approach. Moreover, since it requires K applications of the method, as a consequence, it is slower: ≈ 2 hours vs ≈ 1.5 hours for joint approach (although the individual estimates might be parallelized).

Overall, the simulation clearly shows that a joint estimation is the best practice in the multiple datasets framework compared to concatenation and voting approaches.

5.3 Performance as a function of various parameters

In this section, we will explore the performance of *jewel* as a function of various parameters of input data or the algorithm itself. We will start by exploring the behavior of *jewel* under different dimensional regimes and for different structures of the scale-free graph. We will then see how the simulated precision matrices' entries impact the performance (in short – not significantly). In the end, we will prove that the value of the parameter *tol* in the stopping criteria does not affect the results whatsoever. Therefore, in practical applications we can choose *tol* not too small to provide an additional runtime advantage.

5.3.1 Influence of dimension and graph structure

The simulation settings for this experiment are the following. We considered six different scale-free network scenarios for different values of parameter $m = 1, 2$

and parameter *power* $= 0.5, 1, 1.5$. The first one controls the sparsity, since the number of edges in the resulting graph is equal to $mp - (2m - 1)$. The second parameter *power* controls the hub structure in the graph: bigger *power* implies bigger hubs (and smaller amount of them). Fig. 5.4 shows a random realization of the six graphs when $p = 500$.

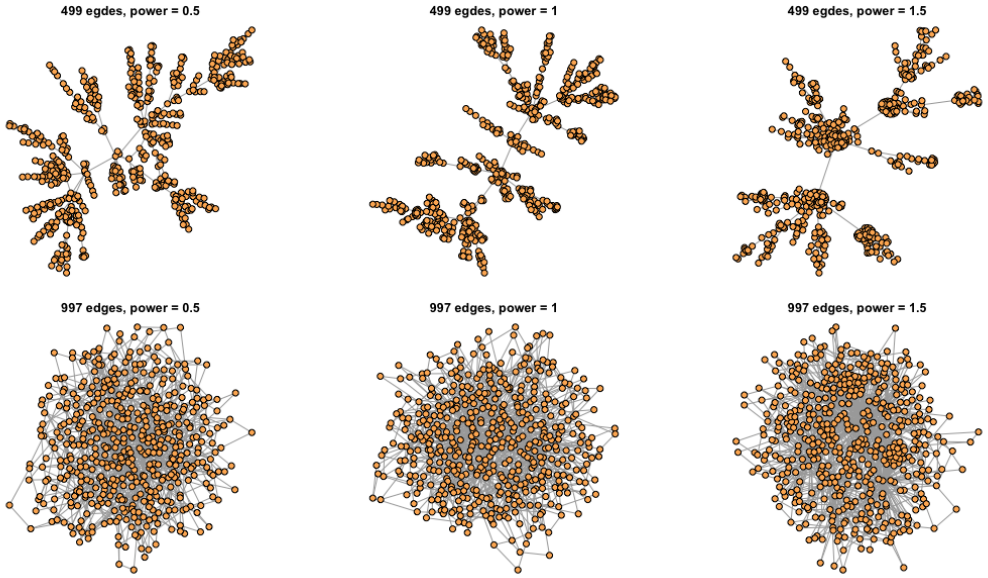


Figure 5.4: Scale-free graphs with $p = 500$ nodes generated with different values of parameters m (in rows) and *power* (in columns). The graphs correspond to one of the 20 random realizations generated in this simulation set-up.

Then, we simulated different dimensional regimes. We generated 20 independent runs as described in Subsection 5.1.1, each with $K = 3$ datasets $\mathbf{X}^{(k)}$, $k = 1 \dots K$. We considered three different values of number of variables $p = 100, 500, 1000$ and two different n_k/p proportions 1/5 and 1/2. Therefore, we got the following five dimensional scenarios: $p = 100$ with $n_k = 20$ and $n_k = 50$; $p = 500$ with $n_k = 100$ and $n_k = 250$; $p = 1000$ with $n_k = 200$ (in this case, simulation for $n_k/p = 1/2$ was not carried since $n_k = 500$ is not typically met in real applications).

Fig. 5.5 shows the resulting ROC-curves with *power* in columns, m in rows, color denoting different values of p and the type of line indicating the n_k/p proportion.

Let us first focus on the dimensional regime (i.e., the n_k/p ratio). We expected that the n_k/p proportion would influence the performance of the method, and although this expectation was confirmed, it turns out that the number of samples n_k plays a much more critical role. In fact, from Fig.5.5, we note that blue solid line, corresponding to $p = 1000, n_k = 200$, far outperforms the red solid one, corresponding to $p = 500, n_k = 100$. Although the n_k/p proportion is the same for both cases, the increase in the sample size leads to better performance. Indeed, the curve with the highest AUC value is the red dashed line, which corresponds to the case $p = 500, n_k = 250$ – case with highest sample size value. Meanwhile, the worst performance is achieved in the $p = 100, n_k = 20$ size setting

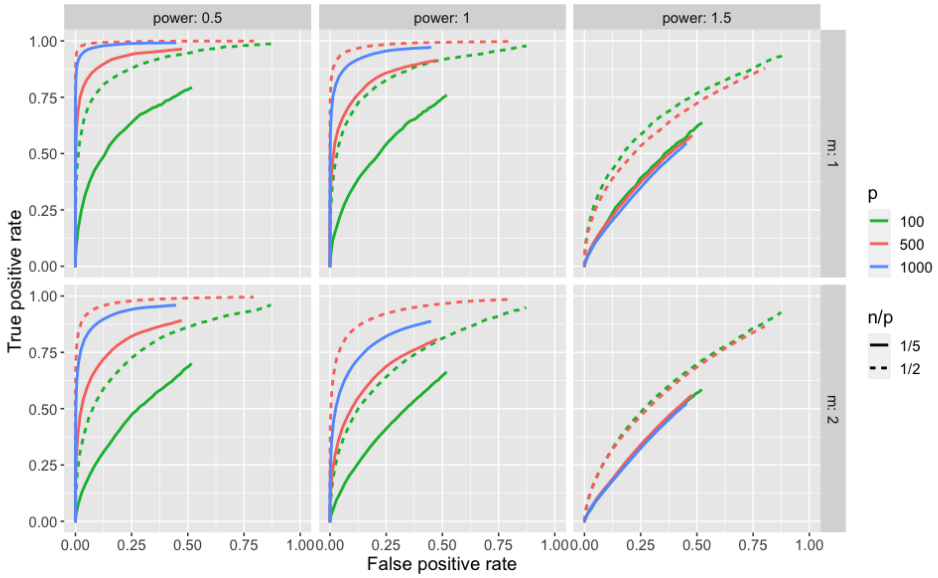


Figure 5.5: ROC-curves for *jewel* evaluated under different dimensional regimes and different structures of the graphs. *power*, controlling hubs, is in columns, m, controlling sparsity, is in rows. Colors denote different values of p, and the type of line stands for different n_k/p proportions.

(solid green line) where the number of samples is the smallest. Increase in the sample size ($p = 100, n_k = 50$, dashed green line) immediately provides gain in performance.

Now let us focus on the influence of $m-power$ parameters. We notice a slight decrease in performance while moving from $m = 1$ to $m = 2$, i.e., increasing the number of edges. This is not surprising since the fewer edges we have, the larger is our chance of identifying them correctly. However, the *jewel* performance is still adequate and in general comparable to the behavior of other methods for joint estimation (as we will see in the next Section 5.4).

Increase of *power*, on the other hand, causes quite drastic drop in performance. In fact, although performance is quite good for $power = 0.5$ and 1, it drops down significantly for $power = 1.5$. Again, in the next section, we will see that this drawback is also present in other joint estimation methods. Moreover, our results agree with the recent paper [62] for the case of one dataset that explores classical methods performance in the application to networks with big hubs and comes to the same discovery. Which is quite distressing since several authors estimate the power of real-world networks, that could be described with scale-free graph, to be in the interval $2 \leq power \leq 3$ which is, of course, even greater than $power = 1.5$. We will talk about the possible ways of overcoming this limitation in the last Chapter 7 but the essential idea is to add degree-induced weights to the penalty to account for hubs. Approach could be extended, for example, from the above mentioned [62] or new procedure can be suggested.

Let us now discuss *jewel* running time under different dimensional settings. In Table 5.2 we report the running time only for $m = 1$, $power = 1$ and $n_k/p = 1/5$ since we noticed that these parameters have little influence on it.

	$p = 100$	$p = 500$	$p = 1000$
$\lambda = 0.01$	20.98 sec	≈7 min	≈28 min
$\lambda = 0.1$	2.94 sec	41.16 sec	≈2.6 min
$\lambda = 0.2$	1.44 sec	26.28 sec	135.82 sec
$\lambda = 0.52$	0.7 sec	26.32 sec	135.84 sec
$\lambda = 0.83$	0.6 sec	26.3 sec	136.67 sec
$\lambda = 1$	0.57 sec	22.65 sec	114.92 sec
grid of 50 λ	≈4.8 min	≈1.5 hours	≈5.8 hours

Table 5.2: *jewel* running time for different dimensional settings: $p = 100$, $p = 500$, $p = 1000$. Other parameters are $n_k/p = 1/5$, $K = 3$, $m = 1$, $power = 1$. Uniform in log-scale grid of 50 parameters λ from 0.01 to 1 was used.

Although running time over the whole grid of parameters can become quite expensive in high dimensions, as discussed in the previous section, we stress that half of the λs are between 0.01 and 0.1 and running *jewel* with very small values of λ constitute the most computationally demanding part. As soon as λ increases (say about 0.1), the running time for fixed λ significantly reduces to just a couple of minutes even for $p = 1000$, and few seconds for smaller dimensions, clearly demonstrating the promising speed of *jewel*.

Moreover, we also note that the running time stabilizes after a certain value of λ. This effect of saturation is due to our initialization strategy where the **Active** matrix contains all 1s. Indeed, if we assume all the variables are "active", we have to perform at least one full cycle iteration over all the nodes. This problem can be ameliorated if we use warm-start (as we do, indeed, for BIC and CV) or consider another **Active** initialization. For example, if there is some prior knowledge on the underlying network structure, we could use such information in the initialization of **Active** matrix, eliminating some edges from the analysis and therefore providing more advantages in the running time.

Additionally, following [19, 53, 80], we can identify independent connectivity component structure of the input datasets and, if present, split the problem of estimating the graph G into several subproblems of smaller dimensions. Such reduction can provide great decrease in running time, not even mentioning possible parallelization. We discuss this idea more in depth in Chapter 7.

In the next subsection, we will explore the indirect influence of precision matrix entries on the methods performance.

5.3.2 Influence of precision matrices

In the data generation procedure, described in Subsection 5.1.1, we constructed K precision matrices $\mathbf{\Omega}^{(k)}$ with the same support as of adjacency matrix of the simulated graph, and with symmetric values sampled from the uniform

distribution with support on $[-b,-a]\cup[a,b]$. Then, we used $\mathbf{\Omega}^{(k)}$ to construct covariance matrices $\mathbf{\Sigma}^{(k)}$ and subsequently, to generate datasets $\mathbf{X}^{(k)}$. This data generation scheme is quite common in the literature. In some papers [19] the choice of the interval is $[-0.4,-0.1]\cup[0.1,0.4]$, while in others [36, 48, 80] it is $[-1,-0.5]\cup[0.5,1]$, and in some others [53] it is $[-0.6,-0.3]\cup[0.3,0.6]$ (see also Appendix A). However, in none of the articles the influence of parameters a and b has been studied, and we would like to fill this gap.

Therefore, in this subsection we will quantify the influence of the different choices of a, b values, i.e., the influence of different uniform distribution supports from which we sample entries of $\mathbf{\Omega}^{(k)}$ matrices. Since in the case of Gaussian distributions, the entries of the precision matrix are proportional to conditional correlation, we expect that bigger values of $\Omega_{ij}^{(k)}$ correspond to "stronger" connection between corresponding variables. And since the entries of $\Theta_{ij}^{(k)}$, estimated by *jewel*, are proportional to $\Omega_{ij}^{(k)}$ we expect that bigger are values $\Omega_{ij}^{(k)}$, easier it is to identify the connections between variables, as stated in the "signal strength" hypothesis of Theorem 3.

The simulation setting for this experiment is the following. We generated a true graph with $m = 1$, $power = 1$ for 20 independent runs. For each graph, we constructed 3 sets of $K = 3$ precision matrices, sampling from different uniform distributions: with support on $[-0.4,-0.1]\cup[0.1,0.4]$, $[-0.8,-0.2]\cup[0.2,0.8]$ and $[-0.9,-0.6]\cup[0.6,0.9]$. Then for each set, we proceeded as described in Subsection 5.1.1.

We consider one dimensional scenario $p = 500$, $n_k = 100\ \forall k$ for two methods – *jewel* and JGL [19]. Comparisons of these methods are set in the next section but here, instead, we focus on their similar behavior in different simulation settings to highlight that influence of a, b is not specific to *jewel* but is rather inherent to the class of maximum-likelihood or regression-based joint estimation methods. The performance of *jewel* and JGL is shown in Fig.5.6.

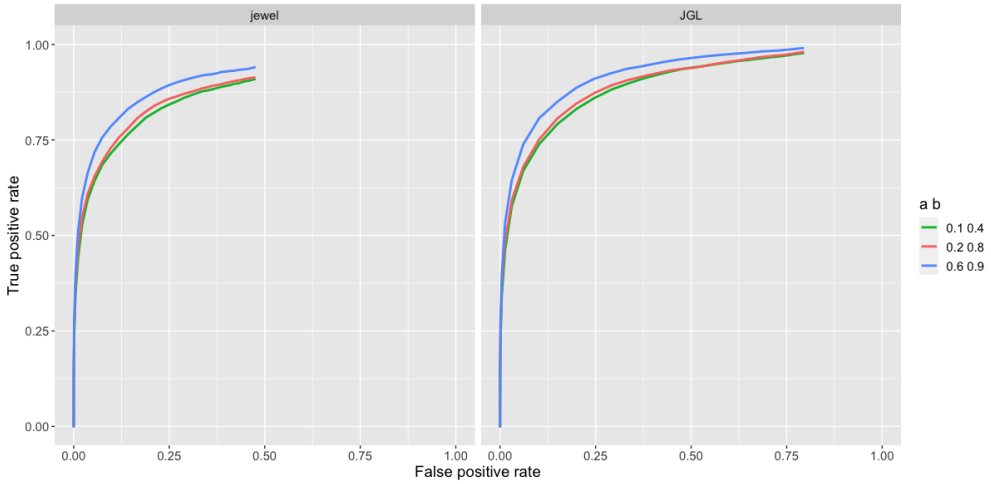


Figure 5.6: ROC-curve for *jewel* and JGL methods applied to simulation data with values of the prevision matrices $\mathbf{\Omega}^{(k)}$, sampled from uniform distributions with different supports $[-b,-a]\cup[a,b]$.

As we can see, our prognosis about the influence of a, b values was correct since the best performance for both methods is obtained for the case $(a, b) = (0.6, 0.9)$ (blue line in Fig.5.6), i.e. when connections are "the strongest". However, the red and green lines, referring respectively to the case $(a, b) = (0.1, 0.4)$ and $(a, b) = (0.2, 0.8)$ are not much worse compared to the blue one, showing that the strength of the signal affects the performance in a subtle way, without changing the overall tendency. The performance difference is certainly not as drastic as we have seen for other data parameters like $p, n_k, m, power$. Therefore, we do not focus on it anymore in our simulations study and from now on we fix parameters $(a, b) = (0.2, 0.8)$ sampling from a uniform distribution over $[-0.8, -0.2] \cup [0.2, 0.8]$ in order to test the performance of *jewel* and other joint estimation methods for connections of any strength.

Now instead of exploring the indirect effect of different data generation parameters through data matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$, we will investigate the choice of *jewel* algorithm parameter in the following subsection.

5.3.3 Influence of the stopping criteria

jewel algorithm is iterative, i.e., we repeat the procedure until the convergence is achieved. Convergence is based on the following stopping criteria

$$\sum_k \left|\boldsymbol{\Theta}^{(k,\,t+1)} - \boldsymbol{\Theta}^{(k,\,t)}\right| / \sum_k \left|\boldsymbol{\Theta}^{(k,\,t)}\right| < tol,$$

where t denotes the iteration counter. In this section, we investigate the influence of the choice of *tol* on the performance and running time of *jewel*.

To explore this point, we generated 20 independent runs as described in Subsection 5.1.1 with all parameters set to their default values studied up to now. We considered two dimensional scenarios: $p = 100$, $n_k = 50\ \forall k$ and $p = 500$, $n_k = 100\ \forall k$. In both cases, we applied *jewel* to each of the 20 runs with different values of threshold $tol = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$ and computed the average TPR and FPR, as well as the running time.

As we can see from Fig.5.7, under both dimensional regimes, the resulting ROC-curves, constructed for different values of *tol* parameter, are indistinguishable. We can conclude that *tol* does not influence the performance of *jewel* – which is not surprising. This is due to the fact that we aim to reconstruct only the support of matrices of regression coefficients $\boldsymbol{\Theta}^{(k)}$, $k = 1 \ldots K$, (which is equivalent to reconstructing the support of precision matrices $\boldsymbol{\Omega}^{(k)}$) and therefore, the actual values of matrix entries, which are controlled by *tol* parameter, are not crucial for the estimation.

The change in running time with the decrease of parameter *tol*, however, is quite significant: it can be of several times for small values of the regularization parameter λ and on average on the whole grid, as we can see from Table 5.3. However, this observation is not valid for bigger values of λ because the number

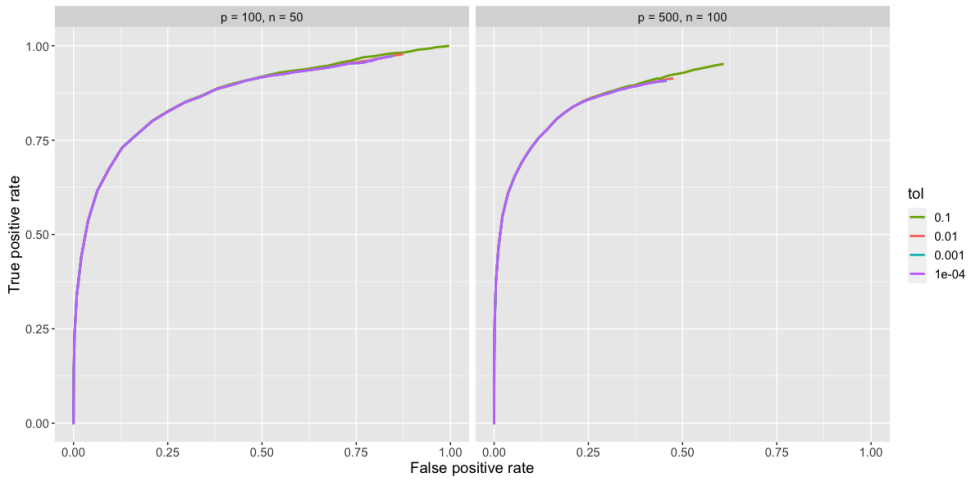


Figure 5.7: ROC-curve for *jewel* applied to data with different value of the threshold of stopping criteria (denoted by different colors) in two different dimensional settings: $p = 100, n_k = 50$ (on the left) and $p = 500, n_k = 100$ (on the right).

	$tol = 10^{-1}$	$tol = 10^{-2}$	$tol = 10^{-3}$	$tol = 10^{-4}$
$\lambda = 0.01$	≈3.2 min	≈7 min	≈12.5 min	≈20 min
$\lambda = 0.1$	34.52 sec	41.16 sec	47.06 sec	53.7 sec
$\lambda = 0.2$	26.03 sec	26.28 sec	26.05 sec	26.16 sec
$\lambda = 0.52$	26.27 sec	26.32 sec	26.04 sec	26.09 sec
$\lambda = 0.83$	26.25 sec	26.3 sec	25.08 sec	23.51 sec
$\lambda = 1$	22.96 sec	22.65 sec	20.41 sec	20.19 sec
50 λ	≈56 min	≈1.5 hours	≈2.4 hours	≈3.5 hours

Table 5.3: *jewel* running time for $K = 3$, $p = 500$, $n_k = 100$ with different value of the stopping criteria threshold. Uniform in log-scale grid of 50 parameters λ from 0.01 to 1 was used.

of iterations naturally reduces. Hence, the parameter *tol* has much less influence on the running time when λ is large.

All these considerations allow us to suggest a reasonably large value of *tol* in practical applications which we set as $tol = 0.01$ by default. This choice ensures great saving in terms of running time without sacrificing the effectiveness and the performance of *jewel*.

Having clarified the influence of the different parameters involved in our method, we can now move on to the next section where *jewel* will be compared with its recent competitors.

5.4 Comparison with other methods for joint estimation

In this section, we compare the performance and runtime of *jewel* with two other methods for joint estimation: joint graphical lasso (JGL) with group penalty [19] and the proposal of Guo et. al [36] (see Section 3.4 of Chapter 3 for details). These two methods were chosen based on code availability (see appendix C for discussion) and their simplicity of application in adapting data structures and/or parameters. The comparison with other joint estimation methods is in progress.

5.4.1 Comparison set-ups

The joint graphical lasso method is implemented in the R package `JGL` [18]. Function `JGL` allows to choose between fused and group penalty, and specify two tuning parameters λ_1 and λ_2. We choose the group lasso penalty being it of our interest due to its similarity with *jewel* penalty. We also set parameter $\lambda_1 = 0$ since it is responsible for differences in the supports of $\hat{\mathbf{\Omega}}^{(k)}$, $k = 1 \dots K$, and under our hypothesis, the patterns of non-zero elements are identical across classes. Therefore, we vary only λ_2.

The code for the implementation of Guo et al. proposal [36] was provided by authors upon my request and I would like to thank them for sharing their code. The estimation function requires only one regularization parameter λ. Since the output of Guo et al. proposal is the set of matrices $\hat{\mathbf{\Omega}}^{(k)}$, $k = 1 \dots K$, we use the OR rule and consider the union of their supports as adjacency matrix of the graph of interest.

For the sake of brevity, in this section, we will discuss only the dimensional setting $K = 3$, $p = 500$, $n_k = 100\ \forall k$ ($n_k/p = 1/5$). The other settings show analogous results. For each method, we use their default parameters, in particular we fix $tol = 0.01$ for both *jewel* and Guo et al. method, while we fix $tol = 10^{-4}$ for JGL method.

5.4.2 Performance for different graphs structures

This section explores results for different values of parameters m and $power$, which control the sparsity of the true graph and its hub structure. The data generation workflow is the same as the one described in Subsection 5.1.1 and the choice of the true network is the same as the ones considered in Subsection 5.3.1. Specifically, we again consider six different $m - power$ scenarios, varying parameter $power = 0.5, 1, 1.5$ and parameter $m = 1, 2$. The first one controls the hub structure of the graph – bigger $power$ implies bigger hubs and smaller amount of them. The second parameter m controls the sparsity, since the number

of edges in the resulting graph is $mp - (2m - 1)$. Therefore, for $p = 500$ we get 499 edges in case $m = 1$ (about 0.4% of all possible edges) and 997 edges in case $m = 2$ (sparsity of about 0.8%). See Fig.5.4 in Subsection 5.3.1 for the illustration of the resulting networks.

In each $m - power$ scenario we simulated 20 independent runs for $K = 3$, $p = 500$ and $n_k = 100$ and then applied *jewel*, JGL and Guo et al. methods to each generated data set, evaluating the mean performance and running time.

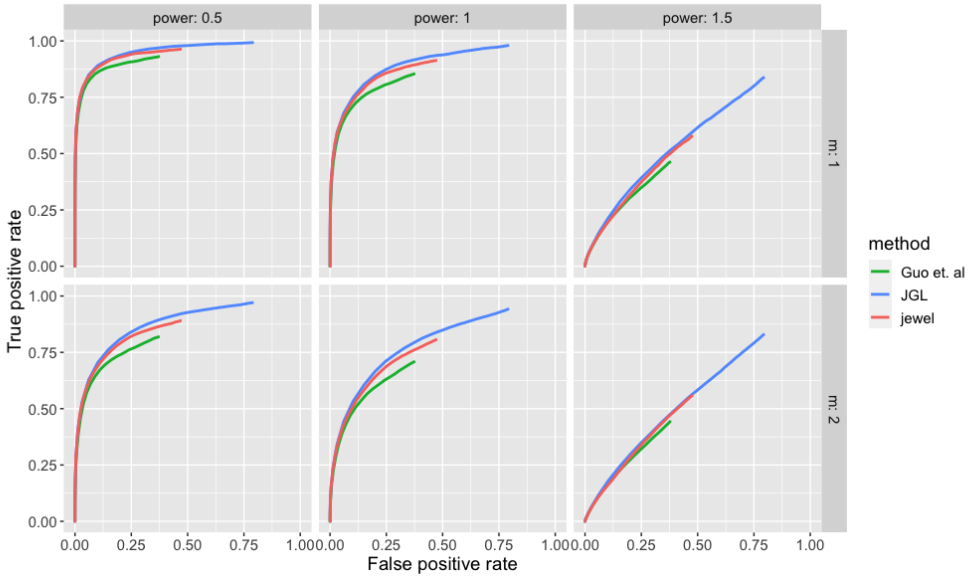


Figure 5.8: ROC-curve for different joint estimation methods: *jewel*, JGL [19] and Guo et. al proposal [36] for $K = 3$, $p = 500, n_k = 100\ \forall k$. Each panel demonstrates performance in different $m - power$ setting.

In Fig.5.8 we can observe that performance of *jewel* in all scenarios is comparable to JGL, and performance of both of these methods is superior to the proposal of Guo et. al. This observation remains valid even in the worst-case scenario, i.e., with $power = 1.5$. Overall, Fig.5.8 illustrates the good performance of this class of methods in the sparse regime, despite an increase of m decreasing the performance (although without destroying it). Let us stress that increasing *power*, i.e., increasing hubs size, leads to a significant loss of performance for all the methods. This behavior was already discussed in Subsection 5.3.1 for *jewel*, and there it was anticipated that this is typical for other studied joint estimation methods. This experiment further confirms our previous observation and matches the results obtained in [62] which explain the inadequate capacity of these methods in the case of large hubs size.

We now analyze the running time results. As before, in Table 5.4, we report the results for some specific values of the parameter λ and for the entire grid of λ. As we can see, *jewel* is approximately two times faster than JGL and several times faster than Guo et al. for small values of parameter λ, while this does not hold for the bigger value of parameter λ where *jewel* has to pay the price of the full initialization matrix **Active**. However, as already observed in Subsection

5.2.2, it is unlikely to use a large value of λ in real data applications since it causes a larger number of false negatives. In practical situations, the values of λ of interests are in the range for which *jewel* is faster than both its competitors. This observation will be confirmed in Section 5.5 where the choice of the optimal λ will be studied.

	jewel	JGL	Guo et. al
$\lambda = 0.01$	≈7 min	≈11.5 min	≈66 min
$\lambda = 0.1$	41.16 sec	80.16 sec	≈2.6 min
$\lambda = 0.2$	26.28 sec	73.76 sec	73.68 sec
$\lambda = 0.52$	26.32 sec	0.31 sec	22.91 sec
$\lambda = 0.83$	26.3 sec	0.099 sec	12.08 sec
$\lambda = 1$	22.65 sec	0.099 sec	10.88 sec
grid of 50 λ	≈1.5 hours	≈3.4 hours	≈8.4 hours

Table 5.4: Running time for different joint estimation methods: *jewel*, JGL [19] and Guo et al. proposal [36] for $K = 3$, $p = 500$, $n_k = 100$, $m = 1$ and $power = 1$ over the uniform in log-scale grid of 50 parameters λ from 0.01 to 1.

Finally, we stress that although Table 5.4 reports results only for the case $power = 1$ and $m = 1$, our conclusions are valid for all combinations of parameters *power* and m, which are omitted only for the sake of brevity, since they do not influence the running time.

To summarize this subsection, we can assert that *jewel* demonstrated performance comparable to JGL and superior to Guo et al. proposal while showing a significant advantage in terms of running time in respect to both methods.

5.5 Regularization parameter selection

In this final section, we discuss results obtained using two procedures for choosing the regularization parameter λ, i.e., Bayesian information criterion (BIC) and cross-validation (CV). We gave the theoretical background of these two methods in Section 4.4 of Chapter 4. The experiment is carried out only for *jewel* since Guo et al. [36] and JGL [19] do not provide the code for procedures of λ estimation from the data. although authors propose the procedures in their manuscripts. In some articles, however, the authors suggest using a user-specific choice of the parameter λ, which is questionable. Our BIC and CV estimations provide an automatic choice guided by the data.

Both BIC and CV procedures are implemented in the `jewel` package. Note that for CV we use 5 folds by default and implement parallelization on an 4-core machine, almost eliminating the running time disadvantage of CV to BIC.

The simulation settings are the following. For $p = 500$, $n_k = 100$, $K = 3$ we generated 20 independent runs as described in Subsection 5.1.1 with default values of $m = 1$ and $power = 1$. We used the uniform in log-scale grid of 50 parameters λ_l, $l = 1 \dots L$, from 0.1 to 1. We note that in all the previous simulations, we started from $\lambda_1 = 0.01$ to obtain a left complete ROC-curve. However, since it is unlikely that λ is chosen extremely small by BIC or CV, here we fix $\lambda_1 = 0.1$ to save running time – but verify that the minimum of the target function was reached inside the chosen interval. We stress that this choice would not be justified if we observed the minimum of BIC or CV at the end of the interval and therefore suspected that the global one could be outside of the observed range. In that case, we would change our choice of the grid, allowing smaller values of λ.

On the other hand, we increase the running time by setting the stopping criterion threshold $tol = 10^{-4}$ instead of the default value $tol = 10^{-2}$. This choice provides higher accuracy for the estimation of regression coefficients $\hat{\mathbf{\Theta}}^{(k)}$ and hence of the residuals $\mathbf{R}^{(k)}$, $k = 1 \dots K$, which are required both for evaluation BIC and CV.

However, the most significant difference is made by using the warm start as far as regards the running time. In the warm start approach, one uses $\left\{\hat{\mathbf{\Theta}}^{(k)}(\lambda_l)\right\}_{k=1\dots K}$ estimated with λ_l as initialization for solving the minimization problem with λ_{l+1}. This trick leads to a decrease of the active pairs in the **Active** matrix. and therefore saves computational cost. Hence, we first investigated if the computational gain is payed in terms of criteria performance. To this aim, we applied BIC and CV criteria with and without the warm start approach and then compared the results. Specifically, in Fig.5.9 we plot values of Bayesian information criterion and cross-validation error for each λ_l value for one realization of data randomly chosen out of twenty independent runs. In Table 5.5 instead, we report results averaged over all 20 runs. In the following, we briefly describe how these average values were obtained. For each run, we first estimated λ_{BIC} and λ_{CV} by the two criteria, then ran `jewel` with these values and evaluated performance in terms of accuracy, precision, recall and running time. These performance measures are defined as

$$\begin{aligned} accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \end{aligned}$$

with TP, TN, FP and FN given in Section 5.1.2.

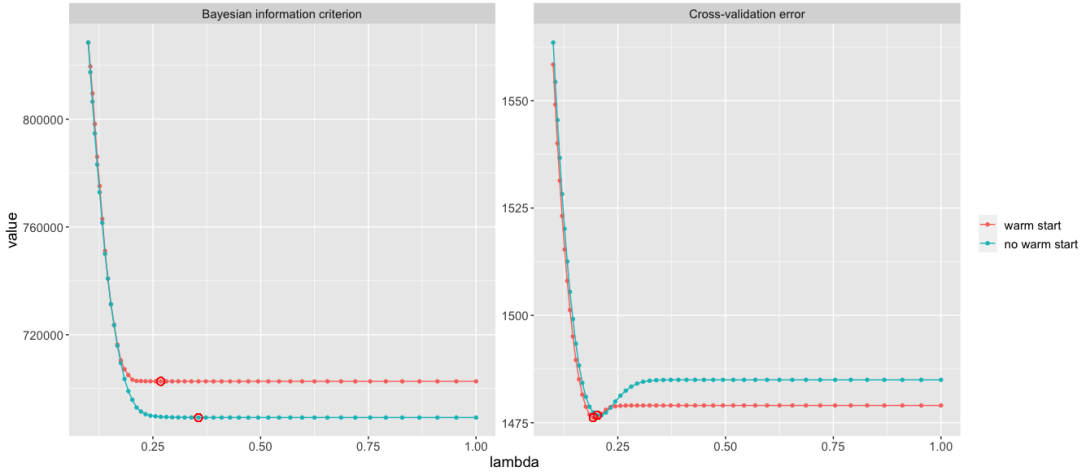


Figure 5.9: Values of BIC error (on the left) and CV error (on the right) obtained with (cyan) and without (red) warm start for *jewel*. Results are obtained for one randomly chosen realization with $K = 3$, $p = 500$, $n_k = 100$, $m = 1$, $power = 1$ over the uniform in log-scale grid of 50 parameters of λ from 0.1 to 1. Red crosses denote the estimated optimal λ_{OPT}.

As we can see from Fig.5.9, the warm start approach changes the curve for both criteria (which is expected), but while for CV it has almost no influence on the final result (with warm start $\lambda_{CV} = 0.193$ and without $\lambda_{CV} = 0.2$), for BIC it makes an important difference since we obtain $\lambda_{BIC} = 0.268$ vs $\lambda_{BIC} = 0.355$. Interestingly, with and without warm start approach, the relative difference $BIC(\lambda_{prev}) - BIC(\lambda_{BIC})/BIC(\lambda_{BIC})$ is incredibly small, of the order of 10^{-5}.

	λ_{OPT}	accuracy	precision	recall	runtime
BIC with w.s.	0.268	0.996	0.548	0.134	≈3.6 min
BIC without w.s.	0.355	0.996	1	0.002	≈3.75 hours
CV with w.s.	0.193	0.992	0.229	0.382	≈3.5 min
CV witouht w.s.	0.2	0.994	0.275	0.348	≈2.6 hours

Table 5.5: Results of Bayesian information criterion and cross-validation procedures with and without warm start for *jewel* for $K = 3$, $p = 500$, $n_k = 100$, $m = 1$, $power = 1$ over the uniform in log-scale grid of 50 parameters of λ from 0.1 to 1. λ_{OPT} is reported on average over 20 runs, performance metrics and runtime were evaluated with estimated λ_{OPT} for each run and then averaged.

Although this inconsistency may lead us to opt out of using the warm start, Table 5.5 convinces us otherwise. Firstly, we note that BIC without the warm start estimates regularization parameter λ_{BIC} too big, "killing" almost all possible edges. Secondly, we note that BIC without the warm start requires significantly larger running time, going from some minutes to several hours (same for CV). These two reasons make the warm star approach very convenient for BIC and CV criteria.

Once we have established that both BIC and CV are preferred when applied with the warm start, let us argue more on the results presented in Table 5.5. We first note that on average, both estimates λ_{BIC} and λ_{CV} are quite close to each other. Our second observation is that they are both very close to $\lambda = \sqrt{\log p/n} = \sqrt{\log 500/100} = 0.249$ which some authors suggest as an estimate of the optimal regularization parameter. Moreover, we observe that *jewel* applied with both λ_{BIC} and λ_{CV} achieves very high accuracy, but relatively low precision and recall. This observation indicates that the chosen regularization parameter is too large, and hence we eliminate too many correct edges during the estimation process. Indeed, from Fig.5.8 the ROC-curve for this setting ($m = 1$ and $power = 1$) shows the optimal λ probably between 0.1 and 0.2. Therefore, we conclude that although the implementation of BIC and CV is fundamental for choosing the regularization parameter λ in the absence of any other information, it must be further explored, as well as with the development of alternative data-driven selection criteria.

Since numerical experimentation of *jewel* has been an important part of my thesis, I would like to list in the following the main results of this chapter:

- more datasets (bigger K) $\Rightarrow$ better performance with adequate runtime;
- joint estimation is better that concatenation and voting;
- less sparse graph (bigger m) $\Rightarrow$ slightly worse performance, but still satisfactory;
- more hubs (bigger *power*) $\Rightarrow$ worse performance, an adjustment of the penalty is required;
- the performance is good for any value of p and any n_k/p ratio as long as n_k is sufficiently large;
- "stronger" connections between variables (bigger a, b) $\Rightarrow$ slightly better performance;
- *tol* has no significant influence on performance, but it has great influence on the runnning time for small λ;
- *jewel* performs better than Guo et al. method [36] and as good as JGL [19];
- *jewel* is faster than Guo et al. method and JGL for most of the range of λ of interest;
- BIC and CV provide a "good enough" estimate of λ_{OPT} to use on real data but need further study.

Now we can move to Chapter 6, where we discuss the real data application of *jewel*.

Chapter 6

Application to real data

In this chapter, I will discuss the application of *jewel* to two case studies arising in genomics applications. I will start with a simple "toy" example of the dataset containing gene expression microarray data for four breast cancer subtypes in which I evaluate a subset of genes belonging to one given functional pathway. Then, I will consider glioblastoma microarray gene expression data obtained in three different studies and will limit my analysis to seven functional pathways.

6.1 TCGA breast cancer dataset

In this section, we considered the application of *jewel* to a small "toy" real data example consisting of a subset of genes taken from a much larger study on breast cancer.

Breast cancer is the most common type of cancer among women. It is a heterogeneous disease meaning that the clinical and genomic characterizations of different subtypes vary. We will consider four different molecular subtypes: luminal A, luminal B, basal-like, and HER2-enriched. Each subtype can have specific regulatory mechanisms but some, however, are common to all subtypes. We aim to reveal such common relationships among genes with this analysis.

We considered the breast cancer gene expression microarray dataset available in The Cancer Genome Atlas and described in [44]. Gene expressions were measured using Agilent G450 microarrays. The original dataset contains 526 breast cancer samples on 17327 genes among which there are $n_1 = 231$ samples of luminal A cancers, $n_2 = 127$ luminal B cancers, $n_3 = 95$ basal-like cancers and $n_4 = 58$ HER2-enriched cancers. These $K = 4$ subtypes were identified in the original paper of [44]. For simplicity of use, we limit our analysis to the preprocessed subset of the original dataset, which is present in the `NETI2` R package developed in [65] (i.e., the `TCGA.BRCA$X` object). In [65], the authors focused only on $p = 139$ genes common to the gene expression datasets and the breast cancer pathway (hsa05224) from the Kyoto Encyclopedia of Genes and Genomes database [42, 41]. Note that the same subset of genes (with some additional information) is also present in the `JEGN` R package [73] by the same author, developed in [74].

Given this data, we estimated the regularization parameter with BIC and cross-validation procedures, described in Chapters 4 and 5, used resulting parameters to infer a network, and then compared it with the one obtained from STRING database (see below).

First, we used BIC (with the warm start) to estimate the optimal value of the regularization parameter λ. To this aim, we used a uniform in the log-scale grid of 50 λ from 0.01 to 1[1]. The procedure revealed $\lambda_{BIC} = 0.20236$ (see Fig.6.1) and we ran *jewel* with this value of the parameter. The resulting graph had 466 edges (about 4.9% of all possible edges). 137 out of 139 vertices were connected to at least one other node.

We then carried the estimation with a cross-validation procedure and obtained $\lambda_{CV} = 0.07197$ (see Fig.6.1). In the network, estimated by *jewel* with this value of regularization parameter, all 139 nodes were connected to at least one other node, and 3541 edges were present. For this dataset, we note that BIC and CV procedures suggested values λ_{BIC} and λ_{CV} that are significantly different. Thus, they lead to the different degrees of sparsity on the estimated graph. Since CV estimated a small value of the regularization parameter, the number of resulting edges is about 37% of all possible edges. This fact suggests that cross-validation might underestimate the regularization parameter for this dataset.

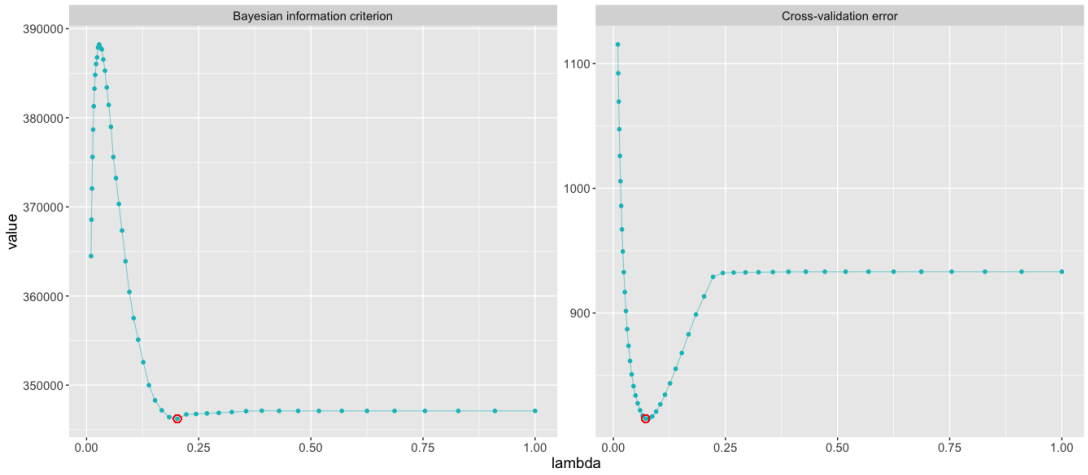


Figure 6.1: Values of Bayesian information criterion (on the left) and cross-validation error (on the right) obtained for breast cancer dataset with $K = 4$, $p = 139$ and $n_1 = 231$, $n_2 = 127$, $n_3 = 95$, $n_4 = 58$ over the uniform in log-scale grid of 50 parameters of λ from 0.01 to 1. Red circles denote the estimated optimal λ_{OPT}.

Note that by construction of the *jewel* algorithm with the active-shooting approach, estimators obtained with different values of regularization parameter λ are supposed to be "nested". Meaning, that if we estimate a graph with some λ_m, then using the value $\lambda_l > \lambda_m$, we will obtain more sparse solution,

[1]As we note in the Subsection 5.5, choice of the grid should be data-driven. Starting from $\lambda_1 = 0.1$ we were not able to observe a global minimum of CV error, therefore, we chose lower starting point $\lambda_1 = 0.01$ for both procedures.

eliminating some edges from $\hat{G}(\lambda_m)$ but not adding any new ones. In fact, this is exactly what we observe as $\hat{G}(\lambda_{BIC}) \subset \hat{G}(\lambda_{CV})$ since $\lambda_{BIC} > \lambda_{CV}$.

The next step of the analysis was to process $p = 139$ genes of the dataset in the STRING database [38]. It is a database of known and predicted protein-protein interactions that can be physical and functional and can be derived from lab experiments, known co-expression and genomic context predictions, as well as knowledge in the databases and even text-mining. It currently contains information on 24584628 proteins from 5090 organisms.

The database is relatively straightforward to use – we imported the list of 139 genes in the "multiple proteins" form and immediately obtained the network. However, we modified the default parameters to limit the retrieval to connections only from "experiments" and "databases" as active interaction sources with the minimum required interaction score set to the highest value of 0.9. The resulting network had 130 out of 139 vertices connected to any other node and 1204 edges. Note that, if we used only "experiments", then even with the medium required interaction score of 0.4, the resulting network would be much more sparse with 119 connected nodes and 337 edges.

The next step was to construct an intersection of the estimated networks and the one provided by STRING database (see Fig.6.2). For the network estimated with λ_{BIC} it revealed 82 edges in common; while for the one obtained with λ_{CV}, there were 434 common edges (see all the results summarized in Table 6.1).

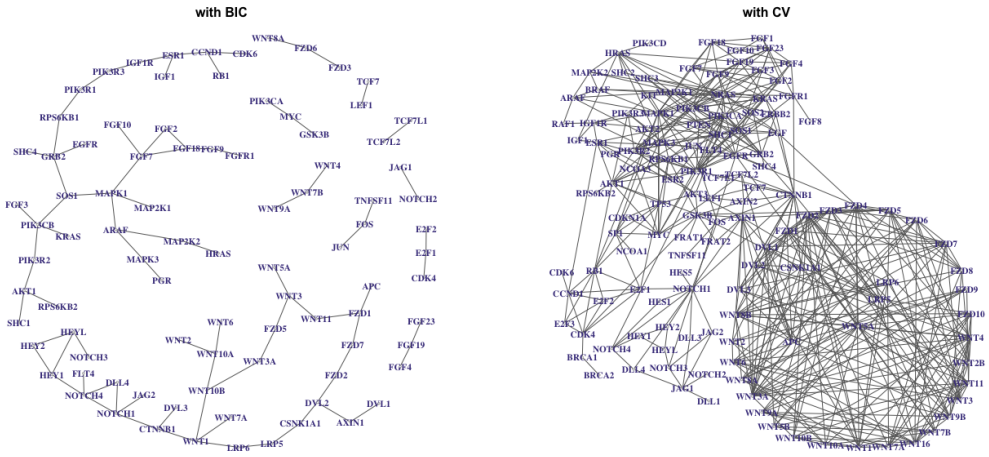


Figure 6.2: Intersection of networks estimated with *jewel* from breast cancer dataset and the one obtained from the STRING database. Regularization parameter, used in the estimation, was obtained with BIC (on the left) and with CV (on the right).

At first sight, such values could be considered relatively low and discouraging. However, we should first note that *jewel* seeks to identify conditional correlation among variables, or equivalently linear relationships between two genes not mediated by other factors (i.e., other genes). Meanwhile, connections from STRING database are not necessarily of such nature. Moreover, since we restricted the analysis to a specific pathway, we eliminated the influence of many

	λ_{OPT}	#conn. nodes	#est. edges	#edges in intersection
BIC	0.20236	137	466/9591	82/1204
CV	0.07197	139	3541/9591	434/1204

Table 6.1: Results of BIC and CV procedures obtained for $K = 4$ breast cancer datasets over the uniform in log-scale grid of 50 parameters of λ from 0.01 to 1.

possible variables that otherwise could have been important in the conditional dependence estimation. We are hoping to overcome this limitation by conducting another analysis in a higher dimension. Secondly, STRING contains general protein-protein interactions, i.e., interactions that are not necessarily present in the tissue/condition studied in [44]. Therefore, we would not be able to identify them. We can use databases with tissue-specific information for further biological validation, such as, for example, HumanBase [35].

Another reason could be that the four chosen subtypes of breast cancer are more different than expected. Hence, the assumption that the underlying conditional dependence network remains the same is violated. Consequently, since *jewel* does not account for class-specific edges, we lose in performance. Indeed, suppose we limit our analysis only to $K = 2$ breast cancer subtypes, i.e., luminal A and luminal B, which are more similar to each other. In that case, we observe in Table 6.2 that the estimated graph identifies many more edges in common to STRING database. Although the overlap with STRING database is not necessarily a sign of goodness, this network might better capture the similarity across the two cancer subtypes. However, again we observe that CV provides an estimate of λ that does not lead to a sparse network.

	λ_{OPT}	#conn. nodes	#est. edges	#edges in intersection
BIC	0.13895	139	912/9591	141/1204
CV	0.05964	139	4272/9591	528/1204

Table 6.2: Results of BIC and CV procedures obtained for $K = 2$ breast cancer datasets over the uniform in log-scale grid of 50 parameters of λ from 0.01 to 1.

Overall, our analysis shows that when comparing disease in different stages or subtypes, it might be interesting to estimate both the common relationships among the networks and the differences between cases. To this purpose, we can modify the *jewel* method so that it would be able to identify the class-specific edges as we will discuss among future work possibilities in the next chapter.

Finally, although we did not perform a specific analysis, from Fig.6.2 we can observe that there are some heavy hubs present in the network and in Chapter 5 we already faced this limitation of *jewel* and other joint estimation methods. Again, we will further discuss this direction of possible method improvement in the next chapter.

Let us make some brief observations about estimated networks. In networks

estimated both with λ_{BIC} and λ_{CV} many groups of genes are identified consistently, like WNT family members, frizzled class receptors, fibroblast growth factors and NOTCH receptors. In the λ_{CV} networks nodes with the highest degrees corresponded to genes encoding GRB2, WNT5A, NRAS and HES1 proteins. GRB2 is often overexpressed in breast cancer and NRAS is implicated in the development of several types of cancers while WNT5A and HES1 are considered by some authors as therapeutic target in breast cancer.

6.2 GEO glioblastoma datasets

This section shows the application of *jewel* to gene expression datasets of patients with glioblastoma, which is the most common type of malignant brain tumor. We used three microarray datasets from Gene Expression Omnibus (GEO) database [7]: GSE22866 [2] (obtained with Agilent Microarray), GSE4290 [63] and GSE7696 [45, 54] (both obtained with Affymetrix Array). Since the data comes from patients with the same type of disease, our assumption about the same underlying graph should hold better than in the previous example and therefore, the analysis should lead to more adequate results.

We annotated the probes (samples) in the gene expression matrices using `biomaRt` R package [24]. To do so, we matched the probe name and the Ensembl gene id from the `biomaRt` annotation. In case of multiple matches between probes and Ensembl gene ids, we gave preference to genes which were in common among all datasets or, in case of further uncertainty, to those present in selected pathways (see below). Afterwards, we converted Ensembl gene ids to gene symbols, and then averaged gene expression over the probes, obtaining $K = 3$ matrices with dimensions 40×20861, 77×16801, 80×16804, respectively. For the sake of simplicity, we considered only the $p = 13323$ genes in common to all three datasets.

For this illustrative analysis, we limited the attention to the genes belonging to seven pathways from the Kyoto Encyclopedia of Genes and Genomes database [42, 41] which were associated with cancer:

- p53 signaling pathway (hsa04115);
- glutamatergic synapse (hsa04724);
- chemokine signaling pathway (hsa04062);
- PI3K-Akt signaling pathway (hsa04151);
- glioma pathway (hsa05214);
- mTOR signaling pathway (hsa04150);
- cytokine-cytokine receptor interaction (hsa04060).

These pathways involve 920 genes in total; out of them, $p = 483$ were present in our datasets. Hence, we applied *jewel* to this subset of genes. As in the previous section, we first selected the regularization parameter λ with both BIC and CV, then estimated the networks with these parameters and compared them with a network obtained from STRING database.

First, when we estimated optimal λ by BIC procedure (with warm start) and obtained the value $\lambda_{BIC} = 0.2223$ (see Fig.6.3). The estimated graph G_{BIC}, which is the solution of *jewel* corresponding to this parameter, has 3113 edges (about 2.7% of all possible edges) and no isolated vertices.

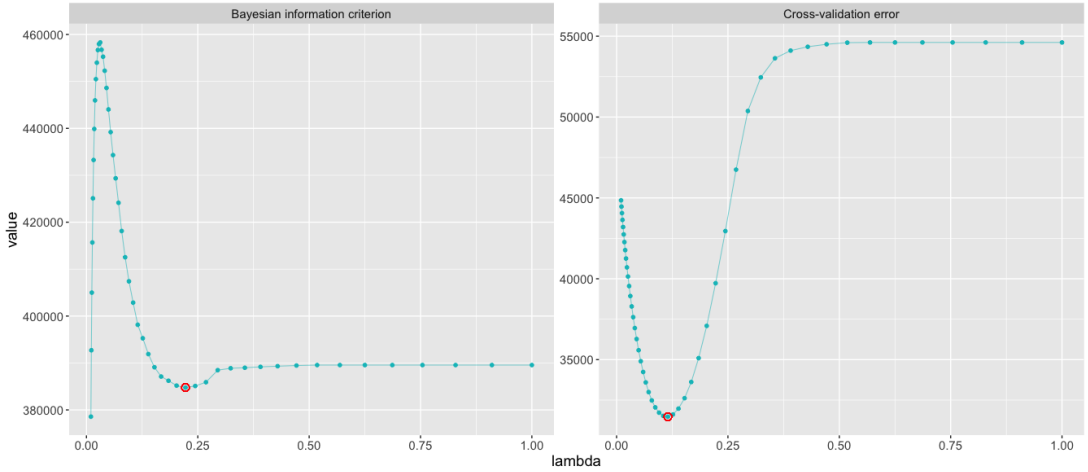


Figure 6.3: Values of BIC (on the left) and CV (on the right) obtained for glioblastoma datasets with $K = 3$, $p = 483$ and $n_1 = 40$, $n_2 = 77$, $n_3 = 80$ over the uniform in log-scale grid of 50 parameters of λ from 0.01 to 1. Red circles denote the estimated optimal λ_{OPT}.

When we used cross-validation (again, with the warm start) to estimate the optimal value of λ, we obtained $\lambda_{CV} = 0.1151$ (see Fig.6.3). We ran *jewel* with this value of the regularization parameter. Resulting graph G_{CV} has 7272 edges (about 6.2% of all possible edges) and no isolated vertices. Since in this example $\lambda_{CV} < \lambda_{BIC}$, G_{CV} has more connections than G_{BIC}.

Then we analyzed the $p = 483$ genes in the STRING database [38]. As in the previous section, we limited the query to connections from "experiments" and "databases" as active interaction sources setting the minimum required interaction score to the highest value of 0.9. The resulting STRING network had 415 out of 483 vertices connected to any other node and 4134 edges.

We measured the number of connections common to our estimated network and the network from STRING database. For each case, Fig.6.4 show the connections identified by *jewel* that were present also in the STRING database. For G_{BIC}, we observed 170 edges in common; while for G_{CV}, we had 297 common edges (see all the results summarized in Table 6.3).

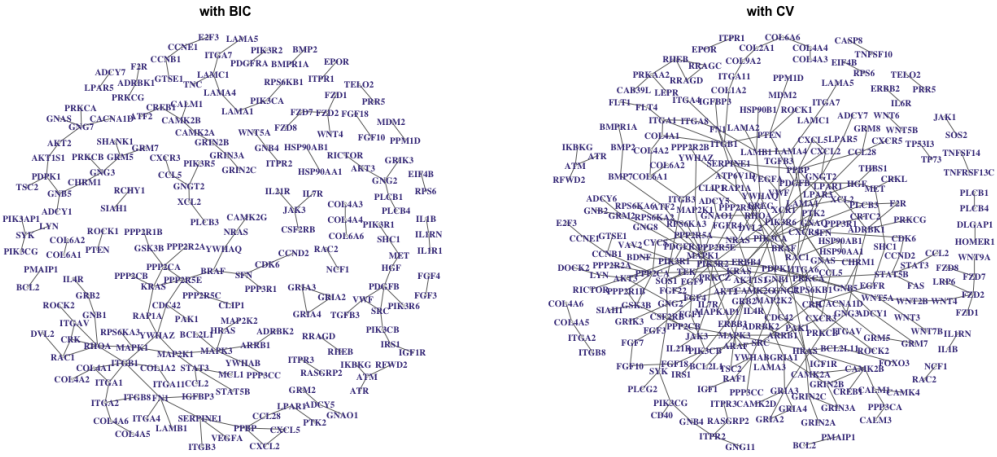


Figure 6.4: Intersection of the networks estimated with *jewel* from glioblastoma datasets and the one obtained from the STRING database. Regularization parameter, used in the estimation, was obtained with BIC (on the left) and with CV (on the right).

	λ_{OPT}	# est. edges	#edges in intersection	p-value
BIC	0.2223	3113/116403	170/4134	3.29255e-08
CV	0.1151	7272/116403	297/4134	0.00697

Table 6.3: Results of BIC and CV procedures obtained for $K = 3$ glioblastoma datasets with $p = 483$ over the uniform in log-scale grid of 50 parameters of λ from 0.01 to 1. The p-values is the results of the hyper-geometric test to assess the significance of the edge overlap.

Although the number of edges in the intersection is lower than in our previous real data application example, it is significant according to the hypergeometric test. Moreover, we notice many groups of genes identified consistently, such as collagen alpha chains, ionotropic glutamate receptors, frizzled class receptors, interleukin 1 receptors, and fibroblast growth factors, collagen, and others. The biggest hubs in G_{BIC} include PPP3CC (frequently underexpressed in gliomas), RCHY1 (vice versa, typically highly expressed in this condition), and IL4R (is associated with better survival rates). In G_{CV}, the biggest hubs are TNN (that is considered a therapeutic target since an increase in expression can suppress brain tumor growth), CALML6, and BCL-2 (that can block apoptosis, i.e., cell death, and therefore may influence tumor prognosis).

To conclude, *jewel* demonstrated its ability to identify connections from the real data, providing more reliable results when applied to the data from a meta-study. In the future we are planning to perform analysis in a higher dimensional context.

Chapter 7

Conclusions and future work

The book "Statistical learning with sparsity" [37] cites Rutherford D. Rogers, a librarian at Yale quoted in New York Times in 1985: *"We are drowning in information and starving for knowledge."* Thirty-five years later, these words are more true than ever. The work carried out with this thesis represents one of the numerous attempts of modern science (and a successful one) of structuring the immense amount of information using statistical methods to derive a meaningful conclusion.

7.1 Thesis results

If we have a dataset consisting of N observations of p variables, how can we describe the direct connections between any two variables, removing the other variables' influence? The graphical model theory tells us to encode these connections into a graph and then estimate the corresponding graphical model. Moreover, if we further assume that the variables follow a Gaussian distribution, then we can estimate the inverse covariance matrix instead of estimating the graph. I reviewed these well-known concepts and properties in Chapter 2.

Then, in the described setting, how exactly do we estimate the inverse covariance matrix? In particular, what are the challenges to face in the high-dimensional regime? To this purpose, I first described the well-known sparsity assumption, i.e., when we assume that the graph of interest has few significant edges, making it more interpretable. After that, I reviewed a few of the methods applicable in high-dimensional sparse settings. In particular, in Chapter 3, I focused on two popular classes of methods: minimization problems based on maximum likelihood or regression approaches. I described the algorithms used to estimate the solution of these minimization problems, mainly focusing on the group descent algorithm.

To meet the novel challenges emerging from technological progress, we can assume that instead of one dataset, we can collect several datasets with observations of mostly the same variables under different conditions. It is then natural to consider an extension of the approaches mentioned above and move from one dataset to the joint analysis of multiple datasets. I discussed the mul-

titask learning framework in the last part of Chapter 3 and Appendix A, giving a review of several available methods for joint estimation and their algorithms, also underlining some specific details about them. Such material gave the pave to formulate my proposal. So, in Chapter 4, I introduced *jewel*, a novel method in this framework, where *jewel* stands for **j**oint nod**e-w**ise estimation of multiple Gaussian graphical mod**el**s, and constitutes the main achievement of my thesis.

In *jewel*, I proposed a regression-based minimization problem with a group lasso penalty to analyze several datasets jointly. The penalty's novelty is that it simultaneously guarantees both the sparsity and the symmetry of the resulting estimator. Therefore, it eliminates the need to use post-processing rules as "AND/OR" to obtain the symmetry. I obtained an algorithm to solve the minimization problem based on the iterative group descent method. I implemented the proposed algorithm using the active-shooting approach, which significantly decreased the algorithm's running time without compromising the performance. For the obtained estimator, I proved the variable selection consistency property. Although this result provides theoretical bounds for the regularization parameter, such conditions are defined through true (unknown) parameters and can not be used in practical applications. Therefore, I introduced suitable modifications of the Bayesian information criterion and cross-validation procedure to estimate the regularization parameter from the observed data. I also described the warm-start "trick" that can be implemented with BIC and CV to decrease the running time. I implemented the proposed method and several auxiliary functions in the novel R package `jewel`.

In Chapter 5, I explored the behavior of *jewel* in different simulations settings. I proved that using several datasets can significantly gain performance and that the joint estimation is better than other naive alternatives. Then, I analyzed how *jewel* performs under different dimensional regimes and with different structures of scale-free graphs. With this study, I discovered the limit in the performance for graphs that contain big hubs (I will discuss this drawback in the next section). I also proved that the stopping criteria threshold does not influence how well *jewel* performs. Therefore, it can be set to a relatively large value to decrease the number of iterations and, hence, the running time. Additionally, I also explored the influence of the simulated precision matrix entries' values on the performance of joint estimation methods in general. It was limited, however, it was never explored before. Finally, I compared *jewel* to other available methods for joint estimation, namely JGL [19] and Guo et al. proposal [36]. I showed that *jewel* performs better than the latter and comparable to JGL in all simulation settings. Moreover, *jewel* shows an advantage in running time to both methods.

In the last chapter, I described applications of *jewel* to the real data. Although, at the moment, I have explored only a small example (consisting of about a hundred variables), observing that the method demonstrates quite promising results, much larger experiments are yet to be conducted in the upcoming weeks.

7.2 Future work

Despite *jewel* demonstrated quite good performance in many contexts, several modifications could be incorporated into the model or added to the implementation to face drawbacks and further improve the results.

The first drawback that I observed was the decrease in performance of all analyzed methods (including *jewel*) when increasing the parameter *power* from scale-free networks' simulations. This parameter determines the preferential node attachment for scale-free graphs and increases the hubs in the graph. Unfortunately, this limitation is relevant since many real networks have a predominant hub-structure. However, our findings agree with the recent paper [62], where the same lack of performance was observed in the case of one data set ($K = 1$) for the following methods: *glasso* [29] (described in Subsection 3.2.3), Meinshausen and Bülhman method [51] (described in Subsection 3.3.1) and *space* [57]. This fact motivated the authors to propose a novel degree-weighted lasso method (DW-Lasso). In DW-Lasso, the authors incorporated into the lasso penalty normalized degree-induced weights, which are updated them iteratively until the algorithm reaches convergence. The simulation results of [62] showed that DW-Lasso performs significantly better for networks with hubs. By contrast, for not highly structured sparse networks with a few hubs, the graph can be better reconstructed using the traditional methods. Since the degree-induced approach is more computationally expensive compared to the classical approach, it should be used when we expect that the network contains several hubs to justify the increase in runtime with a significant performance gain.

Interestingly, regression-based method *space* [57], mentioned in the previous paragraph, also incorporates weights into the minimization problem, although only in the goodness-of-fit part. These weights are not necessarily degree-induced (for example, the authors also suggest using the residual variance). Although this approach demonstrated better performance than *glasso* and Meinshausen and Bülhman method, it performs worse than DW-Lasso, where weights are incorporated in the penalty.

In [64] the difficulty in inferring networks with hubs is addressed using different approach. In their maximum likelihood minimization problem, the authors proposed a hub penalty function, which is based on a specific decomposition of the inverse covariance matrix into the sum of two sparse symmetric matrices. The first one represents the edges between non-hub nodes, the second one, whose columns are either entirely zero or almost entirely non-zero, represents the edges of hubs (non-zero columns represent hubs' edges). This approach also demonstrated an advantage in performance compared to *glasso* and Meinshausen and Bülhman method, mentioned above, and some other proposals. Similar decomposition of the precision matrix can also be found in JRmGRN method [23].

To conclude this first discussion on my future work, I aim to extend the approaches proposed in [62] or [64] to the case of multiple datasets or incorporate suitable weights into *jewel* to enforce hub-structures to emerge.

Another interesting direction of the future work is the modification of the penalty to account for differences between underlying classes. I assumed that the underlying graph G is the same across the K datasets. However, such an assumption might not always hold. For example, imagine we measure gene expression and infer the conditional dependency network among the genes. If we use datasets collected from different studies or laboratories but related to the same experimental condition, we might assume that the underlying graph G remains the same, despite the different high-throughput technologies. If we instead collect gene expression from different disease subtypes, such as different cancer stages, then the common mechanism G still would be present, and therefore, we could still aim to estimate it, but case-specific differences $G^{(k)}$ might be present as well.

To face this problem, many joint estimation methods, described in Section 3.4, both derive the common underlying graph G from K datasets and simultaneously identify the edges that are specific only to some categories $G^{(k)}$ (see [19, 48, 74]). They usually achieve such a structure by combining two kinds of penalties – one accounting for commonalities and another for differences. Hence, one way to extend *jewel* from this point of view is to add such second penalty term that accounts for differences between classes. However, it is worth noting that most of the methods mentioned above are maximum-likelihood, not regression-based. Therefore, this modification of *jewel* penalty and subsequent solution of the resulting minimization problem is yet to be explored.

As pointed out in Section 3.4, there are already many methods in the literature that incorporate prior (biological) information into the learning process. For example, in [75], the specific use of the *graphical lasso* incorporates some cancer genetic information. I also refer to [77] for an interesting broad review of knowledge-guided supervised and unsupervised statistical methods specific for omics data. In the following, I will only provide an example about the type of prior knowledge I would like to incorporate in *jewel* in my future work, namely the gene pathways.

Nowadays it is well known that genes interact in functional pathways to form gene regulatory networks. Each pathway consists of several genes or subnetworks. Genes in a given pathway perform specific biological functions, are often deeply connected and much less connected with those belonging to other pathways. This knowledge is being derived from thousands of experiments and analyses with different levels of reliability. It is then structured and stored in specific databases, such as the Kyoto Encyclopedia of Genes and Genomes (KEGG) [42], Gene Ontology [1], Reactome etc. (see [58] for more). How can we use such information to improve the performance or to speed-up the execution of a method? Imagine we investigate K subtypes of diseases, but from previous studies and databases we know that there is a given pathway of genes that is mostly active only in some disease subtypes – or, in other words, that for some pair of variables i, j connection is present only in some specific classes $M \subset \{1, \dots, K\}$. Then, instead of the group of variables $(\Theta_{ij}^{(1)}, \Theta_{ji}^{(1)}, \dots, \Theta_{ij}^{(K)}, \Theta_{ji}^{(K)})$, we would consider the groups $\left(\Theta_{ij}^{(k)}, \Theta_{ji}^{(k)}\right)_{k \in M}$ and $\left(\Theta_{ij}^{(k)}, \Theta_{ji}^{(k)}\right)_{k \in M^c}$. In principle, this group rearrangement

could incorporate our prior (biological) knowledge into the learning process, accounting for specific differences among classes. In the same spirit, if we knew that some genes are highly connected, i.e., they represent hubs, we could initialize their weights to enforce their importance during the learning process (in a minimization problem with the degree-induced weights). Alternatively, if we assume that we knew some genes are never present in the specific pathway and/or have no confirmed interactions, we could set the corresponding entries of **Active** to zero in our initialization step.

To summarize, incorporating prior biological knowledge is not yet very "popular" in graphical model inference. However, it seems very promising since it can provide advantages both in performance and running time. So, I think it will undoubtedly be one of my future extensions.

Another direction for my future work is to face the computational burden of *jewel* when the number of variables increases. One of my first attempts will be to consider a screening pre-processing of the data to reduce the network's complexity and the number of variables to handle. At this step, some connections can be set to zero before starting the estimation, reducing the number of "active" groups to process during the minimization procedure and therefore improving the running time accordingly. More specifically, the idea of screening is to perform a preliminary analysis of the input data before applying the inference method, identifying somehow the connections that are not "important" with high probability. Those connections that do not "pass" the screening are eliminated by zeroing the corresponding entry of **Active**. It is also possible to remove variables that are screened as isolated and correspond to potential singletons in a graph.

We can implement a naive example of the screening process as following. As a first step, we construct empirical covariance matrices $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(K)}$ from the input datasets. Then, we analyze the "strength" of the connections between pairs of variables through the entries of these matrices. If the entry i, j is below a chosen threshold $|\mathbf{S}_{ij}^{(k)}| < \mu$, we consider the connection between variables i, j in k-th class "weak" and put the element to zero $\mathbf{S}_{ij}^{(k)} = 0$. Then, the active variables can be initialized as $\mathbf{Active} = \cup_k \operatorname{supp} \mathbf{S}^{(k)}$ instead of being all 1s. One can choose some other rule for the construction, like voting, always preserving the symmetry of matrix **Active**. The screening threshold choice μ can be empirical or obtained by a model selection procedure or other.

While it is evident that screening procedures can lead to a decrease in running time, its influence on efficiency is yet to be explored. Discussions on more sophisticated screening, including those that preserve performance, can be found in [26, 70].

Another way to face the computational burden of *jewel* could be to modify its implementation. One idea is to adapt the block-diagonalization approach suggested, for example, in [19, 53]. What if we have biological knowledge, as discussed earlier in this section, and are aware that the genes are operating in distinct pathways? Intuitively, we could apply *jewel* pathway-by-pathway to move from the inference on a large graph to the inference of several subgraphs of

smaller dimension. More in general, can we identify the unconnected components in the underlined graph? In both cases, after a proper rearrangement of the row/column indices, the underlying graph's adjacency matrix results to be block-diagonal. Hence, we can apply *jewel* to several independent subproblems of smaller dimensions, i.e., to each block of the input data, and derive a block-diagonal estimator. We can further benefit from this approach if we parallelize the application to each block, especially for large matrices. Such improvement will be crucial to face problems with tens of thousands of variables that can be reduced to subproblems, each of them of dimensions of a few hundred.

The block-diagonalization approach is very appealing since it has no influence on performance. In JGL [19] the authors built and inspected the empirical covariance matrices $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(K)}$ to determine whether the underlying graph is block-diagonal after some permutation of the features. They proved two theorems – one for each of the proposed penalties, fused and group – which provided necessary and sufficient conditions on the entries of empirical covariance matrices and regularization parameters to establish if there is a block-diagonal structure. If these conditions hold, they applied the JGL algorithm to each block separately, obtaining exactly the same solution as they would by applying the JGL to the whole $p \times p$ matrix (but much more efficiently). Note that JGL method requires eigendecomposition of a $p \times p$ matrix, hence with the block-diagonalization approach authors were are able to achieve a great advantage in running time. There are no similarly "expensive" procedures in *jewel*, however, we can still achieve a decrease of running time by reducing the dimensionality of the problem.

I am also considering extending the implementation of the method for a different number of variables p_k across datasets. Although theoretically there are no limitations, up to my knowledge, there are yet no implementations of joint estimation methods that have already accounted for this fact and do not set $p_k = p \ \forall k$. Future version of `jewel` will fill this gap.

Lastly, it is fundamental to mention that estimating conditional dependency relationships between variables is of interest without the Gaussian hypothesis. This observation is particularly relevant for the recent development of single-cell omics technologies that produce count data, which, of course, can not be modeled by Gaussian variables. However, while I acknowledge the need for such methods, this task would probably require not only to extend or modify *jewel* but to completely redefine the whole framework and probably develop an entirely new method compatible with discrete data. This future work is the more demanding, and some attempts can be found in [49].

To conclude, this thesis aimed to develop a novel method for inferring a graphical model from several datasets, overcoming the limitation of the absence of symmetry in previous proposals. To this aim, I proposed *jewel*, that demonstrated good performance compared to its competitors. However, as I have underlined in this section, there is still room for improvements, and there are many possibilities for further developments of the proposed method.

Appendix A

Joint estimation methods review

In this appendix I present a condensed table-form review of methods for joint estimation of Gaussian graphical models in multiple datasets setting; some of them were described in the last subsection of Chapter 3.

The 1st column of the table reports the reference paper. From 2nd to 5th columns I give a brief description of the method: used approach, type of penalty, algorithm and some other points of particular interest presented in the paper.

Then from column 6 to column 8, I describe the simulation setting: availability of the code, what types of graphs and what combinations of K, p and n_k parameters were used. Note that the notation like $K = 2$, $p = 500/1000$, $n_k = 50/200/500$ means that for $K = 2$ two simulations were conducted: with $p = 500$ and $p = 1000$. For each p, again, three simulations were carried out with $n_k = 50$, $n_k = 200$ and $n_k = 500$ $\forall k$. In the 9th column, I give other notes on simulation settings, for example the support of the uniform distribution from which entries of the precision matrices were sampled or tolerance threshold for the iterative algorithm.

Finally, in the last two columns, I summarize information about the real data on which the methods were tested. Here the notation $n_k = 544, 374, 310$ and 168 rather means that we had $K = 4$ datasets with sample sizes $n_1 = 544$, $n_2 = 374$, $n_3 = 310$ and $n_4 = 168$.

Papers, included in the table, are

- *Joint estimation of multiple graphical models* by J. Guo, E. Levina, G. Michailidis and Ji Zhu [36];
- *The joint graphical lasso for inverse covariance estimation across multiple classes* by P. Danaher, P. Wang, D. M. Witten [19];
- *Node-based learning of multiple Gaussian graphical models* by K. Mohan, P. London, M. Fazel, D. Witten and Su-In Lee [53];
- *Joint structural estimation of multiple graphical models* by J. Ma and G. Michailidis [48];
- *Structural pursuit over multiple undirected graphs* by Yu. Zhu, X. Shen and W. Pan [80];

- *Joint graphical model for inferring gene networks across multiple subpopulations and data types* by X.F. Zhang, Le Ou-Yang, X.T. Hu and H. Yan [74].

Joint estimation methods review

1	2	3	4	5	6	7	8	9	10	11
	The problem				Simulation				Real data	
Paper	Approach	Penalty	Algorithm	Other	code	Types of graph	K, p, n_k	Other	Data	K, p, n_k
Joint estimation of multiple graphical models by J. Guo, E. Levina, G. Michailidis and Ji Zhu	Penalized log-likelihood	$\lambda \sum_{i \neq j} \sqrt{\sum_{k=1}^{K} \left\lvert \Omega_{ij}^{(k)} \right\rvert}$	iterative local linear approximation	• consistency and sparsistency proof	R, by request	• chain • nearest-neighbour • scale-free	K = 3, p = 100, n_k = 100	• [-1, -0.5] + [0.5, 1] • tol = 10^{-2} • #runs = 50	Terms on webpages of computer science departments of different universities.	K = 4, p = 100 (orig. 4800), n_k = 544, 374, 310 and 168.
The joint graphical lasso for inverse covariance estimation across multiple classes (JGL) by P. Danaher, P. Wang and D.M. Witten	Penalized log-likelihood	Group penalty $\lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} \left\lvert \Omega_{ij}^{(k)} \right\rvert + \lambda_2 \sum_{i \neq j} \sqrt{\sum_{k=1}^{K} \Omega_{ij}^{(k)2}}$ Fused penalty $\lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} \left\lvert \Omega_{ij}^{(k)} \right\rvert + \lambda_2 \sum_{k<k'}^{K} \sum_{i \neq j} \left\lvert \Omega_{ij}^{(k)} - \Omega_{ij}^{(k')} \right\rvert$	alternating directions method of multipliers algorithm	• how to block-diagonalize the solution	R package JGL on CRAN	• block-structured scale-free	• K = 3, p = 500, n_k = 150 • K = 2, p = 500/1000 n_k = 50/200/500	• sparsity ~ 0.4% • [-0.4, -0.1] + [0.1, 0.4] • tol = 10^{-5} • #runs = 100	Lung cancer microarray gene expression data (patients and controls)	K = 2, p = 17826 reduced to 278 by block-diagonalisation (orig. 22 283) n_k = 97 and 90.
Node-based learning of multiple Gaussian graphical models (PNJGL and CNJGL) by K. Mohan, P. London, M. Fazel, D. Witten and Su-In Lee	Node-based penalized log-likelihood	$RCON_q$ – row-column overlap norm induced by l_1/l_q matrix norm. PNJGL $\lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} \lvert \Omega_{ij}^{(k)} \rvert + \lambda_2 \sum_{k<k'}^{K} RCON_q(\Omega^{(k)} - \Omega^{(k')})$ CNJGL $\lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} \lvert \Omega_{ij}^{(k)} \rvert + \lambda_2 RCON_q(\Omega^{(1)} - diag(\Omega^{(1)}), \ldots, \Omega^{(K)} - diag(\Omega^{(K)}))$	alternating directions method of multipliers algorithm	• row-column overlap norm • how to block-diagonalize the solution	Matlab, link in the paper	• Erdos-Renyi • scale-free • community	• K = 2, p = 100/200/500, n_k = p/2 • K = 2, p = 100, n_k = 25/50/100/200	• [-0.6, -0.3] + [0.3, 0.6] • tol = 10^{-4} • #runs = 20	mRNA expression levels for two subtypes of glioblastoma multiforme Also university webpages dataset, described above.	K = 2 (orig. 4), p = 34 (orig. 11 861), n_k = 53 and 56 (orig. 220 in total).

1	2	3	4	5	6	7	8	9	10	11
	The problem				Simulation				Real data	
Paper	**Approach**	**Penalty**	**Algorithm**	**Other**	**code**	**Types of graph**	**K, p, n_k**	**Other**	**Data**	**K, p, n_k**
Joint structural estimation of multiple graphical models (JSEM) by J. Ma and G. Michailidis	combined: 1st step regression-based, 2nd step – maximum likelihood refitting	$\mathcal{G}_{ij}$– partition of {1, …, K} $g \in \mathcal{G}_{ij}$ – group $\boldsymbol{\theta}_{ij}^{[g]} = (\Theta_{ij}^{(k)}, \Theta_{ji}^{(k)})_{k \in g}$ $\lambda_{ij}^{[g]} \|\|\boldsymbol{\theta}_{ij}^{[g]}\|\|$	coordinate descent + glasso	• estimation and graph selection consistency proof	R, by request	• scale-free	• K = 5, p = 100 n_k = 50/200 • K = 10, p = 50, n_k = 100/200	• [-1, -0.5] + [0.5, 0.1] • #runs = 20	Measurements of different climate variables over US locations RNA-seq measurements for subtypes of breast cancer	K = 27, p = 16, n_k = n = 17 K = 2, p = 800 (orig. 17 296), n_k = 403 and 117.
Structural pursuit over multiple undirected graphs by Yu. Zhu, X. Shen and W. Pan	Penalized log-likelihood	$\mathcal{E}_{ij}$ – prior knowledge about simularity of ij entry of precision matrix in different classes $\lambda_1 \sum_{k=1}^{K} \min\left(\left\|\Omega_{ij}^{(k)}\right\|, \tau\right) +$ $\lambda_2 \sum_{k,k' \in \mathcal{E}_{ij}} \min\left(\left\|\Omega_{ij}^{(k)} - \Omega_{ij}^{(k')}\right\|, \tau\right)$	difference convex programming, augmented Lagrangian method and the block-wise coordinate descent	• clustering and sparsness structures • necessary and sufficient partition rule • finite-sample bound error	?	• chain • nearest-neighbour • exponen-tially decaying networks	• K = 4, p = 30/200 n_k = 120/300 • K = 30, p = 20, n_k = 120/300 • K = 90, p = 10, n_k = 120/300 • K = 4, p = 1000, n_k = 120 • K = 4, p = 2000, n_k = 500	• [-1, -0.5] + [0.5, 0.1] • #runs = 100	Single cell flow cytometry data	K = 2, p = 11, n_k = 7466 and 4206.
Joint graphical model for inferring gene networks across multiple subpopulations and data types (JEGN) by X.-F. Zhang, Le Ou-Yang, X.T. Hu and H. Yan	Penalized log-likelihood	K data types, L subpopulations $\boldsymbol{\Omega}^{(k,l)} = \mathbf{R}^{(k)} + \mathbf{M}^{(k,l)}$ $\lambda_1\lambda_2 L \sum_{i \neq j} \sqrt{\sum_{k=1}^{K} \mathbf{R}_{ij}^{(k)2}} +$ $\lambda_1(1-\lambda_2) \sum_{l=1}^{L} \sum_{i \neq j} \sqrt{\sum_{k=1}^{K} \mathbf{M}_{ij}^{(k,l)2}}$	alternating directions method of multipliers algorithm	• extension to nonparanor-mal case	R package JEGN on github	• Erdos-Renyi • scale-free	• K = 3, L = 4 (12 in total), p = 100 n_k = 25/50/100	• #runs = 50	Microarray and RNA-seq gene expression for subtypes of breast cancer	K = 2, L = 4, p = 139 (orig. 15 178), n_k = 214, 117, 55 and 90.

Appendix B

`jewel` package

This appendix consists of three parts. In the first part, I provide motivations for the development of `jewel` package, namely why I chose the R language and why I arranged the code in a package. In the second part, I describe the package structure. Finally, in the third part, I provide the package documentation.

The package `jewel` is open-source and freely available at `https://github.com/annaplaksienko/jewel`.

B.1 Why R package?

As explained in the Chapter 1, our primary motivation was to develop a method for estimating graphical models using data from data omics studies (although, theoretically, there are no limitations for using *jewel* in any other applied field). For this reason, we chose the R language since it is among the more popular languages in both bioinformatics and statistics. An illustration of this statement is the fact that apart from the official CRAN repository[1] of R packages, there exists a specific project – Bioconductor – which is a repository of R packages devoted to biological applications. Currently, Bioconductor contains almost 2000 packages, most of them very well documented (since in the last few years, the standards for publishing Bioconductor packages are very rigorous). Such a large number highlights the importance of Bioconductor among practitioners in biological applications and contributs to the R language's widespread use.

R is a high-level language, meaning that it is relatively easy to write a code in and interpret other colleagues' code. Moreover, R is open source, and it makes available many powerful statistical models and visualization tools. However, R has a drawback: its speed. In "Advanced R" H. Wickham [67] goes into details about how the R design language imposes fundamental constraints on its speed and how its GNU-R implementation[2] has not yet reached the theoretical maximum performance speed. The main reason is that R was developed not to

[1]The Comprehensive R Archive Network (CRAN) – archive of the latest and previous versions of the R distribution, documentation, and ≈17 000 contributed R packages.

[2]Implementation of R from r-project.org.

make data analysis fast but instead to make it understandable and accessible, especially for users who do not have special training in this area. The speed drawback is usually overcome by implementing the core package functions in C++ and using R only as an external interface. For example, the `huge` package that contains an efficient implementations of *glasso* and the Meinshausen and Bülhman algorithm [51] is made in C++. Interfacing R with C++ can be achieved with the help of `Rcpp` package. In this version of `jewel` we used only R, therefore it can not be compared with C++ code in terms of speed but only with other R codes. Future versions can face this limitation.

Another point we would like to emphasize in this section is why one should consider building a package. First, we firmly believe that it is essential to accompany published research with the related code, as it allows for reproducibility of results and simplifies the comparison process with competitors - comparison and reproducibility are vital in science. On the other hand, organizing the code as a package rather than as raw files increases accessibility for users who might otherwise have difficulty, and in general, simplifies the usage for everyone. Unfortunately, although almost all authors nowadays conduct simulation studies for their novel methods, developing a package or even providing code online is still not a must, and we aimed to overcome this flaw.

Therefore, we highlight that implementation of `jewel` package is one of the most important results of this thesis, as this is not always true for other competitor methods for the joint graph estimation. One direction of our future work will be devoted to the package's upgrades.

B.2 Package structure

Package `jewel` and its documentation were built using `devtools` [69] and `roxygen2` [68] R packages. The 0.1.0 version of `jewel` contains six functions:

- `jewel()` – implementation of *jewel* method;
- `estimateLambdaBIC()` – Bayesian information criterion for choosing λ;
- `estimateLambdaCV()` – cross-validation for choosing λ;
- `generateLambda()` – generation of λ grid;
- `evaluatePerformance()` – comparison of the true and estimated graphs;
- `generateData()` – generation of data according to Subsection 5.1.1.

The main function is `jewel()` that implements the method proposed in Chapter 4. It requires a list structure of data matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ and a regularization parameter λ as input; instead as optional input parameters one can furnish the list of starting values for the regression matrices $\mathbf{\Theta}^{(1)}, \ldots, \mathbf{\Theta}^{(K)}$, the value of stopping criteria threshold and the maximum allowed number of iterations. `jewel()` produces as main output the estimated adjacency matrix; however, it also provides the list of estimated regression coefficient matrices $\hat{\mathbf{\Theta}}^{(1)}, \ldots, \hat{\mathbf{\Theta}}^{(K)}$,

the list of residual matrices $\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(K)}$ and BIC value for the given λ. Last three outputs are mostly auxiliary for the evaluations of BIC and CV procedures.

Functions `estimateLambdaBIC()` and `estimateLambdaCV()` implement the two selection criteria presented in Subsection 4.4. They both require as input a list of data matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ and a grid of parameters λ_l, $l = 1 \ldots L$, over which the selection procedures is performed. The grid of parameters λ_l can be user provided or generated with `generateLambda()` function. Both `estimateLambdaBIC()` and `estimateLambdaCV()` functions are implemented with the warm start approach for the reasons explained in Subsection 5.5. CV also takes advantage of parallelization over folds which greatly speeds-up the computation, especially in the case when the number of available cores is greater than the number of folds (which is 5 by default but can be changed by the user). To this aim we use `parallel` package.

Function `evaluatePerformance()` is an auxiliary function, and it evaluates the method's performance assuming that we know the truth. In particular, it takes as input two adjacency matrices, the true one and the estimated one, and it gives as outputs the TP, TN, FP, and FN values defined in Subsection 5.1.2. Of course, the two input adjacency matrices need to have the same dimension. Function `evaluatePerformance()` in principle can also be used in other contexts to compare two graphs. However, it cannot be used in real applications because the ground truth is not available.

Finally, function `generateData()` is also auxiliary because it gives no contribution to the inference, but allows generation of the synthetic data. However, it permits to study the performance of *jewel* and other competitor methods, possibly reproducing results presented in Chapter 5 of this thesis. Specifically, it generates a "true" scale-free graph and a list of data matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ as described in details in Subsection 5.1.1 of Chapter 5. Function `generateData()` requires as input the number of vertices p, the *power* of preferential attachment and parameter m which controls the sparsity (default values are $power = 1$ and $m = 1$, the resulting graph having $mp - (2m - 1)$ edges). These input choices define the "true" scale-free graph generated by function `generateData()`. Moreover, function `generateData()` requires as input the number of matrices K, the size of each data matrix $n_k = n \; \forall k = 1 \ldots K$ and parameters a and b which specify the interval $[-b, -a] \cup [a, b]$ where the entries of the true precision matrices are uniformly sampled (default values are $a = 0.2$, $b = 0.8$). We recall from Subsection 5.1.1 that once we have the "true" underlying graph it is possible to generate K precision matrices with support of the given graph. Hence, by inverting the precision matrices, the "true" covariance matrices are obtained and finally by each of them a data matrix is sampled from a p-variate Gaussian distribution. Note that one can provide several sample sizes and therefore get several sets of K datasets with the same underlying graph but different sample sizes.

Function `generateData()` outputs not only the data matrices, but also the "true" underlying graph and the "true" covariance matrices.

The documentation of the `jewel` packages follows.

Package 'jewel'

March 9, 2021

Title Joint node-wise estimation of Gaussian graphical models from multiple datasets

Version 0.1.0

Author Anna Plaksienko, Claudia Angelini, Daniela De Canditiis

Maintainer Anna Plaksienko <anna@plaxienko.com>

Description jewel is a method for joint estimation of the graph of conditional dependencies between variables given multiple classes of data. Package also allows to estimate regularization parameter with Bayesian information criterion and cross-validation and to generate simulation data. Reference paper to be added.

Depends R (¿= 3.6.0)

Imports Matrix, matrixcalc, MASS, SMUT, igraph, rlist, parallel, purrr

URL https://github.com/annaplaksienko/jewel

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

R topics documented:

`estimateLambdaBIC` *Estimation of the optimal regularization parameter for jewel method with Bayesian information criterion*

Description

Given a grid of regularization parameters, function evaluates Baeysian information criterion (BIC) for each element. Optimal lambda is chosen as the one for which BIC's minimum is obtained. Warm start is implemented.

Usage

```
estimateLambdaBIC(X, lambda, makePlot = TRUE)
```

Arguments

`X`	list of `K` numeric data matrices of size `n_k` by `p` (`n_k` can be different for each matrix)
`lambda`	vector of parameters for which function evaluates BIC
`makePlot`	If makePlot = FALSE, plotting of BIC is disabled. The default value is TRUE.

Value

The following list is returned

- `lambda_opt` - a number, optimal value of regularization parameter according to BIC procedure;
- `BIC` - a vector of BICs for each element of input vector `lambda`.

`estimateLambdaCV` *Estimation of the optimal regularization parameter for jewel method based on cross-validation*

Description

Given a grid of regulariation parameters, function performs cross-validation and estimates the optimal parameter as the one for which the cross-validation error's minimum is obtained. Parallelization over folds and warm start are implemented.

Usage

```
estimateLambdaCV(X, lambda, k_folds = 5, verbose = TRUE, makePlot = TRUE)
```

Arguments

`X`	list of `K` numeric data matrices of size `n_k` by `p` (`n_k` can be different for each matrix)
`lambda`	vector of parameters over which cross-validation is performed
`k_folds`	number of folds in which data is divided. The default value is 5.
`verbose`	If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.
`makePlot`	If makePlot = FALSE, plotting of CV error is disabled. The default value is TRUE.

Value

The following list is returned

- `lambda_opt` - a number, optimal value of regularization parameter according to cross-validation procedure;
- `CV_err` - a vector of cross-validation errors for each element of input vector `lambda`

`evaluatePerformance` *Evaluation of graph estimation methods perfomance if true graph is known*

Description

Function compares adjacency matrices of true and estimated simple graphs and calculates the number of true positives (correctly estimated edges), true negatives (correctly estimated absence of edges), false positives (edges present in the estimator but not in the true graph) and false negatives (failure to identify an edge).

Usage

```
evaluatePerformance(G, G_hat)
```

Arguments

`G`	True graph's adjacency matrix
`G_hat`	Estimated graph's adjacenct matrix. Must have the same dimensions as `G`.

Value

performance - vector of length 4 with TP, TN, FP, FN

`generateData` *Generation of a scale-free graph and corresponding datasets using the graph as their Gaussian graphical model*

Description

Function generates a scale-free graph with `p` vertices and `K` corresponding precision and covariance matrices, all of the size `p` by `p`. Then for each `l`-th element of vector `n` it generates `K` data matrices, each of the size `n_l` by `p`, i.e., for the same underlying graph we can generate several sets of `K` datasets with different sample sizes.

Usage

```
generateData(
  p,
  K,
  n,
  power = 1,
  m = 1,
  a = 0.2,
  b = 0.8,
  makePlot = TRUE,
  verbose = TRUE
)
```

Arguments

`p`	Number of nodes in the true graph
`K`	Number of data matrices
`n`	Vector of the sample sizes for each desired set of `K` data matrices. Can be a vector of one element if one wishes to obtain only one dataset of `K` matrices.
`power`	Power of preferential attachment for Barabasi-Albert algorithm for generation of the scale-free graph. The defaut value is 1.
`m`	Number of edges to add at each time step of Barabasi-Albert algorithm for generation of the scale-free graph. Resulting graph has `mp -(2m -1)` edges. The default value is 1 and graph has `p-1` edges.
`a`	Entries of precision matrices are sampled from the uniform distribution on the interval `[-b,-a] + [a,b]`. The default value is `a = 0.2`.
`b`	Entries of precision matrices are sampled from the uniform distribution on the interval `[-b,-a] + [a,b]`. The default value is `b = 0.8`.
`makePlot`	If makePlot = FALSE, plotting of the generated true graph is disabled. The default value is TRUE.
`verbose`	If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

Value

The following list is returned

- `trueGraph` - sparse adjacency matrix of the true graph
- `data` - list of lists, for each sample size (`l`-th element of the input vector `n`) one obtains `K` data matrices, each of the size `n_l` by `p`
- `Sigma` - list of `K` covariance matrices of the size `p` by `p`

---

generateLambdaGrid *Generation of the sequence of regularization parameters*

Description

Function generates a uniform in logarithmic space grid of regularization parameters. `Lambda_max` is `max({1 / (n -1)} max(X^TX))`, `lambda_min = lambda_max * eps`.

Usage

```
generateLambdaGrid(X, nlambda = 50, eps = 0.1, scale = TRUE)
```

Arguments

`X`	list of `K` numeric data matrices of size `n_k` by p (`n_k` can be different for each matrix)
`nlambda`	desired number of parameters. The default value is 50.
`eps`	`lambda_min = lambda_max * eps`. The default value is 0.1
`scale`	if TRUE and hence data is scaled, then resulting grid is independent of `X` and goes from `eps` to 1 uniformly in log-scale. The default value is TRUE.

Value

`lambda` - vector of regularization parameters of length `n`

---

jewel *Joint node-wise estimation of Gaussian graphical model from multiple datasets*

Description

Implementation of the jewel method for estimation of the graph of conditional dependencies between the variables given multiple datasets, i.e. when observations of variables are collected under different conditions.

Usage

```
jewel(X, lambda, Theta = NULL, tol = 0.01, maxIter = 10000, verbose = TRUE)
```

Arguments

`X`	List of `K` numeric data matrices of size `n_k` by `p` (`n_k` can be different for each matrix).
`lambda`	Regularization parameter which controls the sparsity of the resulting graph - bigger it is, less edges one gets.
`Theta`	List of `K` starting regression coefficient matrices of size `p` by `p`. If not provided, initialized as all zeros and adjacency matrix is initialized as all ones.
`tol`	Convergence threshold controlling the relative error between iterations. The default value is 0.01.
`maxIter`	Maximum allowed number of iterations. The default value is 10 000.
`verbose`	If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

Value

The following list is returned

- `EstAdjMat` - adjacency matrix of the estimated graph
- `Theta` - list of `K` estimated covariance matrices of size `p` by `p`
- `residual` - list of `K` matrices of residuals of size `n_k` by `p` (`n_k` can be different for each matrix)
- `BIC` - value of Bayesian information criterion

Appendix C

Implementation discussion

This appendix contains a brief description of the computational improvements I managed to do when developing `jewel` package. Although I suppose that my discoveries are minor to anyone who is advanced in R programming, I hope this discussion may be of use to someone who, like me at the beginning of my Ph.D. studies, has little to no programming experience.

For more "algorithmic" computational improvements refer to the description of the active shooting and warm start techniques in Chapter 4.

C.1 How to make you code faster?

There are two key steps of improving the speed of the code, regardless of the logical algorithm itself: *profiling* – identifying where exactly does the code slow down and *benchmarking* – measuring the effectiveness of possible improvements compared to the existing solutions. Although it is not guaranteed these would help to achieve a great reduction in running time, it is nonetheless useful to be aware of how your code performs.

A profiler is a tool that stops the execution of code every few milliseconds and records the function that is currently "active". It manages to measure the running time and the memory occupied for each individual line of code. Profiler could be used through `profvis` package [13]) or directly through the menu at Rstudio (select lines to profile, then click "Profile" at the top menu bar and then "Profiles selected line(s)" at the drop-down menu). After profiling, one gets a flame graph with call stack (i.e. which function called what) and bar graphs of memory and execution time. This allows to visually identify so-called bottlenecks – lines of code occupying much more time compared to the rest of the code.

Once the weak point is found, one starts to think about possible improvements. Benchmarking is the comparison of several functions applied to the same data with execution time averaged over multiple runs. It allows to clearly see if the solution is in fact useful or provides little to no improvements. Package `microbenchmark` [52] can be used for benchmarking: simply pass functions to compare to the main function, like `microbenchmark(sum(1:100)/100, mean(100))`. The result would show the mean running time for each function. We advise the reader to be careful to consider if it is displayed in nano-, milli- or standard

seconds and if it is big enough to be relatable for their work.

For in-depth discussion on profiling and benchmarking we recommend chapters 23 and 24 of "Advanced R" By H. Wickham [67]. In the next section, we will demonstrate the improvements we were able to achieve with these techniques.

C.2 Examples of improvements

Lists vs long

Since *jewel* is applied to multiple datasets $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$, it felt natural to use list structure in R: store datasets as a list of matrices and access each of them simply by their index `X[[k]]`, mimicking the mathematical notation. However, we discovered that concatenating the matrices into one long matrix and accessing $\mathbf{X}^{(k)}$ by using a specifically constructed index vector is much more efficient.

```
#X is an input list of K matrices
n_k <- sapply(X, function(x) dim(x)[1])
nindex <- rep(1:K, n_k)
X_long <- do.call(rbind, X)
```

In fact, for $p = 500$ the difference in running time could be of three times. This is due to the fact that elements of the matrix are stored consecutively in the memory, while elements of the list are not. Therefore, it takes longer to access them and perform any manipulations.

Embarrassingly parallel

Problem is called *embarrassingly parallel* if it requires little or no effort to separate into parallel tasks, often because there is little or no dependency/-communication between them. This was exactly the case of *jewel* simulation: performance and running time were averaged over 20 independent runs, which had no dependency at all. Therefore, we parallelized the simulation with the help of the `parallel` package. Note that we used Windows OS with distributed memory (individual memory for master and each worker), so we had to first export libraries and all the data to workers to run parallelization. OS X and Linux systems, however, have shared memory, so these steps are redundant.

```
library("parallel")
nruns <- 20
ncores <- detectCores();
cl <- makeCluster(ncores, type = "SOCK")

#export the library and the data to workers
clusterEvalQ(cl, library("jewel"))
#G_true_list 20 true graphs
#X_list 20 sets of K data matrices
clusterExport(cl, varlist = c("G_true_list", "X_list", "lambda"))

#parallel_runs_jewel() function with jewel method, measuring
    running time, evaluating performance etc.
data <- clusterApply(cl, 1:nruns, parallel_runs_jewel)
```

One has to be careful with parallelization, since the process of master-worker communication does require some time itself, so in smaller dimensions or for faster functions it can outweigh the benefits. This, however, was not the case for *jewel* or any JGL and Guo et al proposal. With 4 physical cores, parallelization reduced the total running time for simulation almost five times.

We have also implemented parallelization in cross-validation estimation of regularization parameter λ (see Subsection 4.4.2) since evaluation of errors is performed independently for each fold.

Matrix inversion

One of the steps of generating the data for simulation studies requires to invert the "true" precision matrices $\mathbf{\Omega}^{(k)}$, $k = 1 \dots K$, (see Subsection 5.1.1). Although we could have used standard `solve` function, for symmetric positive definite square matrices there is a faster option `chol2inv` that performs the inversion from matrix's Choleski decomposition. Since $\mathbf{\Omega}^{(k)}$ by construction satisfies all the conditions, we chose `chol2inv`. For $p = 500$ it runs for about 15 milliseconds while `solve` takes around 105.

Matrix multiplication

Initialization of residual matrices $\mathbf{R}^{(k)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Theta}^{(k)}$, $k = 1 \dots K$, (see Section 4.3) at the beginning of the *jewel* algorithm requires matrix multiplication. Although we perform this operation just once, it was still desirable to speed it up. The fastest implementation we discovered was `EigenMapMult` function from `SMUT` package [78]. It uses C++ via `RcppEigen` package and demonstrates advantage both to `%*%` and `crossprod` functions.

Vector norm

Solution of minimization problem, described in Section 4.3, involves evaluation of the l_2-norm of vector $\mathbf{z}$. First intuition could be to use `norm(z, type = "2")` instead of `sqrt(sum(z^2))` for the sole reason of it being the designated function for purpose of evaluating the norm. However, `norm()` is designed for matrix norm evaluation with `"2"` parameter specifying the spectral norm of a matrix. In `norm(z, type = "2")` vector $\mathbf{z}$ is first converted into a matrix, then its singular value decomposition is found and largest singular value identified. Instead, `sqrt(sum(z^2))` does the straightforward computation and gains the running time advantage.

The difference is extremely small, about $\sim 22 \times 10^3$ nanoseconds but it accumulates in the simulation. For the case of $p = 500$, we can gain at least 3 seconds in one launch of `jewel`.

Bibliography

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, et al., *Gene ontology: tool for the unification of biology*, The Gene Ontology Consortium. Nat. Genet., 25 (2000).

[2] E. Atcheverry, M. Aubry, M. de Tayrac, E. Vauleon, R. Boniface, F. Guenot, et al., *Dna methylation in glioblastoma: impact on gene expression and clinical outcome*, BMC Genomics, 11 (2010), pp. 287–300.

[3] C. Backes, A. Rurainski, G. W. Klau, O. Müller, D. Stöckel, A. Gerasch, et al., *An integer linear programming approach for finding deregulated subgraphs in regulatory networks*, Nucleic Acids Research, 40 (2012).

[4] O. Banerjee, L. El Chaoui, and A. d'Aspremont, *Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data*, Journal of Machine Learning Research, 9 (2008), pp. 485–516.

[5] A. Barabasi, *Network Science*, Cambridge University Press, 2016.

[6] A. Barabasi and R. Albert, *Emergence of Scaling in Random Networks*, Science, 286 (1999), pp. 509–512.

[7] T. Barrett, T. Suzek, and D. B. T. et al., *Ncbi geo: mining millions of expression profiles—database and tools*, Nucleic Acids Res., 33 (2005), pp. D562–D566.

[8] S. Basu, A. Shojaie, and G. Michailidis, *Network Granger causality with inherent grouping structure*, Journal of Machine Learning Research, 16 (2015), pp. 417–453.

[9] R. Ben-Hamo and S. Efroni, *Gene expression and network-based analysis reveals a novel role for hsa-miR-9 and drug control over the p38 network in glioblastoma multiforme progression*, Genome Medicine, 3 (2011).

[10] J. Bilmes and C. Bartels, *A Review of Graphical Model Architectures for Speech Recognition*, Graphical Models, (2005), pp. 1–23.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 3 (2010), pp. 1–122.

[12] P. Breheny and J. Hunag, *Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors*, Stat Comput., 25 (2015), pp. 173–187.

[13] W. Chang, J. Luraschi, and T. Mastny, *profvis: Interactive Visualizations for Profiling R Code*, 2020. R package version 0.3.7.

[14] L. Chen, J. Xuan, R. B. Riggins, R. Clarke, and Y. Wang, *Identifying cancer biomarkers by network-constrained support vector machines*, BMC Systems Biology, 5 (2011).

[15] H. Y. Chuang, E. Lee, Y. T. Liu, D. Lee, and T. Ideker, *Network-based classification of breast cancer metastasis*, Molecular Systems Biology, 3 (2007), pp. 1–10.

[16] F. R. K. Chung, *Spectral graph theory*, in CBMS Regional Conference Series in Mathematics, 1997.

[17] G. Csardi and T. Nepusz, *The igraph software package for complex network research*, InterJournal, Complex Systems (2006), p. 1695.

[18] P. Danaher, *JGL: Performs the Joint Graphical Lasso for Sparse Inverse Covariance Estimation on Multiple Classes*, 2018. R package version 2.3.1.

[19] P. Danaher and D. M. Witten, *The joint graphical lasso for inverse covariance estimation across multiple classes*, Journal of the Royal Statistical Society. Series B: Statistical Methodology, 76 (2015), pp. 373–397.

[20] D. de Canditiis and A. Guardasole, *Learning Gaussian Graphical Models by symmetric parallel regression technique*, in MASCOT2018 IMACS Series in Computational and Applied Mathematics, 2018.

[21] M. Dehmer, L. A. Mueller, and F. Emmert-Streib, *Quantitative network measures as biomarkers for classifying prostate cancer disease states: A systems approach to diagnostic biomarkers*, PLoS ONE, 8 (2013), pp. 2–9.

[22] A. Dempster, *Covariance Selection*, Biometrics, 28 (1972), pp. 157–175.

[23] W. Deng, K. Zhang, S. Liu, P. X. Zhao, S. Xu, and H. Wei, *JRmGRN: Joint reconstruction of multiple gene regulatory networks with common hub genes using data from multiple tissues or conditions*, Bioinformatics, 34 (2018), pp. 3470–3478.

[24] S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber, *Biomart and bioconductor: a powerful link between biological databases and microarray data analysis*, Bioinformatics, 21 (2005), pp. 3439–3440.

[25] F. Emmert-Streib, M. Dehmer, and B. Haibe-Kains, *Gene regulatory networks and their applications: Understanding biological and medical problems in terms of networks*, Frontiers in Cell and Developmental Biology, 2 (2014), pp. 1–7.

[26] J. Fan and J. Lv, *Sure independence screening for ultrahigh dimensional feature space*, Journal of the Royal Statistical Society. Series B: Statistical Methodology, 70 (2008), pp. 849–911.

[27] A. Farasat, A. Nikolaev, S. N. Srihari, and R. H. Blair, *Probabilistic graphical models in modern social network analysis*, Social Network Analysis and Mining, 5 (2015), pp. 1–18.

[28] K. Fortney, W. Xie, M. Kotlyar, J. Griesman, Y. Kotseruba, and I. Jurisica, *NetwoRx: Connecting drugs to networks and phenotypes in Saccharomyces cerevisiae*, Nucleic Acids Research, 41 (2013), pp. 720–727.

[29] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics, 9 (2008), pp. 432–441.

[30] ——, *Applications of the lasso and grouped lasso to the estimation of sparse graphical models*, technical report, (2010).

[31] J. Friedman, T. Hastie, and R. Tibshirani, *glasso: Graphical Lasso: Estimation of Gaussian Graphical Models*, 2019. R package version 1.11.

[32] X. Gao, W. Shen, C. M. Ting, S. C. Cramer, R. Srinivasan, and H. Ombao, *Modeling brain connectivity with graphical models on frequency domain*, arXiv, (2018).

[33] S. Ghosh and A. Basu, *Network medicine in drug design: implications for neuroinflammation*, Drug Discovery Today, 17 (2012), pp. 600–607.

[34] C. Giraud, *Introduction to High-dimensional Statistics*, Chapman and Hall/CRC, 2015.

[35] C. S. Greene, A. Krishnan, A. K. Wong, E. Ricciotti, A. Rene, D. S. Himmelstein, et al., *Understanding multicellular function and disease with human tissue-specific networks*, Nat Genet., 47 (2015), pp. 569–576.

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

[37] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity*, Chapman and Hall/CRC, 2015.

[38] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, et al., *STRING 8 – a global view on proteins and their functional interactions in 630 organisms*, Nucleic acids research, 37 (2009).

[39] H. Jiang, X. Fei, H. Liu, K. Roeder, J. Lafferty, L. Wasserman, X. Li, and T. Zhao, *huge: High-Dimensional Undirected Graph Estimation*, 2020. R package version 1.3.4.1.

[40] P. Jie, W. Pei, Z. Nengfeng, and Z. Ji, *Partial Correlation Estimation by Joint Sparse Regression Models*, Journal of the American Statistical Association, 104 (2009), pp. 735–746.

[41] M. Kanehisa, M. Furumichi, Y. Sato, M. Ishiguro-Watanabe, and M. Tanabe, *KEGG: integrating viruses and cellular organisms*, Nucleic Acids Res., 49 (2021).

[42] M. Kanehisa and S. Goto, *KEGG: kyoto encyclopedia of genes and genomes*, Nucleic Acids Res., 28 (2000).

[43] Y. Kim, J. Hao, Y. Gautam, T. B. Mersha, and M. Kang, *DiffGRN: Differential gene regulatory network analysis*, International Journal of Data Mining and Bioinformatics, 20 (2018), pp. 362–379.

[44] D. C. Koboldt, R. S. Fulton, M. D. McLellan, H. Schmidt, J. Kalicki-Veizer, J. F. McMichael, et al., *Comprehensive molecular portraits of human breast tumours*, Nature, 490 (2012), pp. 61–70.

[45] W. L. Lambiv, I. Vassallo, M. Delorenzi, T. Shay, A.-C. Diserens, A. Misra, et al., *The wnt inhibitory factor 1 (wif1) is targeted in glioblastoma and has a tumor suppressing function potentially by induction of senescence*, Neuro-Oncology, 13 (2011).

[46] S. L. Lauritzen, *Graphical Models*, Oxford Science Publications, 1996.

[47] J. Loscalzo, A.-L. Barabási, and E. K. Silverman, eds., *Network Medicine Complex Systems in Human Disease and Therapeutics*, Harvard University Press, 2017.

[48] J. Ma and G. Michailidis, *Joint structural estimation of multiple graphical models*, Journal of Machine Learning Research, 17 (2016), pp. 1–48.

[49] A. McDavid, R. Gottardo, N. Simon, and M. Drton, *Graphical models for zero-inflated single cell gene expression*, Annals of Applied Statistics, 13 (2019), pp. 848–873.

[50] L. Meier, S. Van De Geer, and P. Bühlmann, *The group lasso for logistic regression*, Journal of the Royal Statistical Society. Series B: Statistical Methodology, 70 (2008), pp. 53–71.

[51] N. Meinshausen and P. Bühlmann, *High-dimensional graphs and variable selection with the Lasso*, Annals of Statistics, 34 (2006), pp. 1436–1462.

[52] O. Mersmann, *microbenchmark: Accurate Timing Functions*, 2019. R package version 1.4-7.

[53] K. Mohan, P. London, M. Fazel, D. M. Witten, and S. I. Lee, *Node-based learning of multiple Gaussian graphical models*, Journal of Machine Learning Research, 15 (2014), pp. 445–488.

[54] A. Murat, E. Migliavacca, T. Gorlia, W. L. Lambiv, T. Shay, M.-F. Hamou, et al., *Stem cell-related "self-renewal" signature and high epidermal growth factor receptor expression associated with resistance to concomitant chemoradiotherapy in glioblastoma*, Journal of Clinical Oncology, 26 (2008).

[55] M. C. Nascimento and A. C. P. L. F. de Carvalho, *Spectral methods for graph clustering – A survey*, European Journal of Operational Research, 211 (2011), pp. 221–231.

[56] F. Novomestky, *matrixcalc: Collection of functions for matrix calculations*, 2012. R package version 1.0-3.

[57] J. Peng, P. Wang, N. Zhou, and J. Zhu, *Partial correlation estimation by joint sparse regression models*, Journal of the American Statistical Association, 104 (2009), pp. 735–746.

[58] S. Richardson, G. C. Tseng, and W. Sun, *Statistical Methods in Integrative Genomics*, Annu Rev Stat Appl., 3 (2016), pp. 181–209.

[59] I. Rish and G. Y. Grabarnik, *Sparse Modelling: Theory, Algorithms and Applications*, Chapman and Hall/CRC, 2015.

[60] A. J. Rothman, P. J. Bickel, E. Levina, and J. Zhu, *Sparse permutation invariant covariance estimation*, Electronic Journal of Statistics, 2 (2008), pp. 494–515.

[61] N. Simon and R. Tibshirani, *Standartization and the group lasso penalty*, Statistica Sinica, 22 (2012), pp. 983–1001.

[62] N. Sulaimanov, S. Kumar, F. Burdet, M. Ibberson, M. Pagni, and H. Koeppl, *Inferring gene expression networks with hubs using a degree weighted Lasso approach*, Bioinformatics, 35 (2019), pp. 987–994.

[63] L. Sun, A.-M. Hui, Q. Su, A. Vortmeyer, Y. Kotliarov, S. Pastorino, et al., *Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain*, Cancer Cell, 9 (2006).

[64] K. M. Tan, P. London, K. Mohan, S. I. Lee, M. Fazel, and D. M. Witten, *Learning graphical models with hubs*, Journal of Machine Learning Research, 15 (2015), pp. 3297–3331.

[65] J. J. Tu, L. Ou-Yang, H. Yan, X.-F. Zhang, and H. Qin, *Joint reconstruction of multiple gene networks by simultaneously capturing inter-tumor and intra-tumor heterogeneity*, Bioinformatics, 36 (2020), pp. 2755–2762.

[66] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, Springer, New York, fourth ed., 2002. ISBN 0-387-95457-0.

[67] H. Wickham, *Advanced R*, Chapman and Hall/CRC, 2019.

[68] H. Wickham, P. Danenberg, G. Csárdi, and M. Eugster, *roxygen2: In-Line Documentation for R*, 2020. R package version 7.1.1.

[69] H. Wickham, J. Hester, and W. Chang, *devtools: Tools to Make Developing R Packages Easier*, 2020. R package version 2.3.2.

[70] D. M. Witten, J. Friedman, and N. Simon, *New Insights and Faster Computations for the Graphical Lasso*, Journal of Statistical Software, 33 (2010), pp. 1–22.

[71] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, Journal of the Royal Statistical Society. Series B: Statistical Methodology, 68 (2006), pp. 49–67.

[72] ——, *Model selection and estimation in the Gaussian graphical model*, Biometrika, 94 (2007), pp. 19–35.

[73] X.-F. Zhang, *JEGN: JEGN*, 2020. R package version 1.0.1.

[74] X.-F. Zhang, L. Ou-Yang, T. Yan, X. T. Hu, and H. Yan, *A Joint Graphical Model for Inferring Gene Networks Across Multiple Subpopulations and Data Types*, IEEE Transactions on Cybernetics, 51 (2021), pp. 1043–1055.

[75] H. Zhao and Z. H. Duan, *Cancer genetic network inference using gaussian graphical models*, Bioinformatics and Biology Insights, 13 (2019).

[76] T. Zhao, H. Liu, K. Roeder, J. Lafferty, and L. Wasserman, *The huge package for high-dimensional undirected graph estimation in R*, Journal of Machine Learning Research, 13 (2012), pp. 1059–1062.

[77] Y. Zhao, C. Chang, and Q. Long, *Knowledge-Guided Statistical Learning Methods for Analysis of High-Dimensional Omics Data in Precision Oncology*, JCO Precision Oncology, (2019), pp. 1–9.

[78] W. Zhong, *SMUT: Multi-SNP Mediation Intersection-Union Test*, 2019. R package version 1.1.

[79] X. Zhou and X. Cai, *Inference of differential gene regulatory networks based on gene expression and genetic perturbation data*, Bioinformatics, 36 (2020), pp. 197–204.

[80] Y. Zhu, X. Shen, and W. Pan, *Structural pursuit over multiple undirected graphs*, Journal of the American Statistical Association, 109 (2014), p. 1683–1696.

[81] H. Zou and R. Li, *One-step sparse estimates in nonconcave penalized likelihood models*, Annals of Statistics, 36 (2008), pp. 1509–1533.