Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

Learning Interpretable Gene Regulatory Networks via Merging Bayesian Networks

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial

AUTOR: Nikolas Bernaola
TUTORES: Concha Bielza y
Pedro Larrañaga

Julio 2019

Acknowledgements

To Mario for his incalculable help in implementing FGES-Merge and his work on the Neurosuites visualization tool with which most images in this work were made. Also to Sergio for his contributions to the visualization tool.
To my professors, Pedro and Concha, for their help and guidance through the year.
To all my lab mates at CIG, Bojan, Fernando, David, David, Gabriel and Mario for their help and their company which kept me sane and solved a lot of the problems I kept having.
This project has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under Specific Grant Agreement No. 785907 (HBP SGA2).

Resumen

Nuestro trabajo empieza con la necesidad de reconstruir una red de regulacion genética para el genoma humano usando datos del cerebro. Para conseguirlo, estudiamos el problema biológico de como aprender una red de regulación genética y revisamos la literatura para ver cuales son los metodos mas populares para resolver este problema, junto con sus ventajas y limitaciones. Al final, decidimos que el metodo que mejor se ajusta a nuestras necesidades son las redes bayesianas, sobre todo por su interpretabilidad. En este trabajo presentamos un nuevo algoritmo, *FGES-Merge*, capaz de aprender la estructura de una red de regulación genética mediante la unión de varias redes bayesianas aprendidas localmente alrededor de cada uno de los genes, utilizando una variante del *Fast Greedy Equivalence Search* (FGES). El método es competitivo con el estado del arte en su capacidad de recuperar la estructura original y ademas es mucho más rápido y escala a decenas de miles de variables. Tambien presentamos una solución al problema de inferencia para redes bayesianas con miles de variables. *FGES-Merge* y la herramienta de inferencia estan disponibles publicamente en Neurosuites y pueden ser utilizadas por la comunidad de biología para guiar su investigación hacia las interacciones entre genes que nuestro modelo predice.

Abstract

Our work was motivated by the need of learning a genome-wide regulatory network for the human brain from the Allen Human Brain Atlas dataset. To achieve this, we studied the biological problem and we reviewed the literature for different methods for learning gene regulatory networks, noting their advantages and limitations. We decided to use Bayesian networks because of their interpretability and so we present a new method for learning the structure of gene regulatory networks via merging of locally learned Bayesian networks, based on the Fast Greedy Equivalent Search algorithm. The method is competitive with the state of the art in recall of the true structure while also being much faster and scaling up to the tens of thousands of variables. We also solved the problem of inference for large numbers of variables in Bayesian networks. Both the structure learning algorithm and the inference tool are available publicly at Neurosuites and can be used to guide biological research by testing new gene interactions that are predicted with high confidence by the model.

Contents

Chapter 1

Introduction

1.1 Introduction

Since the advent of high-troughoutput measurement technologies in biology in the 1990s, like in-situ hybridization[1] [2] or RNA microarrays [3], it has been possible to collect information for tens of thousands of genes from every tissue sample or even at the level of a single cell. Since most of the information about the development and function of every living being is codified in its genome, we can study the level of expression of each gene in different conditions. This makes it possible to reconstruct the underlying regulatory relationships between the genes and, therefore, understand their function [4][5]. Due to the combinatorial nature of gene regulation [6] and the size of the genome, it would be intractable to experimentally determine all of the regulatory links. To solve this problem, many computational methods have been proposed to infer the gene regulatory network (GRN) from expression data. The models learned can then be used to guide biological research by letting researchers test the interactions predicted by the net.

1.2 Motivation

The main objective of this work is to present an algorithm capable of reconstructing the GRN for the whole human genome using gene expression data from the human brain to obtain a model that is accurate, easily interpretable and capable of quantitative prediction. To this end we decided to use Bayesian networks (BNs) which are a very powerful model backed by probability theory that is also very intuitive to understand. Their main limitation is the computational cost of learning them and so being able to scale them to the massive sizes required for full genome networks without using massive high-performance computing resources would be a boon for biological research.

1.3 Objectives

The main goal is to create a BN that represents the relationships between all protein-coding genes in the human brain. The algorithm that generates this BN should have a reasonably good accuracy that is comparable with the state of the art. It should also be able to learn the network in a reasonable time without using high-performance computing and should be capable of performing inference in real time.

To achieve this we first studied the literature to see the advantages and limitations of different

methods to learn GRNs. Then we decided that BNs had the properties we wanted if we could scale the learning algorithms to the required number of variables. In addition, we present an algorithm, the FGES-Merge, based on the Fast Greedy Equivalence Search (FGES) [7] that solves the usual problems that BNs have when dealing with very big networks. Finally, we tested the algorithm in a popular benchmark for the problem to fine tune it and then apply it to human brain data and obtain regulatory networks for the full brain.

1.4 Structure of this document

Chapter 2 starts with a review of the literature in which we first discuss the biological background required to understand gene regulation and pose the problem of learning GRNs. Then we review different methods of learning with a summary of their advantages and limitations. Finally we introduce BNs, their theoretical background and applications and review their application to the field of GRNs focusing mainly on the limitations that need to be overcome.

Chapter 3 defines the problems we want to solve: scaling a learning algorithm to the required number of variables and being able to compare networks derived from different conditions.

Chapter 4 presents our proposed solution, reviewing the limitations of previous learning algorithms for BNs and how we overcome then to achieve our desired goal.

Chapter 5 shows our results when using our algorithm with a common benchmark for GRNs to compare our accuracy with other methods. We also compare ourselves in the time we require to solve the problem against other BN structure learning algorithms. Finally we present the networks we obtain when applying our algorithm to real human brain data.

Chapter 6 concludes the work with some final remarks on our algorithm, the results obtained and some ideas on what to improve and some further research that could be done.

Chapter 2

Literature Review

2.1 Literature review

In this chapter we will introduce the biological background to the problem of reconstructing GRNs from data. Then we will briefly summarize some of the most important methods for learning GRNs with their main advantages and limitations. Finally, we will introduce BNs as the model of choice for the problem at hand, discuss the theoretical background of the model, previous work done with BNs in the field and the main limitations we need to overcome to solve the problem of inferring a full genome network from data for the human brain.

2.1.1 Genetics

One of the most important scientific advancements of last century was the discovery that all heritable information in a living organism is encoded biochemically and stored in chromosomes, very long polymers of double stranded, helical DNA [8]. Information stored in DNA can be dynamically read in a process which is also biochemically consistent among most species. One of the DNA strands gets transcribed into RNA, a single stranded polymer of nucleic acids, which acts as an information carrier which gets in turn translated into proteins in ribosomes. Proteins are polymers of amino-acids which depending on the folding structure can carry out almost all cellular functions. The flow of information through the cell is of the uttermost importance in biology and has been names the *central dogma of molecular biology* [9] 2.1 One of the most important facts about gene expression is that every RNA strand encodes one, and only one, protein. This is because every triplet of nucleic acids in the RNA strand maps to a unique amino-acid in a way that is consistent across species. This mapping is called the *genetic code*. [10]

The full complexity of every living organism has to arise from interactions between these biochemical components. Therefore, there is a strong scientific interest in understanding these interactions both to improve our fundamental understanding of how life arises from biochemistry and because of the important practical applications in medicine and drug manufacturing. The main problem is that probing live cells to measure interactions directly is incredibly difficult so studying them directly is almost out of the question. However, thanks to the unique mapping between RNA strands and proteins, we can substitute the questions about protein interactions with questions about the concentration of the messenger RNA (mRNA) that encodes for each of the proteins. And, thanks to the advances in high-throughoutput sequencing technology during the last two decades, measuring the abundances of different biochemical components (including mRNA) is much more manageable so it is possible to do it at a very large scale for relatively cheap. The abundances

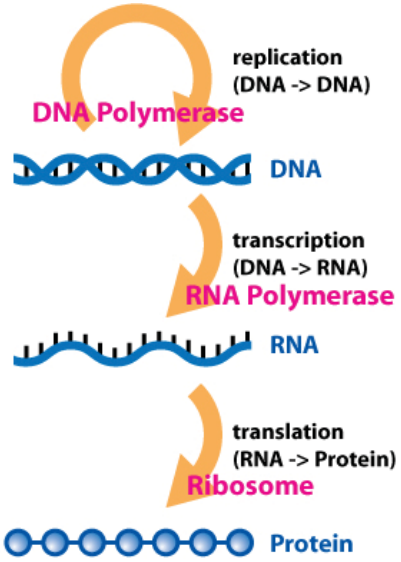


Figure 2.1: Flow of information in a cell. Diagram explaining the Central Dogma of Molecular Biology.

don't give us the full picture of the interactions, but this trend in the availability of data has given a powerful motivation to try to reconstruct the interaction structure that underlies gene expression computationally. These structures are called GRNs and their reconstruction is one of the central efforts in the field of Systems Biology.

Data collection

As was mentioned in the previous section, direct measurements of interactions at the cellular level are almost impossible and direct measurement of the protein products requires a very complex analysis pipeline which makes it worse than using transcriptomic measurements [11]. Micro arrays and in-situ hybridization are the most common methods to collect data. Microarrays are an older method that works by lining a chip with microprobes that puncture a biological sample and take a sample of cytoplasm. Each probe in the array is lined with the complementary chain to the RNA we want to detect (a different one in each probe) in such a way that after washing out the array, only the bound RNA will be found in each probe. The bonding strands are made as to induce a fluorescent molecule to emit light when bound so that each probe will emit light corresponding to the amount of binding RNA found in the sample. By a process of calibration, the level of light measured at each point on the chip can be mapped to a concentration level for the bound gene. The main limitations of microarrays are that we can only measure known transcripts (since the complementary strand has to be designed onto the chip) and that they give a measurement of a population of cells in the tissue sample and since there might be many different cell types in the population the measurement will not be representative of any one of them.
In-situ hybridization (ISH) is a newer method that consists in taking a sample of cytoplasm from a single cell, filtered to keep only mRNA, reverse transcribed into DNA. Finally, a PCR process is used to amplify the amount of DNA in the sample up to a level where it can be more precisely measured. By knowing how much the amplification factor is we can then estimate the original level of RNA in the cell. This process is much more precise and can give measurements for single cells which makes it much more interesting since a single tissue sample used for microarrays can have multiple types of cells each of which would have different levels of expression for each gene.

So microarrays give an average expression level for the tissue sample while ISH can give accurate, single-cell measurements [12].
The dataset we used was the Allen Human Brain Atlas dataset [13], which was created using microarrays. Although there was also some ISH data available it was not complete and we decided not to include it to avoid problems of mixing two data sources (specially since we did not have ISH data for all areas of the brain)

Genetic regulation

The processes that regulate which genes are expressed and how much of the protein is synthesized are the topic of study of genetic regulation. In general we are interested in seeing how different environmental conditions (internal or external) change the amount by which each protein is synthesized. Gene expression can change due to hormonal responses, any physical or chemical changes in the external environment, internal chemical changes, or the expression of other genes. The whole process of gene regulation is thus combinatorial, taking into account many factors for each gene. The process is dynamic since it is always responding to environmental changes and feedback (usually negative), and stochastic, since even in the same conditions we can only say that the level of expression will also be similar due to the imperfections of biological processes and the inherent instability of chemical processes.

Although the most correct model we have for the behaviour of gene expression is as a system of stochastic partial differential equations [14] this model is generally intractable for the study of even the simplest of organisms if we are interested in more than a few tens of genes. One common way of simplifying the model onto one which is usually good enough is to assume that all regulations can be modeled as genetic interactions. This means that we assume that any non-genetic factor (hormones, temperature, chemical changes in the environment...) will not have an effect on the level of expression of genes except indirectly, via mediation by another gene. This way, we can eliminate all external factors from our model and be left with only interactions between genes. This network structure of interactions is the GRN. In this work we will make the further very common assumption of taking the steady-state expression level, this means we will work with static instead of dynamic models.

2.1.2 GRNs and their properties

As we saw in the last section, we have ways to collect massive amounts of data that can be used to reverse engineer the structure of the GRN. We are interested in both the topology of the network to be able to see the interactions between genes but we are also interested in being able to accurately predict changes in the level of expression of some genes given changes in the level of other genes. In this section, we will briefly discuss the mathematical formalism required to properly define these GRNs, some of the most important biological properties that can be used to constraint the space of possible structures and the multiple available methods for learning them. Finally, we will discuss why we have chosen BNs as our prefered method, review the work done on using BNs for GRNs, address their limitations and show how we have solved them with our new method. For a more in depth review of these methods we recommend [5].

Notation

For the representation of a GRN, we will use a directed or undirected graph $\mathcal{G}$ (depending on the method). $\mathcal{G}$ is a pair (V, E), where V is the finite set of vertices or nodes indexed by $\mathcal{I}$ and E is a subset of $\mathcal{I} \times \mathcal{I}$, with element (i, j) indicating an edge between nodes i and j. If the network is undirected, then the set of edges is symmetric under swapping the indices of its members, that is $(i, j) \in E \iff (j, i) \in E$.

In the context of GRNs, the set of nodes always represents the level of expression of genes with each node associated to one gene. Edges are interpreted as relationships between genes, but the precise definition of the relationship will depend on the mathematical model being used. Nevertheless, the topological structure of the network is useful by itself as it gives a visual intuition of the interactions at play. Some things we can know just from the structure are the presence or absence of hubs, which are nodes with higher than average number of edges attached; the density of the network, which is the ratio of edges per node; or calibrating with known relationships by checking if edges exist between the nodes they should.

Most models will add more structure to the network, both to the nodes (usually a base level of expression, but sometimes more information, i.e., the standard deviation in a Gaussian BN 4.2.1) and to the edges (i.e., regression coefficients, correlations...). This information will be used to predict the changes in level of expression along the network and to estimate the strength of relationships between genes.

We will be learning the GRNs from a dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\} \in \mathbb{R}^{G \times N}$ where $\mathbf{x}^{(i)} = (x_1^{(i)}, ..., x_G^{(i)})$, with N the number of measurements and G the number of genes.

Topological properties of GRNs

Some of the most important information we can extract from the topology of the network is the degree distribution which is given by the number of edges attached to each node of the network. In the case of directed networks we can further distinguish between the in-degree and out-degree, which relate to the number of edges going into or out of each node respectively. This information can be compared to the known properties of real GRNs to see if the methods are working correctly or the information can be used beforehand to restrict the space of structures that will be searched over.
For a more in depth overview of topological properties of GRNs see [6], [15] and [16]. We will summarize them here:

- Locally dense but globally sparse: GRNs have a small number of edges compared with the maximum possible. Let n be the number of genes in the network. Then, the maximum number of edges would be n^2 (if we allow for self-edges). The actual number of edges in GRNs varies depending on the network but is $O(n)$ normally with a small ($\simeq$ 10) constant. This means that the network is sparse. However, the degree distribution is very fat-tailed, so that instead of finding that most nodes have one edge, we find many with no edges and we find hubs with many edges. This means that, connected components of the network are dense but there are many disconnected components which makes the network sparse. This translates to a limited number of edges in the network and although it seems like it should make the problem more tractable it is very problematic for many methods of learning that require sparsity, since, to the best of our knowledge, they usually require local sparsity.

- The in-degree distribution is a Laplace distribution with an upper limit: The in-degree of a node in a GRN is the number of regulators a gene has. This number is usually small since most genes require just one regulatory factor, although it can be higher in more complex processes that require multiple things at the same time. However, there is a physical restriction to the number of regulators. In the case of transcription factors, the only way they can affect the expression of a gene is to be physically close to that gene and affect the DNA directly. Since there is limited space around each gene, there can only be a limited number of transcription factors and thus, a limited number of regulators. This number is lower in prokaryotes than in eukaryotes, since there are other mechanisms in eukaryotes that can be affected from a distance (i.e. hystone coiling). This translates to an upper bound in the number of parents for each node.

- The out-degree distribution is scale-free distributed at the tail: As mentioned before, we can find many more hubs with more edges going out than would be expected from a Laplace distribution. This means that these hubs are regulators of transcription and that they are used in many processes at the same time. Biologically, this is due to using hubs being more evolutionary stable. Any useful adaptation can be constructed on top of already existing regulatory machinery which is easier than having a new regulatory network emerge simultaneously with the new adaptation. The more processes pile up on the same regulatory hub, the more likely it is that any deleterious mutation to the hub will kill the individual early instead of just making the process unusable. This makes it so that there are less points of failure for the genes in the process. (From the point of view of the gene, individuals that die early matter much less than individuals that don't carry a copy of the gene since dead individuals mostly don't compete for resources.)

2.1.3 Methods for learning GRNs

Now that we have discussed some basics of the structure and notation used in the field of learning GRNs we will dedicate this section to summarizing the different approaches there are for the task of reconstruction with a brief explanation of some of the most important methods and current work. We will also emphasize the advantages and disadvantages of each method and explain why, in the end, we decided to work with BNs. We will not adress some more complex methods that involve combinations of multiple approaches or methods for modelling dynamic GRNs, a recent review of these methods can be found in [17].

Basic statistical methods

This first group of methods is based on basic statistical methods that can be used to measure dependencies between variables. They start from a fully connected network and then associate a weight to each edge. The output can then be thresholded to try to get a reasonable approximation of the topology of the network. The main advantage of these methods is that they use very common statistical techniques and so are readily available and computationally cheap. They are also reasonably accurate in finding the network topology which makes them some of the most popular.

Correlation networks This method is based on calculating the pairwise Pearson correlation (although other measures are possible) between all pairs of genes. For two genes, $\mathbf{x}_i, \boldsymbol{x}_j$, the pairwise Pearson correlation is given by:

$$\rho_{ij} = \frac{\mathbb{E}\left[(\mathbf{x}_i - \mathbb{E}(\mathbf{x}_i))(\mathbf{x}_j - \mathbb{E}(\mathbf{x}_j))\right]}{\mathrm{Var}\mathbf{x}_i \mathrm{Var}\mathbf{x}_j} \tag{2.1}$$

where the norm is the Euclidean norm. For a dataset with N measurements and G genes, we have a data matrix $D \in \mathbb{R}^{N\times G}$. Then we compute the pairwise correlation between the columns of D to obtain a correlation matrix, $C \in \mathbb{R}^{G\times G}$. This matrix is used to assign a weight to each of the edges of the network. Then, we can apply a threshold which will depend on the level of sparsity we want to obtain the GRN structure. Since correlation is symmetric, the network will be undirected. The advantages of this method are that the complexity is $O(N)$ and $O(G^2)$ [Complexity of correlation] which makes it very popular for genome-wide or other massive studies. The underlying biological assumption that interacting genes should have correlated expressions is plausible and correlation methods are consistently reliable [18]. The main limitations of correlation networks are that they fail to distinguish between direct and indirect regulation, they don capture non-linear interactions well, they are undirected and they are not predictive. They can't distinguish between direct and indirect regulation since if gene A regulates gene B which in turn regulates gene C, it is very likely that the correlation between A and C will be high. This is worsened by the presence

of hubs in GRNs which make all their children correlated to each other and to them so that it is really hard to discern which of them is the true regulator. Since Pearson correlation is linear, it cannot capture more complex types of interaction but this is usually not a problem in practice. The lack of direction makes it impossible to distinguish regulator from regulated without other expert knowledge. Finally, they are not predictive since correlation is just a statistical measure of association and so it cannot be used to make quantitative predictions about expression levels.

Mutual information networks As a way to relax the assumption of linearity implicit in correlation methods some groups have considered an alternative measure based on information theory. They use mutual information in the same way correlation was used in the previous method. Let X and Y be two discrete random variables and let $P(X, Y)$ be their joint probability distribution. The mutual information between the two random variables is defined as:

$$\begin{aligned} \mathrm{MI}[X, Y] &= \sum_{x_i, y_i} P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)} \\ &= \sum_{x_i, y_j} P(x_i, y_j) \log \frac{P(x_i|y_j)}{P(x_i)} \end{aligned} \tag{2.2}$$

Where x_i, y_j are the values that X and Y can take, $P(X), P(Y)$ are the marginal probability distributions for X and Y and $P(X|Y)$ is the conditional probability distribution of X given Y. Mutual information is zero when two variables are independent.
The main idea is to estimate the probability distributions from the data and then calculate the mutual information for each pair of genes. The resulting mutual information matrix gives a score to each edge in the fully connected network which will be undirected. This can be thresholded to obtain a so called relevance network. [19] Some of the most popular methods are ARACNE [20], CLR [21] and MRNET [22].
The advantage of MI networks is that they are almost as computationally cheap as correlation networks while being able to capture non-linear relationships. The main drawbacks are that, again, they don't offer a predictive framework, that the estimation of the probability distributions will be highly sensitive to noise in the samples when the sample size is small and that they overestimate the relationships since they can't distinguish between direct or indirect regulation.

Regression networks The previous methods used basic statistic measures to compute the dependencies between genes. A different way to approach the problem is to try to predict the expression level of one gene given the rest. One obvious way to do this is with a regression model, the simplest of which would be a linear regression model. In the context of GRN the model would be learnt by regressing each gene in turn against all others and the coefficient for each gene in the regression would be used as the weight for the edges of the network. That is, for every gene x, we would have an expresion level x_i for sample i given by:

$$x_i = \sum_{j \neq i \in G} w_{ji} x_j + \epsilon_i \tag{2.3}$$

where ϵ_i is a noise term and solving the regression problem would give the $w_j i$ associated to the edge between genes x_j and x_i. Note that unlike previous methods, regression based networks are directed (and can even have bidirectional edges). As in the other methods, a threshold can be applied to the weights of the edges to prune the network.

Regression based methods are generally very powerful since they give both a structure for the network and a model that can predict gene expression. They were considered to be the state of the art in the last DREAM (Dialogue for Reverse Engineering of Models) challenge [23]. In particular, TIGRESS [**?**] uses a linear regression method with L1 regularization to force some of the $w_j g$ to

be zero which avoids using an arbitrary threshold. Another variation on this theme is GENIE3, which was also used in the DREAM challenge, uses random forest regression to make the method more flexible and non-parametric. [24][25][26].

Regression models are only slightly more computationally intensive than the previous methods and can, theoretically, capture higher order conditional dependencies between genes. In practice, however, regression models tend to fail with limited data, since the highly correlated structure of gene expression makes it so that regression networks (even when regularized) give spurious results in the same way as the methods discussed above.

Deep learning methods

Nowadays we can also find deep learning methods to outperform linear regression based methods for inferring gene expression [27]. However there are two downsides to these approaches: they require a huge amount of data, which is not always easily available and they don't give the structure of the network, just a black box approach to the inference problem. Since we are interested in the structure of the network and don't have an abundance of data for the problems we wish to work on, we don't use deep learning methods.

Probabilistic models

The models above work by defining some measure of dependency in a pairwise or all-to-one way. However, none of them define a probabilistic model of the data explicitly (although there is an implicit model in the regression). In this section we briefly introduce Gaussian graphical models and then introduce and review the work done with BNs which will be the main focus of the article afterwards.

Gaussian graphical models One of the simplest probabilistic models we can consider is a multivariate normal distribution. The probability density distribution for a multivariate normal vector $\mathbf{x} \in \mathbb{R}^G$ is given by:

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}} \exp -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) \tag{2.4}$$

where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix.
Gaussian models give us the whole power of probabilistic inference, which allows us not only to make predictions about gene expression but to quantify our uncertainty. Furthermore, they have a very important property in that the inverse of the covariance matrix, the precision matrix $\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}$, contains the *partial correlations* between the entries in $\boldsymbol{x}$. The partial correlation is the residual correlation between two variables once the effect of all other variables has been substracted. Therefore, it is a better measurement of the strength of the relationship between genes and is less vulnerable to making spurious associations due to the highly correlated nature of the GRN.
This result is used by Gaussian graphical models [28] which are learnt by treating the measurements of expression as a multivariate normal random vector and inferring the precision matrix from the samples using maximum likelihood estimation. The number of parameters is of the order of G^2 so regularization techniques are used. Mainly sparse regularization like L1 because they have the benefit of having a topological interpretation to build the network structure, that is, the non-zero entries of the precision matrix are the edges of the underlying GRN.
Gaussian models are generally a very good model that has all the properties we want. It's probabilistic, interpretable and has both a topological and a predictive component. However, it is limited mainly by the fact that it is generally very difficult to estimate a high-dimensional precision matrix from limited data even being theoretically impossible when the number of samples is

less than the number of dimensions of the matrix (although an estimate can be found but with no guarantees of correctness). There is also the assumption of normality for the expression data which implies linearity in the relationships. Although this is a strong assumption we have seen in correlation networks, linear regression networks and Gaussian graphical models that it seems to be a reasonable approximation which is very effective in practice and simplifies computation enormously.

2.2 Bayesian Networks

Every method we have discussed until now uses a top-down approach to the task of learning the topology of the network, in the sense that we start with a fully connected network and prune it afterwards or start with the full joint distribution of all the parameters and then threshold it to make it sparse. The class of methods we will focus on, BNs, have a similar objective to Gaussian models in that they try to build a joint probabilistic model but they approach it in a bottom-up way; building the model up from local conditional parts.

Bayesian networks [29] are probabilistic models that combine probability and graph theory to be able to efficiently represent the probability distribution of a set of variables $\mathcal{X} = \{X_1, X_2, ..., X_n\}$. BNs model probabilistic conditional dependencies and independencies between the variables in $\mathcal{X}$ in terms of directed acyclic graphs (DAGs) and a series of conditional probability distributions (CPDs) [**?**]. Each of the nodes in the graph represents a variable in $\mathcal{X}$ with the arcs representing conditional dependencies between the variables. Each of the CPDs is associated to a variable X_i and gives the probability distribution of that variable conditioned on its parents in the graph, that is, the nodes that have edges directed towards X_i which we call $\text{Pa}(X_i)$. The formula for the joint distribution of the variables given all the CPDs is:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|\text{Pa}(X_i)) \tag{2.5}$$

2.2.1 Structure Learning

The structure of a BN can be learned from a dataset in many ways, which are usually categorized into two different groups:

- Constraint-based These methods treat BNs as a representation of the conditional independencies between triplets of the variables. These class of methods was originated with the Inductive Causation algorithm [30] which uses conditional independence tests to build the structure of the DAG attempting to reflect as accurately as possible the true underlying independencies. [31] [32] [33] Some of the most popular methods in this group are *parents and children*, (PC) [32] and *light mutual min* (LMM) [34]. These methods usually start with a fully connected network and remove edges every time a conditional independence test passes. Most methods try to do independence tests with an increasingly big conditioning set of variables to try to find separating variables for every pair of nodes. This is computationally intractable for a large number of variables and so there is usually a limit to the size of the conditioning set. Furthermore, these methods are very sensitive to false positives in the tests since they can confound the process of construction. This makes the methods not work well when the sample size is small and the number of variables high since it is more likely to make mistakes the more tests it has to run and the less data there is.

- Score and search Score-based methods explore a predefined space of structures with a score function that assigns a score to each of the structures. The objective is to return the structure

that maximizes the score. When the number of nodes is not very big we can enumerate and score all possible networks to return the maximum but the number of possible structures grows super-exponentially with the number of nodes [Robinson, 1977] which implies that searching for the structure with the maximum score is an NP-hard problem [35]. Because of this, the usual way to carry out the search is via heuristic algorithms. Some of the most commonly used scoring functions are the Bayesian information criterion (BIC) [36] [37], the minimum description length principle (MDL) [38], Bayesian Dirichlet equivalence (BDe) [39], Akaike's information criterion (AIC) [40] and K2 [41]. The main advantage of score-based methods is that they take into account the whole structure of the network to score so they are less sensitive to individual edge failures. Furthermore, some search algorithms like the Greedy equivalent search (GES) [42] can be guaranteed to reach the optimal structure. Generally, most algorithms work by doing local changes to the structure of the network and accepting them if they increase the score which means that they avoid the computational cost of having to calculate the whole score for the network at each step. Some limitations of score-based methods are that they cannot separate between all structures with the same optimal score and that they can overfit the dataset (since there is no guarantee that the optimal network for a given score is the true underlying network). One important algorithm which we will discuss in chapter 4 is the fast greedy equivalence search [7] which scales well to thousands of variables or up to hundreds of thousand in sparse networks.

2.2.2 Parameter learning

Learning the parameters of a BN is usually the simplest part. It requires a modelling assumption which gives the type of distribution that each node will have. The most common might be discrete with any umber of states, continuous following a Gaussian or a mixture of Gaussians. Other distributions are possible and, although it's usually the case that all nodes follow the same type of distribution, hybrid networks can be built too. Once the distributions have been chosen we usually just do maximum likelihood estimation for the parameters of the distributions given the dataset, although some of the scores mentioned in the previous section, like K2 and BDe are based on Bayesian approximations with prior distributions over the variables.

2.2.3 Probabilistic inference

BNs contain all information to compute the joint probability distribution of all the variables and any distribution (conditional, marginal or joint) for any subset of the variables. One of the main objectives of building a BN model is precisely to answer any probabilistic queries about the variables. BNs can perform three types of inference:

- Deductive inference Answers questions about the consequences of a given series of events.
- Inductive inference Answers questions about the causes of a given series of events.
- Abductive inference Answers questions about the most probable hypothesis about the state of the variables given a series of events.

More concretely, the most common queries posed to a BN model are:

- Diagnosis and prediction: The inference process used for both inductive and deductive reasoning is probability propagation. Given a series of events (observed values for some of the variables in the model) we calculate the conditional probability distribution of any other subset of variables conditioned on the evidence.
- *Maximum a posteriori* (MAP): An abductive inference problem in which given some evidence we are to find the most likely values for a subset of the variables.

- *Most probable explanation* (MPE): An abductive inference problem in which given some evidence we are to find the most likely values for all the variables for which we have no evidence.

Ideally, we would like our methods of inference to be able to return the exact results for the queries given the structure and CPDs of the network. This type of inference is called *exact inference*. One of the most important methods in this category is the *message passing algorithm* (MP) [43] which is very efficient for evidence propagation but only works if the structure of the network is that of a polytree. This is a very strong restriction which does not generally apply for real world data. Instead, a clustering method is used to group nodes in a general network to give it the structure of a polytree and then apply MP. The intermediate structure is called a *junction tree* [44].
When it's not possible to do exact inference due to the size of the network, we use approximate inference [41]. This reduces the complexity substantially, but will obviously introduce some error in the results. Some of the most important algorithms for approximate inference are *probabilistic logic sampling*,[45] *likelihood weighting* [46] and *Markov Chain Montecarlo* [47].

2.2.4 Applications of Bayesian networks

BNs are models with a very solid theoretical backing and many algorithms to choose from depending on the task at hand. Futhermore, they are easily interpretable since every node maps to a single variable and edges indicate probabilistic dependence. Thanks to all these advantages BNs have been the models of choice for many different applications like medicine [48], sports [49] or neuroscience [50].

In the field of GRNs one of the earliest approaches was by Friedman in [51] which used a simple approach of searching the whole space of structures for the one with the maximum likelihood. We also have Spirtes et al. [52] which used the PC algorithm on microarray data to obtain a GRN. Some more recent advances include [53] which uses a variant of the Chow-Liu algorithm to be able to learn a BN in quadratic time but with a severe limitation on the structure since it must be a tree. In [23], there are several BN based methods of which the best use simulated annealing to add a stochastic element to the reconstruction and average the results to increase their resiliency against possible errors. In [16], the authors use topological information to restrict the space of structures and accelerate the search. Work like [54] presents a paralelized approach to learning a genome-wide network that is implemented in a supercomputer. One of the most recent advances is [55] in which the authors learn small local networks around each node and then combine them.

2.2.5 Advantages and limitations of Bayesian networks

Bayesian networks did not show a good overall performance in the DREAM 5 challenge [23] but their interpretability makes them a good model to try to improve. They readily encode the regulatory network in their graph structure and the way they are built avoids the problem of finding a complete network, like in regression or pairwise methods. This reduces the need to use an arbitrary threshold to cut some edges, since they are taken out in a less arbitrary way by testing for conditional independence and it can capture indirect regulation well. Their probabilistic nature is specially important in this domain because it allows us to run inference through the learned structure which is done by setting some genes as evidences and then querying another genes to check their variability, sometimes helping us view how some genes influence the expression of the rest (although not always since BNs do not generally represent causal influences).

As we have seen above the main disadvantage we face when using these models is that for them to be scalable to genome-wide datasets we require either restrictions on the structure or high-performance computing to deal with genome-wide networks. The fastest method that doesn't use

either is [55] but it's biggest network is orders of magnitude smaller than what we need and their code was not made available so we could not replicate their results. We take the two best BN methods in the DREAM 5 challenge, which use a simulated annealing approach, and the method in [55] which learns and merges local networks as the state of the art for learning GRN. These approaches gave us some hints as to what lines of work to pursue to improve on them.

Chapter 3

Problem statement

3.1 Learning genome-wide regulatory networks for the human brain

The main goal of this work is to find an efficient way of reconstructing full genome regulatory networks and apply it to learning a GRN for the human brain. Full genome networks for humans can have from 20,000 to 50,000 nodes (depending on whether non protein coding genes are considered or not). As discussed in the previous chapter, this size of network is usually very hard to work with due to the computational cost. Not many algorithms scale well to this size and most need high-performance computing resources to be able to deal well with it.
As our dataset we have the Allen Brain Institute Human Brain Atlas of which we used the microarray data. The dataset has measurements for 20,708 protein-coding genes with 3500 samples gathered from different areas of the brain.
We want our network to have some good properties for analysis so that it is useful for biological research. That means we prefer models that offer a predictive framework for gene expression and are easy to interpret.

3.2 Differential analysis

One of the most common uses of genetic data is to test different conditions and see which genes have a different level of expression. This is usually done with simple hypothesis testing to see if both expression levels could be samples from the same distribution or not.
Comparing GRNs built in different conditions would allow for a more nuanced analysis where instead of just saying if two expression levels are different we could quantify how different the expression level is and also see if the relationships to other genes have changed.

Chapter 4

Proposed solution

As is obvious from the title of this work, we chose Bayesian networks as our model to solve the problem. Bayesian networks fulfill all of our criteria of offering a predictive framework, being easy to interpret and being useful for nuanced differential analysis.
In Chapter 2 we discussed some of the main limitations of Bayesian networks and some of them apply here. Mainly, we are dealing with massive networks of 20,000 nodes which are usually intractable to learn and even harder to do inference in. In this Chapter we will review these limitations and discuss how we addressed them with a new algorithm for structure learning that learns smaller local networks and merges them and restricting the CPDs of the network to be linear Gaussian distributions.

4.1 Learning a genome-wide regulatory network

The first problem we need to solve is how to scale learning algorithms so that they can deal with the number of nodes we have in a genome-wide GRN. For this, we combine two main insights. The first is that we use a score-based algorithm based on the FGES algorithm [7] which is an improved version of the greedy equivalence search (GES) [42]. The GES algorithm starts with an empty graph and explores the space of the equivalence classes of DAGs to search for edge additions (or deletions in the second phase) maximizing a decomposable score like the Bayesian Information Criterion (BIC). The equivalence class of DAGs that represent the same probability distribution is called the Markov Equivalence Class (MEC) and can be represented by a *completed partial directed acyclic graphs* (CPDAGs), and it has directed and undirected edges but no directed cycles. A CPDAG is defined as the set of all DAGs G having the same undirected adjacencies as G and the same "v-structures" (i.e. non-shielded colliders) — substructures $X \rightarrow Y \leftarrow Z$, where $X, Y, Z \in G$ and X, Z are not adjacent.

This algorithm learns the structure by maximizing a decomposable score like the BIC. Doing the search in the space of CPDAGs reduces the size of the space and makes the search faster, but the speed-up is not big enough since the space of equivalence classes still grows super exponentially with the number of nodes [56]. The main problem with this approach is that BIC and similar scores need a DAG to be calculated so we need to transform the CPDAG to a DAG and back at every step. Fortunately, this can be easily done since the local nature of the algorithm means that we don't change much of the network on each step (so we only need to work on small parts of the graph) and by using Meek's rules [57] which guarantee that we preserve the v-structures when undirecting the DAG to get the CPDAG.

FGES aims to improve the GES speed by including some modifications over it. The first main

modification consists in caching the possible edges addition scores to reutilize them in the future (or discard them in case of breaking any rule that would result in a graph not being a CPDAG). The second main modification sets up a searching phase where parallelization can be used. This is possible because the scores calculation for every edge addition doesn't depend on the other edges addition for the current graph. If then, some possible edges additions are no longer valid, they are discarded by a check of minimum graph properties to maintain a correct CPDAG.

FGES, like GES, has two main steps, the forward equivalence search phase (FES) and the backward equivalence search (BES) phase. In the first one, it follows a sequential procedure that at each point computes the possible edge additions and adds the single one with the greater BIC score. The procedure finishes when there are no possible arc additions that increase the BIC. Then in the backward phase, the same procedure is done but this time removing one edge at each step, until no more edges deletion can improve the BIC. Finally, the resulting CPDAG is transformed to a specific DAG by removing the cycles and directing the possible edges while not creating any cycle.

Even with FGES, the scale of the problem at hand is still enormous and we need something else to be able to get a solution in a reasonable time. This second insight is that we can divide the large problem into many small ones and then combine the local results as in [55]. This method has no theoretical guarantee of optimality but in practice works well enough and is, to the best of our knowledge, one of the only ways to deal with massive networks in reasonable times. The method consists in creating n local graphs (being n the number of features) and the combining the graphs with the pure union method (i.e. every arc in a local graph will always be in the global graph).
For each feature, a set of the most probable related features to it is selected. This selection is done by computing the pairwise continuous mutual information matrix and then running a hypothesis test for every row that will identify the point of separation between the distribution of the related genes and the not related ones. Once every feature has its set of nodes to learn its local graph, the learning is done by running greedy equivalence search of which FGES is an improvement.

None of these two methods alone, FGES or local networks, could deal with the problem on their own. Ramsey et al [7] claim to be able to scale up to a million variables but we were not able to replicate this even after personal communication with the authors to set up the algorithm. We expect that this is because GRNs are not locally sparse and it seems to be a requirement for FGES to work well. Liu et al. [55] don't try networks of the size we wanted to test and did not release their code so we couldn't replicate their results, however since, to the best of our knowledge, FGES is much faster than GES in all instances we expect that our version combining both will be faster.

4.1.1 FGES-Merge: Our implementation

The original FGES implementation by [7] is not scalable in terms of CPU and memory when the graph is not sparse (when the graphs have more than 60.000 edges it causes out of memory errors for regular computers with 64 GB of RAM storage). On the other hand, the local-to-global method in [55] is scalable but still doesn't have a great speed as it uses a regular greedy score-based method. To combine the best of both worlds, modifications are necessary.

Here we propose the FGES-Merge method that consists in learning n local graphs, one per gene, with a modification of the FGES algorithm. This allows our algorithm to deal with much more dense networks than the original FGES which is necessary to be able to learn the networks for the full human genome in reasonable time. The combination of the local graphs has been also modified to be able to prune the not so relevant edges and uses a simulated annealing approach to induce variation in the arcs that are learned on each local graph. Then, only the arcs that appear consistently are added to the global network thus increasing the robustness of the final result.

As our algorithm learns the graphs by maximizing their BIC score, we run a hypothesis test that separates the BICs for each gene assuming they are drawn from two separate distributions. Then, we use the genes that have been determined to be drawn from the higher scoring distribution assuming they will be more relevant for predicting our target. By using the BICs instead of the mutual information which [55] use, we improve the speed as the initial BIC scores are already calculated and will then be reutilized by the FGES algorithm in each graph. Note that the number of related features for a specific node can still be a large number so it is also possible to limit the maximum number of related features if necessary.

To combine all the local graphs, we propose a modification of the union method used in [55]. The union method has the problem of adding every arc even if it is only present in one of the n local graphs. We implemented a greedy pruning strategy to not join every arc. Once the global graph has been created with the union method, we have the option to run a loop iterating through every local graph. We identify the less important arcs (by the same hypothesis test describe before, checking how much BIC is added by each arc) and also by the number of appearances in local graphs and we remove them from the global graph. The most important speed-up of FGES-Merge is the embarrassingly parallel nature of the local graphs learning. See figure 4.1 for a diagram explaining FGES-Merge.

We had to use an additional scaling strategy consisting in limiting the exhaustive search of the subset of possible parents for each node, following [58] who implemented a greedy search to solve this. We included a fixed maximum length of the subset to constrain this exponential cost.

Finally, we followed a parallelization strategy to take advantage of all the computing resources. The lower level of parallelization consists of using multiple cores in a single processor. The most computationally costly parts are the initial BIC calculation for the initial arrows and the forward pass of the FGES, the FES. To implement this kind of parallelism we chose multithreading as it is more efficient than multiprocessing in terms of memory.

Python can not use multithreading in a pure concurrent way because of the Global Interpreter Lock (GIL) so we opted to use the Numba library to compile the Python code to optimized machine. Numba does this by using their just in time compilation (JIT) using the LLVM compiler library. This way allows us to make use of concurrent multithreading by avoiding the GIL and using the OpenMP threading layer, while we still preserve the clear Python syntax.

The higher level of parallelism is cluster computing (e.g. HPC environments). The same multithreading parts plus the local graphs learning were also implemented using cluster parallelism so all the computer resources are fully used. We opted to use the MPI protocol because of its adoption in most of the HPC centres, and its efficiency compared to another software like Spark or Hadoop. We include this feature with the mpi4py package for Python [59]. For the results presented in this work we did not need to use HPC but we did divide the full work load between three desktop computers at our laboratory.

The robustness of local-to-global method can be further improved by running the algorithm multiple times and then combining the resulting global graphs. During the DREAM5 competition, two of the Bayesian network based methods with the highest scores were built using the simulated annealing package for R, catnet. We implemented a modification in the FGES to add a stochastic factor for choosing the next arc to add in the forward phase, so this will prevent the algorithm to always add the arc with the greatest BIC in every step. This random factor to not always select the arc with greatest BIC, decreases with every arc we include, so the first arcs are more prone to be discarded in favor of a random arc with positive BIC. Since we take into account the BIC added in the global network when we do the final pruning pass this randomness doesn't have a

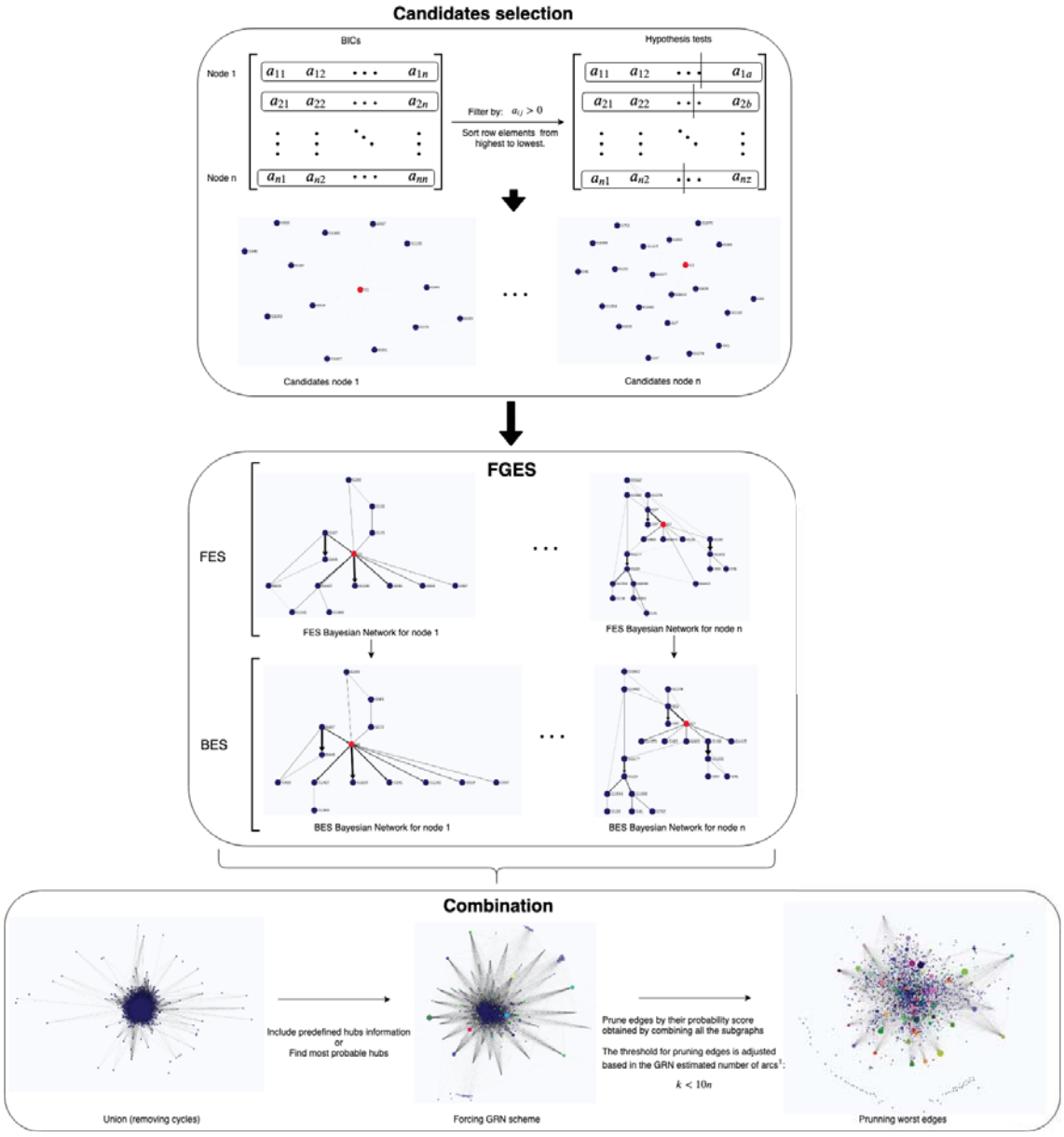


Figure 4.1: FGES-Merge algorithm diagram.

deleterious effect in the result because bad arcs can be easily eliminated but arcs which are better in the global network than the local ones have a chance to be explored.

4.2 Probabilistic inference in a massive regulatory network

The second problem we need to solve is that even if we have learnt the structure of the network in an efficient way we need to be able to do probabilistic inference in a reasonable time for the model to be useful. Fortunately this problem is very easy to solve if we make a very strong but biologically plausible and empirically tested assumption: gene expression is a random variable distributed like a Normal distribution. More concretely, we assume that our Bayesian network is a linear Gaussian

Bayesian network.

4.2.1 Gaussian Bayesian networks

Gaussian Bayesian networks are a type of Bayesian networks where all the variables are continuous and all of the CPDs are linear Gaussians. That is, that given a node X with j parents $\text{Pa}(X) = \{X_1, X_2, ..., X_j\}$:

$$p(X|\text{Pa}(X)) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \text{Pa}(X); \sigma^2\right) \tag{4.1}$$

where $_0$ is the average expression level of X. As a result of this restriction on the form of the CPDs we have that a linear Gaussian Bayesian network is a factorized representation of a multivariate Gaussian. This relates this Bayesian networks to the Gaussian graphical models we discussed previously, inheriting all the advantages of that model without the main limitation of the precision matrix being too hard to learn with limited data. This is beacause the factorization of the distribution makes the required number of learnt parameters smaller than quadratic on the number of nodes, with the gains depending on how sparse the network is. That is, the sparser the network, the less parameters we have to learn compared to estimating the full joint from the beginning.
This restriction on the form of the CPD brings another advantage to the method in that when calculating the BIC for learning the structure with FGES if the distribution of the variables is a linear Gaussian, the BIC is a lot easier to calculate since, for a given edge, it only depends on the variance of the child given its parents and the number of parents. More concretely, let $\text{Pa}(Y)$ be the set of parents of Y in the graph. Then the BIC difference of adding X to $\text{Pa}(Y)$ is given by:

$$\text{BIC}(X, Y, \text{Pa}(Y)) = -m/2 \ln s^2 - ck \ln m \tag{4.2}$$

where n is the sample size, s^2 is the mean squared error of the regression of $\text{Pa}(X) \cup X$ on Y (which is also the likelihood of the distribution) c is the penalty discount which is a free parameter of the algorithm and k is the cardinality of $\text{Pa}(Y) \cup X$ which is the number of parameters to learn. The higher the penalty term c is chosen, the more sparse the network will be since only arcs with positive BIC difference will be added.

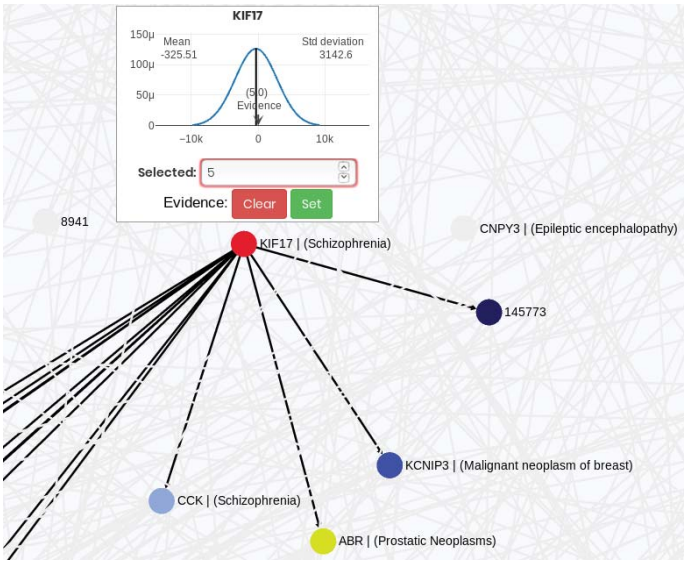
Parameter Learning

Parameters for Linear Gaussian Bayesian Networks are very easy to estimate once the structure of the network is known. For each node we do a multilinear regression against the set of its parents. The parameters of the regression give the mean of the CPD for the node while the mean square error of the regression is the variance of the CPD.

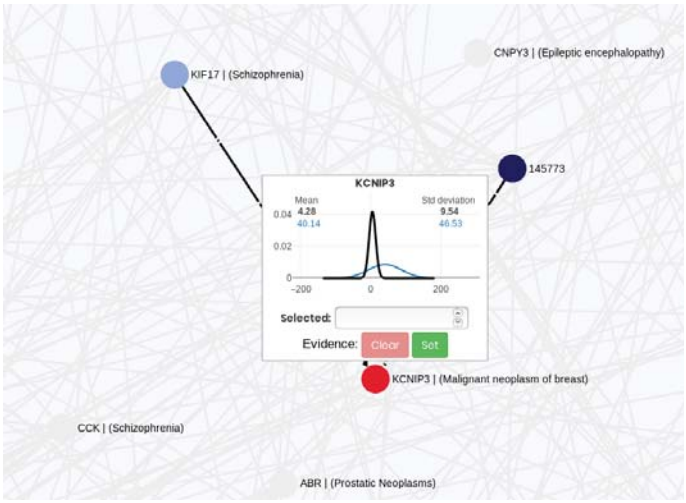
Inference in Gaussian Bayesian networks

The main advantage of Gaussian bayesian networks is that the joint distribution is a multivariate Normal distribution and although we saw that exact inference in discrete Bayesian networks is a very hard problem in general it is only $O(n^3)$ for a Gaussian Bayesian network. Following [**?**] we implemented our own version of an algorithm to move from the factorized network to the joint distribution and back to a conditional distribution with a way to input the inference. This is now implemented in Neurosuites, a platform where all the neuroscience tools developed at the Computational Intelligence Group [1] are available for public use. Both the implementation of FGES-Merge and Gaussian network inference are available here [2]. For an example, see figure 4.2

[1] http://cig.fi.upm.es/
[2] https://neurosuites.com/morpho/ml$_b ayesian_n etworks$



(a) Set the evidence in a specific gene using the Neurosuites platform.



(b) $P(Q|E)$ marginal distribution of a specific gene. In this case the network predicts a relationship between schizophrenia and malignant breast neoplasm. We have found some evidence for this in the literature [60]

Figure 4.2: Example of inference using the Neurosuites tool. The backend was implemented as part of this work and the frontend was done by Mario Michiels and Sergio Paniego.

Chapter 5

Results

5.1 Benchmark tests

We decided on two ways to test our method. We wanted to compare ourselves in our ability to recover the underlying structure against a variety of algorithms, even those which don't use BNs. But we also wanted to compare ourselves to other BN methods as a way to show that our algorithm transcends the usual difficulty for learning them and is capable of scaling to thousands of nodes.

For both benchmarks of our model we decided to use the data from the DREAM challenge in its fifth edition [23] so that we could compare ourselves to the networks recovered by many other methods without having to recompute them ourselves (which would have been almost impossible since not all of the methods are publicly available). Following [23] we did not use experiment 2, so we compared ourselves with the networks extracted for experiments 1, 3 and 4.

Most Bayesian network methods do not assign a confidence level to their arcs and so cannot be correctly evaluated by simply using a threshold on their predicted edge probabilities. Proper calibration for this would require repeating the construction with some random variation and using frequency analysis of the arcs to estimate probabilities.

Prior probability of arcs should be around 1/nodes since regulatory networks are known to be sparse with nodes arcs. If we assign a threshold of 0.5 as is usual for a binary classification level, we are making the accuracy of well calibrated methods incredibly high since its very easy to just guess no arcs exist (an empty net has 99% accuracy for big networks.) There is a huge class imbalance and so we need to either use a smaller threshold or use a score that accounts for this.

In [23] the score used is AUPRC (Area under the precision recall curve) which is usually a good score for imbalanced problems but since it tries multiple thresholds, a good score here doesn't directly translate to usefulness of the method. When given one of these networks a biologist has to make a choice on what interactions to test, whith the corresponding cost, so we need to make sure we have a high ratio of true positives to edge predictions (high precision). At the same time, we also need to be able to explore as many possible interactions as possible, so we are interested in a mixture of recall and precision which is as high as possible. We use the *Matthews Correlation Coefficient* (MCC)[61] since it takes into account the imbalance between the classes. The formula for the MCC canbe calculated from the confusion matrix with the expression:

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}} \tag{5.1}$$

MCC goes from 0 to 1, with 0 being the worst possible score and 1 meaning a perfect prediction. In

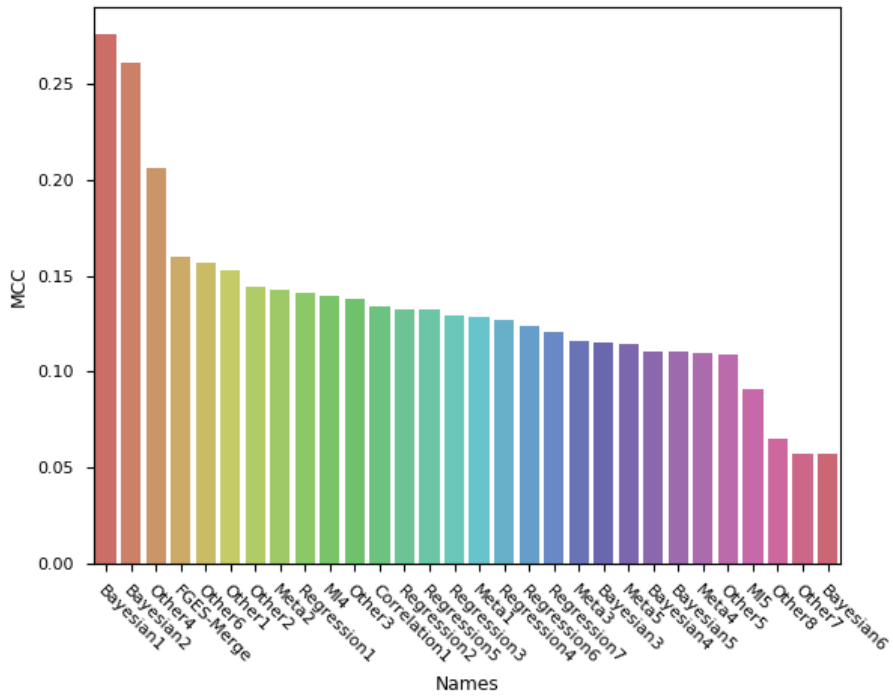


Figure 5.1: MCC scores for network 1 (in silico) from the DREAM 5 challenge. Here FGES-Merge is one of the top rating algorithms.

balanced problems it behaves similarly to F-score, but in imbalanced problems it punishes trying to just guess the bigger class without trying to actually solve the prediction problem.

Our results are competitive with the best Bayesian network methods and, for some networks, competitive with the whole set of methods as can be seen in Figures 5.1, 5.2 and 5.3. As can be seen there, although we are not always the best Bayesian network method, we are much more consistent even in the harder networks.

As for our time comparison, we tested some of the most common Bayesian network learning algorithms on the smallest network from the DREAM challenge. We were goin to show the performance of various algorithms across different network sizes to see if FGES-Merge scaled better. We expected that we would be beaten by Chow-Liu's algorithm and similar due to those being quadratic in time. In the end, we could not test more than one net since although our method finished in slightly more than an hour, the other methods worked for more than a day and didn't finish. The results for network 1 are in Figure 5.4

5.2 Human brain regulatory network

Finally, we applied the method to solve the problem we were interested in: Learning the genome-wide regulatory network for the human brain.

The main objective was to find the genome-wide GRNs for various areas of the brain (i.e. cortex, white matter, cerebellum, hypothalamus...). These networks would be very useful tools for biologists who are interested in studying how the functional differences between brain areas might arise from the differences in genetic expression.

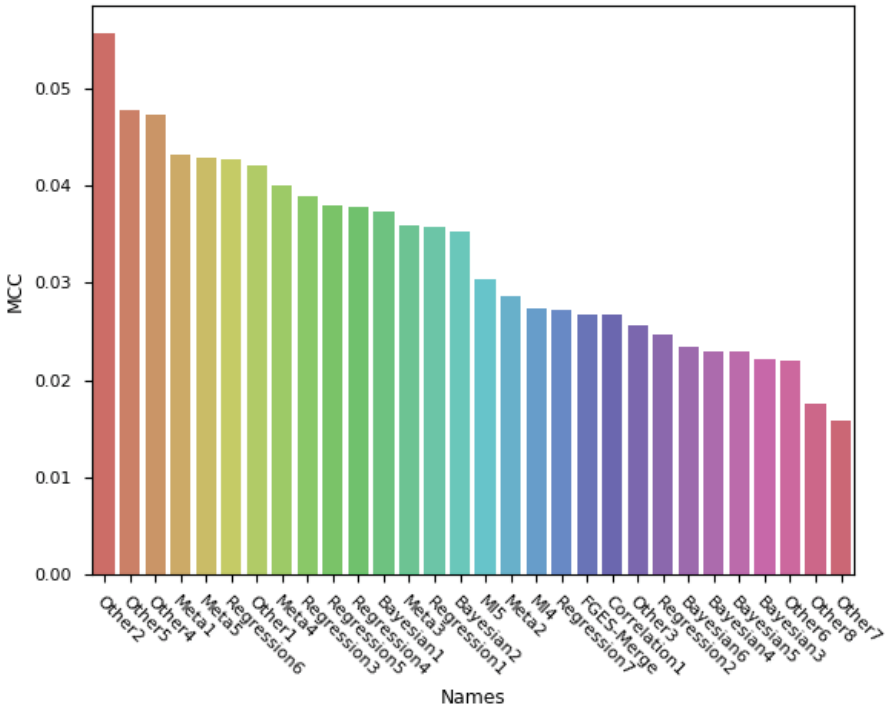


Figure 5.2: MCC scores for network 3 (*Escherichia Coli*) from the DREAM 5 challenge. All methods perform much worse in the in-vitro networks. FGES-Merge is average.

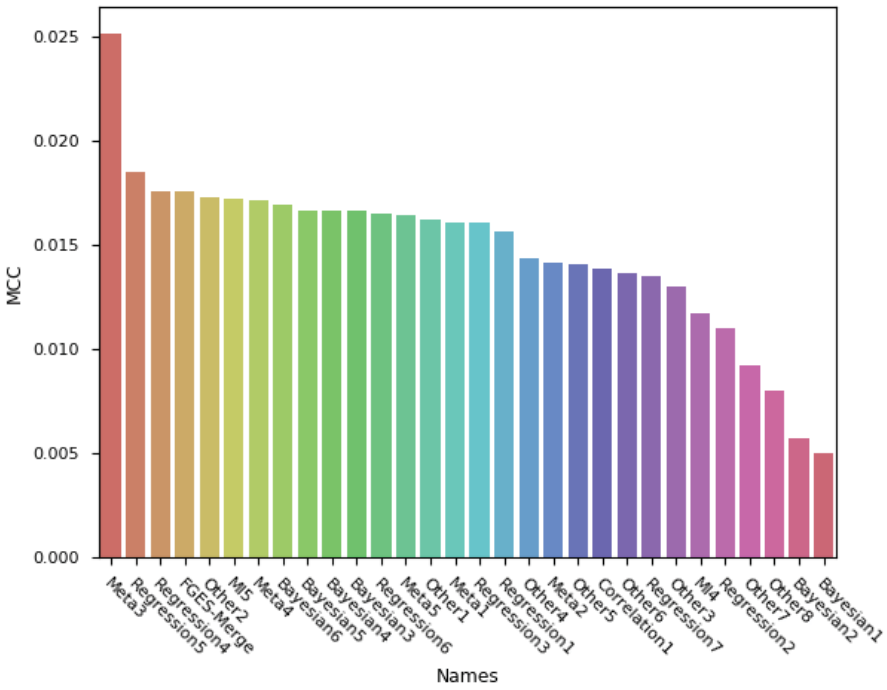


Figure 5.3: MCC scores for network 4 (*Sacharomyces Cerevisiae*) from the DREAM 5 challenge.

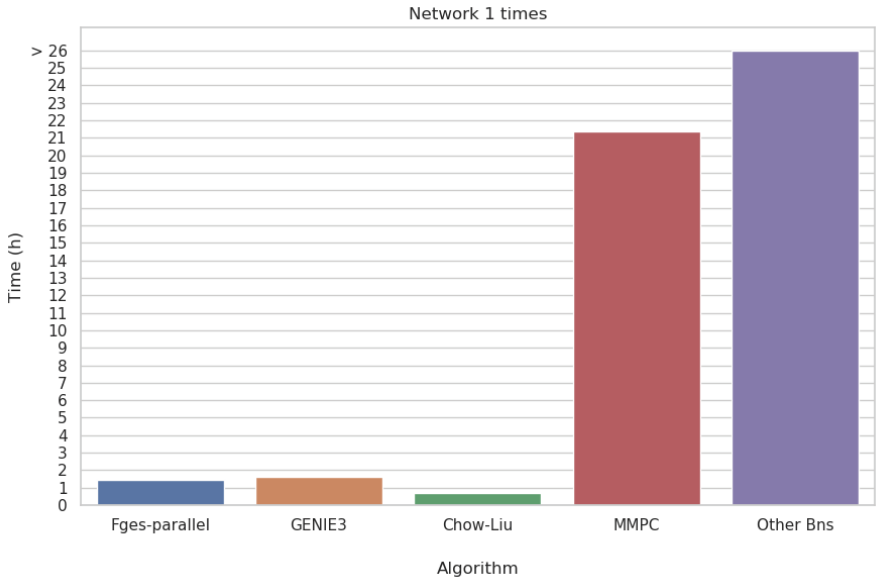


Figure 5.4: Running times for network 1 (in silico) from the DREAM5 challenge for various Bayesian network learning methods.

Thanks to the application of FGES-Merge, we managed all of our objectives. We were able to obtain networks for various areas of the brain and an extra network for the average expression of the whole brain. In this section we will show the networks generated (using software developed by Mario Michiels and Sergio Paniego and available in Neurosuites). We will comment on the topological properties of the learnt networks to see if they respect the known empirical properties of GRNs as discussed in Section 2.1.2.

5.2.1 Network examples

Figure 5.5 shows two networks obtained with the whole dataset. They represent the average GRN for the whole brain. They have been learned with different penalty parameters and thresholds for the number of arcs. We can readily see how the higher penalty network presents various disconnected components unlike the lower penalty network. This is what we would expect since a higher penalty forces sparsity.

5.2.2 Topological properties of the learned GRNs

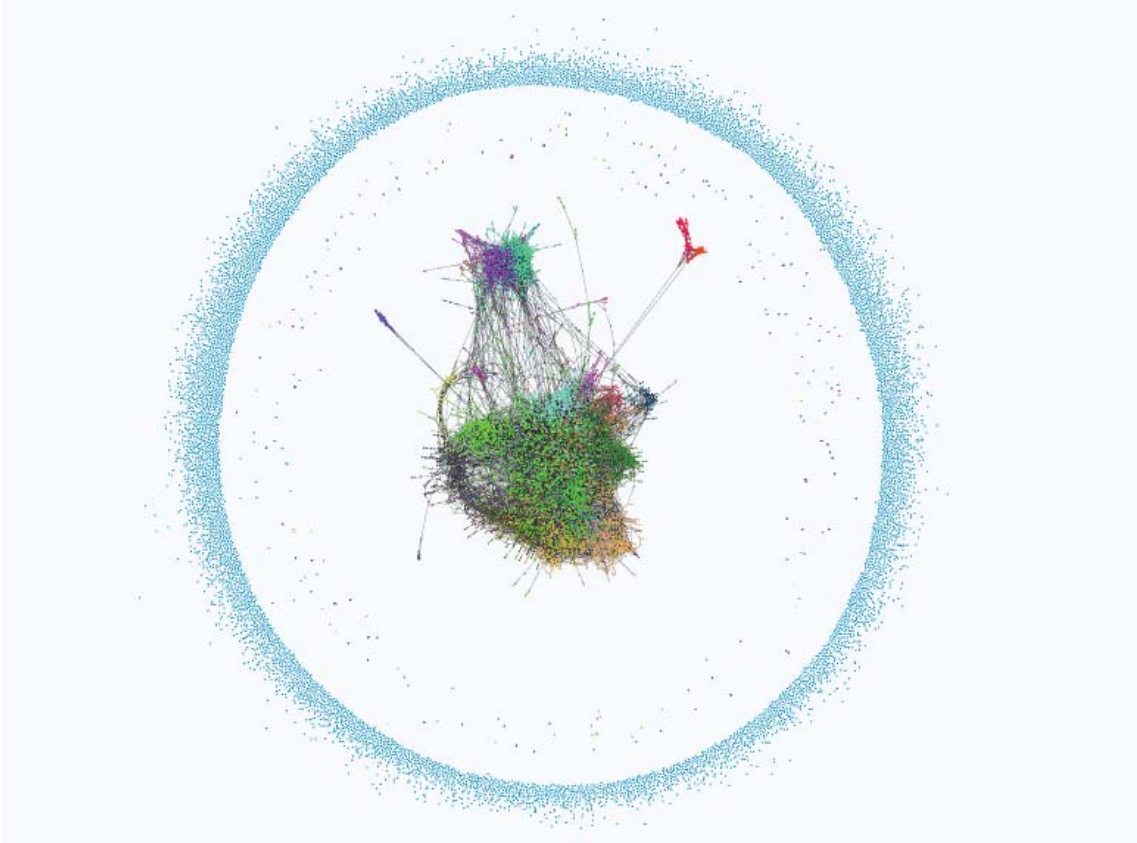
Figure 5.6 shows the in-degree and out-degree distributions for the human brain GRN and the tail of the total degree distribution. As we saw in section 2.1.2, most nodes have an out-degree of 0 (they are not regulators) but the ones that have children have many of them as expected from the evolutionary argument (regulators have an average of over 600 children).

The in-degree distribution is also as expected. Most genes are regulated by a small number of regulators and even the most regulated don't have more than 80. Again, this is in agreement with the biological argument that gene regulation requires physical interaction and there is not enough room for having hundreds or more regulators.

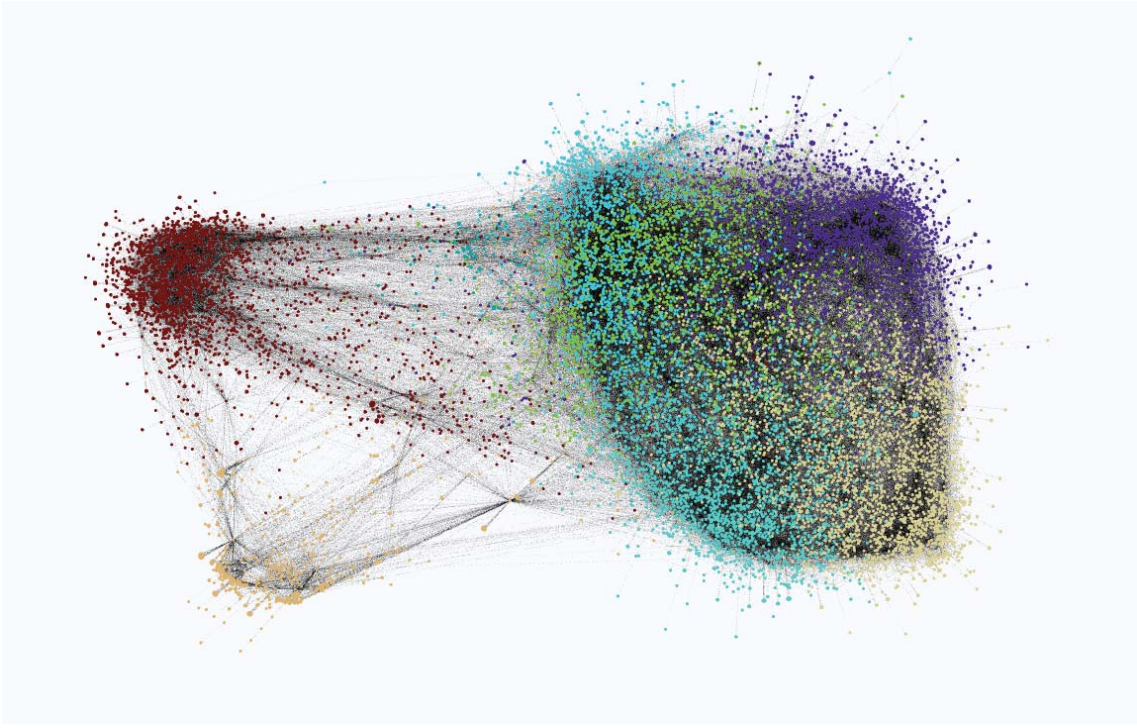
Finally, from all three distributions we can see that the network is globally sparse but locally

dense. Most nodes have no children and around half of the nodes have less than 14 parents (with most having just one or two). However, the tail of the total degree distribution shows that the 1000 nodes with the highest degree have an average of 500 neighbours with some of them having well over a thousand of them.

All of these shows that our algorithm respects the topological properties of GRNs and although it would be unfeasible to test all the interactions in the learned network this topological analysis gives us strong reasons to believe that the inferred network is sound.

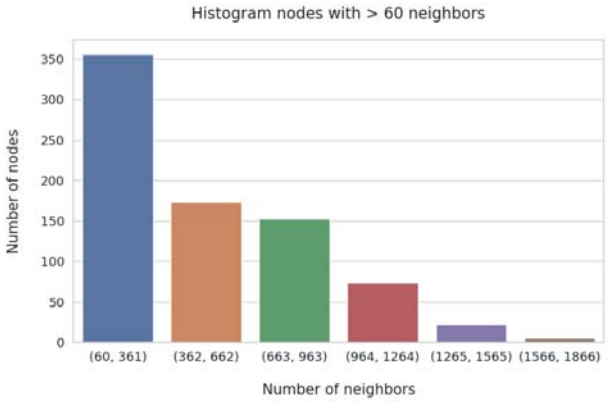


(a) Our learned full human brain network. FGES penalty: 65, not removing any learned arc, Louvain algorithm for coloring the nodes.

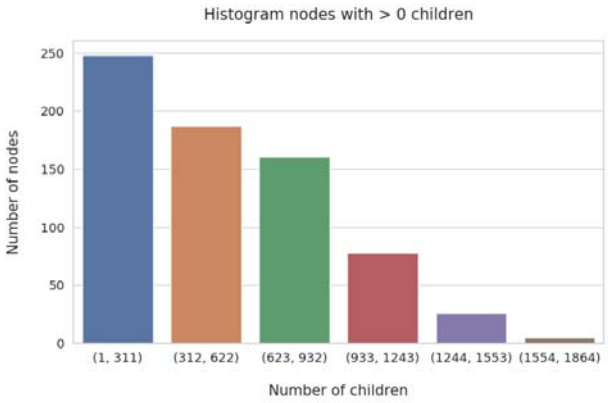


(b) Our learned full human brain network. FGES penalty: 45, number of arcs: $6n$ being n the number of genes, only showing the connected arcs, Louvain algorithm for coloring the nodes.

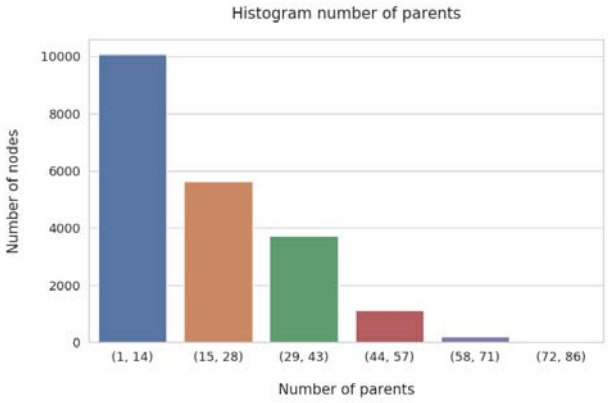
Figure 5.5: FGES-Merge human genome network for the human brain with two different penalties.



(a) Histogram nodes with > 60 neighbors.



(b) Histogram nodes with > 0 children.



(c) Histogram number of parents

Figure 5.6: Nodes degree histograms for our learned full human brain network. The threshold for the minimum number of neighbors and children has been adjusted for a proper visualization of the histograms. FGES penalty: 45, with pure union as merging method. That is, no pruning.

Chapter 6

Conclusions and future research

6.1 Conclusions

After reviewing the field of GRNs and presenting a novel scalable algorithm for learning them with Bayesian Networks; in the final chapter of this document we will summarize the main conclusions obtained both from the field of GRNs and from the development and application of our algorithm using Bayesian networks.

- Learning GRNs is in general a very hard problem. Even the best methods have very low scores on reconstruction, make many mistakes and scale poorly to the genome-wide environments we are interested in.
- Any approach that tries to tackle genome-wide networks will have to use massive computational resources or make very strong assumptions on the structure and parameters of the GRN.
- Methods that allow for quantitative predictions of expression levels require a very large amount of data that is not available most of the time. This is one of the reasons why deep learning methods were not chosen for this work.
- We have presented a method for learning GRNs that is competitive with the state of the art in general methods while also beating most other methods that use Bayesian networks and giving consistently good results even for the harder networks in the benchmarks. Furthermore, our method can deal with the tens of thousands of genes involved in genome-wide networks and gives results that respect the topological properties of real GRNs.

6.2 Future research

Given the results obtained, we can see there are various ways to continue with this work both in terms of improving it or in direct applications. On the one hand, we have the multiple possibilities of using the algorithm to build regulatory networks in different conditions to obtain biological insights. One first obvious line of work would be to use the networks we have built for the human brain to try to determine which genes are involved in specialized functions of each brain region.

On the other hand, there are various variations to the algorithm that could be investigated. Maintaining scalability to genome-wide networks is hard and a lot of limiting assumptions have been made. One way forward that presents itself is to relax the assumption of gaussianity and use different distributions, most likely, a mixture.

We suspect that simple methods like correlation and mutual information networks perform equally or even better than more complex methods because of the lack of data. More complex methods have too many parameters and overfit the dataset and so fail to generalize to new data or make correct predictions in novel situations. This would also explain why making simplifying assumptions for these methods, i.e. Gaussian Bayesian networks, work so well. This suspicion could be tested by trying various methods with synthetic datasets of different sizes and seeing if and how their relative performances change.

6.3 Scientific Dissemination

An early version of the algorithm presented here was presented as a poster at the HBPSC (Human Brain Project Student Conference) in Ghent in February, 2019. An article including the algorithm and a visualization tool is being prepared.

Bibliography

[1] G. J. Nuovo, "PCR *in situ* hybridization," in *In Situ Hybridization Protocols*, Methods in Molecular Biology, pp. 223–241, Humana Press, 1995.

[2] C. Thisse and B. Thisse, "High-resolution *in situ* hybridization to whole-mount zebrafish embryos," *Nature Protocols*, vol. 3, pp. 59–69, 2008.

[3] V. Trevino, F. Falciani, and H. A. Barrera-Saldaña, "DNA microarrays: A powerful genomic tool for biomedical and clinical research," *Molecular Medicine*, vol. 13, pp. 527–541, 2007.

[4] P. Larrañaga, I. Inza, and J. L. Flores, "A Guide to the literature on inferring genetic networks by probabilistic graphical models," in *Data Analysis and Visualization in Genomics and Proteomics*, pp. 215–238, Wiley-Blackwell, 2005.

[5] G. Sanguinetti and V. A. Huynh-Thu, eds., *Gene Regulatory Networks: Methods and Protocols*. Methods in Molecular Biology, Springer, 2019.

[6] U. Alon, "Appendix C: Graph properties of transcription networks," in *An Introduction to Systems Biology: Design Principles of Biological Circuits.*, Chapman and Hall/CRC, 2006.

[7] J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour, "A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images," *International Journal of Data Science and Analytics*, vol. 3, pp. 121–129, 2017.

[8] J. Watson and F. Crick, "Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid," *Nature*, pp. 737–738, 1953.

[9] F. Crick, "Central dogma of molecular biology," *Nature*, pp. 561–563, 1970.

[10] M. Nirenberg, P. Leder, M. Bernfield, *et al.*, "RNA codewords and protein synthesis, vii. on the general nature of the RNA code," *Proceedings of the National Academy of Sciences of the United States of America*, pp. 1161–8, 1965.

[11] M. Bantscheff, M. Schirle, G. Sweetman, J. Rick, and B. Kuster, "Quantitative mass spectrometry in proteomics: a critical review," *Analytical and bioanalytical chemistry*, vol. 389, no. 4, pp. 1017–1031, 2007.

[12] Z. Wang, M. Gerstein, and M. Snyder, "RNA-seq: a revolutionary tool for transcriptomics," *Nature reviews genetics*, vol. 10, no. 1, p. 57, 2009.

[13] M. J. Hawrylycz, E. S. Lein, A. L. Guillozet-Bongaarts, E. H. Shen, L. Ng, J. A. Miller, L. N. Van De Lagemaat, K. A. Smith, A. Ebbert, Z. L. Riley, *et al.*, "An anatomically comprehensive atlas of the adult human brain transcriptome," *Nature*, vol. 489, no. 7416, p. 391, 2012.

[14] K.-C. Chen, T.-Y. Wang, H.-H. Tseng, C.-Y. F. Huang, and C.-Y. Kao, "A stochastic differential equation model for quantifying transcriptional regulatory network in Saccharomyces cerevisiae," *Bioinformatics*, vol. 21, no. 12, pp. 2883–2890, 2005.

[15] A. Angelin-Bonnet, P. Biggs, and M. Vignes, "Gene regulatory networks: A primer in biological processes and statistical modelling," in *Gene Regulatory Networks: Methods and Protocols*, ch. 15, pp. 347–378, Springer, 2019.

[16] A. Nair, M. Chetty, and P. P. Wangikar, "Improving gene regulatory network inference using network topology information," *Molecular BioSystems*, vol. 11, no. 9, pp. 2449–2463, 2015.

[17] M. Grzegorczyk, A. Aderhold, and D. Husmeier, "Overview and evaluation of recent methods for statistical inference of gene regulatory networks from time series data," in *Gene Regulatory Networks*, pp. 49–94, Springer, 2019.

[18] B. Zhang and S. Horvath, "A general framework for weighted gene co-expression network analysis," *Statistical applications in genetics and molecular biology*, vol. 4, no. 1, 2005.

[19] Butte, A.J. and I. Kohane, "Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements.," in *Pacific Symposium on Biocomputing, 2000*, p. 418–429, World Scientific.

[20] A. A. Margolin *et al.*, "ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," in *BMC bioinformatics*, vol. 7, p. S7, BioMed Central, 2006.

[21] J. J. Faith *et al.*, "Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles," *PLoS biology*, vol. 5, no. 1, p. e8, 2007.

[22] P. E. Meyer, K. Kontos, F. Lafitte, and G. Bontempi, "Information-theoretic inference of large transcriptional regulatory networks," *EURASIP journal on bioinformatics and systems biology*, vol. 2007, pp. 8–8, 2007.

[23] D. Marbach, J. C. Costello, *et al.*, "Wisdom of crowds for robust gene network inference," *Nature Methods*, vol. 9, pp. 796–804, 2012.

[24] A. Irrthum, L. Wehenkel, P. Geurts, *et al.*, "Inferring regulatory networks from expression data using tree-based methods," *PloS one*, vol. 5, no. 9, p. e12776, 2010.

[25] V. Huynh-Thu, L. Wehenkel, and P. Geurts, "Gene regulatory network inference from systems genetics data using tree-based methods.," in *Gene network inference: verification of methods for systems genetics data.*, p. 63, Berlin: Springer, 2013.

[26] V. Huynh-Thu and G. Sanguinetti, "Combining tree-based and dynamical systems for the inference of gene regulatory networks.," *Bioinformatics*, p. 1614–1622, 2015.

[27] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie, "Gene expression inference with deep learning," *Bioinformatics*, vol. 32, pp. 1832–1839, 2016.

[28] J. Schäfer and K. Strimmer, "An empirical bayes approach to inferring large-scale gene association networks," *Bioinformatics*, vol. 21, no. 6, pp. 754–764, 2004.

[29] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Elsevier, 1988.

[30] T. Verma and J. Pearl, "An algorithm for deciding if a set of observed independencies has a causal explanation," in *Uncertainty in Artificial Intelligence*, pp. 323–330, Elsevier, 1992.

[31] P. Spirtes and C. Meek, "Learning Bayesian networks with discrete variables from data.," in *KDD*, vol. 1, pp. 294–299, 1995.

[32] P. Spirtes, C. N. Glymour, *et al.*, *Causation, prediction, and search*. MIT press, 2000.

[33] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu, "Learning Bayesian networks from data: An information-theory based approach," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 43–90, 2002.

[34] R. Mahdi and J. Mezey, "Sub-local constraint-based learning of Bayesian networks using a joint dependence criterion," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1563–1603, 2013.

[35] D. M. Chickering, "Learning Bayesian networks is NP-complete," in *Learning from Data: Artificial Intelligence and Statistics V*, Lecture Notes in Statistics, pp. 121–130, Springer, 1996.

[36] G. Schwarz *et al.*, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[37] J. Rissanen, "Stochastic complexity and modeling," *The Annals of Statistics*, pp. 1080–1100, 1986.

[38] R. R. Bouckaert, "Probabilistic network construction using the minimum description length principle," in *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pp. 41–48, Springer, 1993.

[39] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.

[40] H. Akaike, "A new look at the statistical model identification," in *Selected Papers of Hirotugu Akaike*, pp. 215–222, Springer, 1974.

[41] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.

[42] D. M. Chickering, "Optimal structure identification with greedy search," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 507–554, 2002.

[43] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[44] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 50, no. 2, pp. 157–194, 1988.

[45] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Machine Intelligence and Pattern Recognition*, vol. 5, pp. 149–163, Elsevier, 1988.

[46] R. Fung and K.-C. Chang, "Weighing and integrating evidence for stochastic simulation in Bayesian networks," in *Machine Intelligence and Pattern Recognition*, vol. 10, pp. 209–219, Elsevier, 1990.

[47] A. Golightly and D. J. Wilkinson, "Bayesian parameter inference for stochastic biochemical network models using particle markov chain monte carlo," *Interface focus*, vol. 1, no. 6, pp. 807–820, 2011.

[48] M. B. Sesen, A. E. Nicholson, R. Banares-Alcantara, T. Kadir, and M. Brady, "Bayesian networks for clinical decision support in lung cancer care," *PloS ONE*, vol. 8, no. 12, p. e82349, 2013.

[49] A. C. Constantinou, N. E. Fenton, and M. Neil, "pi-football: A Bayesian network model for forecasting association football match outcomes," *Knowledge-Based Systems*, vol. 36, pp. 322–339, 2012.

[50] C. Bielza and P. Larrañaga, "Bayesian networks in neuroscience: A survey," *Frontiers in Computational Neuroscience*, vol. 8, 2014.

[51] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[52] P. Spirtes, C. Glymour, *et al.*, "Constructing Bayesian network models of gene expression networks from microarray data." https://kilthub.cmu.edu/articles/Constructing$_B$ayesian$_N$etwork$_M$odels$_o$f$_G$ene$_E$xpression$_N$etworks$_f$rom$_M$icroarray$_D$

[53] D. Edwards, G. C. De Abreu, and R. Labouriau, "Selecting high-dimensional mixed graphical models using minimal AIC or BIC forests," *BMC Bioinformatics*, vol. 11, no. 1, pp. 1–8, 2010.

[54] C. J. Needham, I. W. Manfield, A. J. Bulpitt, P. M. Gilmartin, and D. R. Westhead, "From gene expression to gene regulatory networks in Arabidopsis thaliana," *BMC Systems Biology*, vol. 3, no. 1, p. 85, 2009.

[55] F. Liu, S.-W. Zhang, W.-F. Guo, Z.-G. Wei, and L. Chen, "Inference of Gene Regulatory Network Based on Local Bayesian Networks," *PLoS Computational Biology*, vol. 12, 2016.

[56] S. B. Gillispie and M. D. Perlman, "Enumerating Markov equivalence classes of acyclic digraph dels," in *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence*, pp. 171–177, Morgan Kaufmann Publishers Inc., 2001.

[57] C. Meek, "Causal inference and causal explanation with background knowledge," *arXiv:1302.4972*, 2013.

[58] J. I. Alonso-Barba, L. delaOssa, J. A. Gámez, and J. M. Puerta, "Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes," *International Journal of Approximate Reasoning*, vol. 54, pp. 429–451, 2013.

[59] H. Asaadi, D. Khaldi, and B. Chapman, "A comparative survey of the HPC and big data paradigms: Analysis and experiments," in *2016 IEEE International Conference on Cluster Computing*, pp. 423–432, IEEE, 2016.

[60] V. Ajdacic-Gross, A. Tschopp, M. Bopp, F. Gutzwiller, and W. Rössler, "Cancer comortality patterns in schizophrenia and psychotic disorders: A new methodological approach for unique databases," *International Journal of Methods in Psychiatric Research*, vol. 23, no. 1, pp. 19–24, 2014.

[61] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, pp. 442–451, 1975.