ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

UNIVERSIDAD POLITÉCNICA DE MADRID

TESIS FIN DE MÁSTER
MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL

A Novel Multi-dimensional Regression Model based on Gaussian Networks.

Autor: Milton Llera Montero

Tutores: Pedro Larrañaga Múgica
Concha Bielza Lozoya

July, 2017

Agradecer en primer lugar a mis tutores Concha Bielza y Pedro Larrañaga por la oportunidad de trabajar en este proyecto, el apoyo y la guía que me han dado durante su realización.

A mis compañeros del CIG por toda la ayuda que me han prestado durante mi adaptación al trabajo en el grupo.

Este proyecto no hubiera sido posible sin el apoyo financiero del proyecto Cajal Blue Brain.

Finalmente quiero agradecer a mis amigos y en especial a mi familia a quienes dedico este trabajo, sin cuyo apoyo hubiese sido imposible realizar esta labor.

Abstract

Modeling and prediction in continuous domains are one of the most important and studied problems in Mathematics and Computer Science. Models that can not only solve regression tasks, but also expose the interdependencies inside the domain are of high value for researchers in many fields. One of the most popular methods for learning the relations between variables in a continuous domain are Gaussian Networks. In this thesis we present a new model that can learn a Gaussian Network. This model can later be used for regression or analysis of the relations in the domain, with a particular interest in its application in the field of Neuroscience.

Resumen

Modelar y predecir en dominios continuos es uno de los problemas más estudiados en el campo de las Matemáticas y la Ciencia de la Computación. Modelos que no solamente puedan resolver problemas de regresión, si no también exponer las relaciones entre las variables de un dominio son de gran valor para los investigadores de muchos campos científicos. Uno de los modelos más populares usados para resolver estos problemas son las Redes Gaussianas. En esta tesis se presenta un nuevo modelo basado en Redes Gaussianas que puede ser ajustado a partir de datos para luego ser utilizado en tareas de regresión y análisis, con un especial interés en su aplicación al campo de la Neurociencia.

Content

List of figures

List of tables

LIST OF TABLES

1

Introduction

Discovering patterns and making predictions from data have become one of the main problems in the area of Machine Learning (ML) and Computer Science in the last decade and a half. Scientists have been working on developing models that can uncover the hidden knowledge buried in the heaps of data that have been created by the explosion in the use of computational devices in many aspects of human life. Some of the more evident applications of these techniques have been found in social networks, email services, search engines and online retail. This has been spurred by the huge revenues that can be achieved if the information can is to bridge the gap between consumers and the products/information they want.

Recently however, new areas of inquiry where these methods can be applied have emerged in various scientific fields like Physics and Medicine where it is hoped that the accumulated data can be exploited to enhance our comprehension and predicting capability of a wide range of phenomena. For example ML has been used to identify stars in high resolution photographs taken by satellites and to detect particle collisions in experiments conducted at the Large Hadron Collider at the CERN. More prominent uses have been found in Medicine and Biology to predict diseases and analyze genetic material. These efforts have mostly been in the application of classical ML methods to these problems with the help of experts to curate the data.

In recent years a new field has grab the attention of the ML community: Neuroscience. The study of how the brain works and all its implications relative to consciousness, intelligence, learning etc. has gained a significant boost in the last decade with several huge initiatives like the Human Brain Project in Europe, the creation of the Allen Institute for Brain Science, Janelia Farm and more recently

the BRAIN Initiative, all of these in the USA, which dedicate a huge amount of resources to this endeavor. Neuroscientists have realized that ML could be used to analyze the various data acquired through experiments and recordings, which could spark a breakthrough in a very complex domain, where a multitude of biological, chemical and electrical phenomena occur at the same time.

The challenges in Neuroscience are somewhat different than in Computer Vision or social network analysis where the amount of data (relative to the dimensionality of the problem) is enormous and the main focus of improvements lie in learning speed and accuracy. Here data is scarce by those standards (often several hundreds or thousands at most) and predictive capacity alone is insufficient as scientists also want to gain insight into the underlying processes (for example what genetic markers are mostly associated with a certain disease). However, these are often diverging goals for it is difficult to balance predictive performance and intuitive models that expose the relations between variables of the given problem. For example, popular models used to boost predictive performance like ensemble methods combine several base models to improve on their individual performance, but consequently are less interpretable. Another extreme example would be Deep Learning, whose techniques have brought a revolution to many fields like Computer Vision and Natural Language Processing but we still lack a proper understanding of why they work.

These algorithms thus become less useful in a problem where prediction is not the only goal. This, in turn, provides the ML and Artificial Intelligence community with a new challenge where the conditions are different from most mainstream applications nowadays and new approaches are needed.

In this work we focus on the problem of modeling the relation between morphology and electro-physiology of neurons, which is a fundamental problem as the electric signals determine all our actions and it is not very well understood how these vary across brain areas (and neuron structure) and what implications these have for higher brain functions. Using this problem as a practical use case as motivation we propose the Multidimensional Bayesian Gaussian regressor, a novel model that is trained from data using a new algorithm which can be used to discover relations between variables and make predictions of their values.

1.1. Relation between morphology and electrophysiology in neurons

More explicitly, the question we wish to explore in this thesis is the following: How does the morphological structure of a neuron relate to the electrical signals it produces? This question can be broken down into two subproblems: 1) What relationship exists between the variables that describe the morphology (or passive properties) and electrical (active) properties of neurons? and 2) How effectively can we use these relationships to predict one from the other?

Neuroscientists have long studied how cell morphology modulates the electrical response of neurons to input stimuli since Hodgkin and Huxley proposed the HH model for action potential generation in the giant squid axon. Even though the relation exist and several phenomena can be precisely explained with mathematical formulations (see Cable Theory in Rall et al. (1995)), quantitative analyses are harder to determine due to the many interacting factors. This has thus become a focus of studies in the field of Neuroscience, as they try to map them to functional properties across regions of the brain.

Studies addressing these issues became prominent in the early 1990s, for example with Larkman and Mason (1990) who study the correlation between these sets of features in pyramidal neuron slices of the rat visual cortex. Correlations where also studied for neurons of the neocortex in Mainen and Sejnowski (1996). Computational models have also been developed, with emphasis on Cable Theory of W. Rall (Rall et al. (1995)), which give a perspective on the relation between properties of dendrites (branching patterns, length, thickness) and how electrical signals propagate through them. Yuste and Tank (1996) present a review of the work done on pyramidal cells and also conclude that dendrites have complicated information processing properties that are derived from their complex structures. Dendritic arbor size has a strong influence on the shape of action potential (Eyal et al. (2014)) which is important for example in determining spike onset rapidness which in turn defines the ability of a neuron to encode changes in the input (Fourcaud-Trocmé et al. (2003)). The architecture of these arbors and of the axons also varies across layers (Mohan et al. (2015)) which means that these might have different active properties (and thus produce distinguishable electrical signals).

The definition of a computational model that can relate these properties of neurons in a precise way would constitute an important step in improving our unders-

tanding of their functional purpose across different regions of the brain and how this relates to their physical characteristics, which in turn would probably allow precise classification of neurons not only by shape (at times a somewhat ambiguous task as found in DeFelipe et al. (2013) and López-Cruz et al. (2014)).

1.2. Objectives

In this master's thesis we propose using more robust methods from the fields of ML and Bayesian probability theory to estimate qualitative and quantitative relations between continuous variables with the interest of applying them specifically. The aim is to generate sensible hypotheses that can later be tested experimentally. These models should also improve on the predictive capability on these sets of features. We focus specifically on Bayesian networks as a mathematical modeling tool (we explain our reasons for this selection in the next chapter).

The objectives for this work can be enumerated as follows:

1. To make a review of techniques often employed to model domains from data where there are several variables of interest which are continuous in nature.

2. To determine which of these tools are most useful for the given problem based on parameters like their reliability, computational efficiency and interpretability.

3. To propose an algorithm that can discover the relevant relations between variables from data and predict new instances accurately .

4. To design experiments that evaluate the capabilities of the proposed method.

5. To discuss possible deficiencies and suggest further improvements.

In other to accomplish these objectives, this thesis is structured in the following way:

- A state of the art on the models used for prediction in continuous domains and the methods used to fit them are presented in Chapter 2.

- Chapter 3 develops the background knowledge of the relevant methods that will be used.

- The proposed model is defined in Chapter 4 also describing implementation and design decisions.
- Experiments conducted to test the proposed algorithm along with the corresponding discussion are in Chapter 5.
- Finally, Chapter 7 contains the conclusions and proposals for future work.

2

Modeling continuous domains

Tackling the problem of discovering the relationship between the variables that describe active and passive properties of neurons can be encompassed in the broader task of modeling domains where variables are continuous. These kinds of problems have long being studied in the ML community although this has often been restricted to single output regression of variables. Given our desire to also emphasize interpretation, we decided that Bayesian Networks are a good fit for this task. To make this claim more robust, in this chapter we will review the existent techniques used to model continuous domains and then proceed to the relevant algorithms used to train Bayesian networks.

From what has been said so far we can establish that whatever model we use, it needs to posses several properties:

1. It must handle uncertainty.
2. We need it to work with continuous variables.
3. It must predict more than one target variable, which may be interrelated.
4. It has to work with a low number of samples relative to problem dimensionality.
5. We would like it to ba as interpretable as possible.

The first point is clear since biological processes and measurements are inherently noisy and this must be taken into account. It necessarily follows that a probabilistic model is required. Next we review the exiting algorithms that address properties 1 to 3.

2.1. Multi-Output Regression

The task of predicting several variables (Multi-Output, MO) simultaneously is called Multi-Output Classification (MOC) when the target variables are discrete and Multi-Output Regression (MOR) when they are continuous, though similar methods can be applied to both.

More formally, in MO we have the following problem statement: given a training set D of N instances where each instance $\mathbf{x}_i = (x_1, \ldots, x_l, y_1, \ldots, y_m)$ is divided into two sets of variables $\mathbf{X}$ and $\mathbf{Y}$. The former are called feature variables, and the latter are the targets. The task is to find a function h such that:

$$h : \Omega_{X_1} \times, \cdots, \Omega_{X_l} \rightarrow \Omega_{Y_1} \times, \cdots, \Omega_{Y_l} \tag{2.1}$$

where Ω_x denotes the domain of variable $\mathbf{X}$ and h has the best possible performance according to some metric. As expected, if the variables t_j are continuous then it's a MOR problem and if they are discrete it's a MOC. Since we are focusing on continuous variables we will elaborate models that deal mainly with these problems.

Multi-Output (and specifically MOR) problems have not been studied too much in the literature, since it has attracted attention mostly recently. The review presented here is based on Borchani et al. (2015), where a more extensive treatment of each algorithm can be found.

Although the related problem of Multi-task learning was studied by Caruana (1998) in the context of Neural Nets (NN) using back-propagation, these are usually black-box models and so are not suited for the current task. Often the way MO problems are solved is by learning one problem for each target variable. This is the simplest way to do this but it ignores the possible relations between the targets, which can contain valuable information that can boost model performance.

A more elaborated algorithm is based on the use of Stacking-Generalization (Spyromitros-Xioufis et al. (2012)). The process is a straight-forward adaptation of this classic meta-classifier where in the fist step each target is learned separately and these predictions are used with the original features to adjust the meta-classifier at in the second level. The goal is that the model will capture the relation between the targets through their predicted values. However this does not work as well as one would expect for regression problems.

In the same article another approach is based on chaining single target models (Multi-Label chain Classifiers and Regressor chains). The idea is to select a random

permutation of the targets and for each one of them construct a feature vector consisting of the feature variables X_i and the real values of the previous targets in the chain. As expected this approach depends a lot on the permutation so the authors also propose the creation of an ensemble by selecting several chains randomly.

Support Vector Regression has been extended to the MO case in several works (Vazquez and Walter (2003), Sánchez-Fernández et al. (2004)). The fist one uses a method called Cokriging to exploit the correlations between target variables. The main drawback is the need to select a proper covariance model but if this is done well the results are better that in the single output case. The second one introduces a generalization of SVR to the MO case, which involves the definition of a loss function that takes into account the multi-dimensional case.

Another approach is to extend trees, something that has been done in De'Ath (2002) which presents an extension to the CART algorithm (his work is done in the context of relations between species and environments) by changing the impurity measure to take into account the multivariate response, greedily splitting at each node in order to minimize the sum of the squared error. The value at the leafs is defined as the mean value of the instances at the leaf.

Appice and Džeroski (2007) presented a multi-target stepwise model tree induction where at each step the algorithm either splits the nodes or adds a regression variable associated with one of the target variables. This means that instead of storing the mean of the instances at the leaf node we get a regression model for each of the variables. Kocev et al. (2007) on the other hand explores the use of bagging and random forests of regression trees for the multi-output case.

The main drawback of these algorithms are the following. Single target models that simply solve for each variable separately do not take into account relations between the targets, something that is surely present in our domain of interest and that we also want to model. The other models, though admittedly more complex, are in general no better than the single target ones, or require fine tunning of several hyper-parameters at the same time and finding a good combination can be a difficult task. There is also the problem of how to interpret the fitted model. These do not expose explicit relations between the variables since their main purpose is prediction of targets.

An approach that could solve these issues has already been applied to the MOC case. Multi-dimensional Bayesian Network Classifiers (MBCs), which where introduced in Van Der Gaag and De Waal (2006) and extended in Bielza et al. (2011),

tackle the MOC problem using Bayesian Networks q(BNs). This family of models would provide the desired interpretabillity since this is one of the main reasons for using graphical models and can also be used for prediction and other procedures such as sensitivity analysis. Learning BNs however is a hard problem, which we describe below.

2.2. Learning Bayesian Networks

Using BNs would give us a model which satisfies properties 1-3 and 5. As a remainder, BNs (Pearl (2014), Koller and Friedman (2009), Lauritzen (1996)) are graphical models where variables are the nodes of the graph and the edges encode the relations between them. BNs are the special case when the graph is a Directed Acyclic Graph (DAG) which use directed edges (arcs) and prohibit loops in the structure. The direction of the arc denote the way in which Bayes rule is applied to calculate conditional probabilities and make inferences. In BNs variables follow the Local Markov property, meaning that each variable is conditionally independent of the rest given its non-descendants. This means that BNs represent the distribution in a compact way which makes parameter estimation and inference easier. Mathematically it can be written as:

$$p(\mathbf{X} = \mathbf{x}) = \prod_i p(X_i = x_i | \mathbf{pa}(X_i)) \tag{2.2}$$

where $\mathbf{X}$ is a vector of d variables and $\mathbf{pa}(X_i)$ are the parents of variable X_i in the graph. An example of two graphs, one who is a BN and one who is not can be seen in Figure 2.1.

BN learning (Cooper and Herskovits (1991)) can be divided into two problems, structure learning, and parameter learning, where the first is considered somewhat harder than the second. This is a consequence of the number of possible networks that can explain a set of observed data which is superexponential in the number of variables (Robinson (1978)). Actually it can be shown that the general problem of learning the structure of a BN is NP-hard (Chickering (1996)). Learning BNs usually requires three elements to be defined: a search space of possible structures, a scoring function that determines the fitness of each structure and an algorithm that traverses the search space. For a review of parameter learning we refer to Heckerman (1998), Koller and Friedman (2009) and Murphy (2012). In this section we present

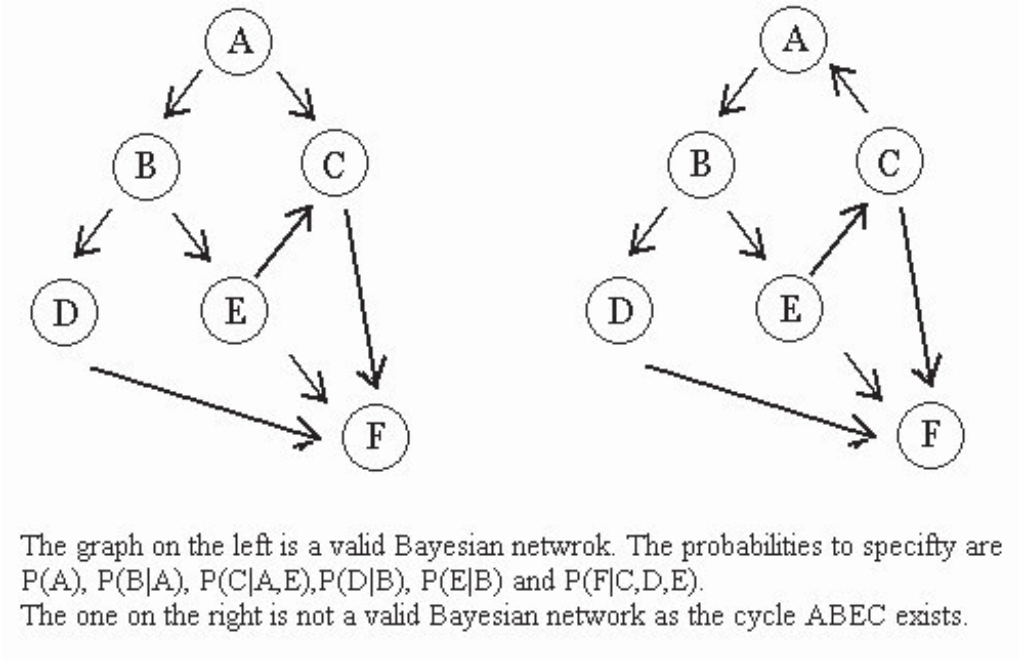


Figura 2.1

a review of BN structure learning so we can determine which one is best suited for our problem.

2.2.1. Search space

Search spaces are usually limited to one of three options: general network structures, equivalent classes of network structures and orderings over network variables. The first one is the most intuitive one but exploring it can be a wast of computational resources since networks that are çlose"have similar or even identical scores if they are in the same equivalence class Madigan et al. (1995) (networks in the same equivalence class should score the same since they encode the same conditional independences). The solution is to move in this space instead (Chickering (2002)). Even in this case though there are many possible structures and so it can still be hard to converge to a good model. Exploring over partial orderings of variables was proposed by Friedman and Koller (2003) in conjunction with Markov chain Monte Carlo (MCMC) where orderings are used to restrict their attention to certain directed graphs at a time. Although their approach is more efficient than the other two variants, it is considerably harder to encode prior knowledge into the distribution (Ellis and Wong (2006)), which is convenient when using MCMC as a search algorithm (see Section 2.2.3.3).

2.2.2. Scoring functions

The second element we need to define is a scoring function. These are usually separated into two groups: information-theoretic and Bayesian scoring functions (Carvalho (2009) is a good review on the subject). The first group is composed of the metrics which are extensions to the basic log-likelihood function that try to limit overfitting: MDL/BIC (Minumum Description Length, Bayesian Information Criterion, Grünwald (2000)), Akaike Information Criterion (Akaike (2011)), normalized minimum likelihood codes (Roos et al. (2008)) and mutual information tests (Campos (2006)). These are general purpose scoring metrics that are not exclusive to the BN setting and so don't take into account some useful properties and assumptions that are often made when learning BNs structures.

The second group contains a more probabilistic approach to the problem of scoring networks and is the result of extensive research into the topic in the early 1990s when Spiegelhalter and Lauritzen (Spiegelhalter and Lauritzen (1990)) defined the problem of learning BN from data and introduced the Global Parameter Independence assumption that allow the likelihood function to be conveniently decomposed. This led to the creation of a series of scoring functions the first of which was the famous K2 score (Cooper and Herskovits (1992)) along with the homonymous algorithm. Buntine (1991) defined the BDeu (Bayesian Dirichlet under score equivalence with prior uniform joint distribution) score which has the equivalent sample size (the number of observed samples prior to seeing any data) as its only hyper-parameter. Heckerman et al. (1994), derive another similar function, this time assuming Score Equivalence, which means that networks that encode the same independence assumptions should score equally and name it (conveniently) the BDe score. This function enables a concise specification of the prior distribution where hyper-parameters are set as a function of the equivalent sample size.

Both functions were generalized in Chickering et al. (1995) as the BD (Bayesian Dirichlet) score, where they also introduced another key assumption called Parameter Modularity which allows to reuse scores for similar networks by only recomputing certain factors (eg. removing an edge only changes the score of the node at the head of that arc). Although this was an important advance, BD score is hard to use because of the number of hyper-parameters that have to be specified.

Even though these scoring functions where developed for the discrete case (and we are concerned with the continuous case), the theoretical results they presented were extended to the continuous case, specifically to Gaussian networks (GNs) where

they were used to define the BGe (Geiger and Heckerman (1994), Kuipers et al. (2014)) score (Bayesian metric for GNs under score equivalence). This is a completely Bayesian score function for GNs that allows the specification of prior probabilities and is computationally efficient.

2.2.3. Search algorithms

The last element needed to perform structure learning is a search algorithm. These are often divided into 3 groups depending on the strategy they employ: greedy hill-climbing algorithms, heuristic optimization and MCMC over some structure space.

2.2.3.1. Greedy hill-climbing and independence tests

Greedy hill-climbing algorithms are the simplest ones as they just move greedily through the search space, always aiming at maximizing the score of the network. This was the procedure used in Cooper and Herskovits (1992) with the K2 score (together they are referred as the K2 algorithm). It requires an order to be given to the variables so that variable X_i can only have parents in the set $\{X_j | j < i\}$. This greedy nature means that the algorithm cannot guarantee it will find a global maximum. To reduce the chance of finding a bad solution, multiple restarts can be used or using a more complex approach, genetic algorithms can be applied to optimize over the orders used (Larrañaga et al. (1996b)).

Margaritis (2003) proposes the grow-shrink algorithm. The idea here is to discover the Markov blanket of each node with statistical independence test and then integrating them to form a network, removing any cycles that remain as a byproduct.

A combination of these ideas is the one found in Tsamardinos et al. (2006), which proposes de max-min hill-climbing algorithm. This method combines search and score along with constrain-based and local search. First they use a method to construct a skeleton that only contains undirected edges. To give orientations to the edges they then perform greedy search ensuring the resulting graph maintains the DAG property.

2.2.3.2. Heuristic optimization algorithms

A second class of methods uses heuristic optimization and genetic algorithms to find an good network structure. The first work that proposed using genetic algo-

rithms was Larrañaga et al. (1996b). They defined the fitness function as the data marginal log-likelihood given the BN which was represented using an adjacency matrix. Operators where defined so the DAG property was never violated, avoiding the need for corrections after each step of the optimization. Larrañaga et al. (1996a) applies genetic algorithms to the task of finding an optimal node order and then applies the K2 algorithm to this ordering. They tested with operators that had been used for the Traveling Salesman Problem with good results.

Tabu search was adapted to this setting in Bouckaert (2001) who described how to use it for structure learning using a tabu list of excluded states so that the state space is better explored. Delaplace et al. (2006) were they used the BDeu score as a fitness function with adjacency matrix representation and they move in equivalence classes to avoid equivalent DAGs. In a second method they dynamically adapt the mutation rate according to the fitness of the population. Simulated Annealing, another popular heuristic algorithm which has connections with MCMC, was used by Heckerman and Geiger (1995) and Wang et al. (2004).

While this class of algorithms is popular, they are also quite general and require being tunned to the particular problem. Also, except for some cases it is difficult to prove any useful theoretical results since they are ad-hoc procedures.

2.2.3.3. MCMC

The third method consists of using MCMC with a proposal distribution over graph structures or node orders to find an optimal solution. MCMC is a general purpose method for sampling from a given distribution (see Section 3.2). Algorithms using this approach are grouped into two classes: those that operate on the space of structures and those that do so in the space of node orderings. This distinction has important computational implications which we will discuss as we explore the different methods.

Working over the structure space was the initial idea in Madigan et al. (1995) where they also defined the basic movements that serve to traverse the space, namely arc addition, removal and inversion while maintaining the DAG restriction. This simple approach proved to not be adequate for high dimensional domains since the number of possible arcs is too great. Also simulations showed that the distribution over possible DAGs is very jagged (Friedman and Koller (2003)), with ridges of equivalent DAGs with equal probability surrounded by "valleys.ºf low probability structures. This means that using these simple movements can make it hard to

escape from high probability regions. Also not all graphs despite having different arcs encode different independences. This means that moving naively might not be productive since you could stay in the same equivalence class of graphs (sets of graphs that encode the same independences).

To solve this last problem Madigan et al. (1995) restricted their focus to what they called essential graphs. These are used to encode all the graphs that belong to the same equivalence class by substituting arcs that can be reversed without changing the independence relations in the graph for edges, and defining moves for both directed and undirected relations. Moving in this space they eliminated spurious moves that don't really generate new conditional independences, accelerating convergence of the underlying Markov chain. A similar idea is to move in the space of junction trees (Giudici and Green (1999), Giudici et al. (2000)). This space has its own complexities since the number of parameters (and thus dimensions) varies, so it was necessary to adapt the reversible-jump MCMC algorithm (Green and Hastie (2009)) to this specific instance.

Another approach is to sample in the space of total orders. This was proposed by Friedman and Koller (2003) as a way to speed up the convergence of the Markov chain. To achieve this they decomposed the problem into two simpler ones: sampling node orders, and determining the best structure for that order (they used the K2 algorithm for this purpose). The authors argued that this space is smaller and more regular thus allowing for faster convergence. On the other hand, the computations associated with calculating the posterior over node orders are highly demanding. Koivisto and Sood (2004) and Parviainen and Koivisto (2009) used dynamic programming to lower the upper bound on time and space requirements for these computations, which was further improved by Koivisto (2012), but it still remained exponential, and it thus becomes useless in domains with more than a few dozen variables.

A problem that was already pointed out in the original article was that DAGs which are compatible with more orders will have higher posterior probability since they could be repeated across all these orderings. They argued that this is not that problematic since sparser DAGs are more desirable anyway and tend to have higher probabilities. A second problem is that only modular priors can be used, which restricts the types of priors that can be used being a notable example the uniform prior over equivalence classes.

Ellis and Wong (2006) tried to solve the bias incurred when sampling node orders by introducing a correction step which scales down the probability of graphs that appear in more orders. They demonstrated that this step accomplishes its intent but the computational cost associated is high which reduces its usefulness. Eaton and Murphy (2007) attempted to improve the computational aspects by combining both approaches (sampling in structure space and node orders) and used the marginalization algorithm of Koivisto (2012) along with the traditional moves in the structure space. They showed experimentally that this approach leads to superior convergence and diminished bias and is thus a good compromise.

Niinimaki et al. (2012) on the other hand worked with a particular type of partial orders called bucket orders. The idea is that the order of nodes in the same bucket cannot be established so the focus is on arcs between nodes in different buckets.

While working in the space of node orders improves convergence, it is not ideal for hight dimensional domains where researchers have a low amount of samples like Biology and Genetics. In these cases the superior computational efficiency and ability to easily specify prior distributions motivated further research in order to improve convergence. This was exemplified in the work of Grzegorczyk and Husmeier (2008), who introduced the more sophisticated New edge reversal (REV) move. Their observation was that arc reversal could be made more useful by resampling the parent sets of the arc's end nodes. This accomplished two things: moves where more drastic and thus increased the probability of escaping a local maximum and secondly all arcs can be reversed since both nodes are orphaned (avoidance of cycles is moved to the parent set selection step, see 3.2.3.1).

More recently, Su and Borsuk (2016) exploited a similar idea and proposed a new move that complements this REV and the classical ones (addition and removal). They proposed a more drastic move called Markov Blanket resamping which selects a node and resamples it's parents and that of his children (though it does not change the children so it's not exactly a resampling of the Markov Blanket) according to its score as given by some predefined metric. The hope is that the chain will "jump"to another high scoring region and skip the low probability "valleysïn between more easily.

In both of the previous works the authors used the BGe score, since it allows to efficiently compute the ratio of the score of two networks when preforming a move, though any metric that obays the Parameter Modularity property can be used (see section 3.2.3).

Masegosa and Moral (2013) tried to restrict possible structures by focusing on samples with a common skeleton of high probability edges. This is a useful way to limit the size of the sampling space if there is a subset of arcs with a much higher probability than the rest but it requires setting a threshold to determine what edges belong to said skeleton which can be hard or useless if no clear value exists, i.e if most edges have similar probabilities or if these are not independent.

Recently, Kuipers and Moffa (2017) proposed to sample in the space of partitions. They define this space so as to not over represent DAGs but so that it groups several network configurations (just as in order MCMC) thus making the space more regular. Nodes in the same partition are not connected to each other and only arcs from lower numbered to higher numbered partitions are allowed, similar to partial order MCMC. Their main achievement is not incurring in the bias of structure MCMC but their proposal is computationally costly and for medium sized networks does not perform better than traditional moves (though the authors postulate that more efficient moves could be designed that are allow faster convergence).

2.3. Conclusion

There is also the problem of selecting a distribution for each of the variables in the domain. If they are discrete then this is a simple task. Discrete distributions can be represented efficiently with Conditional Probability Tables which are tabular mappings from values to probabilities. Any distribution can be represented in this way, which means that conditioning and marginalization can be performed no matter the underlying nature of the distribution. However such a convenient representation does not exist in the continuous case since by definition the space cannot be enumerated. Choosing a parametric family for each variable is also not easy because conditioning and marginalization are not always closed operations, even if we choose the same family for every variable, which is a very strong assumption. The other approach would be discretization, but this introduces another problem which is choosing into how many discrete values we partition the domain. Too many and the computational costs grow out of control; too few leads to a significant loss of information.

We have thus chosen a compromise which is to use Gaussian Networks (GN). This seems like a reasonable first approximation, since they are continuous and equivalent to Multivariate Normal distributions (MVNs) which provides several methodological

advantages (for example the MVN are closed under conditioning and marginalization) that will be discussed in chapter 3. Also we can take advantage of the BGe metric that we discussed earlier.

Selecting a type of search algorithm is more complex. Greedy hill-climbing seems like a poor choice for our problem. As we have already explained in cases like this ones, where the amount of observed data is small relative to the dimensionality of the domain, models computed in this fashion which correspond to local maxima tend to not be very good. Posterior distributions tend to be multi-modal and the evidence is not identifiable. The choice is then between heuristic optimization algorithms and MCMC. Both of these enable a less naive exploration of the search space. The latter though has several practical and theoretical advantages.

Remember that we still need to satisfy Property 4. This is essential since with a low number of samples, barring some prior knowledge that exclude most possible models, the posterior is relatively flat (meaning no model is considerably better than the others). this changes as we see more data and it becomes peaked around the MLE. A way to overcome this is to average the predictions of several models in order to compensate for our lack of data.

One approach could be to use some optimization algorithm that uses some previously defined starting condition like an initial graph or order and perform random restarts and then use the combination of each of the trained models to make predictions. Still this presents the question of how many models, and how do we know that they are sufficiently distinct. Another way is to use the samples produced by the random chain of the MCMC sampling, which if done correctly approximate the true posterior, to obtain the models that will constitute the ensemble. Also since we are using a Gaussian distribution for each variable, we can use the fact that we know the posterior predictive distribution in a Bayesian setting so that we don't need to find the precise values of the parameters for each one of them. Instead we can use the fact that GNs and Multivariate Normals are equivalent (Koller and Friedman (2009)) use the data to obtain a posterior over network parameters and use the predictive distribution (which in this case is a Multivariate T distribution) to perform regression.

Combining models in this way allows to compensate for the lack of data. Furthermore, unlike other models when combined BNs can still be interpreted since we can still quantify the importance of an edge (relationship) by simple techniques like

approximating the posterior probability of each of them by simply counting in how many models they appear.

The final concern would be the choice of search space. As we said, sampling over the space of orders is problematic for this setting, so we opt for working with the space of structures. However we wish to speed up the convergence of the Markov chain. First of all we will use the REV move as found in Grzegorczyk and Husmeier (2008) instead of traditional edge reversal. And secondly, we propose to perform the sampling over the space of CB-decomposable graphs as a means of speeding up convergence by performing more drastic changes to the graph structure. To achieve this first we also need to adapt MBCs to the regression setting.

3

Background

Before we can define our model, we first need to give some background knowledge on the theory involved. Namely we will expand the relevant concepts concerning Gaussian Networks, the Multivariate Gaussian distribution and MCMC sampling.

3.1. Gaussian Networks and Multivariate Normal distributions

Gaussian Networks are one of the basic BN models along with the general discrete BNs. This is due to the fact that assuming that the distribution of the variables are linear Gaussians is a simple way of assuring that operations like conditioning and marginalizing are closed (they produce another Gaussian as a result) which is not generally the case for other distributions. The ubiquity of the distribution in many fields of research also make particularly interesting due to its wide range of applications.

Definition: Gaussian Network (GN) A Gaussian Network (also Gaussian Bayesian Network Koller and Friedman (2009) or Gaussian Belief Network Geiger and Heckerman (1994), abbreviated as GN) is a BN where all the variables X_i are continuous and their conditional probability distribution (CPD) are linear Gaussians such that for each of them we have:

$$p(X_i|\mathbf{pa}(x_i)) = \mathcal{N}(\mu_i + \sum_{X_j \in \mathbf{pa}(x_i)} b_{ij}(x_j - \mu_j), \sigma_i^2) \tag{3.1}$$

where μ_i is the unconditional mean of variable X_i, σ_i is its conditional variance, b_{ij} is the influence of each of its parents X_j and μ_j are their corresponding means.

Informally, this means that in GNs each conditional variable is normally distributed with a mean that is influenced by the values of its parents. The degree of the influence is determined by the parameter b_{ij} which graphically represents the existence of an arc from variable X_j to X_i whenever $b_{ij} \neq 0$. The values of the parameter vectors μ, σ for a set of data D with n rows representing the samples of d dimensions can be obtained with the equations:

$$\mu_i = \frac{1}{n}\sum_{l=1}^{n} D_l \tag{3.2}$$

$$\sigma_i^2 = Cov_D[X_i, X_i] - \sum_{X_j, X_k \in \mathbf{pa}(X_i)} b_j b_k Cov_D[X_j, X_k] \tag{3.3}$$

In the case of the coefficients b, these can be obtained by solving a system of $|pa(X_i)| + 1$ linear equations that are the product of taking the derivative of the likelihood respective to each of the coefficients for each of the variables .

An interesting result is the fact that GNs are equivalent to the class of Multivariate Normal distributions (MVN). Shachter and Kenley (1989) formulated a recursive equation that transforms from one representation to the other. This means that the same results from probability theory that are valid for MVNs also apply to GNs. First let us define a MVN with mean vector μ of d elements and precision matrix $\Sigma = \Lambda^{-1}$ of size $d \times d$ as:

$$p(\mathbf{x}|\mu, \Sigma) = \frac{|\Lambda|^{\frac{1}{2}}}{(2\pi)^{\frac{d}{2}}} \exp[-\frac{1}{2}(x-\mu)\Sigma^{-1}(\mathbf{x}-\mu)] \tag{3.4}$$

Given both definitions for the distribution one can convert from the GN representation to MVN in the following way, where $\Lambda[i]$ represents the upper left square submatrix of size $i \times i$:

$$\Lambda = \begin{cases} \Lambda[1] & = \frac{1}{\sigma_1^2} \\ \Lambda[i] & = \begin{bmatrix} \Lambda[i-1] + \frac{b_i b_i^T}{\sigma_i^2} & -\frac{b_i^T}{\sigma_i^2} \\ -\frac{b_i}{\sigma_i^2} & \frac{1}{\sigma_i^2} \end{bmatrix} \end{cases} \tag{3.5}$$

This ability to change between both representations allows us to work with the most convenient for any given task. For example, analyzing the relation between variables in a given domain is easier and more intuitive in the graphical representation. On the other hand scoring a given network, which is necessary for structure learning is much more convenient in matrix form (see Section 3.1.3).

Other important operations on MVN are computing the conditional and marginal distributions. Given a MVN with mean vector μ and covariance matrix $\mathbf{\Sigma}$, it can be showed (DeGroot (2005),page 51) that the conditional distribution of variables in $\mathbf{X}_i$ given that variables $\mathbf{X}_j = \mathbf{x}_j$ is a MVN such that :

$$f(\mathbf{X}_i|\mathbf{x}_j) = \mathcal{N}(\mathbf{x}_i|\mu_{i|j}, \mathbf{\Sigma_{i|j}}) \tag{3.6}$$

$$\mu_{i|j} = \mu_i + \mathbf{\Sigma}_{ij}\mathbf{\Sigma}_{jj}^{-1}(\mathbf{x}_j - \mu_j) \tag{3.7}$$

$$\mathbf{\Lambda}_{i|j} = \mathbf{\Lambda}_i - \mathbf{\Sigma}_{ji}\mathbf{\Lambda}_{ii}\mathbf{\Sigma}_{ij} \tag{3.8}$$

where $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$ and the subscripts denote the sub-matrices of the given rows and columns. Marginalization is very easy if we use $\mathbf{\Sigma}$ instead of $\mathbf{\Lambda}$:

$$f(\mathbf{X}_i) = \mathcal{N}(\mathbf{x}_i|\mu_i, \mathbf{\Sigma}_{ii}) \tag{3.9}$$

3.1.1. Bayesian Probability Theory and the MVN

Before describing the BGe score we will first develop the basic notions of Bayesian probability theory. In this framework, uncertainty related to a hypothesis (like the true parameters of the distribution that produced some data) is encoded using a probability distribution. Initially we may have some assumption about their value and this is our prior distribution. As we observe data we change our beliefs and obtain a posterior (which in turn may become a prior when more data is seen). As the amount of data grows our uncertainty diminishes and the posterior centers around the true value.

In contrast, classical or frequentist statistics revolves around sampling distributions instead of posteriors, which is the distribution that an estimator (a function that computes some statistics from the data) when applied to different data sets that are assumed to have been generated from the same model which is assumed to be fixed an unique. This is the exact opposite view of the Bayesian approach Murphy (2012), which has important consequences for prediction tasks and modeling uncertainty in complex domains.

A distinctive element of Bayesian Statistics is the needed to define a prior distributions. This is the cause of the main controversies between Bayesians and Frequentists. Nonetheless they are very useful since they allow to encode prior knowledge

about a given domain before we have seen any data or when it is scarce and determining a good MLE is thus difficult. Specifying a prior distribution over parameters can help find a better estimate if our prior assumptions are sufficiently correct (though this is a difficult task in most practical domains). Formally, we can define the posterior over parameters θ given the data D, $p(\theta|D)$, with a prior distribution as $p(\theta)$:

$$p(\theta|D) \propto p(D|\theta)p(\theta) \tag{3.10}$$

This means that the posterior distribution is proportional to a multiplication of the likelihood $p(\theta|D)$ and prior distribution $p(\theta)$. It is useful for parameter estimation if this product has a closed form solution. Priors whose multiplication by some probability distribution remain in the same family are called conjugate priors. In the case of data that follows a MVN distribution, the conjugate prior of the mean μ and precision matrix $\mathbf{\Lambda}$ is the Normal-Wishart (NW) distribution:

$$p(\mu, \mathbf{\Lambda}) = \mathcal{N}(\mu|\mathbf{m}_0, k_0 * \mathbf{\Lambda})\mathcal{W}(\mathbf{\Lambda}|\mathbf{S}_0, \nu_0) \tag{3.11}$$

where m_0 is the prior mean, S_0 is the prior scatter matrix, k_0 is the equivalent sample size which controls the strength of the prior mean and ν_0 controls the strength of the prior scatter matrix. The Wishart distribution is defined as:

$$\mathcal{W}(\mathbf{\Lambda}|S, \nu) = \frac{1}{Z}|\mathbf{\Lambda}|^{(\nu-d-1)/2}\exp[-\frac{1}{2}tr(\mathbf{\Lambda}\mathbf{S}^{-1})] \tag{3.12}$$

where d is the dimension of ν, $\mathbf{\Lambda}$ is a matrix of size $d \times d$, $tr(.)$ is the trace of the matrix and Z is the partition function (or normalizing constant) which is defined in terms of the multivariate gamma function as:

$$Z = 2^{vd/2}\Gamma_d(\nu/2)|\mathbf{S}|^{\nu/2} \tag{3.13}$$

$$\Gamma_d(x) = \pi^{d(d-1)/4}\prod_{i=1}^{d}\Gamma(\frac{x+(1-i)}{2}) \tag{3.14}$$

This prior can be used with the MVN to obtain a posterior over the parameters efficiently when presented with new observations since it can be proven that multiplying both distributions reduces to updating the parameters of the NW. Since the NW prior in conjugate to the MVN distribution, the posterior is also a NW with updated parameters k_n, ν_n, $\mathbf{m}_n$, $\mathbf{\Lambda}_n$. The formulas used to obtain these updated values are (DeGroot (2005); Press (2012); Murphy (2012)):

$$k_n = k_0 + n \tag{3.15}$$

$$\nu_n = \nu_0 + n \tag{3.16}$$

$$\mathbf{m}_n = \frac{k_0\mathbf{m}_0 + n\overline{\mathbf{x}}}{k_n} \tag{3.17}$$

$$\mathbf{S}_n = \mathbf{S}_0 + \mathbf{S}_{\overline{x}} + \frac{k_0 n}{k_n}(\overline{\mathbf{x}} - \mathbf{m}_0)(\overline{\mathbf{x}} - \mathbf{m}_0)^T \tag{3.18}$$

where $k0$ and ν_0 is our confidence in the prior mean and covariance respectively, S_0 is the prior scatter matrix, $\overline{x}$ is the sample mean and $S_{\overline{x}}$ is the sample scatter matrix (the covariance multiplied by $n-1$). The scatter matrix is used instead of the covariance since it's easier to update.

Using these formulas it is straightforward to do online updates to the posterior as new information is found. Another useful property of the MVN is that its posterior predictive distribution over variables $\mathbf{X}$ found by integrating over the possible values of the posterior reduces to a Multivariate Student's T(MVT) with mean μ, scale matrix $\mathbf{T}$ and ν degrees of freedom:

$$\tau(\mathbf{x}|\mu, \mathbf{T}, \nu) = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})} \frac{|\mathbf{T}|^{-1/2}}{\nu^{d/2}\pi^{d/2}} \Big[1 + \frac{1}{\nu}(\mathbf{x}-\mu)^T\mathbf{T}^{-1}(\mathbf{x}-\mu)\Big]^{-\frac{v+d}{2}} \tag{3.19}$$

From this equation we can obtain the conditionals as (Ding (2016)):

$$p(\mathbf{x}_1|\mathbf{x}_2) \sim \tau_{d_1}(\mathbf{x}_1|\mu_{1|2}, \frac{\nu + dist(\mathbf{x}_1, \mu_1, \mathbf{T}_{11})}{\nu + d_1}\mathbf{T}_{11|2}, \nu + d_2) \tag{3.20}$$

$$\mu_{1|2} = \mu + \mathbf{T}_{12}\mathbf{T}_{22}^{-1}(\mathbf{x}_2 - \mu_2) \tag{3.21}$$

$$dist(\mathbf{x}, \mu, \mathbf{T}) = (\mathbf{x}-\mu)\mathbf{T}^{-1}(\mathbf{x}-\mu) \tag{3.22}$$

In the case of GN, we define a structure that makes relations between variables explicit so even though the sample mean is the same (it's the unconditional mean in each case), the sample covariance does not translate directly to that of the GN in MVN form. In this case we need to determine the conditional variances of each variable given its parents in the structure. This is easy by using the traditional formula for the conditional variance 3.2. Using these covariances we simply use the recursion 3.5 to obtain a MVN representation of the network. When predicting we condition on the relevant variables using equation 3.6.

3.1.2. Multidimensional Bayesian Network Classifiers and Class-Bridge Decomposition

Multidimensional Bayesian Network Classifiers (MBCs) are a special class of BN introduced by Van Der Gaag and De Waal (2006) and expanded in Bielza et al. (2011), where variables are separated into two subsets one of feature and one of targets. Variables in both subsets can be mutually related between each other and across subsets but the direction of the relation can never go from features to targets. Also target variables can be further separated into mutually independent subsets given the values of the features.

Definition: Multi-Dimensional Network Bayesian Regressor (MBNR) A MBNR is a BN denoted by B=(G, θ) where each variable is continuous, $G = (V, A)$ is a DAG such that $V = V_Y \cup V_X$ ($V_Y \cap V_X = \emptyset$) where Y ($t = |Y|$) are the nodes associated with the target variables and V_X ($f = |V_Y|$) are the ones associated with features. The set of arcs A can also be partitioned into 3 subsets:

- $A_Y \subseteq V_Y \times V_Y$ is composed of the arcs between target variables and is called the target subgraph: $G_Y = (V_Y, A_Y)$.
- $A_F \subseteq V_X \times V_X$ is composed of the arcs between feature variables and is analogously called feature subgraph: $G_X = (V_X, A_X)$.
- $A_{XY} \subseteq V_Y \times V_X$ is composed of the arcs from target variables to feature variables. This is the bridge subgraph denoted: $G_{XY} = (V, A_{XY})$ (a concept introduced in Bielza et al. (2011)).

As noted in Van Der Gaag and De Waal (2006) the connectivity of the feature subgraph is irrelevant to prediction of the target variables.

Another important concept is that of Class-Bridge decomposition (CBd), which was introduced in Bielza et al. (2011) for the context of multidimensional classification problems, where the target variables are discrete, though using it for the Gaussian case is straight forward since it does not depend on the nature of the variables:

Definition: Degree k Class-Bridge decomposable MBNR (k-CBd) Let B be a MBNR with target and bridge subgraphs G_Y and G_{XY} as defined in the previous definition, then we say that B is CB-Decomposable with degree k (k-CBd) if:

- $G_Y \cup G_{XY}$ can be decomposed into its connected maximal components as $\cup_{i=1}^{k} = (G_Y^i \cup G_{XY}^i)$.
- $Ch(V_Y^i) \cap Ch(V_Y^j) = \emptyset$ for $i, j = 1, \ldots, k$, $i \neq j$ and $Ch(.)$ denotes the set of all children of the given nodes in G_{XY}.

This characterization gives rise to a useful property by applying it in conjunction with the independence assumptions of GNs:

Theorem 3.1.1 *Given a k-CBd MBNR where $\mathcal{I} = \prod_{c \in V_i} \Omega_c$ represents the sample space associated with V_i then:*

$$f(y_1, \ldots, y_t | x_1, \ldots, x_f) = \prod_{i=1}^{k} \prod_{Y \in V_Y^i} f(y|\mathbf{pa}(Y)) \prod_{X \in Ch(Y) \cap V_X} f(x|\mathbf{pa}(X)) \qquad (3.23)$$

where Y are the target variables in the set V_Y^i and X are the features in the set V_X^i of component i. This result is easily applied to Gaussian Networks with the traditional representation or with the matrix form, using the recursive formula 3.5 to obtain the other representation.

3.1.3. BGe score

In order to learn GN structures we need to use a scoring function. As we said in Chapter 2, BGe was first presented in Geiger and Heckerman (1994). The BGe relies on a set of assumptions about the structure of the network and Bayesian probability theory in order to score network structures in an efficient and coherent way. Next we present these assumptions:

Definition: Score Equivalence Given two network structures G_1 and G_2 which are isomorphic, i.e., they represent the same assertions of conditional independence, then the scores given to both structures must be equal.

Definition: Parameter Independence For every GN G, $f(\theta, G|\xi) = \prod_{i=1}^{d} f(\theta_i, \mathbf{b}_i|\xi)$ where θ_i are the parameters of variable X_i, $\mathbf{b}_i$ is the vector of weights that indicate the influence of its parents and ξ is our prior knowledge.

Definition: Parameter Modularity If X_i has the same parents in two Gaussian Networks G_1 and G_2 then $f(\theta_1, \mathbf{b}_1|\xi) = f(\theta_2, \mathbf{b}_2|\xi)$.

These assumptions serve several purposes. The first ensures that graphs that represent the same conditional independences are treated the same, avoiding arbitrary preferences. The second and third assumption means that we can score local structures separately and that we don't need to recalculate scores for similar networks completely, which is useful when exploring the structure space as in MCMC.

Using these properties and assumptions Heckerman et al. (1994) derive an expression for the score of a given network if the associated distribution is a MVN with a NW prior:

$$f(D|G,\xi) = \prod_{i=1}^{d} \frac{f(D_{X_i,\mathbf{pa}(X_i)}|G,\xi)}{f(D_{\mathbf{pa}(X_i)}|G,\xi)} \tag{3.24}$$

where $D_{X_i,\mathbf{pa}(x_i)}$ is the dataset restricted to the columns of variable X_i and its parents.

Using equation 3.19 written as in Box and Tiao (2011) we can obtain an expression for the likelihood of a sample after seeing the first $i-1$ samples as:

$$f(y|\xi) = \prod_{i=1}^{N} f(y_i|y_0,\ldots,y_{i-1},\xi) = \Big(\frac{k_0}{k_n}\Big)^{\frac{d}{2}} \frac{\Gamma_d(\frac{\nu_n}{2})}{\pi^{\frac{dN}{2}}\Gamma_d(\frac{\nu_0}{2})} \frac{|\mathbf{S}_0|^{\frac{\nu_0}{2}}}{|\mathbf{S}_n|^{\frac{\nu_{n+1}}{2}}} \tag{3.25}$$

where y_i is the i-th sample in the data.

Using this expression to obtain the values of the factors in equation 3.24 we obtain the expression in 3.26. This equation needs to be corrected for the fact that we use a subset of the variables when scoring, as opposed to when we compute the parameters of the NW prior where we use all of them, as noted in the addendum to the BGe score of Kuipers et al. (2014) wich follows a theorem in Press (2012)). This means that is we set D_Y to be the dataset restricted to the columns in set Y, then we obtain the following expression:

$$f(D_Y) = \Big(\frac{k_0}{k_n}\Big)^{l/2} \frac{\Gamma_l(\frac{\nu_n-d+l}{2})}{\pi^{lN/2}\Gamma_l(\frac{\nu_0-d+l}{2})} \frac{|\mathbf{S}_0^Y|^{(\nu_0-d+l)/2}}{|\mathbf{S}_n^Y|^{(\nu_n-d+l)/2}} \tag{3.26}$$

Setting $Q = \{x_i, \mathbf{pa}(X_i)\}$ and $P = \{\mathbf{pa}(X_i)\}$ with $\mathbf{pa}(X_i)$ the parents of X_i in G, $p = |P|$ and $|Q| = p+1$ and $\mathbf{S}^Y$ as the submatrix of $\mathbf{S}$ restricted to the rows and columns of the variables in set Y, we can substitute in 3.24 and taking into account that each factor in the fraction differs in only one variables the expression inside the

multiplication can be further simplified as:

$$\frac{f(D_Q|G,\xi)}{f(D_P|G,\xi)} = \left(\frac{k_0}{k_n}\right)^{1/2} \frac{\Gamma_l(\frac{\nu_n-d+p+1}{2})}{\pi^{n/2}\Gamma_l(\frac{\nu_0-d+p+1}{2})} \frac{|\mathbf{S}_0^Q|^{(\nu_0-d+p+1)/2}}{|\mathbf{S}_0^P|^{(\nu_0-d+p)/2}} \frac{|\mathbf{S}_n^P|^{(\nu_n-d+p)/2}}{|\mathbf{S}_n^Q|^{(\nu_n-d+p+1)/2}} \tag{3.27}$$

Multiplying this final expression by the prior probability $p(G|\xi)$ of each network structure we obtain the final score for any given network.

3.2. Markov chain Monte-Carlo (MCMC)

We will now describe MCMC algorithms which will be used for learning the structure of Bayesian Networks in conjunction with the BGe score described before. First as a quick aside, Monte-Carlo simulation is a general purpose technique used for approximating integrals with a finite number of samples when exact computation is infeasible. To do this, the idea is to draw random samples with a random variable, perform some computation with them and aggregating the results.

A textbook (though simple) example is determining the value of π. This value is the ratio between the circumference and the diameter of a circle. This value can ba approximated by dividing the area of the unit square and the inscribed circle. Drawing samples uniformly inside the former and dividing by number of points also inside the latter we obtain 4 times the value of π which we use as approximation. Monte Carlo simulation is tightly related to the theory of Markov chains and sampling and has been used widely used since computer came into prominence in the 1950s.

In the following subsections we will introduce the basic concepts related to Markov chains and the Metropolis-Hastings algorithm, which is our choice of sampling algorithm. For this discusion s_i will be taken to be the sate of the chain at step i, S_i is the distribution over possible states at step i and X the space of possible values.

3.2.1. Markov chains

The idea in MCMC is to approximate a given distribution using a random walk through a Markov chain that has the desired distribution as its stationary distribution.

Definition: Markov chain A Markov chain is a stochastic process where the probability of moving to a given s_i state only depends on the current state s_i and not

on the whole history of states that have been visited, i.e., it satisfies the Markov property:

$$p(s_{i+1}|s_i, \ldots, s_0) = p(s_{i+1}|s_i) \tag{3.28}$$

In discrete domains the distribution over states is denoted by a vector π such that $\pi_i \geq 0$ and $\sum_i \pi_i = 1$. The probability of each transition (in the case of discrete spaces) is called the transition matrix (denoted $\mathbf{P}$). There are several properties of Markov chains which are useful in order to determine if the chain converges and the rate at which it achieves this, something that is necessary for the method to be of any practical use. These are (Robert (2004)):

1. Reducibility: A Markov chain is said to be irreducible if it is possible to get from any state to any other state in a finite number of steps:

   $$p(S_k = x'|S_0 = x) > 0, \forall x, x' \in X \tag{3.29}$$

   Where X is the state space and k is any integer greater than zero.

2. Periodicity: A Markov chain is said to be aperiodic if all sates are aperiodic. A given state is periodic with period k if any return to said state must occur in steps that are multiples of k. A state is then aperiodic if $k = 1$. Notably an irreducible chain only needs one aperiodic state to be aperiodic.

3. Transience and Recurrence: A state i is transient if starting at state i there is a nonzero probability of not returning. If on the other hand we return with probability 1, then the state is said to be recurrent. If the mean hitting time (the mean number of steps until we return to the state) is finite we say the state is positive recurrent.

4. Ergodicity: An ergodic Markov chain is one that is irreducible and where every state is ergodic. A state is ergodic if it is aperiodic and positive recurrent. An important theorem states that if the chain is irreducible with a finite number of states and has one aperiodic state then the chain is ergodic (Robert (2004)).

5. Reversibility: A reversible Markov chain is one where there is a probability distribution π over states such that:

   $$\pi_x * p(S_{i+1} = x'|S_i = x) = \pi_{x'} * p(S_i = x|S_{i+1} = x') \tag{3.30}$$

   for all n and all states $x_i, x_j \in X$. This is known as the **detailed balance condition.**

The usefulness of a Markov chain is that under certain conditions it converges to a stationary distribution:

Definition A distribution π is said to be stationary with respect to some chain with transition matrix $\mathbf{P}$ if it is unchanged by the multiplication of the two:

$$\pi\mathbf{P} = \pi \tag{3.31}$$

With these properties several other results can be formalized. For example if a Markov chain is aperiodic and irreducible then there is a unique stationary distribution π. Also an upper bound on the convergence speed can be determined though this tends to be a large value and hence it is only of theoretical value in most cases. A more useful property is the following:

Theorem 3.2.1 *Stationary distribution of a Markov chain: If a Markov chain satisfies the detailed balance condition for some distribution π and the transition matrix P then π is the unique stationary distribution of the Markov chain.*

These properties have been described for discrete domain since it is easier to understand but can be generalized to include non transition matrices where the states are not explicitly enumerated. This is the case when we work in continuous or discrete domains that are too large for explicit explicit enumeration of the transition probability. In these cases, the transition matrix $\mathbf{P}$ is substituted by a transition kernel $K(S_{i+1}|S_i)$ such that $K(x'|x) \geq 0$ for all states and $\int K(x'|x)dx' = 1$. The same definitions and theorems apply to this situation (Robert (2004)). This definition will be used throughout since the space of possible DAGs, even though it is a discrete one, is sufficiently large that a parametric kernel is needed. Of course when working with a kernel, the distribution π must also be substituted with a parametric density function, which we will denote as p.

A useful property for kernels is that they allow for combinations to form more complex transition matrices:

Definition Given a set of weights p_k such that $\sum_k p_k = 1$, then mixture kernel K^+ is a convex combination of kernels such that:

$$K^+(S_{i+1}|S_i) = \sum_k p_k K_k(S_{i+1}|S_i) \tag{3.32}$$

These kernels have a useful property which we summarize with the following theorem (Tierney (1994)):

Theorem 3.2.2 *Let K^+ be a mixture kernel as that of equation 3.32. If all the component kernels have the same stationary distribution π and one of the component kernels K_l is ergodic then the mixture kernel K^+ is also ergodic and converges to the stationary distribution π.*

3.2.2. Metropolis-Hastings algorithm

In order to actually perform the random walk one must use a concrete algorithm that in some way takes samples from the desired target distribution and uses them as an approximation. One of the most widely used is the Metropolis-Hastings (MH) sampler (Metropolis and Ulam (1949), Hastings (1970), Robert (2004), Murphy (2012)). This algorithm relies on the definition of a transition kernel that uses a proposal distribution, which as it name implies proposes a new state s_{i+1} given the current one s_i. It turns out that if defined correctly this method ensures convergence to the target posterior.

The algorithm first defines a proposal $q(s_{i+1}|s_i)$ and uses it to sample a new state from the previous one. This new state is then accepted with a probability that depends on the ratio of probabilities between the two states according to the posterior:

$$\alpha(s_{i+1}|s_i) = \min\left\{1, \frac{p(s_{i+1})}{p(s_i)}\right\} \tag{3.33}$$

where $p(s)$ is the probability of the state according to the target stationary distribution.

If the proposal is asymmetric ($q(s_i|s_j) \neq q(s_j|s_i)$) then we need to adjust the term with the Hastings correction coefficient giving the following general expression where we also substitute $p(x)$ for the unnormalized probability $\widetilde{p}(s)$ since the normalization constants cancel out and thus they do not have to be computed and $q(s_i|s_j)$ is the proposal distribution:

$$\alpha(s_{i+1}|s_i) = \min\{1, r(s_{i+1}|s_i)\} \tag{3.34}$$

$$r(s_{i+1}|s_i) = \frac{p(s_{i+1})q(s_i|s_{i+1})}{p(s_i)q(s_{i+1}|s_i)} = \frac{\widetilde{p}(s_{i+1})q(s_i|s_{i+1})}{\widetilde{p}(s_i)q(s_{i+1}|s_i)} \tag{3.35}$$

Proposal distributions must give a nonzero probability of moving to each of the states that have nonzero probabilities in the target (or stationary) distribution in order to be admissible. For practical purposes it is necessary that it also covers the

density regions in an appropriate way. If it gives too much probability to a given mode, then the rest of the space will probably go unexplored. Too much density for low probability regions will yield a very low mixing rate for the chain, with many rejected samples which means we have to sample for a longer period. Intuitively if the chain converges then the sampler should visit each state (in the limit) a number of times that is proportional to its marginal probability (this is why it is a Monte Carlo method).

The transition kernel defined by the Metropolis-Hastings algorithm can thus written as (Murphy (2012)):

$$K(s_{i+1}|s_i) = \begin{cases} q(s_{i+1}|s_i)r(s_{i+1}, s_i) & \text{if } s_{i+1} \neq s_i \\ q(s_i|s_i) + \sum_{s_{i+1} \neq s_i} q(s_{i+1}|s_i)(1 - r(s_{i+1}, s_i)) & \text{otherwise} \end{cases} \tag{3.36}$$

which follows from the fact that we propose a move with probability $q(s_{i+1}|s_i)$ and accept it with probability $r(s_{i+1}, s_i)$ in the first case, or in the second case we either propose to stay in the same state which is always accepted or the move is rejected. The reason MH works is the following theorem whose proof can be found in Murphy (2012) page 854:

Theorem 3.2.3 *Given a target distribution p^* and the transition kernel induced by the MH algorithm using q, if the underlying chain is ergodic and irreducible then p^* is its unique stationary (limiting) distribution.*

These theorems will be useful in the next section when we apply MCMC for structure learning of Bayesian Networks.

3.2.3. Using MCMC for structure learning of BNs

The idea of using MCMC for structure learning of BNs is to define a target distribution that gives more probability to better structures and a kernel so that the target is also the stationary distribution. In this way we explore the space given preference to networks that we have determined are better than others. Using the BGe score as a metric to determine how good a given network is and a series of moves to transition from one state to another it is possible to achieve this. The other main advantage is that we can approximate the posterior distribution of network structures. This not only allows to analyze the discovered relations more rigorously

by using a better estimate of the true posterior distribution, but we may use different samples from the chain to create combinations of models in order to improve prediction tasks.

In the case of structure learning the values states of the chain s_i and s_j mentioned in 3.33 are graph configurations so in order to make this fact explicit we will denote the states as $G_{<subscript>}$ following standard graph theory notation. The distribution p is the marginal probability of each graph given the data, $\widetilde{p}$ is the corresponding unnormalized probability (the conditioning on the data will be dropped for brevity) and q is the probability of each move. Formally, if G_i is the current state of the chain:

$$p(G_{i+1}|D) = \frac{p(D|G_{i+1})p(G_{i+1})}{p(D)} \tag{3.37}$$

$$q(G_{i+1}|G_i) = \frac{1}{|Nbh(G_i)|} \tag{3.38}$$

where $Nbh(.)$ is the neighborhood of graph G_i in the state space and depends on the allowed moves. The first factor in the numerator of 3.37 is the likelihood term computed as the score of the network, the second is the prior probability of each graph (in the case of a uniform distribution it can be omitted). The denominator can be excluded since it cancels out when computing the ratio.

The moves most commonly used when defining a proposal q are the following:

- Adding an arc $u \to v$ to the graph. This arc must no violate the DAG property.
- Removing an arc $u \to v$. This move can always be made if there are arcs in the graph.
- Reverse the direction of some arc $u \to v$. This can be reduced to a removal followed by addition of the reversed arc.

These moves were introduced by Madigan et al. (1995). Combining them with equations 3.37 and 3.38, and a score function that conforms to the Parameter Independence property to compute r we get:

$$r(G_{i+1}|G_i) = \frac{\exp[\Psi(\pi_n^{i+1})|D)]p(G_{i+1})|Nbh(G_i)|}{\exp[\Psi(\pi_n^{i+1}|D)]p(G_i)|Nbh(G_{i+1})|} \tag{3.39}$$

where $\Psi(.)$ is the log score associated to the parent of variable X_n in graph G_i denoted π_n^{i+1}, which was the one that was modified by the arc addition or removal. Because we

assume that the score function obeys the Parameter Independence assumption, all other parent sets cancel out since they have not changed and only the one associated with the head of the arc needs to be computed (Geiger and Heckerman (1994)). The kernel for these moves is then:

$$K(G_{i+1}|G_i) = q(G_{i+1}|G_i)\alpha(G_{i+1}|G_i) \tag{3.40}$$

By construction this kernel satisfies the detailed balance condition so under ergodicity it will converge to the stationary distribution p:

$$p(G_{i+1}|D)K(G_i|G_{i+1}) = p(G_i|D)K(G_{i+1}|G_i) \tag{3.41}$$

We note that in this kernel reversal has not been included. This move is generally not used in the naive form since it has several disadvantages. First as noted in Robert (2004) and Friedman and Koller (2003) simple arc operations including the above mentioned (and thus reversal), produce reduced mixing rate and slow the convergence of the chain due to the fact that modifications are small and adjacent graphs have similar scores. In addition to this reversing an arc can create an isomorphic one which is undesirable since it is in essence the same graph. In addition, reversal is a computationally expensive move since not all arcs can be reversed without violating the DAG property. Giudici and Green (1999) introduced the non-covered arc to solve the former problem as well as optimizations in the form of an ancestor matrix which they efficiently update so as to easily find the arcs that can be reversed. Still the convergence problem cannot be solved in this way.

3.2.3.1. The REV move for arc reversal

This is what the new edge reversal (REV) move of Grzegorczyk and Husmeier (2008) attempts to fix (though the authors use the term "new edges reversal", the article is from 2008). The idea is that instead of reversing an arc that does not violate the DAG property, which is constrained by the parent sets of both nodes, we can reverse any arc and we resample the parent sets of both nodes. The parent sets are restricted to those that contain the reversed arc and exclude the children of both nodes (including the head of the new arc which is now a children of the former head). This ensures that the DAG property is upheld and we can reverse any arc.

Since the moves involve several removals and additions of arcs it implies more drastic moves across the search space that allow the random walk to escape from

local maximums more easily speeding up the convergence rate. In order to facilitate this, sampling is not done uniformly, instead doing in accordance to their respective scores. Addition and removal are still needed since this move is not irreducible (it can reach every state) and must be embedded in an irreducible framework like the one provided by these basic moves (it is easy to see that starting in any state we can reach any other by removing arcs and the adding back the new ones).

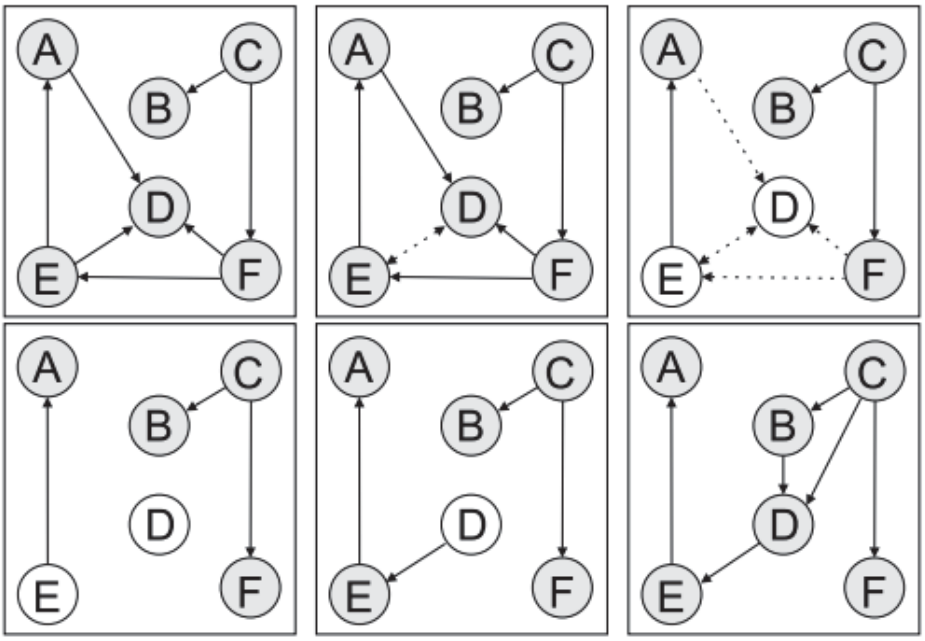


Figura 3.1: The steps of the REV move from left to right and top to bottom. In the first figure the original graph. In step two the arc between $E \rightarrow D$ is selected for reversal. Both nodes are then orphaned, leaving them only connected to their children in step 3 (bottom-left figure). The arc $D \rightarrow E$ is added, notice that since parent sets where removed it does not violate the DAG property. Finally in this intermediate graph, parent sets are sampled for both nodes according to the restriction that must be enforced, namely the graph must be a DAG so all descendants are omitted and D is forced to be in the new parent set of node E.

The proposal distribution thus depends on the number of existing arcs that we can reverse such that the DAG property is not violated (denoted by $|E|$) and the probability of the newly selected parent set for each node. In order to sample a parent set we must then compute the normalizing constant or partition function which implies summing over all possible parent sets. We note that in practice the number of possible parent sets must be limited since otherwise computation of the

partition function becomes unfeasible. This is usually done by imposing a fan in restriction on the number on arcs incident on a node. For the purpose of the following exposition we first define two partition functions:

$$Z^*(X_n|\widetilde{G}_i, X_m) \tag{3.42}$$

$$Z(X_m|\widetilde{G}_i, X_n) \tag{3.43}$$

Where X_u and X_v are the variables corresponding to the arc which will be reversed ($u \to v$) and $\widetilde{G}_i$ is the graph obtained by orphaning both nodes u and v in G_i. The first partition function Z^* excludes the parent sets that contains descendant nodes of u in $\widetilde{G}_i$ and force v to be in the new parent set of node u. The other partition Z excludes parent sets that contain node u and any other of its descendants.

Formalizing how the score of the network changes between states, the ratios can be obtained y simplifying the scores of the two modified parent sets of nodes u and v, which follows from the above informal expositions (for a more visual explanation check figure 3.1:

$$\frac{p(G_{i+1})}{p(G_i)} = \frac{\exp[\Psi(\pi_u^{i+1}|D)]\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_u^{i}|D)]\exp[\Psi(\pi_v^{i}|D)]} \tag{3.44}$$

On the other hand, the proposal distributions is defined using the partition functions and the indicator $\mathbb{I}(.)$ as:

$$q(G_{i+1}|G_i) = \frac{1}{|E(G_i)|}\frac{\exp[\Psi(\pi_u^{i+1}|D)] * \mathbb{I}(X_v \in \pi_u^{i+1})}{Z^*(X_u|\widetilde{G}_i, X_v)}\frac{\exp[\Psi(\pi_v^{i+1}|D)]}{Z(X_v|\widetilde{G}_i, X_u)} \tag{3.45}$$

The first term correspond to the probability of selecting and arc for reversal since it is chose uniformly from the available ones. The second factor corresponds to probability of sampling a new parents set for node u, which is just the score normalized by the sum of all possible parent sets. Analogously, the third term is the corresponding sampling but for node v. Using this formula we can compute the

acceptance ratio by substituting in equation 3.35:

$$
\begin{aligned}
r(G_{i+1}|G_i) &= \frac{p(G_{i+1})}{p(G_i)} \frac{q(G_i|G_{i+1})}{q(G_{i+1}|G_i)} \\
&= \frac{\exp[\Psi(\pi_u^{i+1}|D)] \exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_u^{i}|D)] \exp[\Psi(\pi_v^{i}|D)]} \frac{\frac{1}{|E(G_{i+1}|} \frac{\exp[\Psi(\pi_u^{i}|D)] * \mathbb{I}(X_v \in \pi_u^i)}{Z^*(X_u|\widetilde{G}_{i+1}, X_v)} \frac{\exp[\Psi(\pi_v^{i}|D)]}{Z(X_v|\widetilde{G}_{i+1}, X_u)}}{\frac{1}{|E(G_i)|} \frac{\exp[\Psi(\pi_u^{i+1}|D)] * \mathbb{I}(X_v \in \pi_u^{i+1})}{Z^*(X_u|\widetilde{G}_{i}, X_v)} \frac{\exp[\Psi(\pi_v^{i+1}|D)]}{Z(X_v|\widetilde{G}_{i}, X_u)}} \\
&= \frac{|E(G_i)|}{|E(G_{i+1})|} \frac{Z^*(X_u|\widetilde{G}_i, X_u)}{Z^*(X_v|\widetilde{G}_{i+1}, X_u)} \frac{Z(X_v|\widetilde{G}_i, X_u)}{Z(x_u|\widetilde{G}_{i+1}, X_v)}
\end{aligned}
\tag{3.46}
$$

where the final expression can be obtained by simplifying the scores of the parent sets in both expressions. We see that in practice, only the partition functions for each move must be computed. The last step in order to define the kernel for the MCMC using these moves is combine the kernel for this move obtained by substituting with the one used by the addition and removal ones.

Defining p_s and $p_r = 1 - p_s$ as the probabilities of selecting the standard moves (addition and reversal) or the REV move respectively we obtain the combining kernel defined as:

$$
K^+(G_{i+1}|G_i) = p_s K(G_{i+1}|G_i) + p_r K'(G_{i+1}|G_i) \tag{3.47}
$$

Where K is the kernel with arc additions and removals and K' the one arising form the use of the REV move. As we said the chain associated with REV move is not irreducible since we can't reach every state from any other just by performing it. Combining it with the classical moves solves this problem and ensures irreducibility, since the combination of both kernels uses the basic moves to ensure it can reach any state. In this case, K^+ is ergodic and since K and K' have the same stationary distribution, by theorem 3.2.2 the mixture kernel also converges to the same unique stationary distribution, as is desired.

4

Multi-dimensional Gaussian Network Regressor

In this chapter we will describe our proposed algorithm. As discussed we will make use of MCMC techniques combining them with the MBC DAG structure. MCMC will provide a search mechanism in order to traverse the space of possible networks. The MBC will serve as a mechanism by which to impose constraints on the structure of the graph. The hypothesis is that if the true graph follows a similar structure to the one of MBC this will narrow the search space in a convenient way. This is particularly useful for large graphs where it becomes common practice to impose a fan in restriction. In these cases, being able to narrow the number of parent sets can allow the use of a larger fan in, potentially improving the quality of the discovered networks.

Additionally, instead of selecting a unique model, we will use the generated samples to perform an approximation of the posterior probability of structures which is useful for performing analysis of the posterior probability of arcs and for Bayesian Model Averaging during prediction.

4.1. Using MCMC in CBD-GN

As was said, using MBc graphs first of all implies reducing the possible parent sets to the ones that conform to the MBC structure given target and feature variables. This first step is pretty straightforward. When performing MCMC, the MCB structure affects the size of the neighborhood of each state since arcs in the from feature tu targets are not allowed even if the do not violate the DAG property.

4.1.1. Deleting arcs

The MBC constraint can introduce a minor problem when deleting some arcs depending on how we wish to enforce it. If a feature node v has only one parent who is a target (call it u), then if we select the arc $u \rightarrow v$ for deletion this will leave v with no parents in the set of target variables. For prediction tasks this means that variable v will have no use whatsoever (though it would still be relevant for analysis of the relationships in the domain).

If we wish to enforce the use of all feature variables for prediction then we have two options: disallow the removal of these arcs or reattach v immediately after performing such a removal to a different node w in the set of targets. The second option seems like the best one since not allowing the arc deletion can limit the capacity of the algorithm to efficiently explore the state space.

This operation is simple and does not introduce further difficulties. Reattaching v to a new parent target w is always possible since by construction a cycle will never be formed. Also, it does not violate any fan in restrictions since we always remove an arc before adding a new one. Furthermore this move is clearly reversible since it is simply a removal followed by an addition which are both reversible.

Using the same notation as in the previous chapter the expression for the proposal distribution:

$$q(G_{i+1}|G_i) = \frac{1}{|E_i||V_{-u}|} \tag{4.1}$$

where in this case $V_{-u} = V \setminus u$, and E_i are the arcs in G_i. The proposal is just the probability of selecting any of these arcs in the set E_i uniformly and, in the case that v was a feature and u its only parent, sample a new parent w from the set of targets excluding u uniformly.

Substituting this expression in 3.35 and taking into account that scores of unchanged parent sets cancel out we obtain the following expression of the acceptance probability:

$$\begin{aligned} r(G_{i+1}, G_i) &= \frac{p(G_{i+1}|D)}{p(G_i|D)} \frac{q(G_i|G_{i+1})}{q(G_{i+1}|G_i)} = \frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^i|D)]} \frac{\frac{1}{|E_{i+1}||V_{-w}|}}{\frac{1}{|E_i||V_{-u}|}} \\ &= \frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^i|D)]} \frac{|E_i||V_{-w}|}{|E_{i+1}||V_{-u}|} = \frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^i|D)]} \end{aligned} \tag{4.2}$$

where the last step is a consequence of the fact that the sets of nodes and arcs are the same size. This is evident in the case of the former. In the case of the arcs we can see this is also true since, by the structural property of MCBs, arcs from the targets to the features never add cycles and, furthermore, don't make previously admissible arcs for addition invalid. These could be incident to v from a feature which can't be a parent of w (a target) or from another target, in which case it either is a possible parent of w or not, a situation that is not changed by making it a parent of v.

While we could make this a separate move, it would add unnecessary complexity to the algorithm. So for our implementation we have just made it an extension of the traditional arc removal.a

4.1.2. The Adjacency Resample move

Constraining the possible structures does not resolve the main problem with sampling in a discrete setting like that of graph structures. Mixing in discrete spaces is low and escaping local maxima is hard tasks in many situations due to the jaggedness of the probability landscape.

A common strategy employed to overcome this is to do multiple restarts of the chain from different states. If enough of these restarts are performed and their initial states are sufficiently apart in the state-space then good approximation of the posterior will more likely be found. This of course depends on the particular probability landscape that we are dealing with. There are two problems with this strategy, first it's not always easy to know haw far we are apart these starting points are from each other, i.e. how well they are covering the state space since if they are clustered in one point then these restarts will be redundant. Second, restarting the chain means that any progress we made towards its convergence is lost.

We propose a new move which is compromise between resampling as in the REV move (Figure 4.1), which is done according to the scores of the parent sets, and the random connections usually generated at the start of the chains when arcs are added randomly. We call this new move Adjacency Resample (AR) and its objective is to perform a large reconfiguration of the current state of the chain by jumping to a potentially part of the state-space that is far from the current state without loosing the convergence achieved so far. It achieves this by detaching a node from the graph (except for one case which we will describe below) and reattaching it to a new set of parent and children nodes.

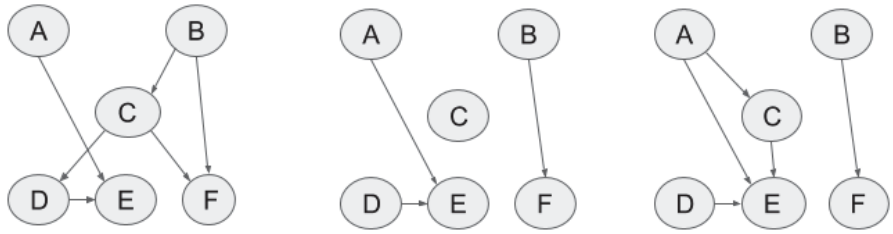


Figura 4.1: The AR move. In the the first figure the original network. In the second, node C is detached from th rest of the graph. Lastly, we reattach it to a new parent set and new children.

Informally, the move works as follows. First we choose one (call it u) of the nodes at random from $V(G)$. Next we proceed to disconnect u from its parents and children (except in the case where a child of u is a feature and u is its only target parent). Then we sample a new parent set with a probability that depends on the score of the parent set (e.g. using the BGe score, equation 3.26) score from those that do not violate neither the DAG nor the MCB properties when the corresponding arcs are added to the graph. Finally, in this intermediate graph we sample some of its admissible arcs that correspond to potential new children nodes of u.

The steps for the move are described in algorithm 1, where Π_u^i are the allowed parent sets of node u in graph G'. The disconnect step is done as mentioned before, without detaching the children who are features and for which u is their only target parent.

Algoritmo 1 AR move

1: **procedure** AR(G_i)
2: $u \leftarrow sample(V(G_i))$ ▷ Pick a node at random
3: $G' \leftarrow disconnect(u, G_i)$ ▷ Disconnect u from G_i
4: $\pi_u \leftarrow sample(\Pi_u^i)|D, G')$ ▷ Sample a parent set according to its score
5: $G'' \leftarrow add\_arcs\pi_u \times \{u\}, G')$ ▷ Add arcs of the new parent set
6: $k \leftarrow sample(|admissible\_arcs(G'', u)|)$ ▷ Sample the number of arcs to add
7: $arcs \leftarrow sample(admissible\_arcs(G'', u), k)$ ▷ Select k arcs randomly
8: $G_{i+1} \leftarrow add\_arcs(arcs, G'')$
9: **return** G_{i+1}
10: **end procedure**

It is important in step 6 that the size k (this is not the same k as when we

discussed the k-CBd in the previous chapter) of the set of arcs that will be added in step 7 is not selected uniformly since the number of combinations is not the same for different set sizes. Selecting k this way would bias the move towards extreme set sizes, either too small or too large which have a lower amount of combinations and hence higher probability for each of them. This is why we instead sample k in accordance with the number of possible sets of that size normalized by the total number of subsets. The probability of selecting subset size k when we have m_i admissible arcs in step 6 if we started from graph G_i is thus computed as:

$$p(k|m_i) = \frac{\binom{m_i}{k}}{\sum_{j=0}^{m_i}\binom{j}{k}} = \frac{\binom{n}{k}}{2^{m_i+1}-1} \tag{4.3}$$

First of all we can see that the this move is reversible, since from G_{i+1} we can simply disconnect node u and we obtain the same graph G'. Furthermore the arcs that could not be disconnected in G_i are the same as in G_{i+1} since their parents never change and if n is one of them then the corresponding arcs are not sampled for addition in steps 6 and 7. In order to reduce the possible arcs to be selected in step 7, we exclude the ones that where present in the previous step. This means that the set of arcs that are considered for admission are also disjoint from those that where removed from the children of node n in G_i so we can always move in the opposite direction no matter the state we started in step i of the chain.

4.1.2.1. Acceptance probability

From the above algorithm we can also see that the ratio of scores between the two graphs depends on the change in parent sets for node u and the set of its children that are different in G_i and G_{i+1} (denoted $C = Ch_i(u) \cup Ch_{i+1}(u)$, where $Ch_i(u)$ are the children of node u in graph G_i):

$$\frac{p(G_{i+1}|D)}{p(G_i|D)} = \frac{\exp[\Psi(\pi_u^{i+1}|D)]}{\exp[\Psi(\pi_u^{i}|D)]} \prod_{v\in C} \frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^{i}|D)]} \tag{4.4}$$

The Hastings correction coefficient (ratio between the probabilities of the move and its inverse) can be computed as follows. Suppose we start in state G_i, the first step after we detach the node as outlined in 1 is to sample the new parent set of node u, which we do with probability:

$$p(\pi_u^{i+1}|G') = \frac{\exp[\Psi(\pi_u^{i+1}|D)]}{Z(X_u|G')} \tag{4.5}$$

Next we sample the number of arcs we are going to add from those that are admissible in G''. The probability of each set size k is obtained with equation 4.3 and the probability of each arc set of the given size is just the inverse of the number of sets of that size since we select them uniformly. Combining both of these expression we obtain the probability of each possible neighbor from G'':

$$p(G_{i+1}|G'') = \frac{\binom{m_i}{k}}{(2^{m_i+1}-1)\binom{m_i}{k}} = \frac{1}{2^{m_i+1}-1} \tag{4.6}$$

Combining equations 4.5 and 4.5 we get the expression for the probability of one move as:

$$q(G_{i+1}|G_i) = p(\pi_u^{i+1}|G')p(G_{i+1}|G'') = \frac{\exp[\Psi(\pi_u^{i+1}|D)]}{Z(X_u|G')2^{m_i+1}-1} \tag{4.7}$$

where we follow the same notation as in the algorithm's outline. To complete the expression of the Hastings correction coefficient we just need to combine the probability of the move and its inverse:

$$\frac{q(G_i|G_{i+1})}{q(G_{i+1}|G_i)} = \frac{\frac{\exp[\Psi(\pi_u^{i}|D)]}{Z(X_u|G')2^{m_{i+1}+1}-1}}{\frac{\exp[\Psi(\pi_u^{i+1}|D)]}{Z(X_u|G')2^{m_i+1}-1}} = \frac{\exp[\Psi(\pi_u^{i}|D)]2^{m_i+1}-1}{\exp[\Psi(\pi_u^{i+1}|D)]2^{m_{i+1}+1}-1} \tag{4.8}$$

where te simplification of the two normalization constants Z is possible since G' is the same in both directions. To see this observe that we always disconnect all arcs from node u (the only exception being the case in which u is the only target parent of a children v. This case however can be safely ignored since we don't modify the parent set of said node in any part of the move). Once we do this we thus have the same graph G' and the only difference between moves is that the new children that we sample are not the same since we exclude the former ones to improve variance.

Multiplying 4.4 and 4.8 we obtain $r(G_{i+1}, G_i)$ for the move:

$$\begin{aligned} r(G_{i+1}|G_i) &= \frac{p(G_{i+1}|D)}{p(G_i|D)}\frac{q(G_i|G_{i+1})}{q(G_{i+1}|G_i)} \\ &= \frac{\exp[\Psi(\pi_u^{i+1}|D)]}{\exp[\Psi(\pi_u^{i}|D)]}\Big(\prod_{v\in C}\frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^{i}|D)]}\Big)\frac{\exp[\Psi(\pi_u^{i}|D)]2^{m_i+1}-1}{\exp[\Psi(\pi_u^{i+1}|D)]2^{m_{i+1}+1}-1} \\ &= \frac{2^{m_i+1}-1}{2^{m_{i+1}+1}-1}\prod_{v\in C}\frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^{i}|D)]} \end{aligned} \tag{4.9}$$

where the scores of the new parent sets for node u cancel out, so only those of the children need to be computed. This expression can be substituted in equation 3.40 along with 4.7 to obtain the final expression of the kernel for the AR move:

$$\begin{aligned} K_{ar}(G_{i+1}|G_i) &= q(G_{i+1}|G_i)\alpha(G_{i+1}|G_i) \\ &= \frac{1}{2^{m_i+1}-1} \min\left\{1, \frac{2^{m_i+1}-1}{2^{m_{i+1}+1}-1} \prod_{v \in C} \frac{\exp[\Psi(\pi_v^{i+1}|D)]}{\exp[\Psi(\pi_v^{i}|D)]}\right\} \end{aligned} \tag{4.10}$$

We next demonstrate that this move is ergodic and thus converges to proper posterior distribution.

Theorem 4.1.1 *Irreducibility of the K_{ar} kernel: The kernel induced by the AR move in the MH sampler is irreducible.*

Proof *Since all arcs could be removed in at most $|V|$ moves (we always sample the empty parent set and no arcs to children for all the nodes in succession) and we can always sample any network from the empty graph, then the move is irreducible. In other words, observe that this move could behave like simple arc addition and removal if the right steps are sampled when assigned a new parent set or children to each node.*

In the special case where we do not allow the removal of an edge that connects a feature to its only target parent we observe that this does not change our demonstration since when we deal with these arcs a move may add another parent to them, which will allow its removal in the next step.

Corollary 4.1.2 *Ergodicity of the K_{ar} kernel: The kernel induced by the AR move in the MH sampler is ergodic.*

Proof *Since the kernel is defined over a finite state space and is irreducible we have positive recurrence for all states and thus ergodicity.*

Corollary 4.1.3 *The AR kernel has $p(G)$ as its limiting distribution.*

Proof *This is a direct consequence of kernels defined by the MH sampler satisfying the detailed balance condition.*

Nonetheless, we do not use this as a standalone transition kernel, since it is intended as a high variance move. Standard addition, removal and the REV move are combined with the AR move to create a combination kernel:

$$K^{+}(G_{i+1}|G_i) = p_{basic}K_{basic}(G_{i+1}|G_i)+p_{rev}K_{rev}(G_{i+1}|G_i)+p_{ar}K_{ar}(G_{i+1}|Gs_i) \quad (4.11)$$

where p_{ad}, p_{rev}, p_{ar} are the probabilities of each move, such that $p_{ad}+p_{rev}+p_{fmb} = 1$. Since the first kernel is irreducible and ergodic then by theorem 3.2.2 $K+$ also converges to the desired stationary distribution and so the proposed MCMC sampling algorithm is correct.

4.1.2.2. Computational Complexity

Analyzing the computational complexity of these moves we find that if we follow the same implementation of Giudici and Green (1999) which uses an ancestor matrix, it takes $O(1)$ to check for if an arc would violate the DAG property. In turn updating this ancestor matrix takes $O(|V|^2)$ for each node. This means that in the worst case, updating after a move takes $O(|V|^3)$ since we need to update for each descendants.

The modified delete that reattaches the feature node does not add further complexity since it's simply an arc removal followed by an addition. In the case of the AR move the cost is $O(|V|^3)$. This is easy to see since we have to update the ancestor matrix of all the descendant of node u which are at most $|V|$ with a cost of $O(|V|^2)$ each. These moves then add no further asymptotic complexity to the existing ones.

4.2. Prediction

To predict with a trained network, we have fit the parameters of the equivalent underlying MVT taking into account the parents of each variable according to the GN. We can determine the parameters of the network using equations 3.2 and then use3.5 to obtain the corresponding MVN. We can then obtain a conditional distribution of the target variables Y given the features X using the conditioning formulas for MVNs(3.6):

$$f(Y|\mathbf{x}, \theta) = \mathcal{N}(Y|\mu_{Y|X}, \mathbf{\Sigma}_{Y|X}) \quad (4.12)$$

Using the fact that GNs may decompose as a k-CBd, we can further decompose the set of variables X and Y as $X = X_0, \ldots, X_k$ and $Y = Y_0, \ldots, Y_k$. This means that the final distribution can be decomposed in its components yielding:

$$f(Y|\mathbf{x}, \theta) = \prod_{i=0}^{k} \mathcal{N}(Y_i|\mu^i_{Y_i|X_i}, \mathbf{\Sigma}^i_{Y_i|X_i}) \tag{4.13}$$

A distinct advantage of using MCMC is that for situations where variance in the models is high, for example when the number of data samples is low, we can use the fact that the samples from the Markov chain approximate the posterior over networks to perform an average over network configurations. This approach has been explored in previous works (Madigan and Hutchinson (1995)) and yields good results at the cost of more computational cost. In this case predicting can be done using the following formula which is an approximation to the true Bayesian posterior predictive distribution:

$$f(Y|X, D) = \sum_{i=1}^{N_s} f(Y|X, G_i, D) \tag{4.14}$$

where N_s is the number of samples extracted from the Markov chain and each term inside the product is equal to 4.13.

4.3. Implementation details

The algorithms were all implemented in Python (**www.python.org**), using Numpy and Scipy (**www.scipy.org**) which provide efficient implementations of arrays and operations over them along with mathematical functions like logarithms and density functions. To operate with the graph structures we used Networkx (**networkx.github.io**), which contains implementations of graphs and graph related algorithms.

Basic operations like adding, removing and deleting arcs take $O(1)$ using dictionaries of dictionaries to the adjacency matrices of the graphs. Further efficiency could be achieved in matrix form combined with GPUs but for the purpose of this work this simple approach seemed sufficient.

To compute the BGe score we used the same factorizations and simplifications presented in Kuipers et al. (2014). These simplified expressions for the score of each parent set and other changes like using the Cholesky decomposition to calculate

determinants of positive definite matrices, provide better numerical stability. This is critical for large networks and databases. In order to operate with the scores which can have quite small values that can cause underflow and precision errors we also work in log-space, avoiding this problem. Finally a fan in restriction is used to limit the number of parents. This is necessary o otherwise pre-computing the score for each one takes to much time (a feature node in a large network has possible $O(2^{|V|} - 1)$ parents).

The conditional probability table (CPTs) used to store the scores of each parent set and from which we sample them where implemented using Pandas (**pandas.pydata.org**), whose Series data structure provide an easy API for using hierarchical indexes and performing operations like selection of values under arbitrary conditions.

5

Evaluation of our method

In this chapter we describe the experiments performed to asses the performance of the model. We will have to rely on several visual tests since statistical hypothesis tests used to determine the convergence of the Markov chain like the Geweke test are designed for MCMC sampling in continuous domains. We also check the performance against the MCMC algorithm using just the traditional moves with the REV edge reversal. We also present the application of our method to the problem of discovering the relation between morphological and electrophysiological variables of neurons described in the Chapter 1.

5.1. Experimental setting

To evaluate the performance we plot the scores of the sampled networks obtained in an MCMC run. This allows us to observe how fast both methods converge to regions of high probability and how the vary once they arrive to those regions. To check how much steps are spent in each region we also plot the distribution of the scores obtained.

To evaluate convergence we monitor the scores of the networks in each state as well as the presence of each arc and the acceptance ratio. In the case of the arc values we use their running means for visualization since they are easier to interpret than the values themselves.

Since we lack benchmark datasets of data generated by MBCs with continuous variables we used simulated data. We randomly generated GNs with different network sizes always respecting the MBC structure. The networks used for testing were fixed at 15 variables with 5 targets and 10 features. The mean of each variable was

set to 0, with variance equal to 0.2 and a strength of the influence of the parents over the children of 2. We could have chosen a random value for this parameter but we wished that the influence was noticeable so check if the algorithm could find the arcs. If the coefficients where very small this would be harder.

The MCMC sampler ran for 100 thousand iterations for each network example with 20 restarts to check how the metrics vary with different initializations, and in the case of the AR to see its ability to escape from local maxima.

5.2. Results

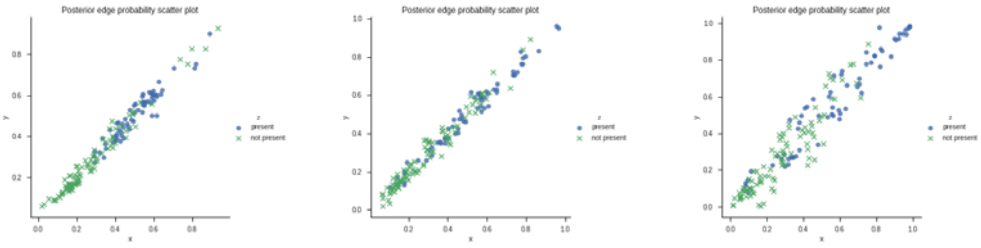
Having established the parameters of the experiments, in this section we present the results along with our comments on their implications.

5.2.1. Convergence Reliabilty

First we will evaluate the convergence of the algorithm. We can run the MCMC sampler from different starting points and trace the values of the parameters and the scores. These provide an estimate of the posterior probability of the arcs. Scatter plots of these values are then used to determine the reliability of the convergence in the same way that it is done in other works on the subject (Friedman and Koller (2003), Grzegorczyk and Husmeier (2008), Su and Borsuk (2016)). Each axis corresponds to the probability value of an arc for two different runs, each point corresponds to one arc. If a point approaches the diagonal then this signals a reliable convergence for that arc since both runs agree on the posterior probability for that arc.

As we can see it shows that both runs tend to arrive at a similar probability for each arc since they both approximate the diagonal. The probabilities however are in many cases not close to the real presence values (though most missing arcs are correctly avoided). This may be due to the fact that we are trying to learn with a fan in restriction from a network without restrictions. As we increase the sample size we observe that the spread of posteriors increases which means that it takes longer for the chains to converge. This is expected since with more samples the posterior landscape becomes more jagged and starting from different positions means that the chains can be trapped in different local maxima.

Another way of visualizing the same results is shown in Figure 5.2, which shows the progression of posterior values across time. We can observe that after an initial



(a) (b) (c)

Figura 5.1: The scatter plot of the arc presence for the same network (network seed=333) using datasets of different sizes. a) size=50, b) size=100, c) size=200. Each point is the posterior probability of one arc obtained with two differently seeded runs. If the points approach the diagonal it means that both runs agree on the probability values.

period where there is a lot of change in the parameter values, they stabilize around the posterior probability. Also noteworthy is that as the sample size increases, the posterior probability of some arcs increases and more arcs start to have higher probabilities, as shown by the accumulation of points in the upper right corner of the plots in Figure 5.1 and the spreading of the posterior traces upward in Figure 5.2 (plot (c)). This is expected since arcs that are important become more identifiable.

We can test how much the fan in restriction could be affecting these posterior values. A different test was thus conducted, in this case generating a network with the same parent set restriction as the one used for learning. We can see in figure 5.3 we can see that in both chains exhibit high convergence to the same posterior (plot a). We can see however that some arcs that are not present in the original graph are assigned a high probability. This is probably a local maximum where these arcs emulate the influence of the real ones. More iterations could probably force the chain to a different part of the posterior landscape but this was not the objective of the test and so it was not tested. With more examples, the algorithm did escape these local maxima, but it also took more time to discover the real arcs (figure not shown).

However since in practice we cannot know the real network structure, it is useful to see if the quality of the discovered structures is too divergent from the original one. We can plot the densities of the scores of sampled networks. For example the network obtained with seed=333 has an unnormalized log-score as computed with BGe metric of around -800, -1400, -2100 for parameters computed from datasets of size 50, 100, and 200 respectively. As we can see in Figure 5.4 these values are

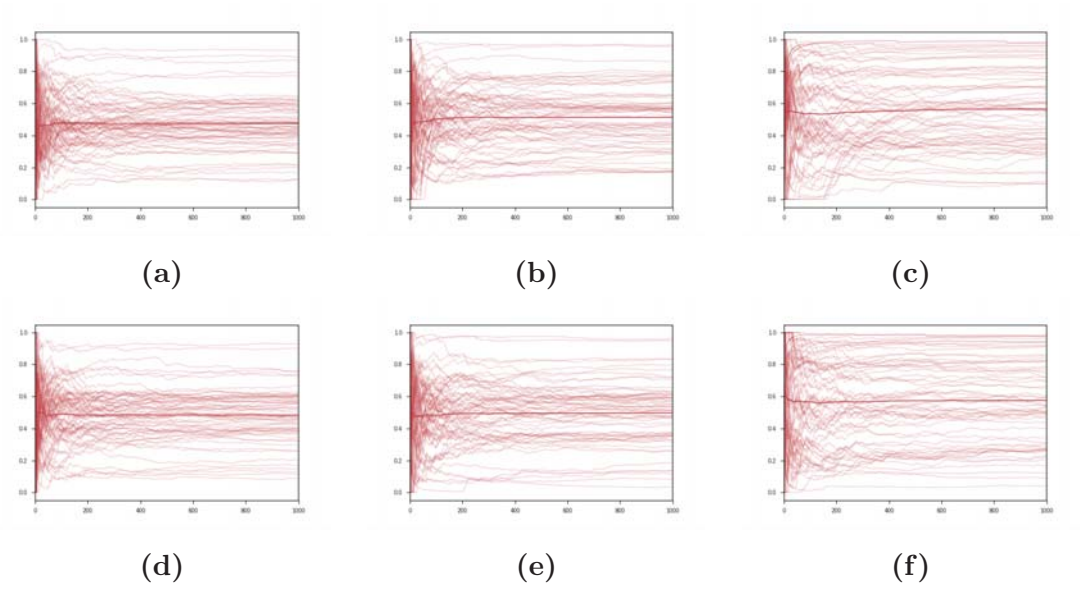


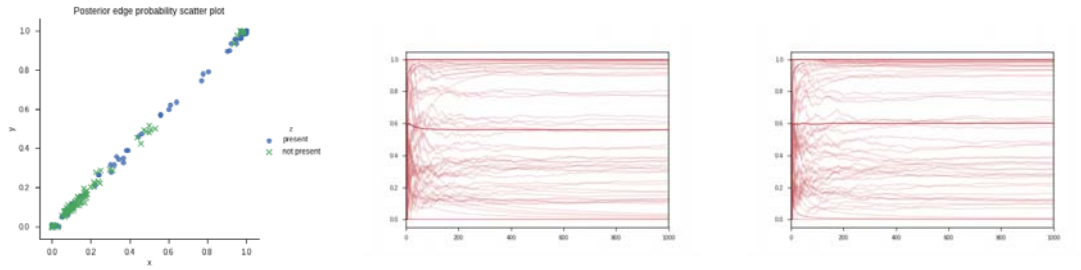
Figura 5.2: The running mean of the parameter values (arc presence in graph for a network with 5 targets and 10 features) for two independently seeded runs (top and bottom) for each dataset size. These plots are easier to read than the presence values at each iteration. The plotted values correspond to the samples taken every 100 iterations. a, d) size=50; b, e) size=100; c, f) size=200.

close to the scores of the structures found, even if these may not have the correct structure.

5.2.2. Comparison to REV

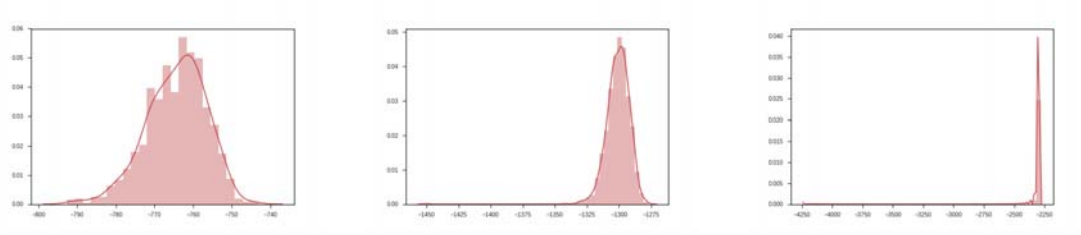
In this section we compare the move proposed move against the sampler using the REV along with the standard addition and removal of arcs. We remind the reader that original (Grzegorczyk and Husmeier (2008)) article uses the uncorrected BGe score, which means that those results could be misleading. We are interested in finding how the convergence rate changes for both approaches and the score of the networks found. In figures 5.5 and 5.6 we see the results of using both methods (REV only and AR with REV) for 4 different networks, averaging 10 restarts using a dataset of 200 samples. Both methods quickly converge to high scoring structures (5.5).

We see that both moves quickly converge to local maxima and enter a phase where they do not vary very much in value. Closer inspection in Figure 5.6 reveals that though in the first and last cases both moves behave particularly better, in



(a) (b) (c)

Figura 5.3: The plots of the test on data generated by networks with the same parent set restriction as the one used in the algorithm (in this case 5). a) Scatter plot of the arc presence for two differently seeded runs (seeds 101 and 102). b, c) the progression of posterior probabilities for the two runs (101 and 102 respectively).



(a) (b) (c)

Figura 5.4: The density plots for the scores of the samples structures for different dataset sizes. a) size=50; b) size=100; c) size=200.

the other instances AR move has outperformed REV. In the second AR is clearly sampling better networks according to score and in the third it managed to jump to a higher probability region right at the end of the MCMC run.

If we plot the distribution of the scores (Figure 5.7) we see that these are nonetheless pretty similar for both moves, with the only major difference being the second example, where the AR move quickly exits a local maximum and proceeds to the higher probability region. In general we find that the sampler with the AR move never under performs relative to the REV and even allows the chain to exit some local maxima and into higher scoring regions where REV was not able to move.

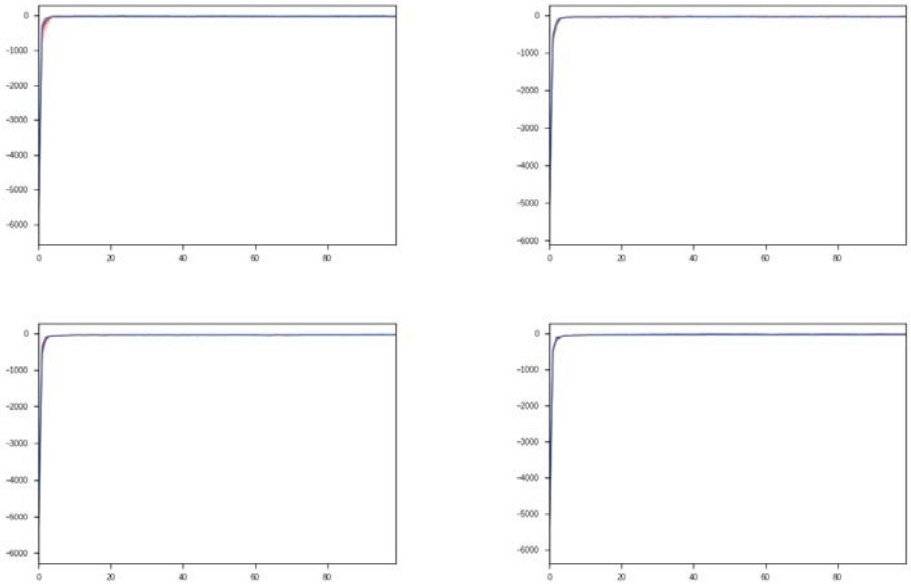


Figura 5.5: The score progression for 4 different random networks without parent set restrictions. In blue the values using the REV move, in red adding the AR move. Plots are from samples taken every 100 samples and include only the first 10000 iterations, after which the scores settle to a stable range of values. Darker shade corresponds to the 67 % confidence interval and lighter to the 95 % confidence interval of the scores across ten restarts.

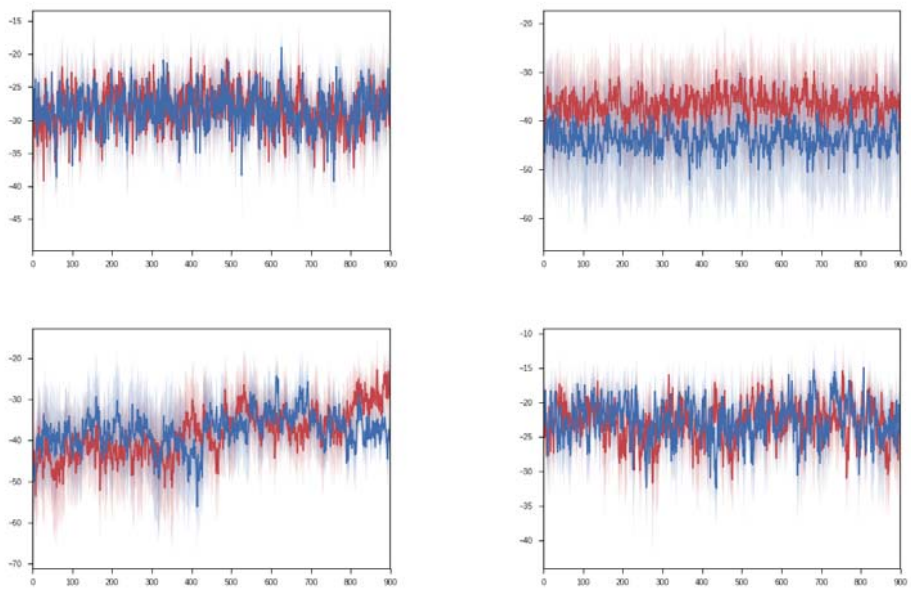


Figura 5.6: The score progression for 4 different random networks without parent set restrictions. In blue the values using the REV move, in red adding the AR move. Plots are from samples taken every 100 samples and after dropping the first 10000 iterations. At this point we estimated that burn in phase had ended since scores where remained stable and close to that of the original network. Darker shade corresponds to the 67 % confidence interval and lighter to the 95 % confidence interval of the scores across ten restarts.

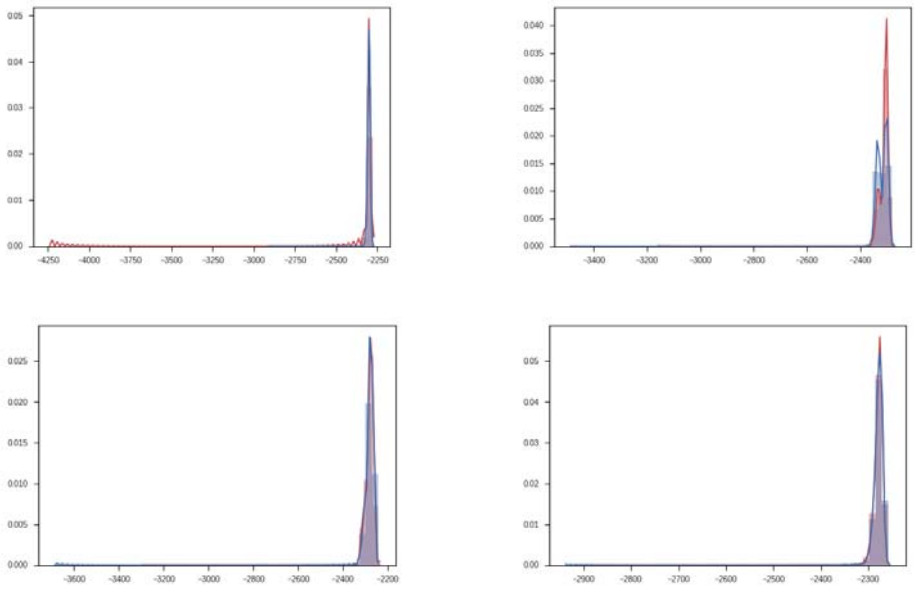


Figura 5.7: The comparison of the density plots for the scores for the sampled structures using the REV (blue) and AR (red) moves for different data sizes. a) size=50; b) size=100; c) size=200.

6

Applying to Electro-Morpho problem

For the purpose of this work we are interested in analyzing the data contained in the Celltype Database of the Allen Institute for Brain Science (Allen Institute (2017a)). This database contains 295 samples consisting of 23 morphological and 48 electrophysiological properties (along with other variables that indicate the position of the cell and other diagnostic labels that are not relevant to our problem). This last group can be further separated into variables that describe the response to different types of stimuli while the in the former we can distinguish between the ones that describe dendritic arborization patterns (number of bifurcations, angle of said bifurcations) and those that describe the soma.

6.1. Data

In the case of morphological variables, we first eliminated those that are not continuous since MVN are not very well suited to model them. This leaves us with a list of 14 variables which we show below along with a description of their meaning as found in Allen Institute (2017b).

Tabla 6.1: Morphological variables

Variable	Description
Soma surface area	The total area of the soma (the center of the cell).
Neuronal height	Height is computed on the y-coordinates and it is the difference of minimum and maximum y-values after eliminating the outer points on either end by using the 95 % approximation of the y-values of the given input neuron.
Neuronal width	Width is computed on the x-coordinates and it is the difference of minimum and maximum x-values after eliminating the outer points on either end by using the 95 % approximation of the x-values of the given input neuron.
Neuronal depth	Depth is measured on the z-coordinates and it is the difference of minimum and maximum z-values after eliminating the outer points on either ends by using the 95 % approximation of the z-values of the given input neuron.
Average diameter (thickness)	The avg. diameter of all compartments of the neuron.
Total length	The total length of the neuron is computed as the sum of distances between two connected nodes for all branches of the input neuron.
Total surface area	The total surface area of the entire neuron.
Total volume	The total volume of the entire neuron.
Maximum Euclidean distance to root	The maximum Euclidean distance of all nodes. The Euclidean distance is the straight line distance from the soma (root) to the node.
Maximum path distance to root	The maximum path distance of all nodes. The path distance is the sum of lengths of all connected nodes from the soma, ending with that node.

Avg. contraction	The avg. ratio between Euclidean distance of a branch and its path length. Euclidean distance of a branch is the straight-line distance from the soma to the branch. Path length is the sum of the lengths between each node along the path.
Average fragmentation	The avg. number of compartments that constitute a branch between two bifurcation points or between a bifurcation point and a terminal tip.
Average parent-daughter diameter ratio	The avg. ratio between the diameter of a daughter branch and its parent branch. One value for each daughter branch is generated at each bifurcation point.
Average local amplitude angle	The avg. angle between the first two compartments (in degree) at a bifurcation. For more information on how the order is established details are in the cited report.

The electrophysiological variables (Allen Institute (2016)) were also subjected to the same criterion. Also since there are several types of stimuli, which accounts for the large number of variables, we choose to focus on the ones that correspond to one type of stimulus, namely the Long Square. This consists of a long constant current injection lasting several milliseconds. This reduces the data set to 19 variables whose description we present in Table 6.2. The first 5 variables describe a single action potential (or spike) while the rest describe features of the trains of actions potentials obtained by applying the described stimuli. An action potential is the electrical response to an input stimulus of a neuron.

Tabla 6.2: Electrophysiological variables

Variable	Description
Action potential peak voltage	Maximum value of the membrane potential during the action potential (i.e., between the action potential's threshold and the time of the next action potential, or end of the response).
Action potential peak time	Time from the start of the action potential.
Action potential trough voltage	Minimum value of the membrane potential in the interval between the peak and the time of the next action potential.
Action potential trough time	Time from the peak of the action potential to the moment in which the minimum trough voltage is achieved.
Action potential fast trough voltage	Minimum value of the membrane potential in the interval lasting 5 ms after the peak.
Action potential fast trough time	Time from the peak of the action potential to the moment in which the minimum fast trough voltage is achieved.
Action potential slow trough voltage	Minimum value of the membrane potential in the interval from 5 ms after the peak until the time of the next action potential. If the time between the peak and the next action potential was less than 5 ms, this value was identical to the fast trough.
Action potential slow trough time	The time from the peak action potential to the moment of the slow trough.
Threshold voltage	The voltage value at which the action potential starts. The start of an action potential is defined as the maximum of the second derivative of the change in voltage in time.
Threshold intensity	The intensity value of the applied current at which an action potential is unleashed.
Threshold time	The time it takes for the start of an action potential when applying a hyperpolarizing current.
Input resistance	The change in voltage in response to an increase in input intensity.
Upstroke downstroke ratio	The ratio between the absolute values of the action potential peak upstroke and the action potential peak downstroke.

F/I curve slope	The change in the rate of response frequency of a neuron when the intensity of the input is changed.
Average ISI	The mean value of all inter-spike intervals in a sweep. The inter-spike interval is the time between the peaks of two contiguous spikes
Latency	Time between the start of the stimulus until the time of the first spike evoked by a stimulus.
Adaptation index	The rate at which firing speed increases or decreases during a stimulus.
Time constant (τ)	The time it takes the neuron to reach 67 % of steady state voltage.
Sag	The ratio between the difference (or decrease) in voltage after a step current and the maximum hyperpolarization.

From the definitions we can see several variables which we would expect to be related. Enumerating all of them could be a challenge but some of the most obvious groups of interrelated features can be extracted from the definitions:

1. Max Euclidean distance and Max path distances along with the avg. contraction: Unless the neuron structure is cramped in a tight space, they should have similar values and thus be highly correlated. The last value also is calculated from the other two so it must be related to them.

2. Neuronal hight, Neuronal width, Neuronal depth and total volume: This is evident from the definition of volume.

3. Total length, total surface area, total volume and soma surface area: All these variables should be related since the larger a neuron the more surface area it should have and the bigger the soma should be.

4. Input resistance, latency, τ: High input resistance should mean high latency and more time to reach a determined level of voltage (in the case of τ, 67 % of the steady state voltage).

5. Time constant and all the time variables: Since the time constant determines how fast the voltage changes in response to variation of stimuli these relations should appear in the resulting network.

These relations give us an idea of the arcs that should be discovered by any algorithm that tries to model the domain. In either case we must remember that we use a fan in restriction when learning a model, which means that some relations may not appear.

6.2. Preprocessing

Two variables (Upstroke/downstroke ratio and Max euclidean distance to root) have a bimodal distribution, which means that modeling them with a Gaussian is inadequate so they were dropped for the experiments. Others resemble exponential distributions or have sharp minimum values (mostly at zero). This means that the support for their empirical distributions is not infinite, so using a Gaussian could be a bad idea if most of the density is outside the corresponding domain.

In order to solve these issues we have tested several transformations for the variables, namely: logarithm, square, square root and fourth root. The logarithm transformation is the ideal since it has infinite support, the same as the Gaussian distribution. One issue with all of them is that real minimum values of variables must be known (since all of them work on positive values and we must shift them by that amount). In the case of physical variables they are positive so we can transform them without any problem. Electrical variables like the voltage have practical limits when discussing action potential of close to $-100mV$ so we can safely shift them even though technically values beyond that point have probability 0.

For a small subset of variables we chose to use square and 4th root transformation even though they do not have infinite support since the shape of the resulting distribution closely resembles a Kernel Density Estimate of the shape of the distribution and the density region that is not covered is small. We could drop these variables instead but we prefer to use as much data as we can. The three variables (soma surface, total volume, total surface) we transformed in this fashion are in plots (d) to (f) of Figure 6.1 after scaling them to zero mean and unit variance.

Of the original 33 variables, 12 were dropped since their shapes were too deviated from normality and transforming them did not improve this (this was the case for variables describing the peak and trough voltage for example). Some variables were kept either way, like the time constant since we think it is an important feature and we wish to see if we can discover how influential it is. The full list of transformations

can be found in Table 6.3, where the selected procedure for the 33 variables is shown. In the end 21 were used in the experiments, 11 targets and 10 features.

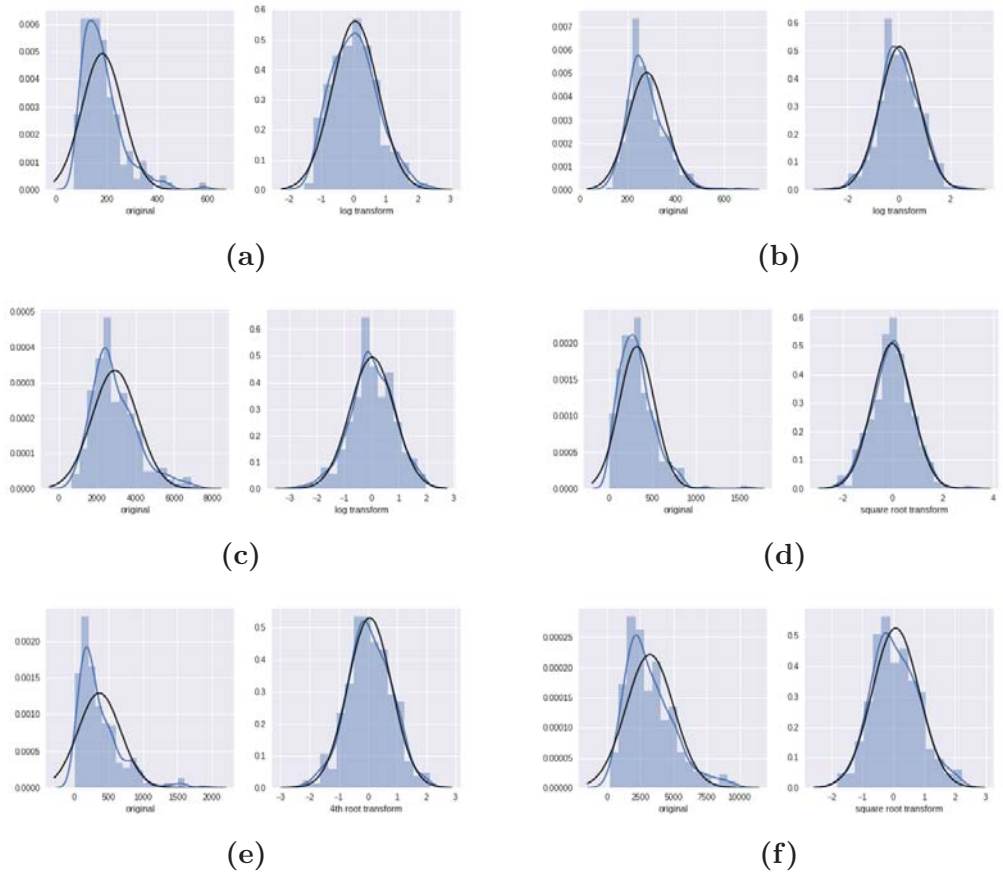


Figura 6.1: The original variable shape and the result of applying the specified transformation for 3 variables. First row: (a) Input resistance; (b) Overall width. Second row: (c) Total length; (d) Soma surface. Third row: (e) Total Volume; (f) Total surface. All values have been standardized

As we can see transformations work better on some variables than others (Figure 6.1). Nonetheless, they all look more Gaussian-like and the problems caused by the truncation at zero that is observed in the three variables in the second row.

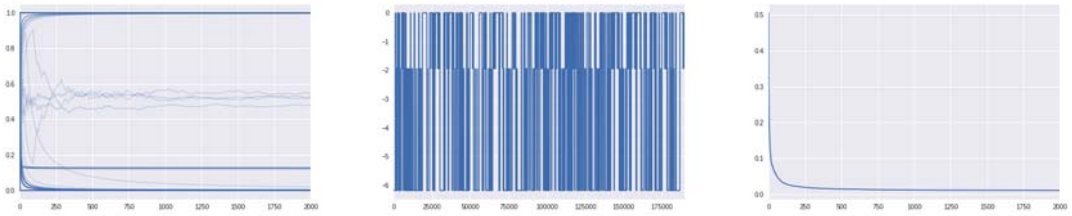
6.3. Applying the model

With the transformed variables we can apply the model to the data and expect better results than just using the original data. The experiments where set up in similar way to the ones in the previous chapter. The fan in restriction was set at 6,

Tabla 6.3: Applied trasformations for each variable

Variable	Transform	Variable	Transform
Soma surface area	Square root	**Neuronal height**	Logarithm
Neuronal width	Logarithm	**Neuronal depth**	Dropped
Average diameter (thickness)	None	**Total length**	Logarithm
Total surface area	Square Root	**Total volume**	4th Root
Maximum Euclidean distance to root	Dropped	**Maximum path distance to root**	Dropped
Average contraction	None	**Average fragmentation**	Logarithm
Average parent daughter diameter ratio	Logarithm	**Average local amplitude angle**	None
Action potential peak voltage	Dropped	**Action potential peak time**	Dropped
Action potential trough voltage	Squared	**Action potential trough time**	Dropped
Action potential fast trough voltage	Dropped	**Action potential fast trough time**	Dropped
Action potential slow trough voltage	None	**Action potential slow trough time**	None
Threshold voltage	None	**Threshold intensity**	Logarithm
Threshold time	Dropped	**Input resistance**	Logarithm
Upstroke downstroke ratio	Dropped	**F/I curve slope**	Dropped
Average ISI	None	**Latency**	Dropped
Adaptation index	None	τ	Logarithm
Sag	Logarithm		

the the number of iterations for the sampler was 200000, saving the state every 100 iterations. The burn in was set at 10000 iterations based on our experiments. In this case we can plot the posterior distributions of the arcs and see how they evolve as the chain progresses 6.2a.



(a) **(b)** **(c)**

Figura 6.2: (a) The evolution of the posterior for each arc as the chain moves forward. (b) The scores of the networks dropping the first 10000 iterations shifted so the maximum is at zero. (c) The acceptance ratio of the chain.

Using this plot we can asses the convergence of the model to the posterior distribution. We can see that contrary to our simulated examples these are much harder since real data has a lot more noise and does not follow the assumed distributions faithfully. Here all possible arcs are plotted since we don't know which ones are the true ones. Nonetheless, we see that most of them have a probability of zero. There are 310 possible arcs and 37 have non-zero probability. In any case most of them have a small probability which is a consequence of the fact that we are using a fan in restriction and that some influences are not very strong. Either way there is a group of arcs with high probability (shown in 6.4).

In the case of this dataset, the algorithm quickly arrives at local maximum and stays there. As we can see in Figure 6.2b after dropping the burn in samples, score of the network remains very close to the maximum found and the accept ratio quickly drops (Figure 6.2c). Most of the few moves performed after reaching this peak correspond to the reversal of one single arc (more below).

For this setting we see that there are only 37 arcs with positive probability, most of them with posterior probability 1. Two arcs are the reverse of one another and together they have probability 1 too. This leaves us with a very specific network where there is really no point selecting more than one model. Only 4 arcs have probability different than 1 and significantly greater than 0 (Figure 6.2a). The others quickly vanish to zero and must thus be considered irrelevant .

Tabla 6.4: The high probability arcs

Arcs	Probability	Arcs	Probability
avg. contraction → avg. bifurcation angle	1.000000	total volume → adaptation index	1.000000
avg. diameter → trough voltage	1.000000	overall height → slow trough voltage	1.000000
soma surface total → sag	1.000000	soma surface total → threshold voltage	1.000000
total surface → tau	1.000000	total surface → threshold intensity	1.000000
adaptation index → input resistance	1.000000	avg. contraction → avg. parent/daughter ratio	1.000000
avg isi → sag	1.000000	sag → tau	1.000000
slow trough voltage → threshold intensity	1.000000	threshold intensity → avg isi	1.000000
threshold voltage → input resistance	1.000000	threshold voltage → trough voltage	1.000000
avg. diameter → avg isi	1.000000	avg. diameter → adaptation index	1.000000
avg. contraction → slow trough time	1.000000	soma surface total length → avg. parent/daughter ratio	1.000000
avg. diameter → avg. contraction	1.000000	avg. diameter → avg. fragmentation	1.000000
avg. diameter → overall width	1.000000	avg. parent daughter ratio → overall height	1.000000
overall height → overall width	1.000000	soma surface total length → avg. diameter	1.000000
soma surface total length → avg. fragmentation	1.000000	trough voltage → slow trough voltage.	1.000000
avg. bifurcation angle local → threshold intensity	1.000000	total surface → avg. bifurcation angle	1.000000
total volume → avg. diameter	1.000000	total volume → overall height	1.000000
total volume → total surface	1.000000	avg. fragmentation → adaptation index	0.563158
soma surface total length → total volume	0.519474	soma surface total length → adaptation index	0.512105
total volume → soma surface total	0.480526		

We can observe several expected relations that we predicted. For example the total surface of the neuron is related to the time constant. This is consistent with the models that describe the electrical properties of neurons. Larger cells mean thicker and larger dendrites which in turn modify resistance and capacitance at each point of the neuron (these are the variables that determine the time constant). Other relations are more evident, for example the total volume is connected to the total volume, avg. diameter. Other expected relations include the ones between the overall width, height and average diameter.

In the case of electrophysiological variables, the trough voltage and the slow trough voltage since they can even be the same under the conditions outlined in the definition. Another expected relation is that of the threshold intensity and the average ISI. Generally, the lower the intensity at which neurons fire the lower the inter-spike interval when subjected to a step current. These are the burst neurons which produce spikes and short groups of spikes.

As for possible hypothesis testing, we could propose to validate the relation between the average fragmentation and the adaptation index for example. This would mean that the number of compartments that constitute a branch between bifurcations is related to how a neuron responds to trains of action potentials.

6.3.1. Final remarks

A main problem in this application was the fact that we could not use a significant part of the variables effectively. This was the result of trying to adhere to the assumptions of the model. We also did not test prediction capabilities since we were

more interested in finding the right arcs. Once this is done, predicting is a much easier task. A greater problem is then that we need to use all variables together without incurring in the difficulties associated with different types of random variables. This would in turn allow a better assesment since we feel that in this case too much information has been lost by dropping the non normal variables. More knowledge could probably be extracted but this falls outside the scope of this work and in the field of Neuroscience.

7

Conclusion

In this work we have presented an algorithm for the discovery of relationships in continuous domain, specifically for the analysis of the dependence between electrophysiological and morphological descriptors of neurons (though the algorithm could be applied to any other field of inquiry). To that end we have presented an adaptation the Multidimensional Bayesian Network Classifier to the continuous domain which in this case allows for a clear separation of both sets of variables, facilitating the learning network structures by reducing the amount of total possible graphs.

In order to create a model that is as precise as possible we have opted for the use of a combination of models instead of a single estimate. To obtain those models we presented a variant of the popular MCMC scheme for structure learning with a new move that promotes exploration of the state space and attempts to escape. This move was applied in conjunction with the CBD structure restriction to accelerate convergence.

Finally we applied the method with data from the Cell Type database of the Allen Institute and showed some of the results obtained and how we can use the model to formulate hypothesis about the domain.

7.1. Future research and improvements

In future works several enhancements could be made. To increase the seep at which the algorithm moves converges and to increase its ability to escape local maximum the acceptance probability could be modified online, increasing its value with each rejection in order to overcome the difference in scores for adjacent states.

In a more algorithmic approach much of these operations like arc addition, deletion and updates of matrices could be implemented with matrix operations and ported to the GPU for faster computation speeding up the learning procedure, specially for larger domains.

A common problem with continuous domains is the difficulty in finding a good way to approximate the probability distributions of the domain variables. In this work we have used the MVN since it posses desirable mathematical properties that make working with it practical. The problem is that many of the variables that describe the properties of neurons do not have normal distributions when we plot the histogram of their values. In particular we found that some followed a distribution that more resembled a Gamma, although other had much more unstructured shapes. This may bebe related to the low number of observations used, at least for some of the variables, though others exhibit to extreme behavior to think that the problem will go away with large sample sizes.

A direction of research them would be to design a convenient way to manage these different distributions together. This problems has been tackled before and has proven very difficult but maybe a solution for a small subset of density functions could be found.

Although the CBD property can be exploited for prediction tasks it was not used in the learning phase. Two ways could be presented: trying to sample in the space of CBDs in a similar manner as order-MCMC or using defining a prior on them to encourage certain types of structures. The first option would give more control of the learning procedure but is also much more complicated. It will require to integrate over all graphs compatible with qa given CBD, which is all the DAGs that are compatible with that CBD. This means somehow enumerating them which is a NP-hard. This and other issues make this approach less appealing so instead working with a prior distribution seems like a better option from this point of view as it can easily be combined with the standard methods.

Bibliography

Akaike, H. (2011). Akaike's information criterion. In *International Encyclopedia of Statistical Science*, pages 25–25. Springer. 12

Allen Institute, B. S. (2016). Technical white paper:electrophysiology. Technical report, Allen Institute for Brain Science. 59

Allen Institute, B. S. (2017a). Celltype database. `http://celltypes.brain-map.org/`. 57

Allen Institute, B. S. (2017b). Technical white paper:morphology. Technical report, Allen Institute for Brain Science. 57

Appice, A. and Džeroski, S. (2007). Stepwise induction of multi-target model trees. In *European Conference on Machine Learning*, pages 502–509. Springer. 9

Bielza, C., Li, G., and Larra ñaga, P. (2011). Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727. 9, 26

Borchani, H., Varando, G., Bielza, C., and Larrañaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233. 8

Bouckaert, R. R. (2001). *Bayesian belief networks: from construction to inference*. PhD thesis. 14

Box, G. E. and Tiao, G. C. (2011). *Bayesian inference in statistical analysis*, volume 40. John Wiley & Sons. 28

Buntine, W. (1991). Learning classification trees. *Artificial Intelligence frontiers in statistics*, pages 182–201. 12

Campos, L. M. d. (2006). A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7(Oct):2149–2187. 12

BIBLIOGRAPHY

Caruana, R. (1998). Multitask learning. In *Learning to learn*, pages 95–133. Springer. 8

Carvalho, A. M. (2009). Scoring functions for learning bayesian networks. *Inesc-id Tec. Rep.* 12

Chickering, D., Geiger, D., and Heckerman, D. (1995). Learning bayesian networks: Search methods and experimental results. In *proceedings of fifth conference on artificial intelligence and statistics*, pages 112–128. 12

Chickering, D. M. (1996). Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer. 10

Chickering, D. M. (2002). Learning equivalence classes of bayesian-network structures. *Journal of machine learning research*, 2(Feb):445–498. 11

Cooper, G. F. and Herskovits, E. (1991). A bayesian method for constructing bayesian belief networks from databases. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pages 86–94. Morgan Kaufmann Publishers Inc. 10

Cooper, G. F. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347. 12, 13

De'Ath, G. (2002). Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117. 9

DeFelipe, J., López-Cruz, P. L., Benavides-Piccione, R., Bielza, C., Larrañaga, P., Anderson, S., Burkhalter, A., Cauli, B., Fairén, A., Feldmeyer, D., et al. (2013). New insights into the classification and nomenclature of cortical gabaergic interneurons. *Nature Reviews. Neuroscience*, 14(3):202. 4

DeGroot, M. H. (2005). *Optimal statistical decisions*, volume 82. John Wiley & Sons. 23, 24

Delaplace, A., Brouard, T., and Cardot, H. (2006). Two evolutionary methods for learning bayesian network structures. In *International Conference on Computational and Information Science*, pages 288–297. Springer. 14

Ding, P. (2016). On the conditional distribution of the multivariate t distribution. *The American Statistician*, 70(3):293–295. 25

Eaton, D. and Murphy, K. (2007). Exact bayesian structure learning from uncertain interventions. In *Artificial Intelligence and Statistics*, pages 107–114. 16

Ellis, B. and Wong, W. (2006). Sampling bayesian networks quickly. *Interface*. 11, 15

Eyal, G., Mansvelder, H. D., de Kock, C. P., and Segev, I. (2014). Dendrites impact the encoding capabilities of the axon. *Journal of Neuroscience*, 34(24):8063–8071. 3

Fourcaud-Trocmé, N., Hansel, D., Van Vreeswijk, C., and Brunel, N. (2003). How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of Neuroscience*, 23(37):11628–11640. 3

Friedman, N. and Koller, D. (2003). Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50(1-2):95–125. 11, 14, 15, 35, 50

Geiger, D. and Heckerman, D. (1994). Learning gaussian networks. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 235–243. Morgan Kaufmann Publishers Inc. 13, 21, 27, 35

Giudici, P., Green, P., and Tarantola, C. (2000). Efficient model determination for discrete graphical models. 15

Giudici, P. and Green, P. J. (1999). Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801. 15, 35, 46

Green, P. J. and Hastie, D. I. (2009). Reversible jump mcmc. *Genetics*, 155(3):1391–1403. 15

Grünwald, P. (2000). Model selection based on minimum description length. *Journal of Mathematical Psychology*, 44(1):133–152. 12

Grzegorczyk, M. and Husmeier, D. (2008). Improving the structure mcmc sampler for bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71(2):265–305. 16, 19, 35, 50, 52

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109. 32

Heckerman, D. (1998). A tutorial on learning with bayesian networks. In *Learning in graphical models*, pages 301–354. Springer. 10

Heckerman, D. and Geiger, D. (1995). Learning bayesian networks: a unification for discrete and gaussian domains. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 274–284. Morgan Kaufmann Publishers Inc. 14

Heckerman, D., Geiger, D., and Chickering, D. M. (1994). Learning bayesian networks: The combination of knowledge and statistical data. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 293–301. Morgan Kaufmann Publishers Inc. 12, 28

Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *European Conference on Machine Learning*, pages 624–631. Springer. 9

Koivisto, M. (2012). Advances in exact bayesian structure discovery in bayesian networks. *arXiv preprint arXiv:1206.6828*. 15, 16

Koivisto, M. and Sood, K. (2004). Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5(May):549–573. 15

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press. 10, 18, 21

Kuipers, J. and Moffa, G. (2017). Partition mcmc for inference on acyclic digraphs. *Journal of the American Statistical Association*, 112(517):282–299. 17

Kuipers, J., Moffa, G., Heckerman, D., et al. (2014). Addendum on the scoring of gaussian directed acyclic graphical models. *The Annals of Statistics*, 42(4):1689–1691. 13, 28, 47

Larkman, A. and Mason, A. (1990). Correlations between morphology and electrophysiology of pyramidal neurons in slices of rat visual cortex. i. establishment of cell classes. *Journal of Neuroscience*, 10(5):1407–1414. 3

Larrañaga, P., Kuijpers, C. M., Murga, R. H., and Yurramendi, Y. (1996a). Learning bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 26(4):487–493. 14

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. H. (1996b). Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE transactions on pattern analysis and machine intelligence*, 18(9):912–926. 13, 14

Lauritzen, S. L. (1996). *Graphical models*, volume 17. Clarendon Press. 10

López-Cruz, P. L., Larrañaga, P., DeFelipe, J., and Bielza, C. (2014). Bayesian network modeling of the consensus between experts: An application to neuron classification. *International Journal of Approximate Reasoning*, 55(1):3–22. 4

Madigan, D. and Hutchinson, F. (1995). Enhancing the predictive performance of bayesian graphical models. In *Communications in Statistics – Theory and Methods*. 47

Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232. 11, 14, 15, 34

Mainen, Z. F. and Sejnowski, T. J. (1996). Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature*, 382(6589):363. 3

Margaritis, D. (2003). *Learning Bayesian network model structure from data*. PhD thesis, US Army. 13

Masegosa, A. R. and Moral, S. (2013). New skeleton-based approaches for bayesian structure learning of bayesian networks. *Applied Soft Computing*, 13(2):1110–1120. 16

Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247):335–341. 32

Mohan, H., Verhoog, M. B., Doreswamy, K. K., Eyal, G., Aardse, R., Lodder, B. N., Goriounova, N. A., Asamoah, B., B. Brakspear, A. C., Groot, C., van der Sluis, S., Testa-Silva, G., Obermayer, J., Boudewijns, Z. S., Narayanan, R. T., Baayen, J. C., Segev, I., Mansvelder, H. D., and de Kock, C. P. (2015). Dendritic and axonal architecture of individual pyramidal neurons across layers of adult human neocortex. *Cerebral Cortex*, 25(12):4839. 3

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. 10, 23, 24, 32, 33

Niinimaki, T., Parviainen, P., and Koivisto, M. (2012). Partial order mcmc for structure discovery in bayesian networks. *arXiv preprint arXiv:1202.3753*. 16

Parviainen, P. and Koivisto, M. (2009). Exact structure discovery in bayesian networks with less space. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 436–443. AUAI Press. 15

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann. 10

Press, S. J. (2012). *Applied multivariate analysis: using Bayesian and frequentist methods of inference*. Courier Corporation. 24, 28

Rall, W., Segev, I., Rinzel, J., and Shepherd, G. M. (1995). *The theoretical foundation of dendritic function: selected papers of Wilfrid Rall with commentaries*. MIT press. 3

Robert, C. P. (2004). *Monte carlo methods*. Wiley Online Library. 30, 31, 32, 35

Robinson, R. W. (1978). *Asymptotic number of self-converse oriented graphs*, pages 255–266. Springer Berlin Heidelberg, Berlin, Heidelberg. 10

Roos, T., Silander, T., Kontkanen, P., and Myllymaki, P. (2008). Bayesian network structure learning using factorized nml universal models. In *Information Theory and Applications Workshop, 2008*, pages 272–276. IEEE. 12

Sánchez-Fernández, M., de Prado-Cumplido, M., Arenas-García, J., and Pérez-Cruz, F. (2004). Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE transactions on signal processing*, 52(8):2298–2307. 9

Shachter, R. D. and Kenley, C. R. (1989). Gaussian influence diagrams. *Management science*, 35(5):527–550. 22

Spiegelhalter, D. J. and Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605. 12

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., and Vlahavas, I. (2012). Multi-label classification methods for multi-target regression. *arXiv preprint arXiv:1211.6581*. 8

Su, C. and Borsuk, M. E. (2016). Improving structure mcmc for bayesian networks through markov blanket resampling. *Journal of Machine Learning Research*, 17(118):1–20. 16, 50

Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728. 31

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78. 13

Van Der Gaag, L. C. and De Waal, P. R. (2006). Multi-dimensional bayesian network classifiers. 9, 26

Vazquez, E. and Walter, E. (2003). Multi-output support vector regression. In *13th IFAC Symposium on System Identification*, pages 1820–1825. 9

Wang, T., Touchman, J. W., and Xue, G. (2004). Applying two-level simulated annealing on bayesian structure learning to infer genetic networks. In *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*, pages 647–648. IEEE. 14

Yuste, R. and Tank, D. W. (1996). Dendritic integration in mammalian neurons, a century after cajal. *Neuron*, 16(4):701–716. 3