Maria Curie-Skłodowska University in Lublin
Faculty of Mathematics, Physics and Computer Science

Maryia Shpak

Structure Learning and Parameter Estimation for Graphical Models via Penalized Maximum Likelihood Methods

PhD dissertation

Supervisor

dr hab. Mariusz Bieniek, prof. UMCS

Institute of Mathematics
University of Maria Curie-Sklodowska

April 2022

Abstract

Probabilistic graphical models (PGMs) provide a compact and flexible framework to model very complex real-life phenomena. They combine the probability theory which deals with uncertainty and logical structure represented by a graph which allows to cope with the computational complexity and also interprete and communicate the obtained knowledge. In the thesis we consider two different types of PGMs: Bayesian networks (BNs) which are static, and continuous time Bayesian networks which, as the name suggests, have temporal component. We are interested in recovering their true structure, which is the first step in learning any PGM. This is a challenging task, which is interesting in itself from the causal point of view, for the purposes of interpretation of the model and the decision making process. All approaches for structure learning in the thesis are united by the same idea of maximum likelihood estimation with LASSO penalty. The problem of structure learning is reduced to the problem of finding non-zero coefficients in the LASSO estimator for a generalized linear model. In case of CTBNs we consider the problem both for complete and incomplete data. We support the theoretical results with experiments.

Keywords and phrases: Probabilistic graphical models, PGM, Bayesian networks, BN, continuous time Bayesian networks, CTBN, maximum likelihood, LASSO penalty, structure learning, Markov Jump Process, MJP, Markov chain, Markov chain Monte Carlo, MCMC, Stochastic Proximal Gradient Descent, drift condition, incomplete data, Expectation-Maximization, EM.

Acknowledgements

Throughout the process of writing this thesis I have received a lot of support and assistance and I wish to express my gratitude.

First, I would like to thank my supervisor, Professor Mariusz Bieniek, who was a great support during this challenging process. His curiosity, open-mindedness and extensive knowledge gave me a chance to research things that are outside of his main field of expertise, and his strive for quality and perfection never let me settle for mediocre results.

Next, I want to thank my second advisor Professor Błażej Miasojedow from University of Warsaw, who introduced us to the field of probabilistic graphical models and some other areas of statistics, stochastic processes and numerical approximation. His great expertise and enormous patience allowed me to gain massive knowledge and understanding of these fields, when sometimes I did not believe I could.

I also would like to thank dr. Wojciech Rejchel from Nicolaus Copernicus University in Toruń , whose expertise in model selection was key in the analysis of theoretical properties of our novel methods for structure learning. I wish to thank mgr. Grzegorz Preisbich and mgr. Tomasz Cąkała for making many numerical results for our methods possible.

I want to thank my university, Maria Skłodowska-Curie University, for an academic leave giving me the opportunity to finish the dissertation and some additional funding. The part of the research was also supported by the Polish National Science Center grant: NCN contract with the number UMO-2018/31/B/ST1/00253. Also I would like to show my appreciation to other people from my university helping me in various ways, among them are Professor Maria Nowak, Professor Jarosław Bylina, Professor Tadeusz Kuczumow, Professor Jurij Kozicki and many others.

Finally, I would like to thank my parents, Pavel and Natallia, who were always there for me to guide me and help me through years of research, without their support this thesis would not be possible. I also wish to extend my special thanks to my dear friends for their emotional support and helping me to stay disciplined, especially I thank Elvira Tretiakova and Olga Kostina.

Contents

Chapter 1

Introduction

1.1 Motivation

It is a common knowledge that we live in the world where data plays crucial role in many areas and applications of great importance for our society and the importance of data is still growing. The amount of data in the world is now estimated in dozens of zettabytes, and by 2025 the amount of data generated **daily** is expected to reach hundreds of exabytes. There is a demand for models and algorithms that can deal with these amounts of data effectively finding useful patterns and providing better insights into the data. On top of it, most environments require reasoning under uncertainty. Probabilistic graphical models (PGMs) provide such a framework that allows to deal with these and many other challenges in various situations. The models combine the *probability theory* which deals with uncertainty in a mathematically consistent way, and *logical structure* which is represented by a graph encoding certain independence relationships among variables allowing to cope with the computational complexity.

PGMs encode joint distributions over a set of random variables (often of a significant amount) combining the graph theory and probabilities, which allows to represent many complex real-world phenomena compactly and overcome the complexity of the model which is exponential in the number of variables. There are also some other advantages that these models have. Namely, because of their clear structure, PGMs enable us to visualize, interprete and also communicate the gained knowledge to others as well as make decisions. Some models, for example Bayesian networks, have directed graphs in their core and offer ways to establish causality in various cases. Moreover, graphical models allow us not only to fit the observed data but also elegantly incorporate prior knowledge, e.g. from experts in the domain, into the model. Besides, certain models take into account a temporal component and consider systems' dynamics in time.

Graphical models are successfully applied to a large number of domains such as image processing and object recognition, medical diagnosis, manufacturing, finance, statistical physics, speech recognition, natural language processing and many others. Let us briefly present here a few examples of various applications.

Bayesian networks, one of the PGMs considered in this thesis, are extensively used in

the development of medical decision support systems helping doctors to diagnose patients more accurately. In the work by Wasyluk et al. (2001) the authors built and described a probabilistic causal model for diagnosis of liver disorders. In the domain of hepatology, inexperienced clinicians have been found to make a correct diagnosis in jaundiced patients in less than 45% of the cases. Moreover, the number of cases of liver disorders is on the rise and, especially at early stages of a disease, the correct diagnosis is difficult yet critical, because in many cases damage to the liver caused by an untreated disorder may be irreversible. As we already mentioned and as it is stressed out in the work above, a huge advantage that these models have is that they allow to combine existing frequency data with expert judgement within the framework as well as update themselves when the new data are obtained, for example patients data within a hospital or a clinic. What is also important in the medical diagnosis is that PGMs, Bayesian networks in particular, efficiently model simultaneous presence of multiple disorders, which happens quite often, but in many classification approaches the disorders are considered to be mutually exclusive. The overall model accuracy, as the authors Wasyluk et al. (2001) claim, seems to be better than that of beginning diagnosticians and reaches almost 80%, which can be used for the diagnosis itself as well as the way to help new doctors to learn the strategy and optimization of the diagnosis process. A few other examples of the PGMs application in medical field are management of childhood malaria in Malawi (Bathla Taneja et al. (2021)), estimating risk of coronary artery disease (Gupta et al. (2019)), etc.

The next popular area of graphical models application is computational biology, for example Gene Regulatory Network (GRN) inference. GRN consists of genes or parts of genes, regulatory proteins and interactions between them and plays a key role in mediating cellular functions and signalling pathways in cells. Accurate inference of GRN for a specific disease returns disease-associated regulatory proteins and genes, serving as potential targets for drug treatment. Chen and Xuan (2020) argued that Bayesian inference is particularly suitable for GRNs as it is very flexible for large-scale data integration, because the main challenge of GRNs is that there exist hundreds of proteins and tens of thousands of genes with one protein possibly regulating hundreds of genes and their regulatory relationship may vary across different cell types, tissues, or diseases. Moreover, the estimation is more robust and easier to compare on multiple datasets. Chen and Xuan (2020) demonstrated this by applying their model to breast cancer data and identified genes relevant to breast cancer recurrence. As another example in this area, Sachs et al. (2005) used Bayesian network computational methods for derivation of causal influences in cellular signalling networks. These methods automatically elucidated most of the traditionally reported signalling relationships and predicted novel interpathway network causalities, which were verified experimentally. Reconstruction of such networks might be applied to understanding native-state tissue signalling biology, complex drug actions, and dysfunctional signalling in diseased cells.

The use of probability models is extensive also in computer vision applications. In their work Frey and Jojic (2005) advocate for the use of PGMs in the computer vision problems

requiring decomposing the data into interacting components, for example, methods for automatic scene analysis. They apply different techniques in a vision model of multiple, occluding objects and compare their performances. Occlusion is a very important effect and one of the biggest challenges in computer vision that needs to be taken into account, and PGMs are considered to be a good tool to handle that effect. PGMs are also used for tracking different moving objects in video sequences, for example long-term tracking of groups of pedestrians on the street (Jorge et al. (2007)), where the main difficulties concern total occlusions of the objects to be tracked, as well as group merging and splitting. Another example is on-line object tracking (Jorge et al. (2004)) useful in real time applications such as video surveillance, where authors overcame the problem of needing to analyze the whole sequence before labelling trajectories to be able to use the tracker on-line and also the problem of unboundedly growing complexity of the network.

1.2 Probabilistic Graphical Models

In the previous subsection we described the advantages of PGMs and why one might be interested in studying them. In this work we focus on two types of PGMs: Bayesian Networks (BN) and Continuous Time Bayesian Networks (CTBN). The first term has rather long history and tracks back to 1980s (Pearl (1985)) whereas the second term is relatively modern (Nodelman et al. (2002)). The underlying structure for both models is a directed graph, which can be treated either as a representation of a certain set of independencies or as a skeleton for factorizing a distribution. In some cases the directions of arrows in the graph can suggest causality under certain conditions and allow not only the inference from the data but also intervene into the model and manipulate desired parameters in the future. BNs are static models, i.e. they do not consider a temporal component, while in CTBNs as the name suggests we study models in the context of continuous time. The framework of CTBNs is based on homogeneous Markov processes, but utilizes ideas from Bayesian networks to provide a graphical representation language for these systems.

A broad and comprehensive tutorial on existing research for learning Bayesian networks and some adjacent models can be found in Daly et al. (2011). The subject of causality is extensively explored in Spirtes et al. (2000) and Pearl (2000), some references are also given in Daly et al. (2011). Several examples of the use of BNs were presented above.

In contrast to regular Bayesian networks, CTBNs have not been studied that well yet. The most extensive work concerning CTBNs is PhD thesis of Nodelman (2007). Some related works include for example learning CTBNs in non-stationary domains (Villa and Stella (2018)), in relational domains (Yang et al. (2016)) and continuous time Bayesian network classifiers (Stella and Amer (2012)). As an example, CTBNs have been successfully used to model the presence of people at their computers together with their availability (Nodelman and Horvitz (2004)), for dynamical systems reliability modeling

and analysis (Boudali and Dugan (2006)), for network intrusion detection (Xu and Shelton (2008)), to model social networks (Fan and Shelton (2012)), to model cardiogenic heart failure (Gatti et al. (2012)), and for gene network inference (Stella et al. (2014) or Stella et al. (2016)).

1.3 Overview of the thesis and its contributions

There are several problems within both the BN and CTBN frameworks. Both of them have graph structures which need to be discovered and this is considered to be one of the main challenges in the field. This thesis is dedicated exclusively to solving this problem in both frameworks. Another problem is to learn the parameters of the model: in the case of BNs it is a set of conditional probability distributions and in the case of CTBNs it is a set of conditional intensity matrices (for details see Chapter 2). The last problem is the statistical inference based on the obtained network (details are in Chapter 3).

The thesis is constructed as follows. In Chapter 2 we provide all the necessary preliminaries for better understanding the frameworks of Bayesian networks and continuous time Bayesian networks. Next, in Chapter 3 we overview known results on learning networks' parameters as well as inference to fully cover the concept of interest. Chapter 4 is dedicated to the structure learning problem for BNs, where we provide novel algorithms for both discrete and continuous data. Chapters 5 and Chapter 6 cover the problems of structure learning for CTBNs in cases of complete and incomplete data, respectively. Finally, Chapter 7 concludes the thesis with the summary and the discussion of obtained results.

Algorithms in both Chapters 4 and 5 lean on feature selection in generalized linear models with the use of LASSO (Least Absolute Shrinkage and Selection Operator) penalty function. It relies on the idea of penalizing the parameters of the model, i.e. adding or subtracting the sum of absolute vales of the parameters of the model with some hyper-parameter, in order to better fit the model and perform a variable selection by forcing some parameters to be equal to 0. The term first appeared in Tibshirani (1996). More on the topic of LASSO can be found for example in Hastie et al. (2015). In Section 2.6 we provide a short description of the concept.

The main contributions of the thesis are collected in Chapters 4, 5 and 6 and they are as follows:

- we provide the novel algorithm for learning the structure of BNs based on penalized maximum likelihood function both for discrete and continuous data;
- we present and prove the consistency results for the algorithm in case of continuous data;
- we compare the effectiveness of our method with other most popular methods for structure learning applied to benchmark networks of continuous data of different sizes;

- we provide the novel algorithm for learning the structure of CTBNs based on penalized maximum likelihood function for **complete** data and present two theoretical consistency results with proofs;
- we provide the novel algorithm for learning the structure of CTBNs based on penalized maximum likelihood function for **incomplete** data where the log-likelihood function is replaced by its Markov Chain Monte Carlo (MCMC) approximation due to inability to express it explicitly;
- we present and prove the convergence of the proposed MCMC scheme and the consistency of the learning algorithm;
- for the mentioned above MCMC approximation we designed the algorithm to produce necessary samples;
- in both cases of complete and incomplete data we provide results of the simulations to show the effectiveness of proposed algorithms.

Part of the content (Chapter 5) in its early stages has been published on arXiv:

Shpak, M., Miasojedow, B., and Rejchel, W., *Structure learning for CTBNs via penalized maximum likelihood methods*, arXiv e-prints, 2020, https://doi.org/10.48550/arXiv.2006.07648.

Chapter 2

Preliminaries

In this chapter we provide theoretical background on Bayesian networks (BNs), Markov processes, conditional Markov processes and continuous time Bayesian networks (CTBNs). We start with the notation common for BNs and CTBNs which we will use through the whole thesis. Then we provide a few basic definitions needed to define and understand the concepts of BNs and CTBNs with their interpretation and examples. Most of the contents of this chapter comes from the Nodelman et al. (2002), Nodelman (2007), Koller and Friedman (2009).

2.1 Notation

First, by upper case letters, for example, X_i, B, Y, we denote random variables. In the case of CTBNs upper case letters represent the whole collection of random variables indexed by continuous time, hence in this case $X_i(t)$, $Y(t)$ are random variables for particular time points t.

Values of variables are denoted by lower case letters, sometimes indexed by numbers or otherwise representing different values of the same random variable - e.g. x_i, s, s'. The set of possible values for a variable X is denoted by $Val(X)$ and by $|X|$ we will denote the number of its elements.

Sets of variables are denoted by bold-face upper case letters - e.g. $\mathbf{X}$ - and corresponding sets of values are denoted by bold-face lower case letters - e.g. $\boldsymbol{x}$ or $\mathbf{x}$. The set of possible values and its size is denoted by $Val(\mathbf{X})$ and $|\mathbf{X}|$.

A pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes a directed graph, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. The notation $u \to w$ means that there exists an edge from the node u to the node w. We will also call them arrows. The set $\mathcal{V} \setminus \{w\}$ is denoted by $-w$. Moreover, we define the set of the parents of the node w in the graph $\mathcal{G}$ by

$$\mathbf{pa}_{\mathcal{G}}(w) = \{u \in \mathcal{V} \ : \ u \to w\}.$$

When there is no confusion, for convenience we sometimes write $\mathbf{pa}(w)$ instead of $\mathbf{pa}_{\mathcal{G}}(w)$. Other useful and relevant locally notation we provide in the corresponding sections.

2.2 Bayesian networks

In this section we provide an overview of Bayesian networks (BNs). We start with the intuition behind BNs followed by the representation of BNs together with its formal definition and notation. The problems of inference and learning for BNs are considered more thoroughly in Section 3.2 and Chapter 4 respectively.

The goal is to represent a joint distribution p over some set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. Even in the simplest case where these variables are binary-valued, the joint distribution requires the specification of $2^n - 1$ numbers - the probabilities of the 2^n different assignments of the values $\{x_1, \ldots, x_n\}$. The explicit representation of the joint distribution is hard to handle from every perspective except for small values of n. Computationally, it is very expensive to manipulate and generally too large to store in computer memory. Cognitively, it is impossible to acquire so many numbers from a human expert; moreover, most of the numbers would be very small and would correspond to events that people cannot reasonably consider. Statistically, if we want to learn the distribution from data, we would need ridiculously large amounts of data to estimate so many parameters robustly (Koller and Friedman (2009)).

Bayesian networks help us specify a high-dimensional joint distribution compactly by exploiting its independence properties. The key notion behind the BN representation is *conditional independence*, which on the one hand allows to reduce amount of estimated parameters significantly and on the other hand, allows to avoid very strong and naive independence assumptions.

Definition 2.1. *Two random variables X and Y are **independent** (denoted by $X \perp Y$) if and only if the equality*

$$\mathbb{P}(X \in A, Y \in B) = \mathbb{P}(X \in A)\mathbb{P}(Y \in B)$$

holds for all Borel sets $A, B \subseteq \mathbb{R}$.

For short, we will write it in the form $\mathbb{P}(X, Y) = \mathbb{P}(X)\mathbb{P}(Y)$. There is also another way to think of independence. If the random variables X and Y are independent, then $\mathbb{P}(X \in \cdot \mid Y) = \mathbb{P}(X \in \cdot)$. Intuitively, this says that having evidence about Y does not change the distribution of our beliefs on the occurrence of X.

If we wish to model a more complex domain represented by some set of variables, it is unlikely that any of the variables will be independent of each other. Conditional independence is a weaker notion of independence, but it is more common in real-life situations.

Definition 2.2. *Two random variables X and Y are **conditionally independent** given a set of random variables $\mathbf{C}$ (symbolically $X \perp Y \mid \mathbf{C}$) if and only if*

$$\mathbb{P}(X \in A, Y \in B \mid \mathbf{C}) = \mathbb{P}(X \in A \mid \mathbf{C})\mathbb{P}(Y \in B \mid \mathbf{C}) \tag{2.1}$$

holds for all Borel sets $A, B \subseteq \mathbb{R}$.

Obviously (2.1) implies

$$\mathbb{P}(X \in A \mid \mathbf{C}, Y) = \mathbb{P}(X \in A \mid \mathbf{C}),$$

which can be written shortly as

$$\mathbb{P}(X \mid \mathbf{C}, Y) = \mathbb{P}(X \mid \mathbf{C}).$$

So intuitively, the influence that X and Y have on each other is mediated through the variables in the set $\mathbf{C}$. It means that, when we have some evidence about variables from $\mathbf{C}$, having any additional information about Y does not change our beliefs about X. Let us demonstrate this definition on a simplified example. Let X be a random variable representing the case if a person has lung cancer and Y representing the case if the same person has yellow teeth. These variables are not independent as having yellow teeth is one of the secondary symptoms of lung cancer. However, when we know that the person is a smoker knowing that they have yellow teeth does not give us any additional insight on lung cancer, and vice versa, as we consider smoking to be the reason of both symptoms.

It is easier to believe that in a given domain most variables will not directly affect most other variables. Instead, for each variable only a limited set of other variables influence it. This is the intuition which leads to the notion of a Bayesian network $\mathcal{B}$ over a set of random variables $\mathbf{B}$ which is a compact representation of a specific joint probability distribution. The formal definition is as follows.

Definition 2.3. *A Bayesian network $\mathcal{B}$ over a set of random variables $\mathbf{B}$ is formed by*

- *a directed acyclic graph (DAG) $\mathcal{G}$ whose nodes correspond to the random variables $B_i \in \mathbf{B}$, $i = 1, \ldots, n$.*
- *the set of conditional probability distributions (CPDs) for each B_i, specifying the conditional distribution $\mathbb{P}(B_i \mid \mathbf{pa}_{\mathcal{G}}(B_i))$ of B_i as a function of its parent set in $\mathcal{G}$.*

The CPDs form a set of local probability models that can be combined to describe the full joint distribution over the variables $\mathbf{B}$ via the chain rule:

$$\mathbb{P}(B_1, B_2, ..., B_n) = \prod_{i=1}^{n} \mathbb{P}(B_i \mid \mathbf{pa}_{\mathcal{G}}(B_i)). \tag{2.2}$$

The graph $\mathcal{G}$ of a Bayesian network encodes a set of conditional independence assumptions. In particular, a variable $B \in \mathbf{B}$ is independent of its non-descendants given the set of its parents $\mathbf{pa}_{\mathcal{G}}(B)$. See for example Figure 2.1 of an Extended Student network taken from Koller and Friedman (2009). As it can be seen, each variable is connected only to a small amount of other variables in the network. In this example according to (2.2) the joint distribution takes the following form:

$$\begin{aligned}&\mathbb{P}(C, D, I, G, S, L, J, H) = \\ &= \mathbb{P}(C)\mathbb{P}(D \mid C)\mathbb{P}(I)\mathbb{P}(G \mid D, I)\mathbb{P}(S \mid I)\mathbb{P}(L \mid G)\mathbb{P}(J \mid L, S)\mathbb{P}(H \mid G, J).\end{aligned}$$

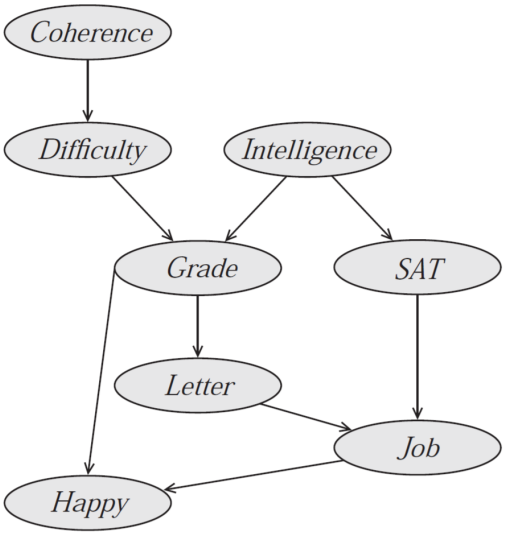


Figure 2.1: The Extended Student network

This example will be considered in more detail further in the thesis.

Now we discuss basic structures for BNs including some examples and give the interpretation of the structures. BNs represent probability distributions that can be formed via products of smaller, local conditional probability distributions (one for each variable). If the joint distribution is expressed in this form, it means that the independence assumptions for certain variables are introduced into our model. To understand what types of independencies are described by directed graphs for simplicity let us start from looking at BN $\mathcal{B}$ with three nodes: X, Y, and Z. In this case, $\mathcal{B}$ essentially has only three possible structures, each of which leads to different independence assumptions.

- *Common parent,* also called *common cause.* If $\mathcal{G}$ is of the form $X \leftarrow Y \rightarrow Z$, and Y is observed, then $X \perp Z \mid Y$. However, if Y is unobserved, then $X \not\perp Z$. Intuitively this stems from the fact that Y contains all the information that determines the outcomes of X and Z; once it is observed, there is nothing else that affects these variables' outcomes. The case with smoking and lung cancer described above is such an example of common cause. See the illustration (c) in Figure 2.2.

- *Cascade, or indirect connection.* If $\mathcal{G}$ is of the form $X \rightarrow Y \rightarrow Z$, and Y is observed, then, again $X \perp Z \mid Y$. However, if Y is unobserved, then $X \not\perp Z$. Here, the intuition is again that Y holds all the information that determines the outcome of Z; thus, it does not matter what value X takes. In Figure 2.2 in (a) and (b) there are shown cases of indirect causal and indirect evidential effects, respectively.

- *V-structure or common effect*, also known as *explaining away.* If $\mathcal{G}$ is of the form $X \rightarrow Y \leftarrow Z$, then knowing Y couples X and Z. In other words, $X \perp Z$ if Y is unobserved, but $X \not\perp Z \mid Y$ if Y is observed. See the case (d) in Figure 2.2.

The last case requires additional explanation. Suppose that Y is a Boolean variable

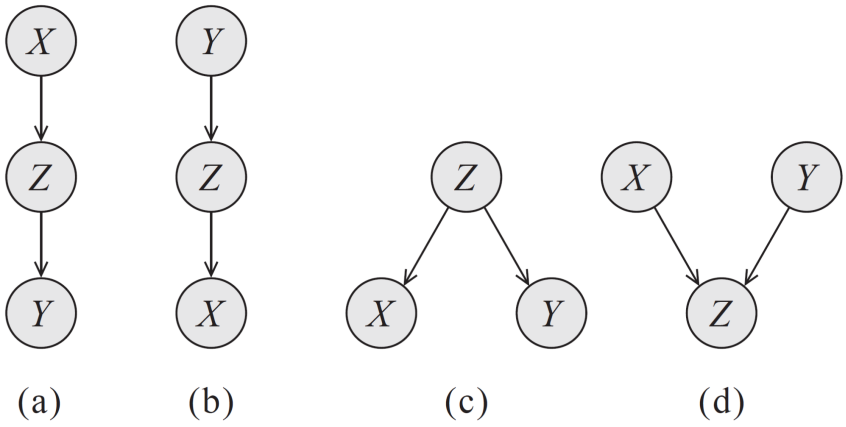


Figure 2.2: The four possible two-edge trails from X to Y via Z: (a) An indirect causal effect; (b) An indirect evidential effect; (c) A common cause; (d) A common effect.

that indicates whether our lawn is wet one morning; X and Z are two explanations for it being wet: either it rained (indicated by X), or the sprinkler turned on (indicated by Z). If we know that the grass is wet (Y is true) and the sprinkler did not go on (Z is false), then the probability that X is true must be one, because that is the only other possible explanation. Hence, X and Z are not independent given Y.

To generalize this for a case of more variables and demonstrate the power but also the limitations of Bayesian networks we will need the notions of d-separation and I-maps. Let $\mathbf{Q}, \mathbf{W}$, and $\mathbf{O}$ be three sets of nodes in a Bayesian network $\mathcal{B}$ represented by $\mathcal{G}$, where the variables $\mathbf{O}$ are observed. Let us use the notation $I(p)$ to denote the set of all independencies of the form $(\mathbf{Q} \perp \mathbf{W} \mid \mathbf{O})$ that hold in a joint distribution p. To extend structures mentioned above to more general networks we can apply them recursively over any larger graph, which leads to the notion of d-separation.

Recall that we say that there exists an undirected path in $\mathcal{G}$ between the nodes u and w if there exists the sequence $v_1, \ldots, v_n \in \mathcal{V}$ such that $v_i \to v_{i+1}$ or $v_i \leftarrow v_{i+1}$ for each $i = 0, 1, \ldots, n$, where $v_0 = u$ and $v_{n+1} = w$. Moreover, an undirected path in $\mathcal{G}$ between $Q \in \mathbf{Q}$ and $W \in \mathbf{W}$ is called *active* given observed variables $\mathbf{O}$ if for every consecutive triple of variables X, Y, Z on the path, one of the following holds:

- *common cause:* $X \leftarrow Y \to Z$ and $Y \notin \mathbf{O}$ (Y is unobserved);
- *causal trail:* $X \to Y \to Z$ and $Y \notin \mathbf{O}$ (Y is unobserved);
- *evidential trail:* $X \leftarrow Y \leftarrow Z$ and $Y \notin \mathbf{O}$ (Y is unobserved);
- *common effect:* $X \to Y \leftarrow Z$ and Y or any of its descendants are observed.

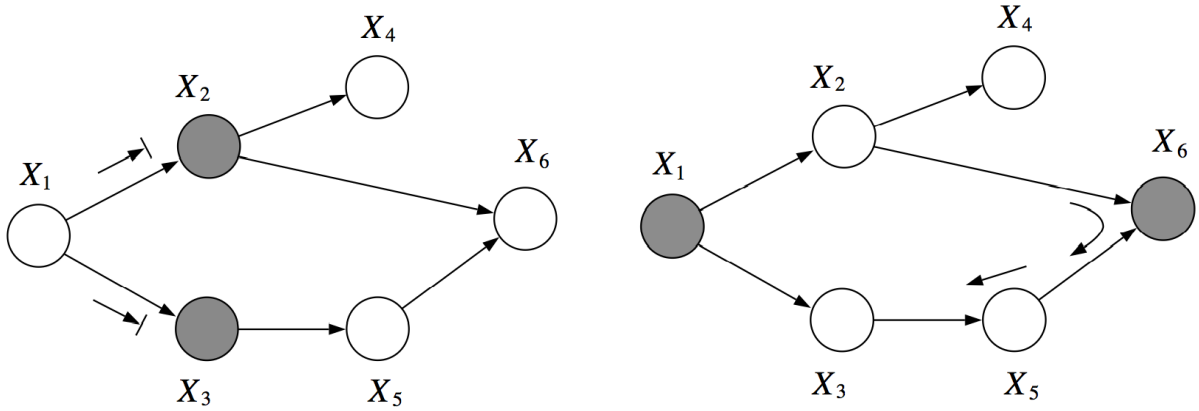


Figure 2.3: An example for d-separation: X_1 and X_6 are d-separated given X_2,X_3 (left), X_2, X_3 are **not** d-separated given X_1, X_6 (right).

Finally, we say that $\mathbf{Q}$ and $\mathbf{W}$ are d-separated given $\mathbf{O}$ if there are no active paths between any node $A \in \mathbf{Q}$ and $B \in \mathbf{W}$ given $\mathbf{O}$. See examples for d-separation in Figure 2.3. In the second example there is no d-separation because there is an active path which passes through the V-structure created when X_6 is observed. The notion of d-separation lets us describe a large fraction of the dependencies that hold in our model. It can be shown that if $\mathbf{Q}$ and $\mathbf{W}$ are d-separated given $\mathbf{O}$, then $\mathbf{Q} \perp \mathbf{W} \mid \mathbf{O}$.

We will write $I(\mathcal{G}) = \{(\mathbf{Q} \perp \mathbf{W} \mid \mathbf{O}) : \mathbf{Q}, \mathbf{W}$ are d-separated given $\mathbf{O}\}$ to denote the set of independencies corresponding to all d-separations in $\mathcal{G}$. If p factorizes over $\mathcal{G}$, then $I(\mathcal{G}) \subseteq I(p)$ and p can be constructed easily. In this case, we say that $\mathcal{G}$ is an I-map for p. In other words, all the independencies encoded in $\mathcal{G}$ are sound: variables that are d-separated in $\mathcal{G}$ are conditionally independent with respect to p. However, the converse is not true: a distribution may factorize over $\mathcal{G}$, yet have independencies that are not captured in $\mathcal{G}$.

So an interesting question here is whether for the probability distribution p we can always find a *perfect* map $I(\mathcal{G})$ for which $I(\mathcal{G}) = I(p)$ or not. The answer is no (see an example from Koller and Friedman (2009)). Another related question is whether perfect maps are unique when they exist. This is not the case either, for example, DAGs $X \rightarrow Y$ and $X \leftarrow Y$ encode the same independencies, yet form different graphs. In a general case we say that two Bayesian networks $\mathcal{B}_1$, $\mathcal{B}_2$ are I-equivalent if their DAGs encode the same dependencies $I(\mathcal{G}_1) = I(\mathcal{G}_2)$. For a case of three variables we can notice that graphs (a), (b) and (c) in Figure 2.2 encode the same dependencies, so as long as we do not turn graphs into V-structures ((d) is the only structure which encodes the dependency $X \not\perp Y \mid Z$) we can change directions in them and get I-equivalent graphs. This brings us to a fact that if $\mathcal{G}_1, \mathcal{G}_2$ have the same skeleton (meaning that if we drop the directionality of the arrows, we obtain the same undirected graph) and the same V-structures, then $I(\mathcal{G}_1) = I(\mathcal{G}_2)$. For the full proof of this statement, other previously made statements and more information about BNs see Koller and Friedman (2009).

2.3 Continuous Time Markov Processes

In this section we collect auxiliary results on Markov processes with continuous time. We can think of a continuous time random process X as a collection of random variables indexed by time $t \in [0, \infty)$. It is sometimes more convenient to view X across all values of t as a single variable, whose values are functions of time, also called paths or trajectories.

Definition 2.4. *The Markov condition is the assumption that the future of a process is independent of its past given its present. More explicitly, the process X satisfies the Markov property iff $\mathbb{P}(X(t+\Delta t) \mid X(s), 0 \leq s \leq t) = \mathbb{P}(X(t+\Delta t) \mid X(t))$ for all $t, \Delta t > 0$ (Chung and Walsh (2005)).*

In this thesis we focus on Markov processes with finite state space which are basically defined by initial distribution and a matrix of transition intensities. The framework of CTBNs is based on the notion of *homogeneous Markov processes* in which the transition intensities do not depend on time.

Definition 2.5. *Let X be a stochastic process with continuous time. Let the state space of X be $Val(X) = \{x_1, x_2, ..., x_N\}$. Then X is a homogeneous Markov process if and only if its behavior can be specified in terms of an initial distribution P_0^X over $Val(X)$ and a Markovian transition model usually presented as an intensity matrix*

$$\mathbf{Q}_X = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1N} \\ q_{21} & -q_2 & \cdots & q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & -q_N \end{bmatrix}, \tag{2.3}$$

where $q_i = \sum_{j \neq i} q_{ij}$ and all the entries q_i and q_{ij} are positive.

Intuitively, the intensity q_i gives the "instantaneous probability" of leaving state x_i and the intensity q_{ij} gives the "instantaneous probability" of the jump from x_i to x_j. More formally, for $i \neq j$

$$\lim_{\Delta t \to 0} \mathbb{P}(X(t + \Delta t) = x_j \mid X(t) = x_i) = q_{ij}\Delta t + O(\Delta t^2), \tag{2.4}$$

and for all $i = 1, \ldots, N$

$$\lim_{\Delta t \to 0} \mathbb{P}(X(t + \Delta t) = x_i \mid X(t) = x_i) = 1 - q_i\Delta t + O(\Delta t^2). \tag{2.5}$$

Therefore, the matrix $\mathbf{Q}_X$ describes the instantaneous behavior of the process X and also makes the process satisfy the Markov assumption since it is defined solely in terms of its current state.

The instantaneous specification of the transition model of X induces a probability distribution over the set of its possible trajectories. To see how the distribution is induced, we must first recall the notion of a matrix function.

Definition 2.6. *The matrix exponential for a matrix* $\mathbf{Q}$ *is defined as*

$$\exp \mathbf{Q} = \sum_{k=0}^{\infty} \frac{\mathbf{Q}^k}{k!}.$$

Now the set of Equations (2.4) and (2.5) can be written collectively in the form

$$\lim_{\Delta t \to 0} \mathbb{P}(X(t+\Delta t) \mid X(t)) = \lim_{\Delta t \to 0} \exp(\mathbf{Q}_X \Delta t) = \lim_{\Delta t \to 0} \left(\mathbf{I} + \mathbf{Q}_X \Delta t + O(\Delta t^2)\right). \tag{2.6}$$

So given the matrix $\mathbf{Q}_X$ we can describe the transient behavior of $X(t)$ as follows. If $X(0) = x_i$ then the process stays in state x_i for an amount of time exponentially distributed with parameter q_i. Hence, the probability density function f and the corresponding distribution function F for the time when $X(t)$ remains equal to x_i are given by

$$\begin{aligned} f(t) &= q_i \exp(-q_i t), \quad t \geq 0, \\ F(t) &= 1 - \exp(-q_i t), \quad t \geq 0. \end{aligned}$$

The expected time of changing the state is $1/q_i$. Upon transitioning, X jumps to the state x_j with probability q_{ij}/q_i for $j \neq i$.

Example 2.7. Assume that we want to model the behavior of the barometric pressure $B(t)$ discretized into three states ($b_1 =$ falling, $b_2 =$ steady, and $b_3 =$ rising). Then for instance we could write the intensity matrix as

$$\mathbf{Q}_B = \begin{bmatrix} -0.21 & 0.2 & 0.01 \\ 0.05 & -0.1 & 0.05 \\ 0.01 & 0.2 & -0.21 \end{bmatrix}.$$

If we view units of time as hours, this means that if the pressure is falling, we expect that it will stop falling in a little less than 5 hours (1/0.21 hours). It will then transition to being steady with probability $0.2/0.21 \approx 0.95$ and to falling with probability $0.01/0.21 \approx 0.0476$.

When the transition model is defined solely in terms of an intensity matrix (as above), we refer to it as using *a pure intensity* parameterization. The parameters for an N state process are $\{q_i, q_{ij} \in \mathbf{Q}_X,\ 1 \leq i, j \leq N,\ i \neq j\}$.

This is not the only way to parameterize a Markov process. Note that the distribution over transitions of X factors into two pieces: an exponential distribution over *when* the next transition will occur and a multinomial distribution over *where* the process jumps. This is called a *mixed intensity* parameterization.

Definition 2.8. *The mixed intensity parameterization for a homogeneous Markov process* X *with* N *states is given by two sets of parameters*

$$\boldsymbol{q}_X = \{q_i,\ 1 \leq i \leq N\}$$

and

$$\boldsymbol{\theta}_X = \{\theta_{ij},\ 1 \leq i, j \leq N,\ i \neq j\},$$

where $\boldsymbol{q}_X$ *is a set of intensities parameterizing the exponential distributions over* ***when*** *the next transition occurs and* $\boldsymbol{\theta}_X$ *is a set of probabilities parameterizing the distribution over* ***where*** *the process jumps.*

To relate these two parametrizations we note the following theorem from Nodelman (2007).

Theorem 2.9. *Let X and Y be two Markov processes with the same state space and the same initial distribution. If X is defined by the intensity matrix $\mathbf{Q}_X$ given by (2.3), and Y is the process defined by the mixed intensity parameterization $\boldsymbol{q}_Y = \{q'_1, \ldots, q'_N\}$ and $\boldsymbol{\theta}_Y = \{\theta'_{ij}, i \neq j\}$, then X and Y are stochastically equivalent, meaning they have the same state space and transition probabilities, if and only if $q'_i = q_i$ for all $i = 1, \ldots, N$ and*

$$\theta'_{ij} = \frac{q_{ij}}{q_i}$$

for all $1 \leq i, j \leq N,\ i \neq j$.

2.4 Conditional Markov Processes

In order to compose Markov processes in a larger network, we need to introduce the notion of a conditional Markov process. This is an inhomogeneous Markov process where the intensities vary with time, but not as a direct function of time. Rather, the intensities depend on the current values of a set of other variables, which also evolve as Markov processes.

Let Y be a process with a state space $Val(Y) = \{y_1, y_2, ..., y_m\}$. Assume that Y evolves as a Markov process $Y(t)$ whose dynamics are conditioned on a set $\mathbf{V}$ of variables, each of which can also evolve over time. Then we have a conditional intensity matrix (CIM) which can be written as

$$\mathbf{Q}_{Y|\mathbf{V}} = \begin{bmatrix} -q_1(\mathbf{V}) & q_{12}(\mathbf{V}) & \ldots & q_{1m}(\mathbf{V}) \\ q_{21}(\mathbf{V}) & -q_2(\mathbf{V}) & \ldots & q_{2m}(\mathbf{V}) \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1}(\mathbf{V}) & q_{m2}(\mathbf{V}) & \ldots & -q_m(\mathbf{V}) \end{bmatrix}.$$

Equivalently, we can view CIM as a set of intensity matrices $\mathbf{Q}_{Y|\boldsymbol{v}}$ one for each instantiation of values $\boldsymbol{v}$ to the variables $\mathbf{V}$, see Example 2.10. Since the framework of CTBNs which we consider in the thesis has a graph at its core, we will refer to the set of variables $\mathbf{V}$ as the set of parents of Y and denote it by $\mathbf{pa}_{\mathcal{G}}(Y)$. Note that if the parent set $\mathbf{pa}_{\mathcal{G}}(Y)$ is empty, then CIM is simply a standard intensity matrix. Just as a regular intensity matrix, CIM induces the distribution of the dynamics of Y given the behavior of $\mathbf{pa}_{\mathcal{G}}(Y) = \mathbf{V}$. If $\mathbf{V}$ takes the value $\boldsymbol{v}$ on the interval $[t, t+\varepsilon)$ for some $\varepsilon > 0$, then as in Equation (2.6)

$$\lim_{\Delta t \to 0} \mathbb{P}(Y_{t+\Delta t} \mid Y_t, \boldsymbol{v}) = \lim_{\Delta t \to 0} \exp(\mathbf{Q}_{Y|\boldsymbol{v}} \Delta t) = \lim_{\Delta t \to 0} \left(\mathbf{I} + \mathbf{Q}_{Y|\boldsymbol{v}} \Delta t + O(\Delta t^2)\right).$$

If we specify an initial distribution of Y, then we have defined a Markov process whose behavior depends on the instantiation $\boldsymbol{v}$ of values of $\mathbf{pa}_{\mathcal{G}}(Y)$.

Example 2.10. Consider a variable $E(t)$ which models whether or not a person is eating (e_1 = not eating, e_2 = eating) conditioned on a variable $H(t)$ which models whether or

not the person is hungry (h_1 = not hungry, h_2 = hungry). Then we can specify exemplary CIM for $E(t)$ as

$$Q_{E|h_1} = \begin{bmatrix} -0.01 & 0.01 \\ 10 & -10 \end{bmatrix} \qquad Q_{E|h_2} = \begin{bmatrix} -2 & 2 \\ 0.01 & -0.01 \end{bmatrix}.$$

For instance, given this model, we expect that a person who is hungry and not eating is going to start eating in half an hour. Also, we expect a person who is not hungry and is eating to stop eating in 6 minutes (1/10 hour).

2.5 Continuous time Bayesian networks

In this section we define the notion of CTBN, which in essence is a probabilistic graphical model with the nodes as variables, the state evolving continuously over time, and where the evolution of each variable depends on the state of its parents in the graph.

Before the formal definition we recall an example from Nodelman et al. (2002). Consider the situation in medical research where some drug has been administered to a patient and we wish to know how much time it takes for the drug to have an effect. The answer to this question will likely depend on various factors, such as how recently the patient ate. We want to model the temporal process for the effect of the drug and how its dynamics depends on other factors. In contrast to previously developed methods of approaching such a problem (e.g. event history analysis, Markov process models) the notion of CTBN introduced by Nodelman et al. (2002) allows the specification of models with a large structured state space where some variables do not directly depend on others. For example, the distribution of how fast the drug takes effect might be mediated through how fast it reaches the bloodstream, which in turn may be affected by how recently the person ate. Figure 2.4 shows an exemplary graph structure for CTBN modelling the drug effect. There are nodes for the uptake of the drug and for the resulting concentration of the drug in the bloodstream. The concentration is also affected by how full patient's stomach is. The drug is supposed to alleviate joint pain, which may be aggravated by falling pressure. The drug may also cause drowsiness. The model contains a cycle, indicating that whether the person is hungry depends on how full their stomach is, which depends on whether or not they are eating.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a directed graph **with possible cycles**, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Further in the context of probabilistic graphical models we use the terms "nodes" and "random variables" interchangeably. For every $w \in \mathcal{V}$ we consider a corresponding space $\mathcal{X}_w$ of possible states at w and we assume that each space $\mathcal{X}_w$ is finite. We consider a continuous time stochastic process on a product space $\mathcal{X} = \prod_{w \in \mathcal{V}} \mathcal{X}_w$, so a state $\mathbf{s} \in \mathcal{X}$ is a configuration $\mathbf{s} = (s_w)_{w \in \mathcal{V}}$, where $s_w \in \mathcal{X}_w$. If $\mathcal{W} \subseteq \mathcal{V}$, then we write $s_\mathcal{W} = (s_w)_{w \in \mathcal{W}}$ for the configuration $\mathbf{s}$ restricted to the nodes in $\mathcal{W}$. We also use the notation $\mathcal{X}_\mathcal{W} = \prod_{w \in \mathcal{W}} \mathcal{X}_w$, so we can write $s_\mathcal{W} \in \mathcal{X}_\mathcal{W}$. In what follows

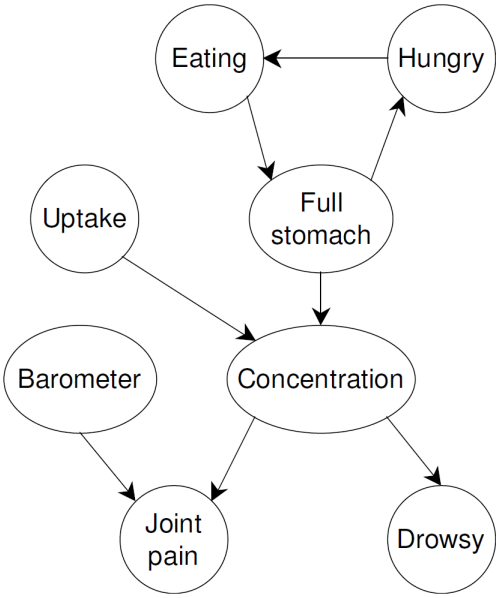


Figure 2.4: (a)

we use the bold symbol $\mathbf{s}$ to denote configurations belonging to $\mathcal{X}$ only. All restricted configurations will be denoted with the standard font s.

Now suppose we have a family of functions $Q_w : \mathcal{X}_{\mathbf{pa}_\mathcal{G}(w)} \times (\mathcal{X}_w \times \mathcal{X}_w) \to [0, \infty)$. For a fixed $c \in \mathcal{X}_{\mathbf{pa}_\mathcal{G}(w)}$ we consider $Q_w(c; \cdot, \cdot\ )$ as a conditional intensity matrix (CIM) at the node w (only off-diagonal elements of this matrix have to be specified, the diagonal ones are irrelevant). The state of CTBN at time t is a random element $X(t)$ of the space $\mathcal{X}$ of all the configurations. Let $X_w(t)$ denote its w-th coordinate. The process $\{(X_w(t))_{w \in \mathcal{V}} : t \geq 0\}$ is assumed to be Markov and its evolution can be described informally as follows: transitions, or jumps, at the node w depend on the current configuration of its parents. If the state of any parent changes, then the node w switches to other transition probabilities. If $s_w \neq s'_w$, where $s_w, s'_w \in \mathcal{X}_w$, then

$$\mathbb{P}\left(X_w(t + \mathrm{d}t) = s'_w \mid X_{-w}(t) = s_{-w}, X_w(t) = s_w\right) = Q_w(s_{\mathbf{pa}_\mathcal{G}(w)}, s_w, s'_w)\,\mathrm{d}t.$$

Definition 2.11. *A continuous time Bayesian network $\mathcal{N}$ over a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ is formed by two components. The first one is an initial distribution $P^0_\mathbf{X}$ specified as a Bayesian network $\mathcal{B}$ over $\mathbf{X}$. The second component is a continuous transition model, specified as*

- *a directed (possibly cyclic) graph $\mathcal{G}$ whose nodes correspond to the random variables X_i;*
- *a conditional intensity matrix $\mathbf{Q}_{X_i|\mathbf{pa}_\mathcal{G}(X_i)}$, specifying the continuous dynamic of each variable X_i given its parents' configuration.*

Essentially, CTBN is a Markov jump process (MJP) on the state space $\mathcal{X}$ with transition intensities given by

$$Q(\mathbf{s},\mathbf{s}') = \begin{cases} Q_w(s_{\mathbf{pa}_\mathcal{G}(w)}, s_w, s'_w), & \text{if } s_{-w} = s'_{-w} \text{ and } s_w \neq s'_w \text{ for some } w, \\ 0, & \text{if } s_{-w} \neq s'_{-w} \text{ for all } w, \end{cases} \tag{2.7}$$

for $\mathbf{s} \neq \mathbf{s}'$. Obviously, $Q(\mathbf{s},\mathbf{s})$ is defined "by subtraction" to ensure that $\sum\limits_{\mathbf{s}'} Q(\mathbf{s},\mathbf{s}') = 0$. For convenience, we will often write $Q(\mathbf{s}) = -Q(\mathbf{s},\mathbf{s})$ so that $Q(\mathbf{s}) \geq 0$. In particular, $Q_w(c; s_w) = -\sum\limits_{s \neq s'} Q_w(c; s, s')$.

It is important to note that we make a fundamental assumption in the construction of the CTBN model: two variables cannot transition at the same time (a zero in the definition of $Q(\mathbf{s},\mathbf{s})$). This can be viewed as a formalization of the view that variables must represent distinct aspects of the world. We should not, therefore, model a domain in which we have two variables that functionally and deterministically change simultaneously. For example, in the drug effect network, we should not add a variable describing the type of food, if any, a person is eating. We could, however, change the value space of the "Eating" variable from a binary "yes/no" to a more descriptive set of possibilities.

Further we omit the symbol $\mathcal{G}$ in the indices and write $\mathbf{pa}(w)$ instead of $\mathbf{pa}_\mathcal{G}(w)$. For CTBN the density of a sample trajectory $X = X([0,T])$ on a bounded time interval $[0,T]$ decomposes as follows:

$$p(X) = \nu(X(0)) \prod_{w \in \mathcal{V}} p(X_w \mid\mid X_{\mathbf{pa}(w)}) \,, \tag{2.8}$$

where ν is the initial distribution on $\mathcal{X}$ and $p(X_w \mid\mid X_{\mathbf{pa}(w)})$ is the density of piecewise homogeneous Markov jump process with the intensity matrix equal to $Q_w(c; \cdot, \cdot\,)$ in every time sub-interval such that $X_{\mathbf{pa}(w)} = c$. Below we explicitly write an expression for the density $p(X_w \mid\mid X_{\mathbf{pa}(w)})$ in terms of moments of jumps and the skeleton of the process $(X_w, X_{\mathbf{pa}(w)})$, as in (2.8), where by skeleton we understand the sequence of states of the process corresponding to the sequence of moments of time.

Let $T^w = (t_0^w \ldots, t_i^w, \ldots)$ and $T^{\mathbf{pa}(w)} = (t_0^{\mathbf{pa}(w)}, \ldots, t_j^{\mathbf{pa}(w)}, \ldots)$ denote moments of jumps at the node $w \in \mathcal{V}$ and at parent nodes, respectively. By convention, put $t_0^w = t_0^{\mathbf{pa}(w)} = 0$ and $t^w_{|T^w|+1} = t^{\mathbf{pa}(w)}_{|T^{\mathbf{pa}(w)}|+1} = t_{\max}$. Analogously, S^w and $S^{\mathbf{pa}(w)}$ denote the corresponding skeletons. Thus we divide the time interval $[0, t_{\max}]$ into disjoint segments $[t_j^{\mathbf{pa}(w)}, t_{j+1}^{\mathbf{pa}(w)})$, $j = 0, 1, \ldots |T^{\mathbf{pa}(w)}|$ such that $X_{\mathbf{pa}(w)}$ is constant and X_w is homogeneous in each segment. Next we define sets $I_j = \{i > 0 : \; t_j^{\mathbf{pa}(w)} < t_i^w < t_{j+1}^{\mathbf{pa}(w)}\}$ with notation j_{beg} and j_{end} for

the first and the last element of I_j. Then we obtain the following formula.

$$
\begin{aligned}
p(X_w \mid\mid X_{\mathbf{pa}(w)}) &= p(T^w, S^w \mid\mid S^{\mathbf{pa}(w)}, T^{\mathbf{pa}(w)}) = \\
&= \prod_{j=0}^{|T^{\mathbf{pa}(w)}|} \Bigg\{ \mathbb{I}(I_j \neq \emptyset) \Bigg[\prod_{i \in I_j} Q_w(s_j^{\mathbf{pa}(w)}; s_{i-1}^w, s_i^w) \times \\
&\times \prod_{i \in I_j \setminus \{j_{\text{beg}}\}} \exp\left(-(t_i^w - t_{i-1}^w) Q_w(s_j^{\mathbf{pa}(w)}; s_{i-1}^w) \right) \times \\
&\times \exp\left(-(t_{j_{\text{beg}}}^w - t_j^{\mathbf{pa}(w)}) Q_w(s_j^{\mathbf{pa}(w)}; s_{j_{\text{beg}}-1}^w) - (t_{j+1}^{\mathbf{pa}(w)} - t_{j_{\text{end}}}^w) Q_w(s_j^{\mathbf{pa}(w)}; s_{j_{\text{end}}}^w) \right) \Bigg] + \\
&+ \mathbb{I}(I_j = \emptyset) \exp\left(-(t_j^{\mathbf{pa}(w)} - t_{j+1}^{\mathbf{pa}(w)}) Q_w(s_j^{\mathbf{pa}(w)}; s_{j_{\text{beg}}-1}^w) \right) \Bigg\} .
\end{aligned}
$$

Below in Figure 2.5 there is an example of a trajectory of the node w with two possible states and of its parent with also two possible states 0 and 1. In this case the sets of indices are $I_0 = \{2, 3, 4\}$, $I_1 = \{\emptyset\}$ and $I_2 = \{7\}$.

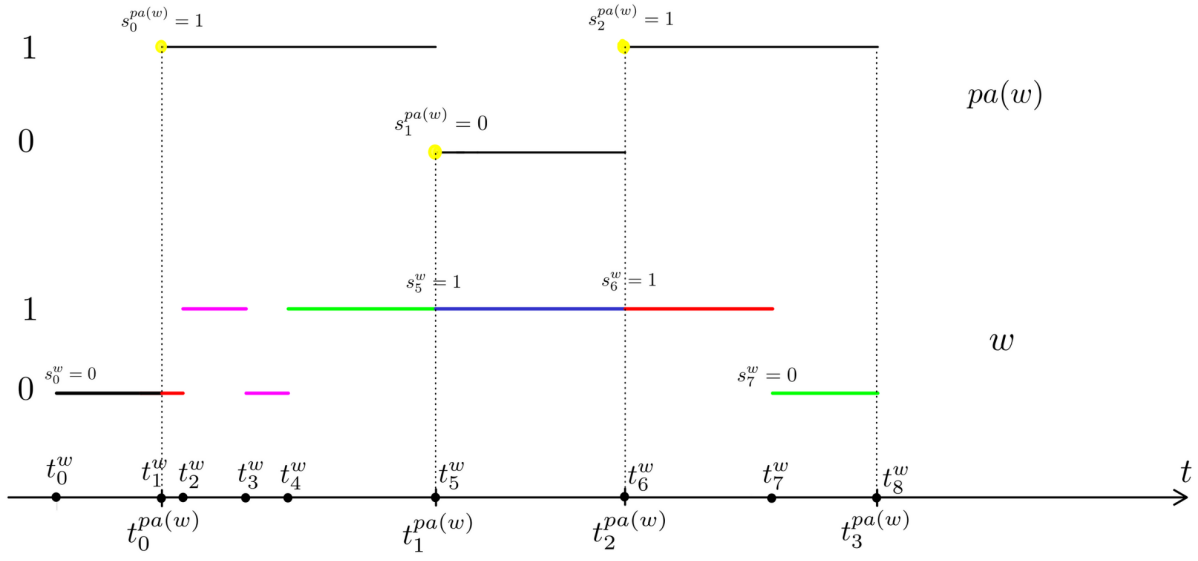


Figure 2.5: An exemplary trajectory of a node w and its parents $pa(w)$.

In consequence, using the fundamental property of the exponential function we may write $p(X_w \mid\mid X_{\mathbf{pa}(w)})$ in the form

$$
p(X_w \mid\mid X_{\mathbf{pa}(w)}) = \prod_{c \in \mathcal{X}_{\mathrm{pa}(w)}} \prod_{s \in \mathcal{X}_w} \prod_{\substack{s' \in \mathcal{X}_w \\ s' \neq s}} Q_w(c;\ s, s')^{n_w^T(c;\, s, s')} \exp\left[-Q_w(c;\ s, s') t_w^T(c;\ s) \right] , \tag{2.9}
$$

where

- $n_w^T(c; s, s')$ denotes the number of jumps from $s \in \mathcal{X}_w$ to $s' \in \mathcal{X}_w$ at the node w on the time interval $[0, T]$, which occur when the parent configuration is $c \in \mathcal{X}_{\mathbf{pa}(w)}$,

- $t_w^T(c; s)$ is the length of time that the node w is in the state $s \in \mathcal{X}_w$ on the time interval $[0, T]$, when the configuration of parents is $c \in \mathcal{X}_{\mathbf{pa}(w)}$.

To simplify the notation we omit the upper index T in $n_w^T(c; s, s')$ and $t_w^T(c; s)$ further in the thesis, except for the part where we consider martingales.

2.6 The LASSO penalty

In this section we shortly describe the notions of the LASSO penalty and LASSO estimators which constitute the base of the novel algorithms for structure learning in the thesis. LASSO is the acronym for Least Absolute Shrinkage and Squares Operator. The term was invented by Tibshirani (1996) though the general concept was introduced even earlier. Most of the contents of this section come from Hastie et al. (2015).

The underlying idea of the LASSO estimators is the assumption of *sparsity*. A sparse statistical model is one in which only a relatively small number of parameters (or predictors) play an important role. Consider a linear regression model with N observations y_i of a target variable and $x_i = (x_{i1}, \ldots, x_{ip})^\top$ of p associated predictor variables which are also called features. The goal is to predict the target from the predictors for future data and also to discover which predictors are *relevant*. In the linear regression model we assume that

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i,$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$ is the vector of unknown parameters and ϵ_i is an error term. The standard way to find β is to minimize the least-squares function

$$\sum_{i=1}^{N} \Big(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \Big)^2.$$

Typically all of the estimates appear to be non-zero, which complicates the interpretability of the model especially with a high number of possible predictors. Moreover, since the data have noise the model will try to fit the training observations too much and the parameters will most probably take extreme values. In case when $p > N$ the estimates are not even unique, so most of solutions will overfit the data.

The solution is to *regularize* the estimation process, i.e. add some constraints on the parameters. The LASSO estimator uses ℓ_1-penalty, which means that we minimize the least-square function with an additional bound on ℓ_1-norm of β, namely $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j| \leq t$. The value t is the user-specified parameter usually called hyperparameter. The motivation to use ℓ_1-penalty instead of any other ℓ_q-penalty comes from the fact that if t is small enough we obtain a sparse solution with only a small amount of non-zero parameters. This does not happen for ℓ_q-norm if $q > 1$, and if $q < 1$ the solutions are sparse but the problem is not convex. Convexity simplifies the computations as well as the theoretical analysis of the properties of the estimator. This allows for scalable algorithms capable of handling

problems with even millions of parameters. Before the optimization process we typically standardize the predictors so that each column is centred, i.e. the mean for each column is 0, and has unit variance, i.e. the mean of squares is equal to 1. We also centre the target column, so in the result we can omit the intercept term β_0 in the estimation process.

The LASSO penalty is used not only in linear regression but in a wide variety of models, for example generalized linear models where the target and the linear model are connected through some link function. Hence, in a more general case we can formulate the optimization problem as

$$\hat{\beta} = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} [\mathcal{L}(\theta, \mathcal{D}) + \lambda \|\theta\|_1],$$

where $\mathcal{L}(\theta, \mathcal{D})$ is the arbitrary loss function for the data $\mathcal{D}$ and the parameter vector θ. The tuning hyperparameter λ corresponds to the constraining value t, there is one-to-one correspondence. This is so-called Lagrangian form for the LASSO problem described above.

In the setting of structure learning for Bayesian networks, both static and continuous, we formulate the problem as an optimization problem for a linear or generalized linear model, where the parameter vectors encode the dependencies between variables in the network. We use the LASSO penalty in all the formulated problems, hence the problem of finding arrows in the graph reduces to recovering certain non-zero parameters in the LASSO estimator. As the loss functions we use the negative log-likelihood function and the residual sum of squares.

Chapter 3

Statistical inference for networks with known structure

There are three main classes of problems concerning Bayesian networks (both static and continuous time). The first one is to discover the structure of the network. Namely, we need to specify the underlying graph of the network, which nodes are the variables of interest, and its edges encode the dependencies between the variables. This problem will be covered in subsequent chapters.

The second problem is to learn the parameters of the network. Namely, knowing the structure of the network we need to specify the behaviour of the network in any specified node given the states of its parents. In the context of static BN this behaviour is encoded by conditional probability distributions (CPD, see (2.2)). The corresponding parameters in case of CTBNs are conditional intensity matrices (CIM, see (2.7)).

The third type of problems is to make statistical inference using the network with known structure and parameters. For instance, we may want to predict the state of some node of interest or, knowing states of some nodes, find which combination of the remaining nodes explains them the best. Finally, we may be interested in prediction of the future dynamics (in time) of some nodes of the network.

In this chapter we discuss well known results concerning the problems of learning the parameters of the network and then the inference based on the fully discovered network. The contents of this chapter are mainly based on Koller and Friedman (2009), Nodelman (2007) and Heckerman (2021) with more detailed references throughout it.

3.1 Learning probabilities in BNs

First we discuss the discrete case. We assume that the Bayesian network with the known underlying graph $\mathcal{G}$ includes n nodes each corresponding to a variable $X_i \in \mathbf{X}$ for $i = 1, \ldots, n$. Also, each variable X_i is discrete, having r_i possible values $x_i^1, x_i^2, \ldots, x_i^{r_i}$. We denote an observed value of X_i in l-th observation as $X_i[l]$. If each node is observed m times, then we obtain the sample dataset $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ with the sample $D_l = (X_1[l], X_2[l], \ldots, X_n[l])$ indicating the observed values of all the nodes in the l-th

sampling. We refer to each D_l as *a case*. If all cases are complete, i.e. no missing values occurred in the dataset $\mathcal{D}$, it is considered as *complete data*; otherwise, it is called *incomplete data*. Missing values in data can occur for many different reasons, for instance, people filling out a survey may prefer not to answer some questions or certain measurements might not be available for some patients in a medical setting.

There are mainly two categories of methods for parameter estimation in BN: one is for dealing with the complete data, and the other is for incomplete data. We will provide concise descriptions of two algorithms for the first category such as maximum likelihood estimation and Bayesian method; and we will briefly discuss algorithms for the second category.

Assume that as in (2.2) we can write the joint distribution of the variables in $\mathbf{X}$ as follows

$$\mathbb{P}(X_1, X_2, ..., X_n \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid \mathbf{pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}_i)$$

for some vector of parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$, where $\boldsymbol{\theta}_i$ is the vector of parameters for the local distribution $\mathbb{P}(X_i \mid \mathbf{pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}_i)$. For shortness, further in this chapter we will write $\mathbf{pa}(X_i)$ instead of $\mathbf{pa}_{\mathcal{G}}(X_i)$. In the case of discrete and completely observed data *categorical distribution* is commonly used. We note that in literature concerning learning Bayesian networks this type of distribution is often referred to as multinomial distribution or in some cases as unrestricted multinomial distribution (for example Heckerman (2021)) to differentiate this distribution from multinomial distributions that are low-dimensional functions of $\mathbf{pa}(X_i)$.

Hence we assume that each local distribution function is a collection of categorical distributions, one distribution for each configuration of its parents, namely

$$\mathbb{P}(X_i = x_i^k \mid \mathbf{pa}_i^j, \boldsymbol{\theta}_i) = \theta_{ijk} > 0, \text{ for } 1 \leq k \leq r_i, 1 \leq j \leq q_i, \tag{3.1}$$

where $q_i = \prod_{X_j \in \mathbf{pa}(X_i)} r_j$ and $\mathbf{pa}_i^1, \mathbf{pa}_i^2, \ldots, \mathbf{pa}_i^{q_i}$ denote all possible configurations of $\mathbf{pa}(X_i)$, and $\boldsymbol{\theta}_i = ((\theta_{ijk})_{k=2}^{r_i})_{j=1}^{q_i}$ are the parameters. Note that the parameter θ_{ij1} is given by the difference $1 - \sum_{k=2}^{r_i} \theta_{ijk}$. For convenience, let us denote the vector of parameters $\boldsymbol{\theta}_{ij} = (\theta_{ij2}, \theta_{ij3}, \ldots, \theta_{ijr_i})$ for all $1 \leq i \leq n$ and $1 \leq j \leq q_i$ so that $\boldsymbol{\theta}_i = (\boldsymbol{\theta}_{ij})_{j=1}^{q_i}$.

As it is well known, the maximum likelihood estimation (MLE) is a method of estimating the parameters of a probability distribution by maximizing the likelihood function, so that under the assumed statistical model the observed data is the most probable. Basically, if C_k is the result of a random test for an event C with several possible outcomes $C_1, C_2, \ldots, C_n$ it will appear in the maximum likelihood for this event. Hence, the estimated value of $\hat{C}$ will be set as parameter θ if it maximizes the value of the likelihood function $\mathbb{P}(C \mid \theta)$.

For the general Bayesian network with n nodes we denote the likelihood function as

$$\begin{aligned} L(\boldsymbol{\theta} : \mathcal{D}) = \mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{l=1}^{m} \mathbb{P}(D_l \mid \boldsymbol{\theta}) = \prod_{l=1}^{m} \mathbb{P}(X_1[l], X_2[l], \ldots, X_n[l] \mid \boldsymbol{\theta}) = \\ = \prod_{l=1}^{m} \prod_{i=1}^{n} \mathbb{P}(X_i[l] \mid \mathbf{pa}_i[l], \boldsymbol{\theta}_i) = \prod_{i=1}^{n} \prod_{l=1}^{m} \mathbb{P}(X_i[l] \mid \mathbf{pa}_i[l], \boldsymbol{\theta}_i) = \prod_i L_i(\boldsymbol{\theta}_i : \mathcal{D}), \end{aligned} \tag{3.2}$$

where by $\mathbf{pa}_i[l] = \mathbf{pa}(X_i)[l]$ we denote the l-th observation of the parents vector of the variable X_i. This representation shows that the likelihood decomposes as a product of independent factors, one for each CPD in the network. This important property is called *the global decomposition* of the likelihood function. Moreover, this decomposition is an immediate consequence of the network structure and does not depend on any particular choice of the parameterization for CPDs (see Koller and Friedman (2009)).

If the conditional distribution of X_i given its parents $\mathbf{pa}_{\mathcal{G}}(X_i)$ is the categorical distribution, then the local likelihood function can be further decomposed as follows

$$\begin{aligned} L_i(\boldsymbol{\theta}_i : \mathcal{D}) = \prod_{l=1}^{m} \mathbb{P}(X_i[l] \mid \mathbf{pa}_i[l], \boldsymbol{\theta}_i) = \prod_{l=1}^{m} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \mathbb{P}(X_i[l] = x_i^k \mid \mathbf{pa}_i[l] = \mathbf{pa}_i^j, \boldsymbol{\theta}_i) \\ = \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ilk}^{N(x_i^k, \mathbf{pa}_i^j)}, \end{aligned} \tag{3.3}$$

where $N(x_i^k, \mathbf{pa}_i^j)$ is the number of cases in $\mathcal{D}$ for which $X_i = x_i^k$ and $\mathbf{pa}(X_i) = \mathbf{pa}_i^j$.

Considering that the dataset is complete for each possible value $\mathbf{pa}_i^j$ of the parents $\mathbf{pa}(X_i)$ of the node X_i, the probability $\mathbb{P}(X_i \mid \mathbf{pa}_i^j)$ is the independent categorical distribution not related to any other configurations $\mathbf{pa}_i^l$ of $\mathbf{pa}(X_i)$ for $j \neq l$. Therefore, as the result of the MLE method we obtain the estimated parameter $\hat{\boldsymbol{\theta}}$ as follows

$$\hat{\theta}_{ijk} = \frac{N(x_i^k, \mathbf{pa}_i^j)}{N(\mathbf{pa}_i^j)},$$

where $N(\mathbf{pa}_i^j)$ denotes the number of cases when the configuration $\mathbf{pa}_i^j$ appears in the full set of observations for the vector of variables $\mathbf{pa}(X_i)$.

Note that in general the MLE approach attempts to find the parameter vector $\boldsymbol{\theta}$ that is "the best" given the data C. On the other hand, the Bayesian approach does not attempt to find such a point estimate. Instead, the underlying principle is that we should keep track of our beliefs about values of $\boldsymbol{\theta}$, and use these beliefs for reaching conclusions. In other words, we should quantify the subjective probability we have initially assigned to different values of $\boldsymbol{\theta}$ taking into account new evidence. Note that in representing such subjective probabilities we now treat $\boldsymbol{\theta}$ as a random variable. Thus, the Bayesian approach is based on the Bayes rule

$$p(\boldsymbol{\theta} \mid C) = \frac{p(C \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(C)}. \tag{3.4}$$

Hence, the basic idea of the Bayesian method for parameter learning is the following. We treat $\boldsymbol{\theta}$ as a random variable with a prior distribution $p(\boldsymbol{\theta})$, and it is very common

to set p as the uniform distribution, especially in the case when we have no prior knowledge about $\boldsymbol{\theta}$. Given a distribution with unknown parameters and a complete set of observed data C, new beliefs about $\boldsymbol{\theta}$, namely $p(\boldsymbol{\theta} \mid C)$, can be estimated according to the previous knowledge. The aim is to calculate $p(\boldsymbol{\theta} \mid C)$ which is called the posterior probability of the parameter $\boldsymbol{\theta}$. For computational efficiency we want to use a conjugate prior, i.e. when the posterior distribution after conditioning on the data is in the same parametric family as the prior one.

Here we assume that each vector $\boldsymbol{\theta}_{ij}$ has the prior Dirichlet distribution, so that

$$p(\boldsymbol{\theta}_{ij}) = Dir(\boldsymbol{\theta}_{ij} \mid \alpha_{ij1}, \ldots, \alpha_{ijr_i}) = \frac{\Gamma(\alpha_{ij})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1}, \tag{3.5}$$

where $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, $\alpha_{ijk} > 0$, $k = 1, \ldots, r_i$, $\alpha_{ij1}, \ldots, \alpha_{ijr_i}$ are hyperparameters and $\Gamma(\cdot)$ is Gamma function. This is the standard conjugate prior to both categorical and multinomial distributions. Hence, the probability of observed samples is

$$\begin{aligned} p(\mathcal{D}) &= \int p(\boldsymbol{\theta}_{ij}) p(\mathcal{D} \mid \boldsymbol{\theta}_{ij}) d\boldsymbol{\theta}_{ij} = \\ &= \int \frac{\Gamma(\alpha_{ij})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1} \times \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} d\boldsymbol{\theta}_{ij} = \\ &= \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}, \end{aligned} \tag{3.6}$$

where for shortness $N_{ijk} = N(x_i^k, \mathbf{pa}_i^j)$ and $N_{ij} = N(\mathbf{pa}_i^j) = \sum_{k=1}^{r_i} N_{ijk}$. The integral is $(r_i - 1)$-dimensional over the set $\{\theta_{ijk} \geq 0, \quad 2 \leq k \leq r_i, \quad \sum_{k=2}^{r_i} \theta_{ijk} \leq 1\}$.

As we have already mentioned, in Bayesian method, if we do not have prior distribution we assume it to be uniform, which is consistent with the principle of maximum entropy in information theory, it maximizes the entropy of random variables with bounded support. Thus, if there is no information used for determination of prior distribution, we set hyperparameters $\alpha_1 = \cdots = \alpha_r = 1$.

Combining (3.4), (3.5) and (3.6) under the assumptions of parameter independence and complete data finally we obtain the posterior distribution as follows

$$p(\boldsymbol{\theta}_{ij} \mid \mathcal{D}) = Dir(\boldsymbol{\theta}_{ij} \mid \alpha_{ij1} + N_{ij1}, \ldots, \alpha_{ijr_i} + N_{ijr_i}). \tag{3.7}$$

Therefore, we have an estimate for each parameter θ_{ijk} from data $\mathcal{D}$ as follows

$$\hat{\theta}_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}, \quad 1 \leq k \leq r_i.$$

Continuous Variable Networks. When we were discussing the MLE method for discrete BNs, we mentioned the global decomposition rule which applies to any type of CPD. That is, if the data are complete, the learning problem reduces to a set of local learning problems, one for each variable. The main difference is in applying the maximum likelihood estimation process to CPD of a different type: how we define the sufficient

statistics, and how we compute the maximum likelihood estimate from them. In this paragraph, we briefly discuss how MLE principles can be applied in the setting of linear Gaussian Bayesian networks.

Consider a variable X with parents $\mathbf{U} = \{U_1, \ldots, U_k\}$ with linear Gaussian CPD:

$$p(X \mid \mathbf{u}) = \mathcal{N}(\beta_0 + \beta_1 u_1 + \cdots + \beta_k u_k, \sigma^2).$$

Our task is to learn the parameters $\hat{\boldsymbol{\theta}}_{X|\mathbf{U}} = (\beta_0, \beta_1, \ldots, \beta_k, \sigma^2)$. To find the MLE values of these parameters, we need to differentiate the likelihood function and to solve the equations that define a stationary point. As usual, it is easier to work with the log-likelihood function. Using the definition of the Gaussian distribution, we have that

$$\begin{aligned}\ell(\boldsymbol{\theta}_{X|\mathbf{U}} : \mathcal{D}) &= \log L_X(\boldsymbol{\theta}_{X|\mathbf{U}} : \mathcal{D}) = \\ &= \sum_l \left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\beta_0 + \beta_1 u_1[l] + \cdots + \beta_k u_k[l] - x[l])^2 \right].\end{aligned}$$

We consider the gradients of the log-likelihood with respect to all of the parameters $\beta_0, \ldots, \beta_k$ and σ^2 and as a result we get a number of equations, which describe the solution to a system of linear equations. From the Theorem 7.3 in Koller and Friedman (2009), it follows that if $\mathcal{B}$ is a linear Gaussian Bayesian network, then it defines a joint distribution that is jointly Gaussian, and the MLE estimate has to match the constraints implied by it.

Briefly speaking, to estimate $p(X \mid \mathbf{U})$ we estimate the means of X and $\mathbf{U}$ and the covariance matrix of $\{X\} \cup \mathbf{U}$ from the data. The vector of means and the covariance matrix define the joint Gaussian distribution over $\{X\} \cup \mathbf{U}$. Then, for example using the formulas provided by Theorem 7.3 in Koller and Friedman (2009), we find the unique linear Gaussian that matches the joint Gaussian with these parameters.

The sufficient statistics we need to collect to estimate linear Gaussians are the univariate terms of the form $\sum_m x[m]$ and $\sum_m u_i[m]$, and the interaction terms of the form $\sum_m x[m] \cdot u_i[m]$ and $\sum_m u_j[m] \cdot u_i[m]$. From these we can estimate the mean and the covariance matrix of the joint distribution.

3.2 Inference in Bayesian networks

In this section we assume that the network structure is known, meaning we know all the existing edges and their directions as well as all the CPDs. The problem of inference for BNs is a challenging task on its own and there is a lot of research done on the subject. We will not go into much of a detail on the inference since our focus is on learning their structure. However, the question of inference is worth mentioning here in order to get a wholesome picture of such a powerful tool as BNs.

First, we discuss what the notion of inference means in the case of BNs. Typically it refers to:

- marginal inference, i.e. finding *the probability of a variable* being in a certain state, *given* that *other variables* are set to certain values; or

- maximum a posteriori (MAP) inference, i.e. finding *the values of a given set of variables that best explain* (in the sense of the highest MAP probability) why a set of other variables have certain values,

Let us demonstrate both categories of questions using an example. We will use the BN structure of a well-known ASIA network (see Figure 3.1) first introduced in Lauritzen and Spiegelhalter (1988). It illustrates the causal structure of a patient having a certain lung disease based on several factors, one being whether or not the patient has recently been to Asia. In this case, an exemplary question on marginal inference might be what is the probability of a patient who is a smoker and has dyspnoea having a certain lung disease, e.g. lung cancer. For the MAP inference, we might want to know what is the most likely set of conditions (with "smoking" and "dyspnoea" excluded) that could have caused the symptoms mentioned above.

Now we provide short descriptions of the most popular exact and approximate inference algorithms for BNs. Among them are variable elimination and belief propagation for the marginal inference, methods for the MAP inference and the sampling-based inference. For the purposes of transparency of the presentation the inference methods for BNs will be demonstrated for the discrete and finite case.

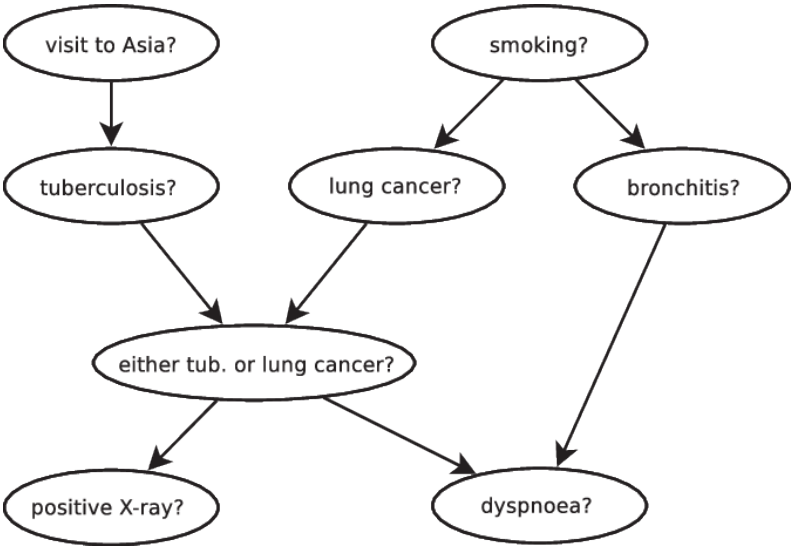


Figure 3.1: The ASIA Bayesian network structure

3.2.1 Variable Elimination

This inference algorithm is defined in terms of so-called factors and is developed to answer questions of marginal inference. Factors generalize the notion of CPDs. *A factor* ϕ is a function of value assignments of a set of random variables $\mathbf{V}$ with positive real values. The set of variables $\mathbf{V}$ is called *the scope* of the factor. There are two operations on factors that are repeatedly performed in a variable elimination algorithm (VE) and hence are of great importance.

- *The factor product.* If $\mathbf{V}_1, \mathbf{V}_2$, and $\mathbf{V}_3$ are disjoint sets of variables and we have factors ϕ_1 and ϕ_2 with scopes $\mathbf{V}_1 \cup \mathbf{V}_2$ and $\mathbf{V}_2 \cup \mathbf{V}_3$ respectively, then we define the factor product $\phi_1 \cdot \phi_2$ as a new factor ψ with the scope $\mathbf{V}_1 \cup \mathbf{V}_2 \cup \mathbf{V}_3$ by

$$\psi(\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3) = \phi_1(\mathbf{V}_1, \mathbf{V}_2) \cdot \phi_2(\mathbf{V}_2, \mathbf{V}_3).$$

  This product is the new factor over the union of the variables defined for each instantiation by multiplying the value of ϕ_1 on the particular instantiation by the value of ϕ_2 on the corresponding instantiation. More precisely,

$$\psi(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3) = \phi_1(\boldsymbol{v}_1, \boldsymbol{v}_2) \cdot \phi_2(\boldsymbol{v}_2, \boldsymbol{v}_3)$$

  for each instantiation, where $\boldsymbol{v}_1 \in \mathit{Val}(\mathbf{V}_1), \boldsymbol{v}_2 \in \mathit{Val}(\mathbf{V}_2)$ and $\boldsymbol{v}_3 \in \mathit{Val}(\mathbf{V}_3)$.

- *The factor marginalization.* This operation "locally" eliminates a set of variables from a factor. If we have a factor $\phi(\mathbf{V}_1, \mathbf{V}_2)$ over two sets of variables $\mathbf{V}_1, \mathbf{V}_2$, marginalizing $\mathbf{V}_2$ produces a new factor

$$\tau(\mathbf{V}_1) = \sum_{\mathbf{V}_2} \phi(\mathbf{V}_1, \mathbf{V}_2),$$

  where the sum is over all joint assignments for the set of variables $\mathbf{V}_2$. More precisely,

$$\tau(\boldsymbol{v}_1) = \sum_{\boldsymbol{v}_2 \in \mathit{Val}(\mathbf{V}_2)} \phi(\boldsymbol{v}_1, \boldsymbol{v}_2), \boldsymbol{v}_1 \in \mathit{Val}(\mathbf{V}_1)$$

  for each instantiation $\boldsymbol{v}_1 \in \mathit{Val}(\mathbf{V}_1)$.

Thus, in the context of factors we can write our distribution over all variables as a product of factors, where each factor presents a CPD as in (2.2):

$$\mathbb{P}(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} \phi_i(\mathbf{A}_i), \tag{3.8}$$

where $\mathbf{A}_i = (X_i, \mathbf{pa}_{\mathcal{G}}(X_i))$ represents a set of variables including the i-th variable and its parents in the network.

Now we can describe the full VE algorithm. Assume we want to find marginal distribution of a fixed variable from $X_1, \ldots, X_n$. First we need to choose in which order O to eliminate remaining variables. The choice of an optimal *elimination ordering* O is an $\mathcal{NP}$-hard problem and it may dramatically affect the running time of the variable elimination algorithm. Some intuitions and techniques on how to choose an adequate ordering are given for example in Koller and Friedman (2009). For each variable X_i (ordered according to the ordering O) we perform the following steps:

- multiply all factors containing X_i (on the first round all the ϕ_i containing X_i);
- marginalize out X_i according to the definition of the factor marginalization to obtain a new factor τ (which does not necessarily correspond to a probability distribution, even though each ϕ is CPD);

- replace the factors used in the first step with τ.

Essentially, we loop over the variables as ordered by O and eliminate them in this order. Performing those steps we use simple properties of product and summation on factors, namely, both operations are commutative and products are associative. The most important rule is that we can exchange summation and product, meaning that if a set of variables $\mathbf{X}$ is not in the scope of the factor ϕ_1, then

$$\sum_{\mathbf{X}} \phi_1 \cdot \phi_2 = \phi_1 \cdot \sum_{\mathbf{X}} \phi_2. \tag{3.9}$$

So far we saw that the VE algorithm can answer queries of the form $\mathbb{P}(\mathbf{V})$, where $\mathbf{V}$ is some subset of variables. However, in addition to this type of questions it can answer marginal queries of the form

$$\mathbb{P}(\mathbf{Y} \mid \mathbf{E} = \boldsymbol{e}) = \frac{\mathbb{P}(\mathbf{Y}, \mathbf{E} = \boldsymbol{e})}{\mathbb{P}(\mathbf{E} = \boldsymbol{e})},$$

where $\mathbb{P}(\mathbf{X}, \mathbf{Y}, \mathbf{E})$ is a probability distribution over sets of query variables $\mathbf{Y}$, observed evidence variables $\mathbf{E}$, and unobserved variables $\mathbf{X}$. We can compute this probability by performing variable elimination once on $\mathbb{P}(\mathbf{Y}, \mathbf{E} = \boldsymbol{e})$ and then once again on $\mathbb{P}(\mathbf{E} = \boldsymbol{e})$ taking into account only instantiations consistent with $\mathbf{E} = \boldsymbol{e}$.

An exemplary run of the VE algorithm is presented in Table 3.1. It corresponds to Extended Student example first mentioned in Section 2.2.

Step	Variables eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C), \phi_D(D, C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G, I, D), \tau_1(D)$	G, I, D	$\tau_2(G, I)$
3	I	$\phi_I(I), \phi_S(S, I), \tau_2(G, I)$	G, S, I	$\tau_3(G, S)$
4	H	$\phi_H(H, G, J)$	H, G, J	$\tau_4(G, J)$
5	G	$\tau_3(G, S), \tau_4(G, J), \phi_L(L, G)$	G, J, L, S	$\tau_5(J, L, S)$
6	S	$\tau_5(J, L, S), \phi_J(J, L, S)$	J, L, S	$\tau_6(J, L)$
7	L	$\tau_6(J, L)$	J, L	$\tau_7(J)$

Table 3.1: A run of variable elimination for the query $P(J)$.

3.2.2 Message Passing Algorithms

Markov random fields. In the framework of probabilistic graphical models there exists another technique for compact representation and visualization of a probability distribution which is formulated in the language of undirected graphs. This class of models (known as Markov Random Fields or MRFs) can succinctly represent independence assumptions that directed models cannot represent and the opposite is also true. There are advantages and drawbacks to both of those methods but that is not the focus of this thesis.

We will introduce and discuss MRFs only to the extent we need to properly describe and explain notions and methods concerning BNs. Note that the methods provided below for marginal and MAP inference are applicable both to MRFs and BNs.

Definition 3.1. *A Markov Random Field (MRF) is a probability distribution over variables $X_1, \ldots, X_n$ defined by an undirected graph $\mathcal{G}$ in which nodes correspond to variables X_i. The probability has the form*

$$\mathbb{P}(X_1, X_2, ..., X_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(X_c),$$

where C denotes the set of cliques (i.e. fully connected subgraphs) of $\mathcal{G}$ and each factor ϕ_c is a non-negative function over the variables in a clique. The partition function

$$Z = \sum_{(x_1,\ldots,x_n)} \prod_{c \in C} \phi_c(X_c)$$

is a normalizing constant that ensures that the distribution sums to one, where the summation is taken over all possible instantiations of all the variables.

Thus, given a graph $\mathcal{G}$, our probability distribution may contain factors whose scope is any clique in $\mathcal{G}$ and the clique can be a single node, an edge, a triangle, etc. Note that we do not need to specify a factor for each clique.

It is not hard to see that Bayesian networks are a special case of MRFs with a normalizing constant equal to 1 where the clique factors correspond to CPDs. One can notice that if we take a directed graph $\mathcal{G}$, add side edges to all parents of a given node and remove their directionality, then the CPDs (seen as factors over each variable and its ancestors) factorize over the resulting undirected graph. The resulting process is called *moralization* (see Figure 3.2). A Bayesian network can always be converted into an undirected network with normalizing constant 1.

Figure 3.2: Moralization of a Bayesian network.

Message passing. As we mentioned above, the VE algorithm can answer marginal queries of the form $\mathbb{P}(\mathbf{Y} \mid \mathbf{E} = \boldsymbol{e})$. However, if we want to ask the model for another query, e.g. $\mathbb{P}(\mathbf{Y}_2 \mid \mathbf{E}_2 = \boldsymbol{e}_2)$, we need to restart the algorithm from scratch. Fortunately, in the process of computing marginals, VE algorithm produces many intermediate factors τ as a side-product of the main computation, which turn out to be the same as the ones that we need to answer other marginal queries.

Many complicated inference problems can be solved by message-passing algorithms, in which simple messages are passed locally among simple elements of the system. An illustrative example was shown in the book MacKay (2003) for a problem of counting soldiers. Consider a line of soldiers walking in the mist. The commander, which is in the line, wishes to count the soldiers. The straightforward calculation is impossible because of the mist. However, it can be done in a simple way which does not require any complex operations. The algorithm requires the soldiers' ability to add two integer numbers and add 1 to it. The algorithm consists of the following steps (for example see Figure 3.3):

- the front soldier in the line says the number 'one' to the soldier behind him,
- the rearmost soldier in the line says the number 'one' to the soldier in front of him,
- the soldier, which is told a number from the soldier ahead or the soldier behind, adds 1 to it and passes the new number to the next soldier in the line on the other side.

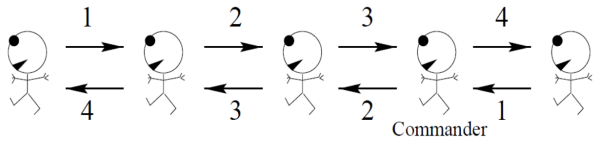
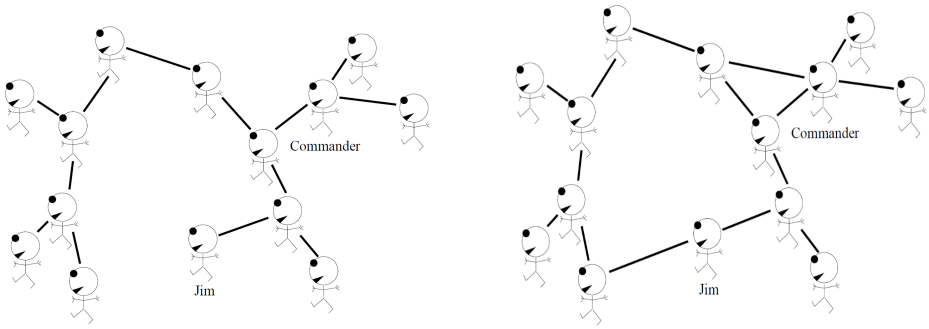


Figure 3.3: A line of soldiers counting themselves using message-passing rule-set.

Hence, the commander can find the global number of soldiers by simply adding together the numbers: heard from the soldier in front of him, from the soldier behind him and 1. This method makes use of a property of the total number of soldiers: the number can be written as the sum of the number of soldiers in front of a point and the number behind that point, two quantities which can be computed separately, because the two groups are separated by the commander. When this requirement is satisfied this message-passing algorithm can be modified for a general graph with no cycles (as an example see Figure 3.4a). When the graph has no cycles (see Figure 3.4a) for each soldier we can uniquely separate the group into two groups, 'those in front', and 'those behind' and perform the algorithm above. However, it is not always possible for a graph with cycles, for instance for a soldier in a cycle (such as 'Jim') in Figure 3.4b such a separation is not unique.

Using the same principle we will now describe the message passing for tree-structured networks (called *belief propagation*, BP for short) and then the modification of the method for general networks (called *clique tree algorithm*).

Belief propagation. Let us first look at tree-structured graphs. Consider what happens if we run the VE algorithm on a tree in order to compute a marginal distribution $\mathbb{P}(X_i)$.



(a) No cycles.

(b) Contains a cycle.

Figure 3.4: A swarm of soldiers.

We can easily find the optimal ordering for this problem by rooting the tree at the node associated with X_i and iterating through the nodes in post-order (from leaves to the root), just like for a swarm of soldiers with no cycles. At each step, we will eliminate one of the variables, say X_j; this will involve computing the factor $\tau_k(x_k) = \sum_{x_j} \phi(x_k, x_j)\tau_j(x_j)$, where X_k is the parent of X_j in the tree. At a later step, the variable X_k will be eliminated in the same manner, i.e. $\tau_k(x_k)$ will be passed up the tree to the parent X_l of X_k in order to be multiplied by the factor $\phi(x_l, x_k)$ before being marginalized out. As a result we obtain the new factor $\tau_l(x_l)$. The factor $\tau_j(x_j)$ can be thought of as a message that X_j sends to X_k that summarizes all of the information from the subtree rooted at the node X_j. We can visualize this transfer of information using arrows on the tree, see Figure 3.5. At the end of the VE algorithm, the node X_i receives messages from all of its children and the final marginal $\mathbb{P}(X_i)$ is obtained by marginalizing those messages out.

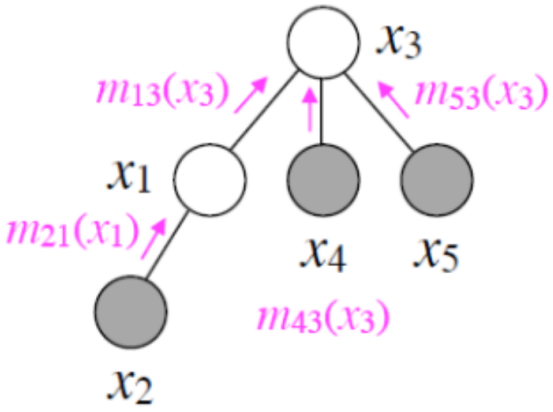


Figure 3.5: Message passing order when using VE to compute $\mathbb{P}(X_3)$ on a small tree.

With the same indices as above, suppose that after computing $\mathbb{P}(X_i)$ we want to compute $\mathbb{P}(X_k)$ as well. We would again run VE for the new tree rooted at the node X_k, waiting until it receives all messages from its children. Note that the new tree consists

of two parts. The first one is the subtree rooted at X_k with all its descendants from the original tree (i.e. rooted at X_i). The other part is the subtree rooted at X_l (which was the parent of X_k in the original tree, but now is the child of X_k). Therefore, this part contains the node X_i. The key insight is that the messages received by X_k from X_j now will be the same as those received when X_i was the root. Thus, if we store the intermediary messages of the VE algorithm, we can quickly compute other marginals as well. Notice for example, that the messages sent to X_k from the subtree containing X_i will need to be recomputed. So, how do we compute all the messages we need? Again, referring to the soldier counting problem, a node is *ready to transmit* a message to its parent after it has received all the messages from all of its children. All the messages will be sent out after precisely $2|\mathcal{E}|$ steps, where $|\mathcal{E}|$ is the number of edges in the graph, since each edge can receive messages only twice.

To define belief propagation (BP) algorithm formally let us see what kind of messages can be sent. For the purposes of marginal inference we will use *sum-product message passing*. This algorithm is defined as follows: while there is a node X_k ready to transmit to X_l it sends the message

$$m_{k\to l}(x_l) = \sum_{x_k} \phi(x_k)\phi(x_k, x_l) \prod_{j\in Nb(k)\setminus\{l\}} m_{j\to k}(x_k),$$

where $Nb(k) \setminus \{l\}$ means all the neighbours of the k-th node, excluding l-th node. Note that this message is precisely the factor τ that X_k would transmit to X_l during a round of variable elimination with the goal of computing $\mathbb{P}(X_i)$, and also note that the product on the RHS of this equation naturally equals to 1 for leaves in the tree.

After having computed all messages, we may answer marginal queries over any variable X_j in constant time using the equation:

$$\mathbb{P}(X_j) \propto \psi(X_j) \prod_{l\in Nb(j)} m_{l\to j}(x_j),$$

where $\psi(X_j)$ is a product of all factors ϕ whose scope contains X_j. In case of BNs we have the equality instead of proportionality.

Clique Tree Algorithm. First let us define what is meant by a clique tree. Clique tree is an undirected tree such that its nodes are clusters $\mathbf{C}_i$ of variables, meaning $\mathbf{C}_i$ is a subset of a set of all variables $\{X_1, \ldots, X_n\}$. Each edge between clusters $\mathbf{C}_i$ and $\mathbf{C}_j$ is associated with a *sepset* (separation set) $\mathbf{S}_{i,j} = \mathbf{C}_i \cap \mathbf{C}_j$. See a simple example demonstrating a clique tree for a chain network in Figure 3.6.

So far we assumed that the graph is a tree. What if that is not the case? Then the clique tree algorithm (also called the junction tree algorithm in the literature) can be used; it partitions the graph into clusters of variables so that interactions among clusters will have a tree structure, i.e. a cluster will be only directly influenced by its neighbours in the tree, we denote it $\mathcal{T}$. Then we can perform message passing on this tree. This leads to tractable global solutions if the local (cluster-level) problems can be solved exactly.

In addition, clique trees must satisfy two following properties:

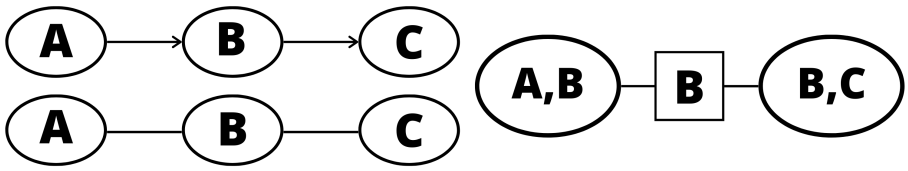


Figure 3.6: An example of a chain network consisting of three variables A, B and C; corresponding MRF and a clique tree with $\mathbf{C}_1 = \{A, B\}$, $\mathbf{C}_2 = \{B, C\}$ and $\mathbf{S}_{1,2} = \{B\}$.

1. *family preservation*, i.e. for each factor ϕ there is a cluster such that factor's scope is a subset of the cluster;

2. *running intersection property (RIP)*, i.e. for each pair of clusters $\mathbf{C}_i$, $\mathbf{C}_j$ and a variable $X \in \mathbf{C}_i \cap \mathbf{C}_j$ all clusters and sepsets on the unique path between $\mathbf{C}_i$ and $\mathbf{C}_j$ contain the variable X.

Note that we may always find a trivial clique tree with one node containing all the variables in the original graph, but obviously such trees are useless. Optimal trees are the ones that make the clusters as small and modular as possible; unfortunately, as in case of VE, the problem of finding the optimal tree is also $\mathcal{NP}$-hard. A special case when we can find it is when we originally have a tree, in this case we can put each connected pair of nodes into a separate cluster, it is easy to check that both conditions are met. One of the practical ways to find a good clique tree is to use a simulation of VE, i.e. the elimination order fixed for VE will induce the graph from which we will take maximal cliques and set them as our clusters and form a tree. RIP will be satisfied automatically. Note that we do not need to run VE, just to simulate it for a chosen ordering and get the induced graph. More formally:

Definition 3.2. *Let Φ be a set of factors (CPDs in the case of Bayesian Networks) over $\mathcal{X} = \{X_1, \ldots, X_n\}$, and $\prec$ be an elimination ordering for some subset $\mathbf{X} \subseteq \mathcal{X}$. The induced graph denoted by $I_{\Phi,\prec}$ is an undirected graph over $\mathcal{X}$, where X_i and X_j are connected by an edge if they both appear in some intermediate factor ψ generated by the VE algorithm using $\prec$ as an elimination ordering.*

In Figure 3.7 there is an example of an induced graph for the Student example using the elimination ordering of Table 3.1, cliques in that graph and a corresponding clique tree. One can see that RIP is satisfied, for a proof that trees corresponding to induced graphs by VE will satisfy RIP see Koller and Friedman (2009).

Now let us define the full clique tree algorithm. First, we define the potential $\psi_i(\mathbf{C}_i)$ of each cluster $\mathbf{C}_i$ as the product of all the factors ϕ in $\mathcal{G}$ that have been assigned to $\mathbf{C}_i$.

By the family preservation property, this is well-defined, and we may assume that our distribution is of the form

$$\mathbb{P}(X_1, \ldots, X_n) = \frac{1}{Z} \prod_i \psi_i(\mathbf{C}_i).$$

Then, at each step of the algorithm, we choose a pair of adjacent clusters $\mathbf{C}_i$, $\mathbf{C}_j$ in a tree graph $\mathcal{T}$ and compute a message whose scope is the sepset $\mathbf{S}_{i,j}$ between the two clusters

$$m_{i \to j}(\mathbf{S}_{i,j}) = \sum_{\mathbf{C}_i \setminus \mathbf{S}_{i,j}} \psi_i(\mathbf{C}_i) \prod_{l \in Nb(i) \setminus \{j\}} m_{l \to i}(\mathbf{S}_{l,i}). \tag{3.10}$$

In the context of clusters, $Nb(i)$ denotes the set of indices of neighboring clusters of $\mathbf{C}_i$. We choose $\mathbf{C}_i$ and $\mathbf{C}_j$ only if $\mathbf{C}_i$ has received messages from all of its neighbors except $\mathbf{C}_j$. Just as in belief propagation, this procedure will terminate in exactly $2|\mathcal{E}_\mathcal{T}|$ steps because this process is equivalent to making an *upward* pass and a *downward* pass. In the upward pass, we first pick a root and send all messages towards it starting from leaves. When this process is complete, the root has all the messages. Therefore, it can now send the appropriate message to all of its children. This algorithm continues until the leaves of the tree are reached, at which point no more messages need to be sent. This second phase is called the downward pass. After it terminates, we will define *the belief* of each cluster based on all the messages that it receives

$$\beta_i(\mathbf{C}_i) = \psi_i(\mathbf{C}_i) \prod_{l \in Nb(i)} m_{l \to i}(\mathbf{S}_{l,i}). \tag{3.11}$$

These updates are often referred to as *Shafer-Shenoy* updates and the full procedure is also referred as *sum-product belief propagation*. Then each belief is the marginal of the clique

$$\beta_i(\mathbf{C}_i) = \sum_{\mathcal{X} \setminus \mathbf{C}_i} \mathbb{P}(X_1, \ldots, X_n).$$

Now if we need to compute the marginal probability of a particular variable X we can select any clique whose scope contains X, and eliminate the redundant variables in the clique. A key point is that the result of this process does not depend on the clique we selected. That is, if X appears in two cliques, they must agree on its marginal. Two adjacent cliques $\mathbf{C}_i$ and $\mathbf{C}_j$ are said to be *calibrated* if

$$\sum_{\mathbf{C}_i \setminus \mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i) = \sum_{\mathbf{C}_j \setminus \mathbf{S}_{i,j}} \beta_j(\mathbf{C}_j).$$

A clique tree $\mathcal{T}$ is calibrated if all pairs of adjacent cliques are calibrated. For a calibrated clique tree, we use the term clique beliefs for $\beta_i(\mathbf{C}_i)$ and sepset beliefs for $\mu_{i,j}(\mathbf{S}_{i,j})$ defined as either side of the above equality.

As the end result of sum-product belief propagation procedure we get a calibrated tree, which is more than simply a data structure that stores the results of probabilistic inference for all of the cliques in the tree, i.e. their beliefs (3.11). It can also be viewed

as an alternative representation of the joint measure over all variables. For sepset beliefs we have that

$$\mu_{i,j}(\mathbf{S}_{i,j}) = m_{i\to j}(\mathbf{S}_{i,j})m_{j\to i}(\mathbf{S}_{i,j}).$$

Using this fact at convergence of the clique tree calibration algorithm, we get the unnormalized joint measure $\tilde{P}$ as

$$\tilde{P}(X_1,\ldots,X_n) = \prod_i \psi_i(\mathbf{C}_i) = \frac{\prod_i \beta_i(\mathbf{C}_i)}{\prod_{(i,j)} \mu_{i,j}(\mathbf{S}_{i,j})}, \tag{3.12}$$

where the product in the numerator is over all cliques and the product in the denominator is over all sepsets in the tree. As a result we get a different set of parameters that captures unnormalized measure that defined our distribution (in case of BNs it is simply the distribution) and there is no information lost in the process. Thus, we can view the clique tree as an alternative representation of the joint measure, one that directly reveals the clique marginals.

The second approach, mathematically equivalent but using a different intuition, is message passing with division. In sum-product belief propagation messages were passed between two cliques only after one had received messages from all of its neighbors except the other one as in (3.10) and the resulting belief was (3.11). Nonetheless, a different approach to compute the same expression is to multiply in all of the messages, and then divide the resulting factor by the message from the other clique to avoid double-counting. To make this notion precise, we must define a factor-division operation.

Let $\mathbf{X}$ and $\mathbf{Y}$ be disjoint sets of variables and let ϕ_1 and ϕ_2 be two factors with scopes $\mathbf{X}\cup\mathbf{Y}$ and $\mathbf{Y}$ respectively. Then we define the division $\frac{\phi_1}{\phi_2}$ as a factor-division ψ with the scope $\mathbf{X}\cup\mathbf{Y}$ as follows

$$\psi(\mathbf{X},\mathbf{Y}) = \frac{\phi_1(\mathbf{X},\mathbf{Y})}{\phi_2(\mathbf{Y})},$$

where we define $\frac{0}{0} = 0$. We now see that we can compute the expression of equation (3.10) by computing the beliefs as in equation (3.11) and then dividing by the remaining message

$$m_{i\to j}(\mathbf{S}_{i,j}) = \frac{\sum_{\mathbf{C}_i\setminus\mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i)}{m_{j\to i}(\mathbf{S}_{i,j})}.$$

The belief of the j-th clique is updated by multiplying its previous belief by $m_{i\to j}$ and dividing it by the previous message passed along this edge (regardless of the direction) stored in sepset belief $\mu_{i,j}$ to avoid double counting. This algorithm is called *belief update* message passing and is also known as the *Lauritzen-Spiegelhalter algorithm.*

3.2.3 MAP inference

The maximum a posteriori (MAP) problem has a broad range of applications, in computer vision, computational biology, speech recognition, and more. By using MAP inference we lose the ability to measure our confidence (or uncertainty) in our conclusions. Nevertheless, there are good reasons for using a single MAP assignment rather than using

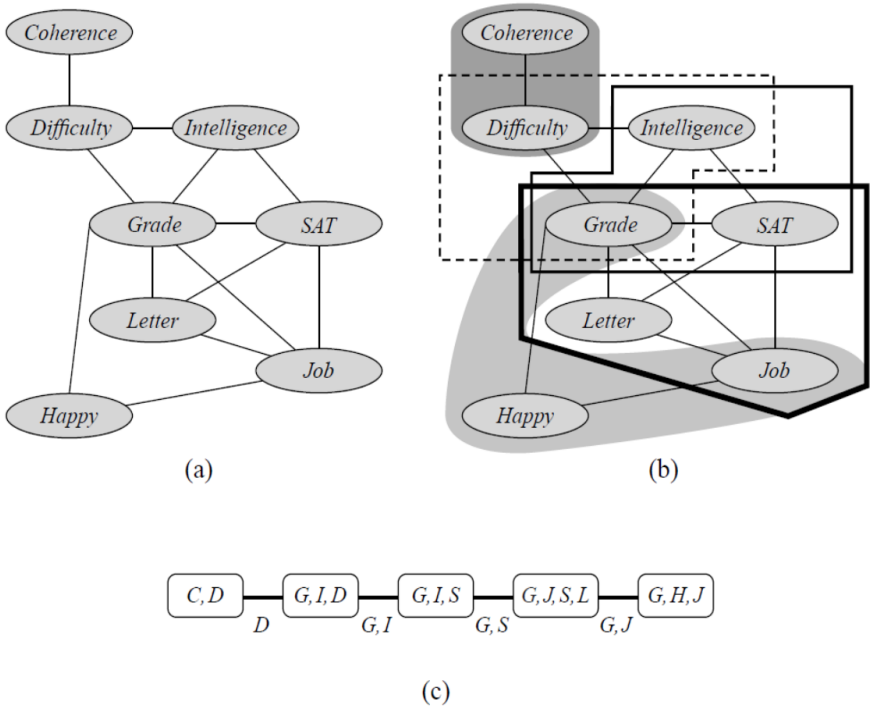


Figure 3.7: (a) Induced graph for VE in the Student example, using the elimination order of Table 3.1 (b) Cliques in the induced graph:$\{C, D\}$, $\{D, I, G\}$, $\{G, I, S\}$, $\{G, J, S, L\}$ and $\{G, H, J\}$. (c) Clique tree for the induced graph.

the marginal probabilities of the different variables. The first reason is the preference for obtaining a single coherent joint assignment, whereas a set of individual marginals may not make sense as a whole. The second is that there are inference methods that are applicable to the MAP problem and not to the task of computing probabilities, so that the former may be tractable even when the latter is not. The problem of finding the MAP assignment in the general case is $\mathcal{NP}-$hard (Cooper (1990)).

There are two types of Maximum a Posteriori (MAP) inference: a MAP query and a marginal MAP query. Assume first that the set of all variables $\mathbf{X} = \mathbf{Y} \cup \mathbf{E}$ consists of two disjoint sets, where $\mathbf{E}$ is the evidence meaning that we know values of those variables. Then a MAP query aims to find the most likely assignment to all of the non-evidence variables $\mathbf{Y}$

$$MAP(\mathbf{Y} \mid \mathbf{E} = \boldsymbol{e}) = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, \mathbb{P}(\mathbf{Y} = \boldsymbol{y} \mid \mathbf{E} = \boldsymbol{e}).$$

Now assume that the set of all variables $\mathbf{X} = \mathbf{Y} \cup \mathbf{W} \cup \mathbf{E}$ consists of three disjoint sets, where $\mathbf{E}$ is still the evidence. In this case a marginal MAP query aims to find the most likely assignment to the subset $\mathbf{Y}$, marginalizing over the rest of the variables $\mathbf{W}$

$$\begin{aligned} MAP(\mathbf{Y} \mid \mathbf{E} = \boldsymbol{e}) &= \underset{\boldsymbol{y}}{\operatorname{argmax}}\, \mathbb{P}(\mathbf{Y} = \boldsymbol{y} \mid \mathbf{E} = \boldsymbol{e}) = \\ &= \underset{\boldsymbol{y}}{\operatorname{argmax}} \sum_{\boldsymbol{w}} \mathbb{P}(\mathbf{Y} = \boldsymbol{y}, \mathbf{W} = \boldsymbol{w} \mid \mathbf{E} = \boldsymbol{e}). \end{aligned}$$

Both tasks can be solved within the same variable elimination (VE) and message passing frameworks as marginal inference, where instead of summation we use maximization. The second type of query is much more complicated both in theory and in practice since it involves both maximization and summation. In particular, exact inference methods such as VE can be intractable, even in simple networks. Hence, first we will briefly discuss them and then introduce some more efficient methods.

Recall that while discussing VE we introduced two operations on factors, which were the foundation in performing the algorithm. Now we need to introduce one additional operation called *the factor maximization*. Let $\mathbf{X}$ be a set of variables, and $Y \notin \mathbf{X}$ a variable not belonging to the set $\mathbf{X}$. Let $\phi(\mathbf{X}, Y)$ be a factor over those variables. We define the factor maximization of Y in ϕ to be a factor ψ over $\mathbf{X}$ such that:

$$\psi(\mathbf{X}) = \max_{Y} \phi(\mathbf{X}, Y).$$

More precisely,

$$\psi(\boldsymbol{x}) = \max_{y \in Val(Y)} \phi(\boldsymbol{x}, y)$$

for each instantiation $\boldsymbol{x} \in Val(\mathbf{X})$. Similarly to the property (3.9) we have that if a set of variables $\mathbf{X}$ is not in the scope of the factor ϕ_1, then

$$\max_{\mathbf{X}}(\phi_1 \cdot \phi_2) = \phi_1 \cdot \max_{\mathbf{X}} \phi_2 \tag{3.13}$$

and

$$\max_{\mathbf{X}}(\phi_1 + \phi_2) = \phi_1 + \max_{\mathbf{X}} \phi_2. \tag{3.14}$$

This leads us to *a max-product variable elimination algorithm* for a general MAP query, which is constructed in the same way as a sum-product variable elimination algorithm in Subsection 3.2.1, but we replace the marginalizing step (summation) with maximization over corresponding variables.

This way we find the maximum value for the joint probability, though the original and more interesting problem is to find the most probable assignment corresponding to that maximum probability. This process is called *a traceback procedure*, which is quite straightforward (details can be found in Koller and Friedman (2009)). In the process of eliminating variables we find their maximizing value given the values of the variables that have not yet been eliminated. When we pick the value of the final variable, we can then go back and pick the values of the remaining variables accordingly.

Recall that the joint distribution P in Bayesian networks is represented by a product of factors, where each factor coincides with a CPD (we introduced this representation in (3.8)). Then we can write the marginal MAP query as

$$\operatorname*{argmax}_{\boldsymbol{y}} \sum_{\mathbf{W}} P(\boldsymbol{y}, \mathbf{W}) = \operatorname*{argmax}_{\boldsymbol{y}} \sum_{\mathbf{W}} \prod_{i} \phi_i,$$

where we skipped the evidence set for the transparency of notation since it does not effect the main point of discussion. First we compute

$$\max_{\boldsymbol{y}} \sum_{\mathbf{W}} \prod_{i} \phi_i.$$

This form immediately suggests an algorithm combining the ideas of sum-product and max-product variable elimination. Specifically, the summations and maximizations outside the product can be viewed as operations on factors. Thus, to compute the value of this expression, we simply have to eliminate the variables in $\mathbf{W}$ by summing them out, and the variables in $\mathbf{Y}$ by maximizing them out. When eliminating a variable X, whether by summation or by maximization, we simply multiply all the factors whose scope involves X, and then eliminate X to produce the resulting factor. The ability to perform this step is justified by the interchangeability of factor summation and maximization with factor product (properties (3.9) and (3.13)). The traceback procedure to find the most probable assignment can also be found in Koller and Friedman (2009).

At first glance it seems that algorithms for both queries have the same complexity but that is not the case. It can be shown that even on very simple networks, elimination algorithms can require exponential time to solve a marginal MAP query (see Example 13.7 in Koller and Friedman (2009)). The difficulty comes from the fact that we are not free to choose an arbitrary elimination ordering. When summing out variables, we can utilize the fact that the operations of summing out different variables commute. Thus, when performing summing-out operations for sum-product variable elimination, we could sum out the variables in any order. Similarly, we could use the same flexibility in the case of max-product elimination. Unfortunately, the max and sum operations do not commute. Thus, in order to maintain the correct semantics of marginal MAP queries, as specified in the equation, we must perform all the variable summations before we can perform any of the variable maximizations.

We can also use the message passing framework, or more general case of clique tree algorithm, to MAP inference. In Subsection 3.2.2 we used clique trees to compute the sum-marginals over each of the cliques in the tree. Here, we compute a set of max-marginals over each of those cliques. By the max-marginal of a function f defined on the set $\mathbf{X}$ relative to a set of variables $\mathbf{Y} \subset \mathbf{X}$ we denote such a factor that for each $\boldsymbol{y} \in \mathbf{Y}$

$$MaxMarginal_f(\boldsymbol{y}) = \max_{\langle \boldsymbol{x} \rangle_{\mathbf{Y}} = \boldsymbol{y}} f(\boldsymbol{x})$$

determines the value of the unnormalized probability of the most likely joint assignment $\boldsymbol{x} \in \mathbf{X}$ consistent with $\boldsymbol{y}$. We compute the whole set for two reasons. First, the set of max-marginals can be a useful indicator for how confident we are in particular components of the MAP assignment. Second, in many cases, an exact solution to the MAP problem via a variable elimination procedure is intractable. In this case, to compute approximate max-marginals we can use message passing procedure in cluster graphs, similar to the clique tree procedure. These pseudo-max-marginals can be used for selecting an assignment; while this assignment is not generally the MAP assignment, we can nevertheless provide some guarantees in certain cases. As before, our task consists of two parts: computing the max-marginals and decoding them to extract a MAP assignment.

As for the first part, in the same way as we modified sum-product VE to sum-product message-passing we modify max-product VE to max-product belief propagation algorithm in clique trees. The resulting algorithm executes precisely the same initialization and

overall message scheduling as in the sum-product belief propagation algorithm. The only difference is that we use max-product rather than sum-product message passing. As a result of running the algorithm we will get a set of max-marginals for every clique of our clique tree.

Each belief is the max-marginal of the clique $\beta_i(\mathbf{C}_i) = MaxMarginal_p(\mathbf{C}_i)$ and all pairs of adjacent cliques are *max-calibrated*

$$\mu_{i,j}(\mathbf{S}_{i,j}) = \max_{\mathbf{C}_i \setminus \mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i) = \max_{\mathbf{C}_j \setminus \mathbf{S}_{i,j}} \beta_j(\mathbf{C}_j).$$

Similarly to sum-product message passing we get reparameterization of the distribution in the form (3.12) with corresponding beliefs of the max-product belief propagation algorithm.

Now we need to decode those max-marginals to get a MAP assignment. In the case of variable elimination, we had the max-marginal only for a single last to be eliminated variable and could identify the assignment for that particular variable. To compute the assignments to the rest of the variables, we had to perform a traceback procedure. Now the situation appears different. One obvious solution is to use the max-marginal for each variable to compute its own optimal assignment, and thereby compose a full joint assignment to all variables. However, this simplistic approach works only in case when there is a unique MAP assignment, equivalently, each max-marginal has a unique maximal value. For generic probability measures this is not a very rigid constraint, thus, we can find the unique MAP assignment by locally optimizing the assignment to each variable separately.

Otherwise, in most cases to break ties we can introduce a slight random perturbation into all of the factors, making all of the elements in the joint distribution have slightly different probabilities. However, there might be cases when we need to preserve the structure in relationships between some variables, for example some variables can share parameters or there might be some deterministic structure that should be preserved. Under these circumstances we find a locally optimal assignment using for example traceback procedure. Afterwards we can verify if this assignment is a MAP assignment (for procedure and verification see Koller and Friedman (2009)).

MAP as Linear Optimization Problem. In MAP inference we search for assignments which maximize a certain measure, in our case either the joint probability over all non-evidence variables or the probability over some set of variables. Therefore, it is natural to consider it directly as an optimization problem. There exists extensive literature on optimization algorithms and we can apply some of those ideas and algorithms to our specific case.

The main idea here is to reduce our MAP problem to an Integer Linear Programming (ILP) problem, i.e. an optimization problem over a set of integer valued variables, where both the objective and the constraints are linear. First, to define ILP problem we need to turn the product representation of the joint probability as in (3.8) into a sum, replacing the probability with its logarithm. It is possible because all the factors (CPDs)

are positive. Hence, we want to compute

$$\operatorname*{argmax}_{\xi} \prod_{i=1}^{n} \phi_i(\mathbf{A}_i) = \operatorname*{argmax}_{\xi} \sum_{i=1}^{n} \log(\phi_i(\mathbf{A}_i)),$$

where ξ is a general assignment for the whole vector of variables in the network, and $\mathbf{A}_i = (X_i, \mathbf{pa}_{\mathcal{G}}(X_i))$ represents a set of variables including the i-th variable and its parents in the network. Note that the whole discussion in this paragraph is actually identical for MRFs with positive factors, the only difference is the number of factors, but since they are not the focus of this thesis, we formulate everything in the Bayesian networks framework.

For variable indices $r \in \{1, \ldots, n\}$ we define the number of corresponding possible vector instantiations $n_r = |\mathit{Val}(\mathbf{A}_r)|$. For any joint assignment ξ, if this assignment constrained to the variables from $\mathbf{A}_r$ takes the value of $\boldsymbol{a}_r^j$, $j = \{1, \ldots, n_r\}$, i.e. $\xi_{\mathbf{A}_r} = \boldsymbol{a}_r^j$, then the factor $\log(\phi_r)$ makes a contribution to the objective of a quantity denoted as $\eta_j^r = \log(\phi_r(\boldsymbol{a}_r^j))$.

We introduce optimization variables $q(\boldsymbol{x}_r^j)$, where r enumerates the different factors, and j enumerates the different possible assignments to the variables from $\mathbf{A}_r$. These variables take binary values, so that $q(\boldsymbol{x}_r^j) = 1$ if and only if $\mathbf{A}_r = \boldsymbol{a}_r^j$ and 0 otherwise. It is important to distinguish the optimization variables from the random variables in our original graphical model; here we have an optimization variable $q(\boldsymbol{x}_r^j)$ for each joint assignment $\boldsymbol{a}_r^j$ to the model variables $\mathbf{A}_r$.

Let $\boldsymbol{q}$ denote a vector of the optimization variables $\{q(\boldsymbol{x}_r^j), \quad 1 \leq r \leq n, \quad 1 \leq j \leq n_r\}$ and $\boldsymbol{\eta}$ denote a vector of the coefficients η_r^j sorted in the same order. Both of these are vectors of dimension $N = \sum_{r=1}^{n} n_r$. With this interpretation, the MAP objective can be rewritten as:

$$\max_{\boldsymbol{q}} \sum_{r=1}^{n} \sum_{j=1}^{n_r} \eta_r^j q(\boldsymbol{x}_r^j) \tag{3.15}$$

or, in shorthand, $\max_{\boldsymbol{q}} \boldsymbol{\eta}^\top \boldsymbol{q}$.

Now that we have an objective to maximize we need to add some consistency constraints that would guarantee that an assignment $\boldsymbol{q} \in \{0, 1\}^N$ we get as a solution of optimization problem is legal, meaning it corresponds to some assignment in $\mathcal{X}$. Namely, first we require that we restrict attention to integer solutions, then we construct two constraints to make sure that these integer solutions are consistent. The first constraint enforces the mutual exclusivity within a factor and the second one implies that factors in our network agree on the variables in the intersection of their scopes. In this way we reformulate the MAP task as an integer linear program, where we optimize the linear objective of equation (3.15) subject to discussed constraints. We note that the problem of solving integer linear programs is itself $\mathcal{NP}$-hard, so that we do not avoid the basic hardness of the MAP problem.

One of the methods often used to tackle ILP problems is the method of linear program relaxation. In this approach we turn a discrete, combinatorial optimization problem into a continuous problem. This problem is a linear program (LP), which can be solved in

polynomial time, and for which a range of very efficient algorithms exists. One can then use the solutions to this LP to obtain approximate solutions to the MAP problem. To perform this relaxation, we substitute the condition that the solutions are integer with a relaxed constraint that they are non-negative.

This linear program is a relaxation of our original integer program, since every assignment to $\boldsymbol{q}$ that satisfies the constraints of the integer problem also satisfies the constraints of the linear program, but not the other way around. Thus, the optimal value of the objective of the relaxed version will be no less than the value of the (same) objective in the exact version, and it can be greater when the optimal value is achieved at an assignment to $\boldsymbol{q}$ that does not correspond to a legal assignment ξ. An important special case are tree-structured graphs, in which the relaxation is guaranteed to always return integer solutions, which are in turn optimal (for proof and more detailed discussion see Koller and Friedman (2009)). Otherwise we get approximate solutions, which in order we need to transform into integer (and legal) assignments.

One approach is a greedy assignment process, which assigns values to the variables X_i one at a time. Another approach is to round the LP solution to its nearest integer value. This approach works surprisingly well in practice and has theoretical guarantees for some classes of ILPs (Koller and Friedman (2009)).

An alternative method for the MAP problem which also comes from the optimization theory is called *dual decomposition*. Dual decomposition uses the principle that our problem can be decomposed into sub-problems, together with linear constraints (the same as in ILP) that enforce some notion of agreement between solutions to the different problems. The sub-problems are chosen such that they can be solved efficiently using exact combinatorial algorithms. The agreement constraints are incorporated using Lagrange multipliers, it is called Lagrangian relaxation, and an iterative algorithm - for example, a subgradient algorithm - is used to minimize the resulting dual. The initial work on dual decomposition in probabilistic graphical models was focused on the MAP problem for MRFs (see Komodakis et al. (2007)).

By formulating our problem as a linear program or its dual, we obtain a very flexible framework for solving it; in particular, we also can easily incorporate additional constraints into the LP, which reduce the space of possible assignments of $\boldsymbol{q}$, eliminating some solutions that do not correspond to actual distributions over $\mathcal{X}$. The problems are convex and in principle they can be solved directly using standard techniques, but the size of the problems is very large, which makes this approach unfeasible in practice. However, the LP has special structure: when viewed as a matrix, the equality constraints in this LP all have a particular block structure that corresponds to the structure of adjacent clusters. Moreover, when the network is not densely connected, the constraint matrix is also sparse, thus, standard LP solvers may not be fully suited for exploiting this special structure. The theory of convex optimization provides a wide spectrum of tools, and some are already being adapted to take advantage of the structure of the MAP problem (see for example, Wainwright et al. (2005), Sontag and Jaakkola (2007)). The empirical evidence suggests that the more specialized solution methods for the MAP problems are

often more effective.

Other methods. Another method for solving a MAP problem is local search algorithms. It is a heuristic-type solution, which starts with an arbitrary assignment and performs "moves" on the joint assignment that locally increase the probability. This technique does not offer theoretical justification; however, we can often use prior knowledge to come up with highly effective moves. Therefore, in practice, local search may perform extremely well.

There are also searching methods that are more systematic. They search the space so as to ensure that assignments that are not considered are not optimal, and thereby guarantee an optimal solution. Such methods generally search over the space of partial assignments, starting with the empty assignment and successively assigning variables one at a time. One such method is known as branch-and-bound.

These methods have much greater applicability in the context of marginal MAP problem, where most other methods are not currently applicable. In the next subsection we discuss sample-based algorithms which can be applied both to marginal and MAP inference.

3.2.4 Sampling-based methods for inference

In practice, the probabilistic models that we use can often be quite complex, and simple algorithms like VE may be too slow for them. In addition, many interesting classes of models may not have exact polynomial-time solutions at all, and for this reason, much research effort in machine learning is spent on developing algorithms that yield approximate solutions to the inference problem. In this subsection we consider some sampling methods that can be used to perform both marginal and MAP inference queries; additionally, they can compute various interesting quantities, such as the expectation $\mathbb{E}[f(\mathbf{X})]$ of a function of the random vector distributed according to a given probabilistic model.

In general, sampling is rather a hard problem. The aim is to generate a random sample of the observations of $\mathbf{X}$. However, our computers can only generate samples from very simple distributions, such as the uniform distribution over $[0, 1]$. All sampling techniques involve calling some kind of simple subroutine multiple times in a properly constructed way. For example, in case of multinomial distribution with parameters $\theta_1, \ldots, \theta_k$ instead of directly sampling a multinomial variable we can sample a single uniform variable previously subdividing a unit interval into k regions with region i having size θ_i. Then we sample uniformly from $[0, 1]$ and return the value of the region in which our sample falls.

Forward sampling. Now let us return to the case of Bayesian networks (BN). We can apply the same sampling technique to BNs with multinomial variables. We start from the nodes which do not have parents, these variables simply have multinomial distribution, and we go down the network to the next generation as arrows point out until we reach the leaves. Therefore, for a particular node we need to wait until all of its parents are

sampled. When we know all the values of parents the variable naturally has multinomial distribution. In the Student example to sample student's grade, we would first sample an exam difficulty d' and an intelligence level i'. Then, once we have samples d' and i', we generate a student grade g' from $\mathbb{P}(g \mid d', i')$. There is one problem though, as we cannot perform it in case of having evidence for any variables besides roots.

Monte Carlo and rejection sampling. Algorithms that construct solutions based on a large number of samples from a given distribution are referred to as Monte Carlo (MC) methods. Sampling from an arbitrary distribution p lets us compute integrals of the form

$$\mathbb{E}_{\mathbf{X}\sim p}[f(\mathbf{X})] = \sum_{\boldsymbol{x}} f(\boldsymbol{x})p(\boldsymbol{x}),$$

where the summation extends over all possible values of $\mathbf{X}$ and p can be thought of as the density of $\mathbf{X}$ with respect to counting measure. Below we follow the same interpretation also with regards to joint and conditional distributions.

If $f(\mathbf{X})$ does not have special structure that matches the BN structure of p, this integral will be impossible to compute analytically; instead, we will approximate it using a large number of samples from p. Using Monte Carlo technique we approximate a target expectation with

$$\mathbb{E}_{\mathbf{X}\sim p}[f(\mathbf{X})] \approx I_T = \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{x}^t),$$

where $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^T$ are samples drawn according to p. It is easy to show that I_T is an unbiased estimator for $\mathbb{E}_{\mathbf{X}\sim p}[f(\mathbf{X})]$ and its variance can be made arbitrarily small with a sufficiently large number of samples.

Now let us consider rejection sampling as a special case of Monte Carlo integration. For example, suppose we have a Bayesian network over the set of variables $\mathbf{X} = \mathbf{Z} \cup \mathbf{E}$. We may use rejection sampling to compute marginal probabilities $\mathbb{P}(\mathbf{E} = \boldsymbol{e})$. We can rewrite the probability as

$$\mathbb{P}(\mathbf{E} = \boldsymbol{e}) = \sum_{\boldsymbol{z}} \mathbb{P}(\mathbf{Z} = \boldsymbol{z}, \mathbf{E} = \boldsymbol{e}) = \sum_{\boldsymbol{x}} \mathbb{P}(\mathbf{X} = \boldsymbol{x})\mathbb{I}(\mathbf{E} = \boldsymbol{e}) = \mathbb{E}_{\mathbf{X}\sim p}[\mathbb{I}(\mathbf{E} = \boldsymbol{e})]$$

and then take the Monte Carlo approximation. In other words, we draw many samples from p and report the fraction of samples that are consistent with the value of the marginal.

Importance sampling. Unfortunately, rejection sampling can be very wasteful. If $\mathbb{P}(\mathbf{E} = \boldsymbol{e})$ equals, say, 1%, then we will discard 99% of all samples. A better way of computing such integrals uses importance sampling. The main idea is to sample from an auxiliary distribution q (hopefully with $q(\boldsymbol{x})$ roughly proportional to $f(\boldsymbol{x}) \cdot p(\boldsymbol{x})$), and then reweigh the samples in a principled way, so that their sum still approximates the desired integral.

More formally, suppose we are interested in computing $\mathbb{E}_{\mathbf{X}\sim p}[f(\mathbf{X})]$. Adopting analogous convention regarding notation for probability distribution we may rewrite this integral as

$$\mathbb{E}_{\mathbf{X}\sim p}[f(\mathbf{X})] = \sum_{\boldsymbol{x}} f(\boldsymbol{x})p(\boldsymbol{x}) = \sum_{\boldsymbol{x}} f(\boldsymbol{x})\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}q(\boldsymbol{x}) =$$
$$= \mathbb{E}_{\mathbf{X}\sim q}[f(\mathbf{X})w(\mathbf{X})] \approx \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{x}^t)w(\boldsymbol{x}^t),$$

where $w(\boldsymbol{x}) = \dfrac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$ and the samples $\boldsymbol{x}^t$ are drawn from q. In other words, instead of sampling from p we may take samples from q and reweigh them with $w(\boldsymbol{x})$; the expected value of this Monte Carlo approximation will be the original integral. By choosing $q(\boldsymbol{x}) = \dfrac{|f(\boldsymbol{x})|p(\boldsymbol{x})}{\int |f(\boldsymbol{x})|p(\boldsymbol{x})d\boldsymbol{x}}$ we can set the variance of the new estimator to zero. Note that the denominator is the quantity we are trying to estimate in the first place and sampling from such q is $\mathcal{NP}$-hard in general.

In the context of our previous example for computing $\mathbb{P}(\mathbf{E} = \boldsymbol{e})$, we may take q to be the uniform distribution and apply importance sampling as follows:

$$\mathbb{P}(\mathbf{E} = \boldsymbol{e}) = \mathbb{E}_{\boldsymbol{z}\sim p}[p(\boldsymbol{e} \mid \boldsymbol{z})] = \mathbb{E}_{\boldsymbol{z}\sim q}\left[p(\boldsymbol{e} \mid \boldsymbol{z})\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right] =$$
$$= \mathbb{E}_{\boldsymbol{z}\sim q}\left[\frac{p(\boldsymbol{e},\boldsymbol{z})}{q(\boldsymbol{z})}\right] = \mathbb{E}_{\boldsymbol{z}\sim q}[w_{\boldsymbol{e}}(\boldsymbol{z})] \approx \frac{1}{T}\sum_{t=1}^{T} w_{\boldsymbol{e}}(\boldsymbol{x}^t),$$

where $w_{\boldsymbol{e}}(\boldsymbol{z}) = \dfrac{p(\boldsymbol{e},\boldsymbol{z})}{q(\boldsymbol{z})}$. Unlike rejection sampling, this will use all the samples; if $p(\boldsymbol{z} \mid \boldsymbol{e})$ is not too far from uniform, this will converge to the true probability after only a very small number of samples.

Markov chain Monte Carlo. Now let us turn to performing marginal and MAP inference using sampling. We will solve these problems using a very powerful technique called Markov chain Monte Carlo (MCMC).

A key concept in MCMC is that of a Markov chain, which is a sequence of random elements having Markov property (see 2.3). A Markov chain $\mathcal{X} = (\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2, \dots)$ with each random vector $\mathbf{X}_i$ taking values from the same state space $Val(\mathcal{X})$ is specified by the initial distribution $\mathbb{P}(\mathbf{X}_0 = \boldsymbol{x})$, $\boldsymbol{x} \in Val(\mathcal{X})$, and the set of transition probabilities

$$\mathbb{P}(\mathbf{X}_{k+1} = \boldsymbol{x}' \mid \mathbf{X}_k = \boldsymbol{x})$$

for $\boldsymbol{x}, \boldsymbol{x}' \in Val(\mathcal{X})$, which do not depend on k (in this case the Markov chain is called homogeneous). Therefore, the transition probabilities at any time in the entire process depend only on the given state and not on the history of the process. In what follows, we consider finite state space only so we may assume $Val(\mathcal{X}) = \{1, \dots, d\}$, unless stated otherwise.

If the initial state $\mathbf{X}_0$ is drawn from a vector of probabilities p_0, we may represent the probability p_t of ending up in each state after t steps as

$$p_t = T^t p_0,$$

where T denotes the transition probability matrix with $T_{ij} = \mathbb{P}(\mathbf{X}_{k+1} = i \mid \mathbf{X}_k = j)$, $i, j \in \{1, \ldots, d\}$, and T^t denotes matrix exponentiation. If the limit $\lim_{t\to\infty} p_t = \pi$ exists, it is called a stationary distribution of the Markov chain. A sufficient condition for π to be a stationary distribution is called detailed balance:

$$\pi(j)T_{ij} = \pi(i)T_{ji}$$

for all $i, j \in \mathit{Val}(\mathcal{X})$.

The high-level idea of MCMC is to construct a Markov chain whose states are joint assignments to the variables in the model and whose stationary distribution is equal to the model probability p. Then, running the chain for a number of times, we obtain the sample from the distribution p. In order to construct such a chain, we first recall the conditions under which stationary distributions exist. This turns out to be true under two sufficient conditions: *irreducibility*, meaning that it is possible to get from any state $\boldsymbol{x}$ to any other state $\boldsymbol{x}'$ with positive probability in a finite number of steps, and *aperiodicity*, meaning that it is possible to return to any state at any time. In the context of continuous variables, the Markov chain must be *ergodic*, which is a slightly stronger condition than the above. For the sake of generality, we will require our Markov chains to be ergodic.

At a high level, MCMC algorithms will have the following structure. They take as an argument a transition operator T specifying a Markov chain whose stationary distribution is p, and an initial assignment $\mathbf{X}_0 = \boldsymbol{x}_0$ of the chain. An MCMC algorithm then performs the following steps:

1. Run the Markov chain from $\boldsymbol{x}_0$ for B burn-in steps.

2. Run the Markov chain for N sampling steps and collect all the states that it visits.

The aim of the burn-in phase is to wait until the state distribution is reasonably close to p. Therefore, we omit the first B states visited by the chain and then we collect a sample from the chain of the size N. A common approach to set the number B is to use a variety of heuristics to try to evaluate the extent to which a sample trajectory has "mixed", i.e. when it is reasonably close to p (see Koller and Friedman (2009)). Also Geyer (2011) advocates that burn-in is unnecessary and uses other ways of finding good starting points. Gelman and Shirley (2012) propose to discard the first half of generated sequences. We may then use these samples for Monte Carlo integration (or in importance sampling). We may also use them to produce Monte Carlo estimates of marginal probabilities. Finally, we may take the sample with the highest probability and use it as an estimate of the mode (i.e. perform MAP inference).

Before we discuss two most important special cases, note that sampling-based methods have theoretical asymptotic justification. Therefore, their application for finite samples of

reasonable size may lead to drastically inaccurate results, especially in sophisticated and complex models. Successful implementation heavily depends on how well we understand structure of the model as well as on intensive experimentation. It can also be achieved by combining sampling with other inference methods.

Metropolis-Hastings Algorithm. The Metropolis-Hastings (MH) algorithm (Hastings (1970)) is one of the first ways to construct Markov chains within MCMC. The MH method constructs a transition operator T from two components:

1. A transition kernel q specified by the user. In practice, the distribution $q(\boldsymbol{x}' \mid \boldsymbol{x})$ can take almost any form and very often it is a Gaussian distribution centered at $\boldsymbol{x}$.

2. An acceptance probability for moves proposed by q, specified by the algorithm as

$$A(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left(1, \frac{p(\boldsymbol{x})q(\boldsymbol{x}' \mid \boldsymbol{x})}{p(\boldsymbol{x}')q(\boldsymbol{x} \mid \boldsymbol{x}')}\right).$$

At each step, if the Markov chain is in the state $\boldsymbol{x}$, then we choose a new point $\boldsymbol{x}'$ according to the distribution q. Then, we either accept this proposed change with the probability $\alpha = A(\boldsymbol{x}' \mid \boldsymbol{x})$, or with the probability $1 - \alpha$ we remain at our current state. Notice that the acceptance probability encourages the chain to move towards more likely points in the distribution (imagine for example that q is uniform); when q suggests that we move into a low-probability region, we follow that move only a certain fraction of time. Given any q the MH algorithm ensures that p is a stationary distribution of the resulting Markov Chain. More precisely, p will satisfy the detailed balance condition with respect to the Markov chain generated by MH algorithm. This is a straight consequence of the definition of $A(\boldsymbol{x}' \mid \boldsymbol{x})$.

As the result we wish to build the Markov chain with a small correlation between subsequent values, which allows to explore the support of the target distribution rather quickly. This correlation consists of two components. The higher the variance of q, the lower the correlation between the current state and the newly chosen one, and the lower the variance of q, the lower the correlation when we stay at the same state hitting the low-probability region. To choose a good kernel q we need to find good balance between the two. For multivariate distributions the covariance matrix for the proposal distribution should reflect the covariance structure of the target.

Gibbs sampling. A widely-used special case of the Metropolis-Hastings methods is Gibbs sampling. It was first described in Geman and Geman (1984). Suppose we have a finite sequence of random variables $X_1, \ldots, X_n$. We denote the i-th sample as $\boldsymbol{x}^{(i)} = (X_1^{(i)}, \ldots, X_n^{(i)})$. Starting with an arbitrary configuration $\boldsymbol{x}^{(0)}$ we perform the procedure below.

Repeat until convergence for $t = 1, 2, 3, \ldots$:

1. Set $\boldsymbol{x} \leftarrow \boldsymbol{x}^{(t-1)}$

2. For each variable X_i

   - Sample $X_i' \sim P(X_i \mid X_{-i})$
   - Update $\boldsymbol{x} \leftarrow (X_1^{(t)}, \ldots, X_{i-1}^{(t)}, X_i', X_{i+1}^{(t-1)}, \ldots, X_n^{(t-1)})$

3. Set $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{x}$

By X_{-i} we denote all the variables in our set except X_i. At each epoch of the step 2 only one site undergoes a possible change, so that successive samples for each iteration can differ in at most one coordinate. Note that at this step we use updated values of the variables for which we have already sampled new values. The sampling step is quite easy to perform because we only condition on variables from X_i-th Markov blanket, which consists of its parents, children and other parents of its children.

In Geman and Geman (1984) it was stated that the distribution of $\boldsymbol{x}^{(t)}$ converges to π as $t \to \infty$ regardless of $\boldsymbol{x}^{(0)}$. The only assumption is that we continue to visit each site which is obviously a necessary condition for convergence. As in case of any MCMC algorithm if we choose an arbitrary starting configuration there is a burn-in phase, for the list of intuitions on how to decide how many samples we want to discard see Casella and George (1992). To avoid the high correlation between successive samples in Gibbs sampler we can also take every r-th sample instead of all of them, which is rather a question of heuristics and experimenting.

3.3 Learning probabilities in BNs for incomplete data

Here we again consider categorical distributions. Suppose we observe a single incomplete case in our data, which we denote as $\boldsymbol{d} \in \mathcal{D}$. Under the assumption of parameter independence, we can compute the posterior distribution of $\boldsymbol{\theta}_{ij}$ for our network as follows:

$$p(\boldsymbol{\theta}_{ij} \mid \boldsymbol{d}) = (1 - p(\mathbf{pa}_i^j \mid \boldsymbol{d}))\{p(\boldsymbol{\theta}_{ij})\} + \sum_{k=1}^{r_i} p(x_i^k, \mathbf{pa}_i^j \mid \boldsymbol{d})\{p(\boldsymbol{\theta}_{ij} \mid x_i^k, \mathbf{pa}_i^j)\}.$$

Each term in curly brackets in this equation is a Dirichlet distribution. Thus, unless both X_i and all the variables in $\mathbf{pa}(X_i)$ are observed in case $\boldsymbol{d}$, the posterior distribution of $\boldsymbol{\theta}_{ij}$ will be a linear combination of Dirichlet distributions, that is a Dirichlet mixture with mixing coefficients $(1 - p(\mathbf{pa}_i^j \mid \boldsymbol{d}))$ and $p(x_i^k, \mathbf{pa}_i^j \mid \boldsymbol{d})$, $1 \leq k \leq r_i$. See Spiegelhalter and Lauritzen (1990) for the details of derivation.

When we observe a second incomplete case, some or all of the Dirichlet components in the previous equation will again split into Dirichlet mixtures. More precisely, the posterior distribution for $\boldsymbol{\theta}_{ij}$ will become a mixture of Dirichlet mixtures. As we continue to observe incomplete cases, where each case has missing values for the same set of variables, the posterior distribution for $\boldsymbol{\theta}_{ij}$ will contain a number of components that is exponential in the number of cases. In general, for any interesting set of local likelihoods and priors, the exact computation of the posterior distribution for $\boldsymbol{\theta}$ will be intractable. Thus, we require an approximation for incomplete data.

One of the possible ways to approximate is Monte-Carlo methods discussed previously, for example the Gibbs sampler, which must be irreducible and each variable must be chosen infinitely often. More specifically for our case, to approximate $p(\boldsymbol{\theta} \mid \mathcal{D})$ given an incomplete data set we start with some initial states of the unobserved variables in each case (chosen randomly or otherwise) and as a result, we have a complete random sample $\mathcal{D}_c$. Then we choose some variable $X_i[l]$ (variable X_i in case l) that is not observed in the original random sample $\mathcal{D}$, and reassign its state according to the probability distribution

$$p(x'_{il} \mid \mathcal{D}_c \setminus \{x_{il}\}) = \frac{p(x'_{il}, \mathcal{D}_c \setminus \{x_{il}\})}{\sum_{x''_{il}} p(x''_{il}, \mathcal{D}_c \setminus \{x_{il}\})},$$

where $\mathcal{D}_c \setminus x_{il}$ denotes the data set $\mathcal{D}_c$ with observation x_{il} removed, and the sum in the denominator runs over all states of the variable X_i. Both the numerator and denominator can be computed efficiently as in (3.6). In the third step we repeat this reassignment for all unobserved variables in $\mathcal{D}$, producing a new complete random sample $\mathcal{D}'_c$. The fourth step is to compute the posterior density $p(\boldsymbol{\theta}_{ij} \mid \mathcal{D}'_c)$ as in (3.7) and, under the assumption of parameter independence, the joint posterior $p(\boldsymbol{\theta} \mid \mathcal{D}'_c)$ will be a product of all densities $p(\boldsymbol{\theta}_{ij} \mid \mathcal{D}'_c)$. Finally, we iterate through last three steps, and use the average of $p(\boldsymbol{\theta} \mid \mathcal{D}'_c)$ as our approximation.

Monte-Carlo methods yield accurate results but they are often intractable, for example when the sample size is large. Another approximation that is more efficient than Monte-Carlo methods and often accurate for relatively large samples is the Gaussian approximation. The idea is that for large amounts of data we can approximate the distribution $p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$ as a multivariate-Gaussian distribution, namely

$$p(\boldsymbol{\theta} \mid \mathcal{D}) \approx p(\mathcal{D} \mid \tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}) \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})H(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top\right),$$

where $\tilde{\boldsymbol{\theta}}$ is the configuration of $\boldsymbol{\theta}$ that maximizes $g(\boldsymbol{\theta}) = \ln(p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta}))$ and H is a negative Hessian of $g(\boldsymbol{\theta})$. The vector $\tilde{\boldsymbol{\theta}}$ is also called the maximum a posteriori (MAP) configuration of $\boldsymbol{\theta}$. There are various methods to compute the second derivatives proposed in literature (Meng and Rubin (1991), Raftery (1995), Thiesson (1995)).

One more way to learn probabilities from incomplete data is the Expectation-Maximization (EM) algorithm. It is an iterative algorithm consisting of two alternating steps - Expectation and Maximization. When the data is incomplete we cannot calculate the likelihood function as in (3.2) and (3.3). Now instead of maximizing likelihood or log-likelihood function we will be maximizing *the expected log-likelihood* of the complete data set with respect to the joint distribution for $\mathbf{X}$ conditioned on the assigned configuration of the parameter vector $\boldsymbol{\theta}'$ and the known data $\mathcal{D}$. The calculation of the expected log-likelihood (Expectation step) amounts to computing *expected sufficient statistics*. For incomplete data the expected log-likelihood takes the following form

$$\mathbb{E}[\ell(\boldsymbol{\theta}) \mid \mathcal{D}, \boldsymbol{\theta}'] = \sum_{i=1}^{n} \sum_{l=1}^{q_i} \sum_{k=1}^{r_i} \hat{N}_{ilk} \log(\theta_{ilk}),$$

where

$$\hat{N}_{ilk} = \mathbb{E}[\mathbb{I}(X_i = x_i^k, \mathbf{pa}(X_i) = \mathbf{pa}_i^l) \mid \mathcal{D}, \boldsymbol{\theta}'] = \sum_{j=1}^{m} \mathbb{P}(X_i = x_i^k, \mathbf{pa}(X_i) = \mathbf{pa}_i^l \mid \boldsymbol{d}_j, \boldsymbol{\theta}').$$

Here $\boldsymbol{d}_j$ is possibly incomplete j-th case in $\mathcal{D}$. When X_i and all the variables in $\mathbf{pa}(X_i)$ are observed, the term for this case requires a trivial computation: it is either zero or one. Otherwise, we can use any Bayesian network inference algorithm discussed above to evaluate the term.

Having performed the Expectation step we want to find the new parameter vector, which is obtained by maximization of the expected log-likelihood (Maximization step). In our case we have new parameters on the r-th iteration

$$\theta_{ilk}^r = \frac{\hat{N}_{ilk}}{\sum_{k=1}^{r_i} \hat{N}_{ilk}}.$$

We start algorithm with an arbitrary (for example, random) parameter configuration $\boldsymbol{\theta}^0$ and iteratively perform two steps described above until the convergence. Dempster et al. (1977) showed that, under certain regularity conditions, iterations of the expectation and maximization steps will converge to a local maximum.

3.4 Learning parameters for CTBNs

The new method we propose in next chapters for learning CTBNs is capable of performing both tasks of parameter learning and structure learning simultaneously, although naturally these tasks can be performed separately. In this section we review selected methods focused only on parameter learning.

3.4.1 Data

In this thesis we discuss both complete and incomplete data. In essence, CTBN models the joint trajectories of its variables, hence having complete, or fully observed, data means that for each point in time of each trajectory, we know the full instantiation to all variables.

By $\mathcal{D} = \{\sigma[1], \ldots, \sigma[m]\}$ we denote a data set of trajectories. In case of complete data each $\sigma[i]$ is a complete set of state transitions and the times at which they occurred. Another way to specify each trajectory is to assign a sequence of states $\boldsymbol{x}_i \in \mathit{Val}(\mathbf{X})$, each with an associated duration.

In contrast to the definition of complete data, an incomplete data set can be represented by a set of one or more partial trajectories. A partially observed trajectory $\sigma \in \mathcal{D}$ can be specified as a sequence of *subsystems* S_i of X, each with an associated duration. A *subsystem* S describes the behaviour of the process over a subset of the full state space, i.e. $\mathit{Val}(S) \subset \mathit{Val}(\mathbf{X})$. It is simply a nonempty subset of states of $\mathbf{X}$, in which we know the system stayed for the duration of the observation. Some transitions are partially observed, i.e. we know only that they take us from one subsystem to another. Transitions

from one state to another within the subsystem are fully unobserved, hence, we do not know how many transitions there are inside of a particular subsystem nor when they do occur.

3.4.2 Learning parameters for complete data

Recall, that CTBN $\mathcal{N}$ consists of two parts. The first is an initial distribution $P_0^{\mathbf{X}}$, specified as a Bayesian network over $\mathbf{X}$. The second is a continuous transition model, specified as a directed (and possibly cyclic) graph and a set of conditional intensity matrices (CIM), one for each variable X_i in the network. For the purposes of this section we abbreviate $\mathbf{pa}_{\mathcal{G}}(X_i)$ as $\mathbf{pa}(X_i)$ and we denote CIMs as $\mathbf{Q}_{X_i|\mathbf{pa}(X_i)}$. Recall that each $\mathbf{Q}_{X_i|\mathbf{pa}(X_i)}$ consists of intensity matrices $\mathbf{Q}_{X_i|\mathbf{pa}_i}$, where $\mathbf{pa}_i$ is a single configuration of $\mathbf{pa}(X_i)$. Strictly speaking, $\mathbf{pa}_i$ is one of the possible parent configurations $\mathbf{pa}_i^1, \ldots, \mathbf{pa}_i^{q_i}$ similar to (3.1). In terms of pure intensity parameterization we denote elements of these matrices as $q_{xx'|\mathbf{pa}_i}$ and $q_{x|\mathbf{pa}_i}$. Note, that by Theorem 2.9 we can divide the set of parameters in terms of mixed intensity into two sets. Then for each variable X_i and each instantiation $\mathbf{pa}_i$ of its set of parents $\mathbf{pa}(X_i)$ the parameters of $\mathbf{Q}_{X_i|\mathbf{pa}(X_i)}$ will be $\boldsymbol{q}_{X_i} = \{q_{x|\mathbf{pa}_i} : x \in Val(X_i)\}$ and $\boldsymbol{\theta}_{X_i} = \{\theta_{xx'|\mathbf{pa}_i} : x, x' \in Val(X_i),\ x \neq x'\}$. More precisely, for each X_i and every $x \in Val(X_i)$ we have

$$\theta_{xx'|\mathbf{pa}_i} = \frac{q_{xx'|\mathbf{pa}_i}}{\sum_{x'} q_{xx'|\mathbf{pa}_i}}, \quad x' \in Val(X_i), \quad x \neq x'.$$

The learning problem for the initial distribution is a Bayesian network learning task, which was discussed previously in this chapter. Therefore, it remains to learn the vector of parameters $(\boldsymbol{q}, \boldsymbol{\theta})$.

Likelihood estimation. Let us start from a fully observed case and a single homogeneous Markov process $X(t)$. As all the transitions are observed, the likelihood of $\mathcal{D}$ can be decomposed as a product of the likelihoods for individual transitions d. Let $d = \langle x_d, t_d, x'_d \rangle \in \mathcal{D}$ be the transition where X transitions to state x'_d after spending the amount of time t_d in state x_d. Using the mixed intensity parameterization, we can write the likelihood for the single transition d as

$$L_X(\boldsymbol{q}, \boldsymbol{\theta} : d) = L_X(\boldsymbol{q} : d) L_X(\boldsymbol{\theta} : d) = q_{x_d} \exp(-q_{x_d} t_d) \cdot \theta_{x_d x'_d}.$$

Then multiplying the likelihoods for each transition d in our data $\mathcal{D}$ we can summarize it in terms of sufficient statistics $T[x]$ which describes the amount of time spent in each state $x \in Val(X)$ and $M[x, x']$ which encodes the number of transitions from x to x', where $x \neq x'$ as follows:

$$\begin{aligned} L_X(\boldsymbol{q}, \boldsymbol{\theta} : \mathcal{D}) &= \left(\prod_{d \in \mathcal{D}} L_X(\boldsymbol{q} : d) \right) \left(\prod_{d \in \mathcal{D}} L_X(\boldsymbol{\theta} : d) \right) \\ &= \left(\prod_x q_x^{M[x]} \exp(-q_x T[x]) \right) \left(\prod_x \prod_{x' \neq x} \theta_{xx'}^{M[x,x']} \right), \end{aligned} \tag{3.16}$$

where $M[x] = \sum_{x'} M[x, x']$.

Now in case of CTBNs, each variable X of the network $\mathcal{N}$ is conditioned on its parent set $\mathbf{Pa} = \mathbf{pa}_{\mathcal{G}}(X)$, and each transition of X must be considered in the context of the instantiation $\mathbf{pa}$ of $\mathbf{Pa}$. With complete data, we know the value of $\mathbf{Pa}$ during the entire trajectory, so at each point in time we know precisely which homogeneous intensity matrix $\mathbf{Q}_{X|\mathbf{pa}}$ governed the dynamics of X.

Thus, the likelihood decomposes into the product of likelihoods, each corresponding to the variable in the network, as

$$L_{\mathcal{N}}(\boldsymbol{q}, \boldsymbol{\theta} : \mathcal{D}) = \prod_{X_i \in \mathbf{X}} L_{X_i}(\boldsymbol{q}_{X_i|\mathbf{U}_i}, \boldsymbol{\theta}_{X_i|\mathbf{U}_i} : \mathcal{D}) = \prod_{X_i \in \mathbf{X}} L_{X_i}(\boldsymbol{q}_{X_i|\mathbf{U}_i} : \mathcal{D}) L_{X_i}(\boldsymbol{\theta}_{X_i|\mathbf{U}_i} : \mathcal{D}).$$

The term $L_X(\boldsymbol{\theta}_{X|\mathbf{Pa}} : \mathcal{D})$ is the probability of the sequence of state transitions, disregarding the times between transitions. These state changes depend only on the value of the parents at the moment of the transition. For each variable $X \in \mathbf{X}$ let $M[x, x' \mid \mathbf{pa}]$ denote the number of transitions from $X = x$ to $X = x'$ while $\mathbf{Pa} = \mathbf{pa}$. Then, with this set of sufficient statistics $M[x, x' \mid \mathbf{pa}]$, we have

$$L_X(\boldsymbol{\theta}_{X|\mathbf{Pa}} : \mathcal{D}) = \prod_{\mathbf{pa}} \prod_{x} \prod_{x' \neq x} \theta_{xx'|\mathbf{pa}}^{M[x,x'|\mathbf{pa}]}.$$

The computation of $L_X(\boldsymbol{q}_{X|\mathbf{Pa}} : \mathcal{D})$ is more subtle since the duration in the state can be terminated not only due to a transition of X, but also due to a transition of one of its parents. The total amount of time where $X = x$ and $\mathbf{Pa} = \mathbf{pa}$ can be decomposed into two different kinds of durations $T[x \mid \mathbf{pa}] = T_r[x \mid \mathbf{pa}] + T_c[x \mid \mathbf{pa}]$, where $T_r[x \mid \mathbf{pa}]$ is the total length of the time intervals that terminate with X remaining equal to x, and $T_c[x \mid \mathbf{pa}]$ is the total length of the time intervals that terminate with a change in the value of X. However, it is easy to show that we do not need to maintain the distinction between the two of them and we can use the set of $T[x \mid \mathbf{pa}]$ as sufficient statistics.

Finally, we can write the log-likelihood as a sum of local variable likelihoods of the form

$$\begin{aligned} &\ell_X(\boldsymbol{q}, \boldsymbol{\theta} : \mathcal{D}) = \ell_X(\boldsymbol{q} : \mathcal{D}) + \ell_X(\boldsymbol{\theta} : \mathcal{D}) = \\ &= \left[\sum_{\mathbf{pa}} \sum_{x} M[x \mid \mathbf{pa}] \log q_{x|\mathbf{pa}} - q_{x|\mathbf{pa}} T[x \mid \mathbf{pa}] \right] + \left[\sum_{\mathbf{pa}} \sum_{x} \sum_{x' \neq x} M[x, x' \mid \mathbf{pa}] \log \theta_{xx'|\mathbf{pa}} \right]. \end{aligned} \tag{3.17}$$

Now we can write the maximum-likelihood (MLE) parameters as functions of the sufficient statistics as follows (for the proof see Nodelman (2007)):

$$\hat{q}_{x|\mathbf{pa}} = \frac{M[x \mid \mathbf{pa}]}{T[x \mid \mathbf{pa}]}, \qquad \hat{\theta}_{xx'|\mathbf{pa}} = \frac{M[x, x' \mid \mathbf{pa}]}{M[x \mid \mathbf{pa}]}.$$

The Bayesian approach. The other way to estimate parameters in case of fully observed data is the Bayesian approach. To perform Bayesian parameter estimation, similarly to the case of Bayesian networks, for computational efficiency we use a conjugate

prior (one where the posterior after conditioning on the data is in the same parametric family as the prior) over the parameters of our CTBN.

For a single Markov process we have two types of parameters, a vector of parameters $\boldsymbol{\theta}$ for categorical distribution and q for exponential distribution. An appropriate conjugate prior for the exponential parameter q is the Gamma distribution $P(q) = Gamma(\alpha, \tau)$, and as we mentioned in Section 3.1, the standard conjugate prior to categorical distribution is a Dirichlet distribution $P(\boldsymbol{\theta}) = Dir(\alpha_{xx_1}, \ldots, \alpha_{xx_k})$. The posterior distributions $P(\boldsymbol{\theta} \mid \mathcal{D})$ and $P(q \mid \mathcal{D})$ given data are Dirichlet and Gamma distributions, respectively.

In order to apply this idea to an entire CTBN we need to make two standard assumptions for parameter priors in Bayesian networks, *global parameter independence*:

$$P(\boldsymbol{q}, \boldsymbol{\theta}) = \prod_{X \in \mathbf{X}} P(\boldsymbol{q}_{X|\mathbf{pa}_{\mathcal{G}}(X)}, \boldsymbol{\theta}_{X|\mathbf{pa}_{\mathcal{G}}(X)})$$

and *local parameter independence* for each variable X in the network:

$$P(q_{X|\mathbf{Pa}}, \boldsymbol{\theta}_{X|\mathbf{Pa}}) = \left(\prod_{x} \prod_{\mathbf{pa}} P(q_{x|\mathbf{pa}}) \right) \left(\prod_{x} \prod_{\mathbf{pa}} P(\boldsymbol{\theta}_{x|\mathbf{pa}}) \right).$$

If our parameter prior satisfies these assumptions, so does our posterior, as it belongs to the same parametric family. Thus, we can maintain our parameter distribution in the closed form, and update it using the obvious sufficient statistics $M[x, x' \mid \mathbf{pa}]$ for $\boldsymbol{\theta}_{x|\mathbf{pa}}$ and $M[x \mid \mathbf{pa}], T[x \mid \mathbf{pa}]$ for $q_{x|\mathbf{pa}}$.

Given a parameter distribution, we can use it to predict the next event, averaging out the event probability over the possible values of the parameters. As usual, this prediction is equivalent to using "expected" parameter values, which have the same form as the MLE parameters, but account for the "imaginary counts" of the hyperparameters:

$$\hat{q}_{x|\mathbf{pa}} = \frac{\alpha_{x|\mathbf{pa}} + M[x \mid \mathbf{pa}]}{\tau_{x|\mathbf{pa}} + T[x \mid \mathbf{pa}]}, \qquad \hat{\theta}_{xx'|\mathbf{pa}} = \frac{\alpha_{xx'|\mathbf{pa}} + M[x, x' \mid \mathbf{pa}]}{\alpha_{x|\mathbf{pa}} + M[x \mid \mathbf{pa}]}.$$

Note that, in principle, this choice of parameters is only valid for predicting a single transition, after which we should update our parameter distribution accordingly. However, as is often done in other settings, we can approximate the exact Bayesian computation by "freezing" the parameters to these expected values, and use them for predicting an entire trajectory.

3.4.3 Learning parameters for incomplete data

Recall, that in case of Bayesian networks one of the methods to deal with missing data was Expectation-Maximization (EM) algorithm. Here we provide a concise description of the algorithm based on EM for CTBNs presented in detail in Nodelman et al. (2012). We start again with reviewing the EM scheme for a single Markov process X, which is the basis of the algorithm for CTBNs. Let $\mathcal{D} = \{\sigma[1], \ldots, \sigma[m]\}$ denote the set of all partially observed trajectories of X.

For each partial trajectory $\sigma[i] \in \mathcal{D}$ we can consider the space $\mathbf{H}[i]$ of possible completions of this trajectory. For every transition of $\sigma[i]$ each completion $h[i] \in \mathbf{H}[i]$ specifies which underlying transition of X occurred. Also it specifies all the entirely unobserved transitions of X. Combining $\sigma[i]$ and $h[i]$ gives us a complete trajectory $\sigma^+[i]$ over X. Note that, in a partially observed trajectory, the number of possible unobserved transitions is unknown. Moreover, there are uncountably many times at which each transition can take place. Nevertheless, we can define the set $\mathcal{D}^+ = \{\sigma^+[1], \ldots, \sigma^+[m]\}$ of completions of all of the partial trajectories in $\mathcal{D}$. For examples of completions see Nodelman et al. (2012).

As we mentioned in the previous subsection, the sufficient statistics of the set of complete trajectories $\mathcal{D}^+$ for a Markov process are $T[x]$, the total amount of time that X stays in x, and $M[x, x']$, the number of times in which X transitions from x to x'. Applying logarithm to (3.16) we can write the log-likelihood $\ell_X(\boldsymbol{q}, \boldsymbol{\theta} : \mathcal{D}^+)$ for X as an expression of these sufficient statistics.

Let r be a probability density over each completion in $\mathbf{H}[i]$ which, in turn, yields a density over possible completions of the data $\mathcal{D}^+$. We can write the expectations of the sufficient statistics with respect to the probability density over possible completions of the data as $\overline{T}[x]$, $\overline{M}[x, x']$ and $\overline{M}[x]$. These expected sufficient statistics allow us to write the expected log-likelihood for X as

$$\mathbb{E}_r[\ell_X(\boldsymbol{q}, \boldsymbol{\theta} : \mathcal{D}^+)] = \mathbb{E}_r[\ell_X(\boldsymbol{q} : \mathcal{D}^+)] + \mathbb{E}_r[\ell_X(\boldsymbol{\theta} : \mathcal{D}^+)] = \\ = \sum_x \left(\overline{M}[x] \ln(q_x) - q_x \overline{T}[x] + \sum_{x' \neq x} \overline{M}[x, x'] \ln(\theta_{xx'}) \right).$$

Now we can use the EM algorithm to find maximum-likelihood parameters $\boldsymbol{q}, \boldsymbol{\theta}$ of X. The EM algorithm begins with an arbitrary initial parameter assignment, $\boldsymbol{q}^0, \boldsymbol{\theta}^0$. It then repeats the two steps, Expectation and Maximization, updating the parameter set, until convergence. After the k-th iteration we start with parameters $\boldsymbol{q}^k, \boldsymbol{\theta}^k$. The Expectation step goes as following: using the current set of parameters, we define for each $\sigma[i] \in \mathcal{D}$, the probability density $r^k(h[i]) = p(h[i] \mid \sigma[i], \boldsymbol{q}^k, \boldsymbol{\theta}^k)$. We then compute expected sufficient statistics $\overline{T}[x]$, $\overline{M}[x, x']$ and $\overline{M}[x]$ according to this posterior density over completions of the data given the data and the model. Using the expected sufficient statistics we just have computed as if they came from a complete data set, we set $\boldsymbol{q}^{k+1}$ and $\boldsymbol{\theta}^{k+1}$ to be the new maximum likelihood parameters for our model as follows

$$q_x^{k+1} = \frac{\overline{M}[x]}{\overline{T}[x]}, \qquad \theta_{xx'}^{k+1} = \frac{\overline{M}[x, x']}{\overline{M}[x]}. \tag{3.18}$$

The difficult part in this algorithm is the Expectation Step. The space over which we are integrating is highly complex, and it is not clear how to compute the expected sufficient statistics in a tractable way.

In Nodelman et al. (2012) and Nodelman (2007) authors provided in detail the algorithm on how to compute expected sufficient statistics for an n-state homogeneous Markov process X_t with intensity matrix $\mathbf{Q}_X$ with respect to the posterior probability density over

completions of the data given the observations and the current model. The statistics are computed for each partially observed trajectory $\sigma \in \mathcal{D}$ separately and then the results are combined.

A partially observed trajectory σ is given as a sequence of N subsystems so that the state is restricted to subsystem S_i during the interval $[t_i, t_{i+1})$ for $0 \leq i \leq N-1$. To conduct all the necessary computations, for each time t, the forward and backward probability vectors α_t and β_t are defined, which include evidence of any transition at time t, and also vectors α_t^- and β_t^+, neither of which include evidence of a transition at time t. The total expected time $\mathbb{E}[T[j]]$ is obtained by summing the integrals over all intervals of constant evidence $[v, w)$ with the subsystem S to which the state is restricted on that interval. Each integrand is an expression containing α_v, β_w and $\mathbf{Q}_S$. The computations for each integral are performed via the Runge-Kutta method of fourth order with an adaptive step size.

Regarding the expected number of transitions $\mathbb{E}[M[x, x']]$ from the state x to x' discrete time approximations of $M[x, x']$ are considered which in the limit as the size of the discretization goes to zero yields an exact equation. As a result we get the sum of expressions where each summand is associated with a time interval. The overall expression for the expected number of transitions consists of two parts: the sum of products corresponding to intervals with partially observed transitions and containing α_t^- and β_t^+ for different time points t and the sum of integrals of practically identical form to those obtained for total expected time.

In order to compute α_t and β_t a forward-backward style algorithm (Rabiner and Juang (1986)) over the entire trajectory is used to incorporate evidence and get distributions over the state of the system at every time t_i. If needed it is possible to exclude incorporation of the evidence of the transition from either forward or backward vector and also obtain α_t^- and β_t^+. We can then write the distribution over the state of the system at time t given all the evidence.

Continuous time Bayesian networks are a factored representation for homogeneous Markov processes, hence, extending the EM algorithm to them involves making it sensitive to a factored state space. As mentioned previously, the log-likelihood decomposes as the sum of local log-likelihoods for each variable. With the sufficient statistics $T[x \mid \mathbf{pa}]$, $M[x, x' \mid \mathbf{pa}]$ and $M[x \mid \mathbf{pa}]$ of the set of complete trajectories $\mathcal{D}^+$ for each variable X in CTBN $\mathcal{N}$ the likelihood for each variable X further decomposes as in (3.17). By linearity of expectation, the expected log-likelihood function also decomposes in the same way. So we can write the expected log-likelihood $\mathbb{E}_r[\ell(\boldsymbol{q}, \boldsymbol{\theta} : \mathcal{D}^+)]$ as a sum of terms, one for each variable X, in a similar form as (3.17), but using the expected sufficient statistics $\overline{T}[x \mid \mathbf{pa}]$, $\overline{M}[x, x' \mid \mathbf{pa}]$ and $\overline{M}[x \mid \mathbf{pa}]$.

The EM algorithm for CTBNs is essentially the same as for homogeneous Markov processes. We need only specify how evidence in the network induces evidence on the induced Markov process, and how expected sufficient statistics in the Markov process give us the necessary sufficient statistics for CTBN.

The Maximization step is practically the same as in (3.18), we just use proper expected

sufficient statistics for the CTBN case:

$$q_{x|\mathbf{pa}}^{k+1} = \frac{\overline{M}[x \mid \mathbf{pa}]}{\overline{T}[x \mid \mathbf{pa}]}, \quad \theta_{xx'|\mathbf{pa}}^{k+1} = \frac{\overline{M}[x, x' \mid \mathbf{pa}]}{\overline{M}[x \mid \mathbf{pa}]}.$$

The Expectation step is again more difficult and could be done by flattening the CTBN into a single homogeneous Markov process with a size of the state space exponential in the number of variables. Then we follow the method described above. However, as the number of variables in the CTBN grows the process becomes intractable, so we are forced to use approximate inference.

We want this approximate algorithm to be able to compute approximate versions of the forward and backward messages α_t and β_s and extract the relevant sufficient statistics from these messages efficiently. In the next subsection we review a cluster graph inference algorithm which can be used to perform this type of approximate inference. Using obtained cluster beliefs (see below) we can compute $\alpha_{t_{i+1}}$ and β_{t_i} and use them in the forward-backward message passing procedure. The cluster distributions are represented as local intensity matrices from which we can compute the expected sufficient statistics over families $X_i, \mathbf{pa}_{\mathcal{G}}(X_i)$ as described above.

3.5 Inference for CTBNs

To gain the perspective on the whole concept of continuous time Bayesian networks and their power, similarly to Bayesian networks, we discuss the questions of inference although it is not the key subject of this thesis. We start with a discussion of the types of queries we might wish to answer and the difficulties of the exact inference.

Inference for CTBNs can take a number of forms. The common types of queries are:

- querying the marginal distribution of a variable at a particular time or also the time at which a variable first takes a particular value,
- querying the expected number of transitions for a variable during a fixed time interval,
- querying the expected amount of time a variable stayed in a particular state during an interval.

Previously we showed that we can view CTBN as a compact representation of a joint intensity matrix for a homogeneous Markov process. Thus, at least in principle, we can use CTBN to answer any query that we can answer using an explicit representation of a Markov process: we can form the joint intensity matrix and then answer queries just as we would do for any homogeneous Markov process.

The obvious flaw is that this approach for answering these queries requires us to generate the full joint intensity matrix for the system as a whole. The size of the matrix is exponential in the number of variables, making this approach generally intractable. The

graphical structure of the CTBN immediately suggests that we perform the inference in a decomposed way, as in Bayesian networks. Unfortunately, the problems are significantly more complex in this setting.

In Nodelman et al. (2002) the authors describe an approximate inference algorithm based on ideas from clique tree inference, but without any formal justification for the algorithm. More importantly, the algorithm covers only point evidence, meaning observations of the value of a variable at a point in time, but in many applications we observe a variable for an interval or even for its entire trajectory. Therefore, we shortly describe an approximate inference algorithm called Expectation Propagation (EP) presented in Nodelman et al. (2005) that allows both point and interval evidence. The algorithm uses message passing in a cluster graph (with clique tree algorithms as a special case), where the clusters do not contain distributions over the cluster variables at individual time points, but over trajectories of the variables through a duration.

As we discussed in this chapter, in cluster graph algorithms we construct a graph whose nodes correspond to clusters of variables and then pass messages between these clusters to produce an alternative parameterization, in which the marginal distribution of the variables in each cluster can be read directly from the cluster. In discrete graphical models, when the cluster graph is a clique tree, two passes of the message passing algorithm produce the exact marginals. In generalized belief propagation (Yedidia et al. (2001)), message passing is applied to a graph which is not a clique tree, in which case the algorithm may not converge, and produces only approximate solutions. There are several forms of message passing algorithm as we have discussed in Subsection 3.2.2. The algorithm of Nodelman et al. (2005) is based on multiply-marginalize-divide scheme of Lauritzen and Spiegelhalter (1988), which we now briefly review.

A cluster graph is defined in terms of a set of clusters $\mathcal{C}_i$, whose scope is some subset of the variables $\mathbf{X}$. Clusters are connected to each other by edges, along which messages are passed. The edges are annotated with a set of variables called a sepset $S_{i,j}$, which is the set of variables in $\mathcal{C}_i \cap \mathcal{C}_j$. The messages passed over an edge between $\mathcal{C}_i$ and $\mathcal{C}_j$ are factors over the scope $S_{i,j}$. Each cluster $\mathcal{C}_i$ maintains a potential β_i, which is a factor reflecting its current beliefs over the variables in its scope. Each edge similarly maintains a message $\mu_{i,j}$ which encodes the last message sent over the edge. The potentials are initialized with a product of some subset of factors parameterizing the model (CIMs in our setting). Messages are initialized to be uninformative. Clusters then send messages to each other, and use incoming messages to update their beliefs over the variables in their scope. The message $m_{i \to j}$ from $\mathcal{C}_i$ to $\mathcal{C}_j$ is the marginal distribution $S_{i,j}$ according to β_i. The neighbouring cluster $\mathcal{C}_j$ assimilates this message by multiplying it into β_i, but avoids double-counting by first dividing by the stored message $\mu_{i,j}$. Thus, the message update takes the form $\beta_j \longleftarrow \beta_j \cdot \frac{m_{i \to j}}{\mu_{i,j}}$.

In the algorithm the cluster beliefs represent not the factors over values of random variables themselves, but rather cluster potentials and messages both encode measures over entire trajectories of the variables in their scope. The number of parameters grows exponentially with the size of the network, and thus we cannot pass messages exactly

without giving up the computational efficiency of the algorithm. To address this issue Nodelman et al. (2005) used the *expectation propagation (EP)* approach of Minka (2001), which performs approximate message passing in cluster graphs. In order to get an approximate message each message $m_{i \to j}$ is projected into a compactly representable space so as to minimize the KL-divergence between the message and its approximation. To encode the cluster potentials CIMs are used. In order to apply the EP algorithm to clusters of this form some basic operations over CIMs need to be defined. They include CIM product and division, approximate CIM marginalization, as well as incorporating the evidence into CIM.

The message propagation algorithm is first considered for one segment of the trajectory with constant continuous evidence. Exactly the same as for Bayesian networks, this process starts with constructing the cluster tree for the graph $\mathcal{G}$. Note that cycles do not introduce new issues. We can simply moralize the graph connecting all parents of a node with undirected edges and then make all the remaining edges undirected. If there is a cycle, it simply turns into a loop in the resulting undirected graph. Next we select a set of clusters $\mathcal{C}_i$. These clusters can be selected so as to produce a clique tree for the graph, using any standard method for constructing such trees. We can also construct a loopy cluster graph and use generalized belief propagation. We did not discuss this topic in the thesis (for more details see Koller and Friedman (2009)). The message passing scheme described in this section is the same in both cases.

The algorithm iteratively selects an edge connecting the clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ in the cluster graph and passes the message from the former to the latter. In clique tree propagation the order in which we chose edges was basically fixed, meaning that we started from leaves to roots performing an upward pass and then going in the opposite direction. In generalized belief propagation, we might use a variety of message passing schemes. Convergence occurs when messages cease to affect the potentials which means that neighboring clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ agree on the approximate marginals over the variables from $S_{i,j}$.

Now we can generalize the algorithm for a single segment to trajectories containing multiple segments of continuous evidence. Nodelman et al. (2005) applied this algorithm separately to every segment, passing information from one segment to the next one in the form of distributions. More precisely, consider a trajectory defining a sequence of time points $t_1, \ldots, t_n$, with constant continuous evidence on every interval $[t_i, t_{i+1})$ and possible point evidence or observed transition at each t_i. Then a sequence of cluster graphs over each segment is constructed. Starting from the initial segment EP inference is run on each cluster graph using the algorithm for a single segment described above, and the distribution at the end time point of the interval is computed. The resulting distribution is then conditioned on any point evidence or the observed transition, and next used as the initial distribution for the next interval.

However, there is one subtle difficulty relating to the propagation of messages from one interval to another. If a variable X appears in two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ in a cluster graph, the distribution over its values in these two clusters is not generally the same, even if the EP computation converges. The reason is that even calibrated clusters only agree

on the projected marginals over their sepset, not the true marginals. To address this issue and to obtain a coherent distribution which can be transmitted to the next cluster graph the individual cluster marginals and sepsets for the state variables at the end time point of the previous interval are recalibrated to form a coherent distribution (the conditioning on point evidence can be done at the same time if needed). Then we can extract the new distribution as a set of calibrated cluster and sepset factors, and introduce each factor into the appropriate cluster or sepset in the cluster graph for the next time interval.

The above algorithm performs the propagation of beliefs forward in time. It is also possible to do a similar propagation backwards and pass messages in reverse, where the cluster graph for one time interval passes a message to the cluster graph for the previous one. Also to achieve more accurate beliefs we can repeat the forward-backward propagation until the entire network is calibrated, essentially treating the entire network as a single cluster graph. Note that since one cluster graph is used for each segment of fixed continuous evidence, then each cluster will approximate the trajectory of all the variables it contains as a homogeneous Markov process for the duration of the entire segment. Therefore, the choice of segments and the resulting subsets of variables, over which we compute the distribution, determine the quality of the approximation.

Chapter 4

Structure learning for Bayesian networks

Recall the Definition 2.3 of Bayesian Networks (BN), the notion of which combines the structure given by a Directed Acyclic Graph (DAG) and the probability distribution encoded by Conditional Probability Distributions (CPD). By far, in Chapter 3 we discussed the problem of finding CPDs and making the inference given the structure. In this chapter we will discuss the problem of learning the structure of Bayesian networks. In Section 4.1 we briefly review known approaches to the problem. In Section 4.2 we recall partition MCMC algorithm for learning the structure of the network, whose part concerning the division of the graph into layers will be the first step of our new method. In Sections 4.3 and 4.4 we present a novel approach to structure learning with the use of the above algorithm and LASSO approach for continuous and discrete data, respectively. Section 4.5 is dedicated to numerical results.

4.1 Problem of learning structure of Bayesian Networks

Structure learning is known to be a hard problem, especially due to the superexponential growth of the DAG space when the number of nodes is increasing. Generally speaking the literature on the structure learning can be divided into three classes: constraint-based methods, score-and-search algorithms and the dynamic programming approach (as discussed for example in Koller and Friedman (2009)), even though this division is not that strict. The contents of this section come mostly from Kuipers and Moffa (2017) and Daly et al. (2011).

Constraint-based methods use conditional independence tests to obtain information about the underlying causal structure. They start from the full undirected graph and then make decisions about removing the edge in the network based on tests of conditional independence. The widely used algorithm of this nature, PC algorithm (Spirtes et al. (2000)), and constraint-based methods in general are sensitive to the order in which they are run. However Colombo and Maathuis (2014) proposed some modifications for PC algorithm to remove either partially or altogether this dependence. These methods scale

well with the dimension but are sensitive to local errors of the independence tests which are used.

One of the most widely studied ways of learning a Bayesian network structure has been the use of so-called 'score-and-search' techniques. These algorithms comprise of:

- a search space consisting of the various allowable states, each of which represents a Bayesian network structure;
- a mechanism to encode each of the states;
- a mechanism to move from state to state in the search space;
- a scoring function assigning some score to a state in the search space which describes the goodness of fit with the sample data.

Also some hybrid methods combining ideas from both techniques were proposed, for example the max-min-hill-climbing of Tsamardinos et al. (2006).

Within the family of search and score methods we can distinguish a separate class of MCMC methods for the graph space exploration. Their main and huge advantage is that they can provide a collection of samples from the posterior distribution of the graph given the data. This means that rather than making the inference based on a single graphical model, we can account for model uncertainty by averaging over all the models in the obtained class. In particular, we can estimate the expectation of any given network feature, such as the posterior probability of an individual edge, by averaging the posterior distributions under each of the models, weighted by their posterior model probabilities (Madigan et al. (1995), Kuipers and Moffa (2017)). This is especially important in high dimensional domains with sparse data where the single best model cannot be clearly identified, so the inference relying on the best scoring model is not justified.

The first MCMC algorithm over graph structures is due to Madigan et al. (1995), later refined by Giudici and Castelo (2003). To improve on the mixing and convergence, Friedman and Koller (2001) instead suggested to build a Markov chain on the space of node orders, at the price of introducing a bias in the sampling. For smaller systems with smaller space and time complexity one of the efficient approaches is the dynamic programming (Koivisto and Sood (2004)), which can be further used to extend the proposals of standard structure MCMC approach in a hybrid method (Eaton and Murphy (2007)). Within the MCMC approach, to avoid the bias while keeping reasonable convergence rate, Grzegorczyk and Husmeier (2008) more recently proposed a new edge reversal move method combining ideas both of standard structure and order based MCMC. Recently Kuipers and Moffa (2017) presented another MCMC algorithm designed on the combinatorial structure of DAGs, with the advantage of improving convergence with respect to structure MCMC, while still providing an unbiased sample since it acts directly on the space of DAGs. Moreover, it can also be combined with the algorithm of Grzegorczyk and Husmeier (2008) to improve the convergence rate even further.

4.2 Partition MCMC method

In this section we describe the Partition MCMC algorithm of Kuipers and Moffa (2017), which will be the base of our novel method for learning the structure of BNs. This algorithm considers combinatorial representation of DAGs to build an efficient MCMC scheme directly on the space of DAGs. Its convergence is better than that of the structure MCMC and does not introduce bias as the order based MCMC. As we mentioned, the authors also proposed a way to combine their method with the new edge reversal move approach of Grzegorczyk and Husmeier (2008) and improve upon their MCMC sampler.

First we need to introduce the notion of *layers* and *partitions* for DAG. Given DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we define layers ℓ_i of the nodes (called interchangeably variables) in the network as follows:

- $\ell_0 = \{v \in \mathcal{V} : \mathbf{pa}_{\mathcal{G}}(v) = \emptyset\}$ is the layer of the nodes which do not have any parents;
- having defined the layer ℓ_i for $i = 0, 1, \ldots, k-1$ we define the next layer as

$$\ell_k = \{v \in \mathcal{V} : \exists w \in \ell_{k-1} \text{ such that } w \in \mathbf{pa}_{\mathcal{G}}(v) \text{ and } \mathbf{pa}_{\mathcal{G}}(v) \subseteq L_{k-1}\},$$

  where $L_{k-1} = \bigcup_{i \leq k-1} \ell_i$.

Note that variables from the same layer do not have arrows between them, and that each variable (except for the layer ℓ_0) has at least one arrow directed towards it from any variable from the adjacent previous layer. For instance, the graph in Figure 4.1 has three layers: $\ell_0 = \{1, 3, 5\}$, $\ell_1 = \{4\}$ and $\ell_2 = \{2\}$.

Suppose that for some arbitrary graph we have $q+1$ layers. Each layer ℓ_i has a certain amount k_i of nodes, which in sum gives the total number of nodes d, i.e. $\sum_{i=0}^{q} k_i = d$. In addition, with each layer representation there is associated a *permutation* of nodes, where we list nodes in the layer order. More precisely, first we write nodes from the first layer, then from the second one, etc. For the graph in Figure 4.1 we have the partition $\lambda = [3, 1, 1]$ and the permutation $\pi_\lambda = [1, 3, 5, 4, 2]$. Together a pair (λ, π_λ) is called *a labelled partition.*

Kuipers and Moffa (2017) proposed an efficient MCMC algorithm for exploring the space of partitions to find the most probable layer representation given the observed data. Although the full algorithm is suited for structure learning, we want to improve on this algorithm and replace the second part of it with the LASSO estimator. The authors define an MCMC algorithm on the space of node partitions avoiding in this way over-representation of certain DAGs. Compared to other MCMC methods mentioned above partition MCMC is faster than structure MCMC of Madigan et al. (1995). It is slower than order MCMC of Friedman and Koller (2001) but does not introduce any bias. The basic move consists of splitting one element of the partition (i.e. layer) into two parts or joining two adjacent elements (the authors also propose an additional move consisting of swapping two nodes in adjacent layers). All the partitions reachable from a given partition in

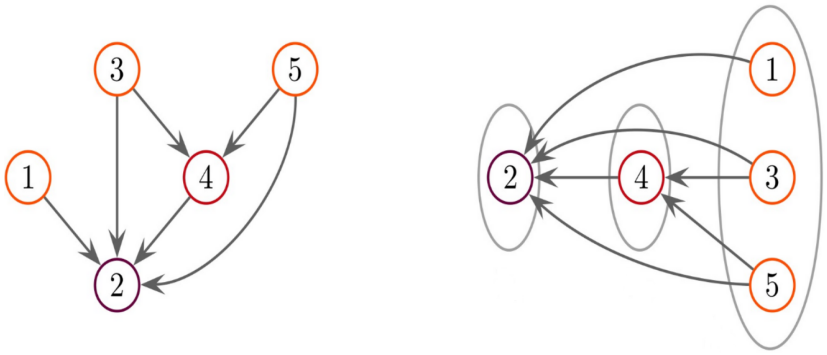


Figure 4.1: An example of partition representation of the DAG.

one basic move are called the neighbourhood of the partition. So the MCMC scheme consists of sampling a partition from the neighbourhood of the previous partition with a small probability to stay still defined by the user. The obtained partition is scored and the score coincides with the posterior probability of the labelled partition. After sampling the partition we sample a single DAG weighted according to its posterior. Then we can average the acquired DAGs in the MCMC chain and choose the model. However, we propose to change the step where we sample DAG from the posterior distribution and average DAGs from the MCMC chain. It is well suited for inference and estimation of network parameters but we believe that we can improve the Bayesian averaging approach in the case of structure learning. We propose to use partition MCMC for finding the best scoring partition and next to use it for recovering arrows with the LASSO estimator where each parameter corresponds to a certain arrow in the network.

4.3 The novel approach to structure learning

We want to combine advantages of partition MCMC and LASSO for linear models. First we find the best layer representation using partition MCMC algorithm. Next we obtain the final DAG solving d LASSO problems, where d is the number of variables (nodes). Having found the most probable layer representation for a DAG we consider two models: one for continuous data and one for discrete data.

4.3.1 Gaussian Bayesian Networks

For the continuous case we consider Gaussian Bayesian Networks (GBN) introduced in Section 3.1. We denote as X_i^m the m-th random variable in the i-th layer, where $m \in \{1, \ldots, k_i\}$. We assume that each ϵ_i^m has the normal distribution $\mathcal{N}(0, \sigma_i^m)$. We also assume that each ϵ_i^m is independent of all X_i^m. Now given the partition $[k_0, k_1, \ldots, k_q]$ we can write the problem of finding the DAG structure as a set of the following d linear

model problems:

$$
\begin{aligned}
X_0^1 &= \beta_{0,0}^1 + \epsilon_0^1 \\
&\vdots \\
X_0^{k_0} &= \beta_{0,0}^{k_0} + \epsilon_0^{k_0} \\
X_1^1 &= \beta_{1,0}^1 + \beta_{1,0}^{1,1} X_0^1 + \cdots + \beta_{1,0}^{1,k_0} X_0^{k_0} + \epsilon_1^1 \\
&\vdots \\
X_1^{k_1} &= \beta_{1,0}^{k_1} + \beta_{1,0}^{k_1,1} X_0^1 + \cdots + \beta_{1,0}^{k_1,k_0} X_0^{k_0} + \epsilon_1^{k_1} \\
X_2^1 &= \beta_{2,0}^1 + \beta_{2,0}^{1,1} X_0^1 + \cdots + \beta_{2,0}^{1,k_0} X_0^{k_0} + \beta_{2,1}^{1,1} X_1^1 + \cdots + \beta_{2,1}^{1,k_1} X_1^{k_1} + \epsilon_2^1 \\
&\vdots \\
X_q^{k_q} &= \beta_{q,0}^{k_q} + \sum_{\substack{j<q, \\ 1\le m_j \le k_j}} \beta_{q,j}^{k_q,m_j} X_j^{m_j} + \epsilon_q^{k_q}.
\end{aligned}
\tag{4.1}
$$

Then the problem of finding DAG's structure is equivalent to the problem of finding non-zero parameters $\beta_{l,i}^{m_l,m_i}$. This corresponds to starting from the full possible graph and removing non-existing edges by shrinking the parameters to 0. It is possible due to the fact that we have a partition, where we know which nodes can be parents for which nodes. This would not be possible otherwise because the graph has to be acyclic and we would have to introduce other constraints to the optimization problem. To solve this problem we will apply LASSO regression to each linear model, which tends to shrink the coefficients to 0 by penalizing those coefficients with ℓ_1-norm.

Let m be the number of observations. Let (X_{ik}) be the matrix of observations, where $i \in \{1, \ldots, m\}$ and $k \in \{1, \ldots, d\}$. Then by X^j we denote the matrix formed by the columns corresponding to the j-th layer. Similarly, by $X^{0:(j-1)}$ we denote the matrix which consists of the columns of the matrix of observations corresponding to the first j layers. By $X^j[i]$ we denote the column of the matrix X^j corresponding to the i-th variable from the j-th layer and by X_i we denote the row of the matrix (X_{ik}) corresponding to the i-th observation. Moreover, let $\beta_j^i = [\beta_{j,1}^{i,1}, \ldots, \beta_{j,1}^{i,k_0}, \beta_{j,2}^{i,1}, \ldots, \beta_{j,j-1}^{i,k_{j-1}}]$, where $j = \{1, \ldots, q\}$ and $i = \{1, \ldots, k_j\}$. To find the required vectors β_j^i we solve the following d optimization problems

$$
\hat{\beta}_j^i = \underset{\theta \in \mathbb{R}^{k_0 + \cdots + k_{j-1}}}{\operatorname{argmin}} [RSS_{j,i}(\theta) + \lambda_{j,i} \|\theta\|_1], \tag{4.2}
$$

where $RSS_{j,i}(\theta) = 1/2 \|X^j[i] - \theta^\top X^{0:(j-1)}\|_2^2$ is a residual sum of squares for the i-th variable in the j-th layer and $\|\theta\|_1 = \sum_{l=0}^{j-1} \sum_{m_l=1}^{k_l} |\theta_l^{m_l}|$ is the l_1-norm of θ. Note, that θ depends both on j and i. The tuning parameters $\lambda_{j,i} > 0$ balance the minimization of the cost function and the penalty function. The form of the penalty is crucial, because its singularity at the origin implies that some coordinates of the minimizer $\hat{\beta}_j^i$ are exactly equal to 0 if $\lambda_{j,i}$ is sufficiently large. Thus, starting from the graph with all possible arrows for the given layer representation (i.e. there are arrows from variables on each

layer towards all the variables in next layers) we remove irrelevant edges. The functions $RSS_{j,i}(\theta)$ and the penalty are convex, so (4.2) is a convex minimization problem. This is an important fact from both practical and theoretical points of view.

4.3.2 Theoretical results for GBNs

By $S_{j,i}$ we denote the support of the true vectors of parameters β^i_j, i.e. the sets of non-zero coordinates of each β^i_j, and by $S = \{S_{1,1}, \ldots, S_{1,k_1}, S_{2,1}, \ldots, S_{q,k_q}\}$. Moreover, $\beta^i_{j,\min}$ is the smallest in the absolute value element of β^i_j restricted to $S_{j,i}$. The set $S^c_{j,i}$ denotes the complement of $S_{j,i}$, that is the set of zero coordinates of β^i_j. For any vector a we denote its l_∞-norm by $\|a\|_\infty = \max_k |a_k|$. For a vector a and a subset of indices I by a_I we denote the vector a restricted to its coordinates from the set I, i.e. $(a_I)_i = a_i$ for $i \in I$ and $(a_I)_i = 0$ otherwise. Moreover, $|I|$ denotes the number of elements of I. For a vector $a = (a_1, \ldots, a_n)$ by $Cov(a)$ we denote the matrix (c_{ij}), where $c_{ii} = Var(a_i)$ and $c_{ij} = Cov(a_i, a_j)$.

Before we state the main results of this chapter we introduce the cone invertibility factor (CIF), which plays an important role in the theoretical analysis of the properties of LASSO estimators. In literature there are three related notions which are commonly used in said analysis and help to provide some constraints on the optimized function so that the estimator is "good" in certain sense. These notions are the cone invertibility factor, compatibility factor and restricted eigenvalue (see Huang et al. (2013) and references therein). For any $\xi > 1$ we define the cones $\mathcal{C}(\xi, S_{j,i}) = \left\{\theta : \|\theta_{S^c_{j,i}}\|_1 \leq \xi \|\theta_{S_{j,i}}\|_1\right\}$. Then CIF is defined as

$$\bar{F}_{j,i}(\xi) = \inf_{0 \neq \theta \in \mathcal{C}(\xi, S_{j,i})} \frac{\|\Sigma^j \theta\|_\infty}{\|\theta\|_\infty}, \tag{4.3}$$

where Σ^j is the covariance matrix for a random vector $(X_1, \ldots, X_{j-1})$ of variables from the first j layers. More precisely,

$$\Sigma^j = \frac{1}{m} \left(X^{0:(j-1)}\right)^\top X^{0:(j-1)}.$$

Our goal will be to show that the estimators $\hat{\beta}^i_j$ are close to the true vectors β^i_j in a certain sense. However, if the curvature of the function in (4.2) around β^i_j is relatively small, then the closeness between its values at $\hat{\beta}^i_j$ and β^i_j does not necessarily imply the closeness between the arguments $\hat{\beta}^i_j$ and β^i_j. Hence, we require some additional conditions, for instance, strong convexity of $RSS_{j,i}$ at β^i_j, i.e. that the smallest eigenvalue of its Hessian is positive. In the high-dimensional case it is too strong of an assumption, therefore one usually considers restricted strong convexity or restricted smallest eigenvalues, where "restricted" means that we take infimum over $\mathcal{C}(\xi, S_{j,i})$ instead of the whole space. CIF (4.3) is an example of such reasoning. We also introduce a non-random version $F_{j,i}(\xi)$ of CIF for each $j \in \{1, \ldots, q\}$ as follows. First we define

$$H^j = Cov(X_1[0 : (j-1)]),$$

where $X_1[0:(j-1)]$ denotes the restriction of X_1 to variables from the first j layers. We assume that each H_j is positive definite and elements on the diagonal are equal to 1, i.e. $H^j_{ii} = 1$, where $i \in \{1, 2, \ldots, k_1 + \cdots + k_{j-1}\}$. Then

$$F_{j,i}(\xi) = \inf_{0 \neq \theta \in \mathcal{C}(\xi, S_{j,i})} \frac{\|H^j \theta\|_\infty}{\|\theta\|_\infty}. \tag{4.4}$$

Since in a Gaussian Bayesian Network the joint probability of all variables is assumed to be Gaussian, then each marginal is Gaussian as well. Hence, for simplicity we can bound the variance for each variable by the same constant τ^2. Also we denote

$$m_{j,i} = \frac{|S|^2 \tau^4 (1+\xi)^2 \log(|L_{j-1}|^2 q k_j/\varepsilon)}{F^2_{j,i}(\xi)} \tag{4.5}$$

for each $j \in \{1, \ldots, q\}$ and $i \in \{1, \ldots, k_j\}$.

Theorem 4.1. *Fix arbitrary $\varepsilon \in (0,1)$ and $\xi > 1$. Assume that $F_{j,i}(\xi)$ defined in (4.4) is positive for each $j \in \{1, \ldots, q\}$ and $i \in \{1, \ldots, k_j\}$. In addition suppose that*

$$m \geq K_1 \max_{j,i} m_{j,i} \tag{4.6}$$

and for each i and j we have

$$\lambda_{j,i} \geq K_2 \frac{\xi+1}{\xi-1} \tau \sigma^i_j \sqrt{\frac{\log(|L_{j-1}| q k_j/\varepsilon)}{m_{j,i}}}$$

for some universal constants K_1 and K_2. Then

$$\mathbb{P}\left(\|\hat{\beta} - \beta\|_\infty \leq \frac{4\xi}{\xi+1} \max_{j,i} \frac{\lambda_{j,i}}{F_{j,i}} \right) \geq 1 - \varepsilon.$$

The second main result is about thresholded version of LASSO estimator. It will be proved after the proof of Theorem 4.1. Consider the Thresholded LASSO estimator with the sets of nonzero coordinates $\hat{S}_{j,i}$. The set $\hat{S}_{j,i}$ contains only those coefficients of the LASSO estimator (4.2), which are larger in the absolute value than some pre-specified threshold $\delta_{j,i}$ for each $j \in \{1, \ldots, q\}$ and $i \in \{1, \ldots, k_j\}$. We denote $\{\hat{S}_{1,1}, \ldots, \hat{S}_{1,k_1}, \hat{S}_{2,1}, \ldots, \hat{S}_{q,k_q}\}$ as $\hat{S}_\delta$.

Corollary 4.2. *Suppose that assumptions of Theorem 4.1 are satisfied. If for each j, i and arbitrary $\xi > 1$ we have $\beta^i_{j,min}/2 > \delta_{j,i} \geq \dfrac{4\xi\lambda_{j,i}}{(\xi+1)F_{j,i}}$, then Thresholded LASSO with $\delta = [\delta_{1,1}, \ldots, \delta_{q,k_q}]$ is consistent, i.e.*

$$P\left(\hat{S}_\delta = S\right) \geq 1 - \varepsilon\,.$$

Before the proof of Theorem 4.1 we state and prove an auxiliary result Proposition 4.3, which is interesting on its own. It describes a slightly more general case and it will be used multiple times for different numbers and sets of predictors and targets in order to prove

Theorem 4.1. Moreover, to avoid any confusion with indices and notation introduced before for convenience we use more general notation in subsequent proofs.

Hence, let $(Y_1, Z_1), \dots, (Y_m, Z_m)$ be i.i.d. random vectors such that $Y_i \in \mathbb{R}^p$ and $Z_i \in \mathbb{R}$. The coordinates of Y_i will be denoted by Y_{ij} for each $j = \{1, \dots, p\}$ and by Y we denote the full $(m \times p)-$matrix of predictors $Y = (Y_1, \dots, Y_m)^\top$. Moreover, let $H = Cov(Y_1)$ is a positive definite matrix with diagonal elements $H_{jj} = 1$ for $j = 1, \dots, p$. We assume that

$$Z_i = \beta^\top Y_i + \varepsilon_i, \quad i = 1, \dots, m, \tag{4.7}$$

where $\varepsilon_1, \dots, \varepsilon_m$ are i.i.d. random variables with $\mathbb{E}\varepsilon_i = 0$, which are subgaussian with the parameter σ^2, and are independent of p predictors $Y_i, \dots, Y_p$. Subgaussianity means that for each i and $a \in \mathbb{R}$

$$\mathbb{E}\exp(a\varepsilon_i) \le \exp(a^2\sigma^2/2).$$

We also assume that predictors are subgaussian with the parameter τ^2, i.e. $\mathbb{E}\exp(aY_{1j}) \le \exp(a^2\tau^2/2)$ for each $j = 1, \dots, p$.

The goal is to find the set of indices of the relevant predictors

$$S = \{j \in \{1, \dots, p\} : \beta_j \neq 0\}. \tag{4.8}$$

The set S^c denotes the complement of S, that is the set of zero coordinates of β. Now consider the LASSO estimator

$$\hat{\beta} = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}}[RSS(\theta) + \lambda\|\theta\|_1], \tag{4.9}$$

where

$$RSS(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(Z_i - \theta^\top Y_i\right)^2.$$

For any $\xi > 1$ we define the cone $\mathcal{C}(\xi, S) = \{\theta : \|\theta_{S^c}\|_1 \le \xi\|\theta_S\|_1\}$. Then CIF is defined as

$$\bar{F}(\xi) = \inf_{0 \neq \theta \in \mathcal{C}(\xi, S)} \frac{\|Y^\top Y\theta/m\|_\infty}{\|\theta\|_\infty},$$

and its non-random version is given by

$$F(\xi) = \inf_{0 \neq \theta \in \mathcal{C}(\xi, S)} \frac{\|H\theta\|_\infty}{\|\theta\|_\infty}.$$

Proposition 4.3. *Fix arbitrary $a \in (0, 1)$ and $\xi > 1$. Suppose that $F(\xi)$ is positive and*

$$m \ge \frac{K_1|S|^2\tau^4(1+\xi)^2\log(p^2/a)}{F^2(\xi)} \tag{4.10}$$

and

$$\lambda \ge K_2\frac{\xi+1}{\xi-1}\tau\sigma\sqrt{\frac{\log(p/a)}{m}}, \tag{4.11}$$

where K_1, K_2 are some universal constants. Then

$$\mathbb{P}\left(\|\hat{\beta} - \beta\|_\infty \le \frac{4\xi\lambda}{(\xi+1)F(\xi)}\right) > 1 - 2a. \tag{4.12}$$

The proof of Proposition 4.3 relies on Lemma 4.4 and 4.6 below.

Lemma 4.4. *In the context of previously defined random variables Y_{ij} and ε_i, where $i = \{1, \dots, m\}$, for arbitrary $j = 1, \dots, p$ and $u > 0$ we have*

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} Y_{ij}\varepsilon_i > 2\tau\sigma\left(2\sqrt{\frac{2u}{n}} + \frac{u}{n}\right)\right) \leq \exp(-u).$$

The proof of Lemma 4.4 uses the following Corollary 8.2 of van de Geer (2016).

Lemma 4.5. *Suppose that $Z_1, \dots, Z_n$ are i.i.d. random variables and there exists $L > 0$ such that $C^2 = \mathbb{E}\exp(|Z_1|/L)$ is finite. Then for arbitrary $u > 0$*

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}Z_i) > 2L\left(C\sqrt{\frac{2u}{n}} + \frac{u}{n}\right)\right) \leq \exp(-u).$$

Proof of Lemma 4.4. Fix $j = 1, \dots, p$ and $u > 0$. We consider an average of i.i.d. centered random variables $Z_j = Y_{ij}\varepsilon_i$ with $\mathbb{E}Z_j = 0$, so we can use Lemma 4.5. We need to find $L, C > 0$ such that $\mathbb{E}\exp(|Y_{1j}\varepsilon_1|/L) \leq C^2$. Note that

$$\mathbb{E}\exp(|Y_{1j}\varepsilon_1|/L) \leq \mathbb{E}\exp(Y_{1j}\varepsilon_1/L) + \mathbb{E}\exp(-Y_{1j}\varepsilon_1/L). \tag{4.13}$$

For the first term on the right-hand side of (4.13) we have

$$\mathbb{E}\exp(Y_{1j}\varepsilon_1/L) = \mathbb{E}\left[\mathbb{E}\left(\exp(Y_{1j}\varepsilon_1/L) \mid Y_{1j}\right)\right].$$

Using independence of Y_{1j} and ε_1, and subgaussianity of ε_1 for each $y \in \mathbb{R}$ we obtain

$$\mathbb{E}\left[\exp(Y_{1j}\varepsilon_1/L) \mid Y_{1j} = y\right] = \mathbb{E}\exp(y\varepsilon_1/L) \leq \exp\left(y^2\sigma^2/(2L^2)\right).$$

Therefore we have

$$\mathbb{E}\left[\mathbb{E}\left(\exp(Y_{1j}\varepsilon_1/L)|Y_{1j}\right)\right] \leq \mathbb{E}\exp\left(Y_{1j}^2\sigma^2/(2L^2)\right),$$

which, using subgaussianity of Y_{1j} and Lemma 7.4 of Baraniuk et al. (2011), we can bound from above by

$$\frac{1}{\sqrt{(1 - \tau^2\sigma^2/L^2)}},$$

provided that $L > \tau\sigma$. The second expectation on the right-hand side of (4.13) can be bounded analogously, hence, we obtain

$$\mathbb{E}\exp(|Y_{1j}\varepsilon_1|/L) \leq \frac{2}{\sqrt{(1 - \tau^2\sigma^2/L^2)}},$$

provided that $L > \tau\sigma$. We can take $L = 2\tau\sigma$ and obtain $C \geq \frac{2}{\sqrt[4]{3}}$, which finishes the proof. □

Lemma 4.6. *Suppose that assumptions of Proposition 4.3 are satisfied. Then for arbitrary $\varepsilon \in (0, 1)$ and $\xi > 1$ with probability at least $1 - \varepsilon$ we have $\bar{F}(\xi) \geq F(\xi)/2$.*

Proof. Fix $\varepsilon \in (0,1)$ and $\xi > 1$. We start with considering the l_∞-norm of the matrix

$$\left\| \frac{1}{m} Y^\top Y - \mathbb{E} Y_1^\top Y_1 \right\|_\infty = \max_{j,k=1,\dots,p} \left| \frac{1}{m} \sum_{i=1}^m Y_{ij} Y_{ik} - \mathbb{E} Y_{1j} Y_{1k} \right|.$$

Fix $j, k \in \{1, \dots, p\}$. Notice that for any two numbers a and b we have the inequality $ab \le \frac{a^2}{2} + \frac{b^2}{2}$. Hence, we can write

$$|Y_{1j} Y_{1k}| \le \frac{Y_{1j}^2}{2} + \frac{Y_{1k}^2}{2}.$$

Therefore, first using the previous inequality and Cauchy-Schwarz inequality afterwards for any positive constant L we obtain

$$\begin{aligned} \mathbb{E} \exp\left(|Y_{1j} Y_{1k}|/L\right) &\le \mathbb{E} \exp\left(Y_{1j}^2/(2L)\right) \exp\left(Y_{1k}^2/(2L)\right) \\ &\le \sqrt{\mathbb{E} \exp\left(Y_{1j}^2/L\right) \mathbb{E} \exp\left(Y_{1k}^2/L\right)}. \end{aligned} \tag{4.14}$$

The variable Y_{1j} is subgaussian, so using Lemma 7.4 of Baraniuk et al. (2011) we can bound the first expectation under the square root in (4.14) from above by $\left(1 - \frac{2\tau^2}{L}\right)^{-1/2}$, provided that $2\tau^2 < L$. The second expectation under the square root in (4.14) can be bounded by the same value when we use the subgaussianity of Y_{1k}. Therefore,

$$\mathbb{E} \exp\left(|Y_{1j} Y_{1k}|/L\right) \le \left(1 - \frac{2\tau^2}{L}\right)^{-1/2},$$

provided that $2\tau^2 < L$. Applying Lemma 4.5 with $L = 3\tau^2$ and $C = 2$ and $u = \log(p^2/\varepsilon)$ we obtain

$$\mathbb{P}\left(\left| \frac{1}{m} \sum_{i=1}^m Y_{ij} Y_{ik} - \mathbb{E} Y_{1j} Y_{1k} \right| > K\tau^2 \sqrt{\frac{\log(p^2/\varepsilon)}{m}} \right) \le \frac{\varepsilon}{p^2},$$

where K is an universal constant. Therefore,

$$\begin{aligned} &\mathbb{P}\left(\left| \frac{1}{m} Y^\top Y - \mathbb{E} Y_1^\top Y_1 \right|_\infty > K\tau^2 \sqrt{\frac{\log(p^2/\varepsilon)}{m}} \right) = \\ &= \mathbb{P}\left(\max_{j,k=1,\dots,p} \left| \frac{1}{m} \sum_{i=1}^m Y_{ij} Y_{ik} - \mathbb{E} Y_{1j} Y_{1k} \right| > K\tau^2 \sqrt{\frac{\log(p^2/\varepsilon)}{m}} \right) \le \\ &\le \sum_{j,k} \mathbb{P}\left(\left| \frac{1}{m} \sum_{i=1}^m Y_{ij} Y_{ik} - \mathbb{E} Y_{1j} Y_{1k} \right| > K\tau^2 \sqrt{\frac{\log(p^2/\varepsilon)}{m}} \right) \le \varepsilon. \end{aligned} \tag{4.15}$$

Proceeding similarly to the proof of Lemma 4.1 of Huang et al. (2013) we first obtain

$$\begin{aligned} &\left| \left\| \frac{1}{m} Y^\top Y \theta \right\|_\infty - \|H\theta\|_\infty \right| \le \left\| \frac{1}{m} Y^\top Y \theta - H\theta \right\|_\infty \le \left\| \frac{1}{m} Y^\top Y - H \right\|_\infty \|\theta\|_1 = \\ &= \left\| \frac{1}{m} Y^\top Y - H \right\|_\infty (\|\theta_S\|_1 + \|\theta_{S^c}\|_1) \le (1+\xi)|S| \cdot \|\theta\|_\infty \left\| \frac{1}{m} Y^\top Y - \mathbb{E} Y_1^\top Y_1 \right\|_\infty. \end{aligned}$$

This implies that

$$\left\|\frac{1}{m}Y^\top Y\theta\right\|_\infty \geq \|H\theta\|_\infty - (1+\xi)|S|\cdot\|\theta\|_\infty \left\|\frac{1}{m}Y^\top Y - \mathbb{E}Y_1^\top Y_1\right\|_\infty .$$

Then by dividing both sides by $\|\theta\|_\infty$, taking infimum with respect to θ over the cone $\mathcal{C}(\xi, S)$ and using (4.15) we derive that

$$\bar{F}(\xi) \geq F(\xi) - K(1+\xi)|S|\tau^2\sqrt{\frac{\log(p^2/\varepsilon)}{m}}$$

with probability higher than $1-\varepsilon$. Finally, using (4.10) we have

$$\bar{F}(\xi) \geq F(\xi) - \frac{K}{\sqrt{K_1}}F(\xi) = \left(1 - \frac{K}{\sqrt{K_1}}\right)F(\xi).$$

We finish the proof by taking sufficiently large K_1. □

Proof of Proposition 4.3. The central part of the proof is to show that

$$\mathbb{P}\left(\|\hat{\beta}-\beta\|_\infty \leq \frac{2\xi\lambda}{(\xi+1)\bar{F}(\xi)}\right) > 1-a. \tag{4.16}$$

Let us denote $\Omega = \{\|\nabla RSS(\beta)\|_\infty \leq \frac{\xi-1}{\xi+1}\lambda\}$. Now we want to bound from below the probability of Ω. For each $j = 1, \ldots, p$ we can calculate j-th partial derivative of $RSS(\theta)$ at true β

$$\nabla_j RSS(\beta) = \frac{\partial RSS}{\partial\theta_j}(\beta) = \frac{1}{m}\sum_{i=1}^m Y_{ij}\epsilon_i$$

and we bound it from above with high probability using Lemma 4.4. Therefore, taking (4.11) into account we have

$$\begin{aligned}
\mathbb{P}(\Omega) &= \mathbb{P}\left(\max_j|\nabla_j RSS(\beta)| \leq \frac{\xi-1}{\xi+1}\lambda\right) = \mathbb{P}\left(\bigcap_{j=1}^p\left\{|\nabla_j RSS(\beta)| \leq \frac{\xi-1}{\xi+1}\lambda\right\}\right) = \\
&= 1 - \mathbb{P}\left(\bigcup_{j=1}^p\left\{|\nabla_j RSS(\beta)| > \frac{\xi-1}{\xi+1}\lambda\right\}\right) \geq 1 - \sum_{j=1}^p \mathbb{P}\left(|\nabla_j RSS(\beta)| > \frac{\xi-1}{\xi+1}\lambda\right) \geq \\
&\geq 1 - \sum_{j=1}^p \mathbb{P}\left(|\nabla_j RSS(\beta)| > K_2\tau\sigma\sqrt{\frac{\log(p/a)}{m}}\right).
\end{aligned}$$

Now applying Lemma 4.4 with $u = \log(p/2a)$ and appropriately chosen K_2 we bound from below this probability by $1-a$.

In further argumentation we consider only the event Ω. Besides, we denote $\tilde{\beta} = \hat{\beta} - \beta$ where $\hat{\beta}$ is a minimizer of a convex function given in (4.9), which is equivalent to

$$\begin{cases}
\dfrac{\partial RSS}{\partial\theta_j}(\hat{\beta}) = -\lambda\ \mathrm{sgn}(\hat{\beta}_j), & \text{if } \hat{\beta}_j \neq 0, \\
\left|\dfrac{\partial RSS}{\partial\theta_j}(\hat{\beta})\right| \leq \lambda, & \text{if } \hat{\beta}_j = 0,
\end{cases} \tag{4.17}$$

where $j = 1, \ldots, p$. Next we show that $\tilde{\beta} \in \mathcal{C}(\xi, S)$. Our argumentation is analogous to Ye and Zhang (2010). From conditions in (4.17) and the fact that $\|\tilde{\beta}\|_1 = \|\tilde{\beta}_S\|_1 + \|\tilde{\beta}_{S^c}\|_1$ we obtain

$$\begin{aligned}
0 \leq \tilde{\beta}^\top Y^\top Y \tilde{\beta}/m = \tilde{\beta}^\top \Big[\nabla RSS(\hat{\beta}) - \nabla RSS(\beta)\Big] = \\
= \sum_{j \in S} \tilde{\beta}_j \nabla_j RSS(\hat{\beta}) + \sum_{j \in S^c} \hat{\beta}_j \nabla_j RSS(\hat{\beta}) - \tilde{\beta}^\top \nabla RSS(\beta) \leq \\
\leq \lambda \sum_{j \in S} |\tilde{\beta}_j| - \lambda \sum_{j \in S^c} |\hat{\beta}_j| + \|\tilde{\beta}\|_1 \|\nabla RSS(\beta)\|_\infty = \\
= [\lambda + \|\nabla RSS(\beta)\|_\infty] \|\tilde{\beta}_S\|_1 + [\|\nabla RSS(\beta)\|_\infty - \lambda] \|\tilde{\beta}_{S^c}\|_1.
\end{aligned}$$

Since we exclusively consider the event Ω, we obtain the following inequality

$$\|\tilde{\beta}_{S^c}\|_1 \leq \frac{\lambda + \|\nabla RSS(\beta)\|_\infty}{\lambda - \|\nabla RSS(\beta)\|_\infty} \|\tilde{\beta}_S\|_1 \leq \xi \|\tilde{\beta}_S\|_1.$$

Hence, we have just proved that $\tilde{\beta}$ belongs to the cone $\mathcal{C}(\xi, S)$. Therefore from the definition of $\bar{F}(\xi)$ we have

$$\|\hat{\beta} - \beta\|_\infty \leq \frac{\|Y^\top Y(\hat{\beta} - \beta)/m\|_\infty}{\bar{F}(\xi)} \leq \frac{\|\nabla RSS(\hat{\beta})\|_\infty + \|\nabla RSS(\beta)\|_\infty}{\bar{F}(\xi)}.$$

Using the second condition in (4.17) and the definition of the event Ω we then obtain (4.16). Finally, having shown (4.16), we apply Lemma 4.6 and obtain (4.12) which finishes the proof. □

Proof of Theorem 4.1. In order to show that our estimator is close to the true parameter vector β we first use union bounds. So here we have

$$\begin{aligned}
\mathbb{P}\left(\|\hat{\beta} - \beta\|_\infty \leq \frac{4\xi}{(\xi+1)} \max_{j,i} \frac{\lambda_{j,i}}{F_{j,i}}\right) \geq \mathbb{P}\left(\bigcap_{j,i} \left\{\|\hat{\beta}_j^i - \beta_j^i\|_\infty \leq \frac{4\xi\lambda_{j,i}}{(\xi+1)F_{j,i}}\right\}\right) = \\
= 1 - \mathbb{P}\left(\bigcup_{j,i} \left\{\|\hat{\beta}_j^i - \beta_j^i\|_\infty > \frac{4\xi\lambda_{j,i}}{(\xi+1)F_{j,i}}\right\}\right) \geq \\
\geq 1 - \sum_{j=1}^{q} \sum_{i=1}^{k_j} \mathbb{P}\left(\|\hat{\beta}_j^i - \beta_j^i\|_\infty > \frac{4\xi\lambda_{j,i}}{(\xi+1)F_{j,i}}\right).
\end{aligned}$$

Then, using Proposition 4.3 separately for each variable X_j^i in each layer ℓ_j for $j = 1 \ldots, q$ with $\lambda = \lambda_{j,i}$, with the number of predictors equal to $p = |L_{j-1}|$ and $a = \dfrac{\varepsilon}{qk_j}$ we obtain that the expression above can be bounded from below by $1 - \varepsilon$. The bound on the number of observations m is chosen according to (4.5) and (4.6). □

To prove Corollary 4.2 we apply the same methodology. Namely, we prove an auxiliary lemma concerning the model described by (4.7), so the set S is defined by (4.8). Additionally, by $\beta_{\min}$ we denote the smallest in the absolute value non-zero coordinate of the true parameter vector β. By $\hat{S}$ we denote the set of non-zero coordinates of the Thresholded LASSO estimator with the level δ, i.e. the coordinates of the vector $\hat{\beta}$, which are greater than δ.

Lemma 4.7. *Fix $a \in (0,1)$ and $\xi > 1$. Then under the assumptions of Proposition 4.3 and*

$$\frac{4\xi\lambda}{(\xi+1)F(\xi)} \le \delta \le \beta_{\min}/2 \tag{4.18}$$

we have

$$\mathbb{P}\left(\hat{S} = S\right) \ge 1 - a\,.$$

Proof. Take any $j \notin S$. Then from Proposition 4.3 and (4.18) with the probability greater than $1-a$ we have

$$|\hat{\beta}_j| = |\hat{\beta}_j - \beta_j| \le \|\hat{\beta} - \beta\|_\infty \le \delta.$$

Therefore, the j-th coordinate of Thresholded LASSO $\hat{\beta}_j^{\mathrm{TH}} = 0$. Next, we take $j \in S$ and obtain, also from Proposition 4.3 and (4.18), that with probability greater than $1-a$

$$|\hat{\beta}_j| \ge |\beta_j| - |\hat{\beta}_j - \beta_j| \ge \beta_{\min} - \|\hat{\beta} - \beta\|_\infty \ge \delta.$$

Hence, $\hat{\beta}_j^{\mathrm{TH}} \neq 0$. □

Proof of Corollary 4.2. From Lemma 4.7 for each $j \in \{1,\dots,q\}$ and $i = \{1,\dots,k_j\}$ under the assumptions of Theorem 4.1 we have that for arbitrary $a_{j,i} \in (0,1)$

$$\mathbb{P}\left(\hat{S}_{j,i} \neq S_{j,i}\right) < a_{j,i}.$$

Now we obtain

$$\mathbb{P}\left(\hat{S}_\delta \neq S\right) = \mathbb{P}\left(\bigcup_{j,i}\{\hat{S}_{j,i} \neq S_{j,i}\}\right) \le \sum_{j=1}^{q}\sum_{i=1}^{k_j}\mathbb{P}\left(\hat{S}_{j,i} \neq S_{j,i}\right).$$

By taking $a_{j,i} = \dfrac{\varepsilon}{qk_j}$ we obtain the bound $\mathbb{P}(\hat{S}_\delta \neq S) < \varepsilon$ and finish the proof. □

4.4 Discrete case

As we discussed in Section 3.1 in the discrete case as the distribution of the model we take a collection of categorical distributions for each variable. First we assume a binary case so that each $X_i \in \{0,1\}$ and we consider the *logistic regression* model. Let us denote the *sigmoid* function as $\sigma(x) = \dfrac{1}{1+e^{-x}}$. In this setting we can write probabilities for each variable in each layer similar to (4.1) as follows

$$\begin{aligned}
&\mathbb{P}(X_1^1 = 1) = \sigma(\beta_{1,0}^1 + \beta_{1,0}^{1,1}X_0^1 + \dots + \beta_{1,0}^{1,k_0}X_0^{k_0})\\
&\quad\vdots\\
&\mathbb{P}(X_1^{k_1} = 1) = \sigma(\beta_{1,0}^{k_1} + \beta_{1,0}^{k_1,1}X_0^1 + \dots + \beta_{1,0}^{k_1,k_0}X_0^{k_0})\\
&\mathbb{P}(X_2^1 = 1) = \sigma(\beta_{2,0}^1 + \beta_{2,0}^{1,1}X_0^1 + \dots + \beta_{2,0}^{1,k_0}X_0^{k_0} + \beta_{2,1}^{1,1}X_1^1 + \dots + \beta_{2,1}^{1,k_1}X_1^{k_1})\\
&\quad\vdots\\
&\mathbb{P}(X_q^{k_q} = 1) = \sigma(\beta_{q,0}^{k_q} + \sum \beta_{q,j}^{k_q,m_j}X_j^{m_j}).
\end{aligned} \tag{4.19}$$

Using the same notation as for the continuous case we need to solve the following d optimization problems

$$\hat{\beta}_j^i = \underset{\theta \in \mathbb{R}^{k_0+\cdots+k_{j-1}}}{\operatorname{argmin}} [\ell_{j,i}(\theta) + \lambda_{j,i}\|\theta\|_1], \quad j = 1, \ldots, q, \quad i = 1, \ldots, k_j,$$

where $\ell_{j,i}$ is the negative log-likelihood for the i-th variable in the j-th layer and has the following form

$$\ell_{j,i}(\theta) = -\sum_{l=1}^{m} X_{(p+i)l} \log\left[\sigma\left(\theta_j^{i\top} X^{0:(j-1)}\right)\right] + (1 - X_{(p+i)l}) \log\left[1 - \sigma\left(\theta_j^{i\top} X^{0:(j-1)}\right)\right].$$

Here we denote by $p = p(j) = k_0 + \cdots + k_{j-1}$ the number of variables in the previous $j-1$ layers.

We can also generalize the above case to the case where each variable has a discrete and finite state space, namely each $X_j^i \in \{1, \ldots, N_j^i\}$. Now instead of the sigmoid function we use the so-called *softmax* function. For any vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ we define the softmax function $\sigma(\boldsymbol{a})$ as the vector $\sigma(\boldsymbol{a}) = (\sigma(\boldsymbol{a})[1], \ldots, \sigma(\boldsymbol{a})[n])$, where $\sigma(\boldsymbol{a})[i] = \dfrac{\exp(a_i)}{\sum_{j=1}^n \exp(a_j)}$. We denote as $\mathbf{X}_j = (X_0^1, X_0^2, \ldots, X_0^{k_0}, X_1^1, \ldots, X_{j-1}^{k_{j-1}})^\top$ for $j = 1, \ldots, q$. Also we denote the vectors of parameters corresponding to the l-th class of the i-th variable in the j-th layer as

$$\beta_j^i[l] = (\beta_{j,0}^{i,1}[l], \ldots, \beta_{j,0}^{i,k_0}[l], \beta_{j,1}^{i,1}[l], \ldots, \beta_{j,j-1}^{i,k_{j-1}}[l])$$

for $j = 0, \ldots, q-1$, $i = 1, \ldots, k_j$ and $l = 1, \ldots, N_j^i$. Then the model analogous to the logistic model in (4.19) takes the form

$$\begin{aligned}
&\mathbb{P}(X_1^1 = 1) = \sigma(\beta_{1,0}^1[1] + \beta_1^1[1]\mathbf{X}_1, \ldots, \beta_{1,0}^1[N_1^1] + \beta_1^1[N_1^1]\mathbf{X}_1)[1] \\
&\quad\vdots \\
&\mathbb{P}(X_1^1 = N_1^1) = \sigma(\beta_{1,0}^1[1] + \beta_1^1[1]\mathbf{X}_1, \ldots, \beta_{1,0}^1[N_1^1] + \beta_1^1[N_1^1]\mathbf{X}_1)[N_1^1] \\
&\quad\vdots \\
&\mathbb{P}(X_1^{k_1} = 1) = \sigma(\beta_{1,0}^{k_1}[1] + \beta_1^{k_1}[1]\mathbf{X}_1, \ldots, \beta_{1,0}^{k_1}[N_1^{k_1}] + \beta_1^{k_1}[N_1^{k_1}]\mathbf{X}_1)[1] \\
&\quad\vdots \\
&\mathbb{P}(X_j^i = l) = \sigma(\beta_{j,0}^i[1] + \beta_j^i[1]\mathbf{X}_j, \ldots, \beta_{j,0}^i[N_j^i] + \beta_j^i[N_j^i]\mathbf{X}_j)[l] \\
&\quad\vdots \\
&\mathbb{P}(X_q^{k_q} = N_q^{k_q}) = \sigma(\beta_{q,0}^{k_q}[1] + \beta_q^{k_q}[1]\mathbf{X}_q, \ldots, \beta_{q,0}^{k_q}[N_q^{k_q}] + \beta_q^{k_q}[N_q^{k_q}]\mathbf{X}_q)[N_q^{k_q}].
\end{aligned}$$

This is called *multinomial logistic regression*. It is not difficult to notice that logistic regression is a particular case of multinomial logistic regression with two possible classes. For each variable X_j^i we denote the full vector of parameters $\beta_j^i = (\beta_j^i[1], \ldots, \beta_j^i[N_j^i])$. Then we need to solve d optimization problems analogous to the case of logistic regression

$$\hat{\beta}_j^i = \underset{\theta \in \mathbb{R}^{(k_0+\cdots+k_{j-1})N_j^i}}{\operatorname{argmin}} [\ell_{j,i}(\theta) + \lambda_{j,i}\|\theta\|_1], \quad j = 1, \ldots, q, \quad i = 1, \ldots, k_j,$$

where $\ell_{j,i}$ is also the negative log-likelihood for the i-th variable in the j-th layer and in this case has the following form

$$\ell_{j,i}(\theta) = -\sum_{l=1}^{m}\sum_{k=1}^{N_j^i} \mathbb{I}(X_{(p+i)l} = k)\left[\theta_j^i[l]X^{0:(j-1)} - \log\left(\sum_{l=1}^{N_j^i}\theta_j^i[l]X^{0:(j-1)}\right)\right],$$

where we again denoted by $p = p(j) = k_0 + \cdots + k_{j-1}$ the number of variables in the previous $j-1$ layers.

4.5 Numerical results

In this section we describe the details of algorithm implementation as well as the results of experimental studies comparing our algorithm to others.

4.5.1 Details of implementation

We provide in details practical implementation of the proposed algorithm. The solution of (4.2) depends on the choice of $\lambda_{j,i}$. Finding the „optimal" parameters $\lambda_{j,i}$ and the thresholds $\delta_{j,i}$ in practice is difficult. We solve it using the information criteria (Xue et al., 2012; Pokarowski and Mielniczuk, 2015; Miasojedow and Rejchel, 2018).

First, recall the function which is being minimized in (4.2)

$$RSS_{j,i}(\theta) + \lambda_{j,i}\|\theta\|_1 = \frac{1}{2}\left\|X^j[i] - \theta^\top X^{0:(j-1)}\right\|_2^2 + \sum_{l=0}^{j-1}\sum_{m_l=1}^{k_l}|\theta_l^{m_l}|,$$

with $X^j[i]$ being the vector of the length m of observations for the i-th variable in the j-th layer. We perform the optimization separately for each variable and the vector θ is from $\mathbb{R}^{k_0+\ldots+k_{j-1}}$ for $j = 1, \ldots, q$ and $i = 1, \ldots, k_j$. In our implementation we use the following scheme. We start with computing a sequence of minimizers on the grid, i.e. for any j and i we create a finite sequence $\{\lambda_k\}_{k=1}^N$ uniformly spaced on the log scale, starting from the largest λ_k, which corresponds to the empty model. Next, for each value λ_k we compute the estimator $\hat{\beta}_j^i[k]$ of the vector β_j^i

$$\hat{\beta}_j^i[k] = \underset{\theta\in\mathbb{R}^{k_0+\ldots,k_{j-1}}}{\operatorname{argmin}} \left\{RSS_{j,i}(\theta) + \lambda_k\|\theta\|_1\right\}. \tag{4.20}$$

To solve (4.20) numerically for a given λ_k we use the FISTA algorithm with backtracking from Beck and Teboulle (2009). The final LASSO estimator $\hat{\beta}_j^i := \hat{\beta}_j^i[k^*]$ is chosen using the Bayesian Information Criterion (BIC), which is a popular method of choosing $\lambda_{j,i}$ in the literature (Xue et al., 2012; Miasojedow and Rejchel, 2018), i.e.

$$k^* = \underset{1\le k\le N}{\operatorname{argmin}}\left\{m\log(RSS(\hat{\beta}_j^i[k])) + \log(m)\|\hat{\beta}_j^i[k]\|_0\right\}.$$

Here $\|\hat{\beta}_j^i[k]\|_0$ denotes the number of non-zero elements of $\hat{\beta}_j^i[k]$ and m is the number of observations of the network. In our simulations we use $N = 100$.

Finally, the threshold δ is obtained using the Generalized Information Criterion (GIC). A similar way of choosing a threshold was used previously in Pokarowski and Mielniczuk (2015); Miasojedow and Rejchel (2018). For a prespecified sequence of thresholds $\mathscr{D}$ we calculate

$$\delta^*_{j,i} = \underset{\delta \in \mathscr{D}}{\operatorname{argmin}} \left\{ m \log(RSS(\hat{\beta}^i_{j,\delta})) + \log(k_0 + \cdots + k_{j-1}) \|\hat{\beta}^i_{j,\delta}\|_0 \right\},$$

where $\hat{\beta}^i_{j,\delta}$ is the LASSO estimator $\hat{\beta}^i_j$ after thresholding with the level δ.

4.5.2 Experiments

In this subsection we compare our algorithm to other algorithms developed for this problem applying them to benchmark networks. We use the `bnlearn` package in `R` (Scutari (2010)), in which many algorithms for learning Bayesian networks including structure learning are implemented. Algorithms of different types discussed in the beginning of this chapter such as constraint-based algorithms, score-and-search algorithms and hybrid algorithms can be found there. The choice of specific algorithms was made empirically, i.e. we selected the best performing ones on the chosen networks. We took the networks with continuous data of a medium, large and very large amount of nodes and arcs. We refer to medium, large and very large sizes as 20-50 nodes, 50-100 nodes or 100-1000 nodes, respectively, adopting this classification from the authors of `bnlearn` package.

We chose a medium-size network $ECOLI70$ with 46 nodes and 70 arcs (Schäfer and Strimmer (2005)), a large network $MAGIC\text{-}IRRI$ * with 64 nodes and 102 arcs and a very large network $ARTH150$ with 107 nodes and 150 arcs (Opgen-Rhein and Strimmer (2007)). The algorithms chosen for comparison are hill-climbing (`hc`) algorithm, tabu search (`tabu`), max-min hill-climbing (`mmhc`) and Hybrid HPC (`h2pc`) algorithm. Hill-climbing (Scutari et al. (2018)) is a greedy search algorithm that explores the space of the directed acyclic graphs (DAGs) by an addition, removal or reversal of a single edge and uses random restarts to avoid local optima. Tabu search (Russell and Norvig (2010)) is a modified hill-climbing method, which is able to escape local optima by selecting a network that minimally decreases the score function. Both methods above use search-and-score approach. Max-min hill-climbing algorithm (Tsamardinos et al. (2006)) is a hybrid method combining a constraint-based algorithm called *max-min parents and children* and hill-climbing. H2PC (Hybrid HPC, Gasse et al. (2014)) algorithm is a hybrid algorithm combining an ensemble of weak PC learners (Spirtes and Glymour (1991)) and hill-climbing. For more different comparisons of methods in `bnlearn` package see Scutari et al. (2018).

*The model $MAGIC\text{-}IRRI$ was developed as an example of multiple trait modelling in plant genetics for the invited talk "Bayesian Networks, MAGIC Populations and Multiple Trait Prediction" delivered by Marco Scutari, the author of `bnlearn` package, at the 5th International Conference on Quantitative Genetics (ICQG 2016).

For each network we used two sizes of the data set with $m = 300$ and $m = 1000$ observations. In the tables with results we denoted them with 2-3 first letters of the name of the network followed by the number of observations so it does not create any confusion. For each algorithm we ran the experiment for 100 times, each time with a new set of m observations, and averaged the results in terms of three performance measures:

- **power,** i.e. the proportion of correctly discovered edges.
- **false discovery rate (FDR),** i.e. the fraction of incorrectly selected edges among all selected edges.
- **structural Hamming distance (SHD),** i.e. the smallest number of operations (such as adding or removing the edge and changing the direction of the arrow) required to match the true DAG and a learned one.

In Tables 4.1-4.3 we provide the results of experiments for mentioned above data sets and methods including ours. In terms of power our algorithm performs right in the middle of score-and-search and hybrid methods for $ECOLI70$ data set, similarly to the hybrid methods in case of small $MAGIC\text{-}IRRI$ data sets. We note that for data sets of 1000 observations it performs worse than other methods, however, with the number of observations growing the algorithm's power grows as well. With the number of observations of 10000 it grows up to 0.5 performing as good as other score methods without any increase in FDR. The same situation we observe with $ARTH150$ data set.

In terms of FDR our algorithm performs the best consistently giving very low numbers for false discoveries. This is especially important when the cost of a false discovery is high and makes obtained discoveries more certain. With the numbers of observations of 10000 we constantly get numbers in the range 0.2-0.4% for all data sets. When it comes to structural Hamming distance (SHD) our algorithm performs the best or close to the best numbers as well. With the growing number of observations it decreases due to increasing power and consistently low FDR. For the number of observations of 10000 it outperforms other algorithms or has close SHDs to hybrid methods and reaches around 28, 62 and 86 for $ECOLI70$, $MAGIC\text{-}IRRI$ and $ARTH150$ data sets, respectively.

We also checked our method on a discrete binary network called $ASIA$, introduced in Chapter 3. It is a small network of 8 nodes and 8 edges. Our algorithm recognizes 6 arrows and makes 2 false discoveries, discovering 8 arrows in total. However, after a closer look we noticed that it could not recognize 2 arrows due to incorrect assignment of layers. Finally, one false discovery was an arrow of an opposite direction to the true one, and the other one was an arrow from the start to the end of a causal trail (we obtained an additional arrow $X \to W$ for the trail $X \to Y \to Z \to W$), hence still recovering dependencies in both cases.

Method	$EC300$	$EC1000$	$MAG300$	$MAG1000$	$AR300$	$AR1000$
`hc`	0.57	0.65	0.28	0.45	0.58	0.67
`tabu`	**0.6**	**0.7**	**0.31**	**0.5**	**0.59**	**0.68**
`mmhc`	0.39	0.45	0.25	0.46	0.48	0.58
`h2pc`	0.4	0.49	0.23	0.45	0.5	0.61
MCMC + LASSO	0.49	0.55	0.24	0.38	0.38	0.46

Table 4.1: Average power for $ECOLI70$, $MAGIC\text{-}IRRI$ and $ARTH150$ networks for 300 and 1000 observations.

Method	$EC300$	$EC1000$	$MAG300$	$MAG1000$	$AR300$	$AR1000$
`hc`	0.049	0.044	0.04	0.037	0.028	0.19
`tabu`	0.047	0.036	0.04	0.036	0.028	0.018
`mmhc`	0.021	0.024	0.022	0.022	0.01	0.008
`h2pc`	0.02	0.023	0.14	0.019	0.006	0.006
MCMC + LASSO	**0.004**	**0.004**	**0.004**	**0.006**	**0.004**	**0.003**

Table 4.2: Average FDR for $ECOLI70$, $MAGIC\text{-}IRRI$ and $ARTH150$ networks for 300 and 1000 observations.

Method	$EC300$	$EC1000$	$MAG300$	$MAG1000$	$AR300$	$AR1000$
`hc`	65.6	51.8	129.5	96.5	214.2	151.1
`tabu`	64.8	46.9	127.1	93.1	215	150.9
`mmhc`	48.1	39.2	101.7	**72.2**	127.3	104.1
`h2pc`	45.9	37.9	91.6	67.21	103.7	**90.3**
MCMC + LASSO	**39.2**	**35.1**	**85.9**	79.3	**103**	98.7

Table 4.3: Average SHD for $ECOLI70$, $MAGIC\text{-}IRRI$ and $ARTH150$ networks for 300 and 1000 observations.

Chapter 5

Structure learning for CTBNs for complete data

In this chapter we consider continuous time Bayesian networks (CTBNs) introduced and defined in Section 2.5. First we consider the fully observed case where we observe the behaviour of the network at each moment of time.

5.1 Notation and preliminaries

In this section we describe the proposed method, using the notation introduced in Section 2.5. First, we consider the full graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, namely we assume that $\mathbf{pa}_{\mathcal{G}}(w) = \mathbf{pa}(w) = -w$ for each $w \in \mathcal{V}$. Then we remove unnecessary edges using the penalized likelihood technique. We start by introducing the new parametrization of the model. For simplicity, in the main part of this chapter we consider the binary graph, i.e. $\mathcal{X}_w = \{0, 1\}$ for each $w \in \mathcal{V}$. The extension of our results to more general case is described in Section 5.5.

Let d be the number of nodes in the graph. Consider a fixed order $(w_1, w_2, \ldots, w_d)$ of nodes of the graph. Using this order we define a $(2d) \times d$-dimensional matrix

$$\beta = \left(\beta_{0,1}^{w_1}, \beta_{1,0}^{w_1}, \beta_{0,1}^{w_2}, \beta_{1,0}^{w_2}, \ldots, \beta_{0,1}^{w_d}, \beta_{1,0}^{w_d}\right)^\top, \tag{5.1}$$

whose rows are vectors $\beta_{s,s'}^{w} \in \mathbb{R}^d$ for all $w \in \mathcal{V}$ and $s, s' \in \{0, 1\}$ such that $s \neq s'$. Obviously, the matrix β can be easily transformed to $2d^2$-dimensional vector in a standard way. In this chapter we assume that for all $w \in \mathcal{V}$, $c \in \mathcal{X}_{-w}$, $s, s' \in \{0, 1\}$, $s \neq s'$ the conditional intensity matrices satisfy

$$\log(Q_w(c, s, s')) = {\beta_{s,s'}^{w}}^\top Z_w(c), \tag{5.2}$$

where $Z_w \colon \mathcal{X}_{-w} \to \{0, 1\}^d$ is a binary deterministic function described below. With the slight abuse of notation, by Z we will denote the set of all functions $Z_{w_1}, \ldots, Z_{w_d}$. In (5.2) the conditional intensity matrix $Q_w(\cdot, s, s')$ is modeled in the analogous way to the regression function in generalized linear models (GLM) and the functions $Z_w(\cdot)$

play the role of explanatory variables (covariates). In our setting the link function is logarithmic. The analogous approach can be found in Andersen and Gill (1982); Huang et al. (2013), where the Cox model is considered. The relation between the intensity and covariates in those papers is similar to (5.2). Since the considered CTBNs do not contain explanatory variables, we introduce them artificially as *any possible representations* of parents' states. Thus, for every $w \in \mathcal{V}$ these explanatory variables are *dummy variables* encoding all possible configurations in $\mathbf{pa}(w) = -w$. To make it more transparent we consider the following example.

Example 5.1. We consider CTBN with three nodes A, B and C. For the node A we define the function Z_A as

$$Z_A(b,c) = [1, \mathbb{I}(b=1), \mathbb{I}(c=1)]^\top$$

for each $b, c \in \{0,1\}$, where $\mathbb{I}(\cdot)$ is the indicator function. Therefore, for each configuration of parents' states (i.e. values in the nodes B and C) the value of the function $Z_A(\cdot,\cdot)$ is a three-dimensional binary vector, whose coordinates correspond to the intercept, the value in the parent B and the value in the parent C, respectively. Analogously, we define representations for remaining nodes: $Z_B(a,c) = [1, \mathbb{I}(a=1), \mathbb{I}(c=1)]^\top$ and $Z_C(a,b) = [1, \mathbb{I}(a=1), \mathbb{I}(b=1)]^\top$ for each $a, b, c \in \{0,1\}$. In this example the parameter vector (5.1) is defined as $\beta = \left(\beta^A_{0,1}, \beta^A_{1,0}, \beta^B_{0,1}, \beta^B_{1,0}, \beta^C_{0,1}, \beta^C_{1,0}\right)^\top$. With slight abuse of notation, the vector $\beta^A_{0,1}$ is given as $\beta^A_{0,1} = \left[\beta^A_{0,1}(1), \beta^A_{0,1}(B), \beta^A_{0,1}(C)\right]^\top$ and we interpret (5.2) as follows: $\beta^A_{0,1}(B) = 0$ means that the intensity of the change from the state 0 to 1 at the node A does not depend on the state at the node B. Similarly, $\beta^A_{0,1}(C)$ describes the dependence between the above intensity and the state at the node C, and $\beta^A_{0,1}(1)$ corresponds to the intercept. For the node B the coordinates of the vector $\beta^B_{0,1} = \left[\beta^B_{0,1}(1), \beta^B_{0,1}(A), \beta^B_{0,1}(C)\right]$ describe the relation between the intensity of the jump from the state 0 to 1 at the node B to the intercept, states at nodes A and C, respectively.

Now what if $Z = \{Z_A, Z_B, Z_C\}$ was defined differently? The new function $\overline{Z}_A$ can be defined in 3 more different ways, for example $\overline{Z}_A(b,c) = [1, \mathbb{I}(b=0), \mathbb{I}(c=1)]^\top$. The same applies to the functions $\overline{Z}_B$ and $\overline{Z}_C$. Having defined $\overline{Z} = \{\overline{Z}_A, \overline{Z}_C, \overline{Z}_C\}$ we obtain the new vector of the parameters $\overline{\beta}$. Then for instance we have $\overline{\beta}^A_{0,1} = \left[\overline{\beta}^A_{0,1}(1), \overline{\beta}^A_{0,1}(B), \overline{\beta}^A_{0,1}(C)\right]$ and so on. Note that both sets Z and $\overline{Z}$ fully describe the state configuration of the network and both $\beta^A_{0,1}$ and $\overline{\beta}^A_{0,1}$ correspond to the same dependencies as above. In particular, it is easy to check that for instance $\beta^A_{0,1}(B) = \beta^A_{1,0}(B) = 0$ if and only if $\overline{\beta}^A_{0,1}(B) = \overline{\beta}^A_{1,0}(B) = 0$.

Analogously as in Example 5.1, for $w \in \mathcal{V}$, $u \neq w$, and $s, s' \in \{0,1\}$, $s \neq s'$ we define the coordinate of the function Z_w corresponding to the node u as an indicator of its state equal to either 0 or 1. Moreover, we denote the coordinate of $\beta^w_{s,s'}$ corresponding to the node u by $\beta^w_{s,s'}(u)$. We interpret $\beta^w_{s,s'}(u)$ as the parameter describing dependence of the intensity of the jump from the state s to s' at the node w on the state at the node u.

Our goal is to find edges in a directed graph $(\mathcal{V}, \mathcal{E})$. We define the relation between

parameters and edges in $(\mathcal{V}, \mathcal{E})$ in the following way

$$\beta^w_{0,1}(u) \neq 0 \text{ or } \beta^w_{1,0}(u) \neq 0 \;\Leftrightarrow\; \text{the edge } u \to w \text{ exists}, \tag{5.3}$$

which makes parameters compatible with the considered CTBNs. Roughly speaking, *the node u is a parent of w* means that the intensity of switching a state at w depends on the state at u. Therefore, the problem of finding edges in the graph is reformulated as the problem of the estimation of the parameter β.

Remark 5.2. As we mentioned previously in the example, the set Z fully describes the parents state configuration and the relation above does not depend on the choice of Z. More precisely, assume we have two different properly defined Z and $\overline{Z}$ and the corresponding vectors of parameters

$$\beta = \left(\beta^{w_1\top}_{0,1}, \beta^{w_1\top}_{1,0}, \beta^{w_2\top}_{0,1}, \beta^{w_2\top}_{1,0}, \ldots, \beta^{w_d\top}_{0,1}, \beta^{w_d\top}_{1,0}\right)^\top,$$
$$\overline{\beta} = \left(\overline{\beta}^{w_1\top}_{0,1}, \overline{\beta}^{w_1\top}_{1,0}, \overline{\beta}^{w_2\top}_{0,1}, \overline{\beta}^{w_2\top}_{1,0}, \ldots, \overline{\beta}^{w_d\top}_{0,1}, \overline{\beta}^{w_d\top}_{1,0}\right)^\top.$$

Then the following is true

$$\beta^w_{0,1}(u) = 0 \wedge \beta^w_{1,0}(u) = 0 \Leftrightarrow \overline{\beta}^w_{0,1}(u) = 0 \wedge \overline{\beta}^w_{1,0}(u) = 0.$$

This means that no matter how we define our explanatory functions Z_w, we will get the same arrows in the underlying CTBN.

Remark 5.3. For simplicity, in the rest of the thesis, we omit the first coordinate $\beta^w_{s,s'}(1)$ in the vector $\beta^w_{s,s'}$ for all w, $s \neq s'$, because it corresponds to the intercept and is not involved in the recognition of the edges in the graph. The first coordinates of representations $Z_w(c)$ are discarded as well.

Remark 5.4. The Markov equivalence/identifiability/non-uniqueness problem is challenging for directed graphical models. However, this problem does not appear here for CTBNs. It is a consequence of our Assumption (5.2), which states that we restrict to models having a conditional intensity in the GLM form. Moreover, under this assumption β is uniquely determined. Moreover, this uniquely defined β determines uniquely the structure of a graph by (5.3). In fact, our main result (Theorem 5.5 below) shows consistency of the estimator of β, which is a much stronger property than identifiability. Finally, in Assumption (5.2) we require that a conditional intensity of a variable is a linear function of the states of its parents. This condition can be easily extended to a polynomial dependence, so it can cover quite general dependence structure.

Our method is based on estimating the parameter β using the penalized likelihood method. In the rest of the thesis the term β is reserved for the true value of the parameter. Other quantities are denoted by θ. First, we consider a function

$$\ell(\theta) = \frac{1}{T} \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{X}_{-w}} \sum_{s \neq s'} \left[-n_w(c; s, s') \theta^{w\top}_{s,s'} Z_w(c) + t_w(c; s) \exp\left(\theta^{w\top}_{s,s'} Z_w(c)\right) \right], \tag{5.4}$$

where the third sum in (5.4) is over all $s, s' \in \mathcal{X}_w$ such that $s \neq s'$. Recall that $n_w(c; s, s')$ and $t_w(c; s)$ were introduced in Section 2.5 to denote the number of jumps from a state s to s' and the total time in the state s for the node w, respectively, while the parents configuration equals to c. Notice that the function (5.4) is the *negative log-likelihood*. Indeed, we just apply the negative logarithm to the density (2.8) combined with (2.9) and (5.2), where $\mathbf{pa}(w) = -w$ for each $w \in \mathcal{V}$. Then we divide it by T and omit the term corresponding to the initial distribution ν, because ν does not depend on β. We define an estimator of β as

$$\hat{\beta} = \underset{\theta \in \mathbb{R}^{2d(d-1)}}{\operatorname{argmin}} \left\{ \ell(\theta) + \lambda \|\theta\|_1 \right\}, \tag{5.5}$$

where $\|\theta\|_1 = \sum\limits_{w \in \mathcal{V}} \sum\limits_{s \neq s'} \sum\limits_{u \in -w} |\theta^w_{s,s'}(u)|$ is the l_1-norm of θ. The tuning parameter $\lambda > 0$ characterizes a balance between minimizing the negative log-likelihood and the penalty function. As we have mentioned, the form of the penalty is crucial, because its singularity at the origin implies that some coordinates of the minimizer $\hat{\beta}$ are exactly equal to 0, if λ is sufficiently large. Thus, starting from the full graph we remove irrelevant edges and estimate parameters for existing ones simultaneously. The function $\ell(\theta)$ and the penalty are convex functions, so (5.5) is a convex minimization problem, which is an important fact from both practical and theoretical point of views.

At first glance, computing (5.5) seems to be computationally complex, because the number of summands in (5.4) is $d2^d$. However, the number of nonzero terms of the form $n_w(c; s, s')$ and $t_w(c; s)$ is bounded by the total number of jumps, which grows linearly with time T. Hence, most of summands in (5.4) are also zeroes and the minimizer (5.5) can be calculated efficiently.

Before we state and prove main results of this chapter we introduce some additional notation. First, for each $w \in \mathcal{V}$ we denote its parents indicated by the true parameter β as

$$S_w = \left\{ u \in -w : \beta^w_{0,1}(u) \neq 0 \quad \text{or} \quad \beta^w_{1,0}(u) \neq 0 \right\}.$$

By S we denote the support of β, i.e. the set of nonzero coordinates of β. Moreover, $\beta_{\min}$ is the smallest in the absolute value element of β restricted to S. The set S^c denotes the complement of S, that is the set of zero coordinates of β. Besides, for each $w \in \mathcal{V}$ we define $-S_w = \mathcal{V} \setminus \{S_w \cup w\}$ and denote $\Delta = \max\limits_{\mathbf{s}, \mathbf{s}' \in \mathcal{X},\, \mathbf{s} \neq \mathbf{s}'} Q(\mathbf{s}, \mathbf{s}')$.

Recall that for a vector a we denote its l_∞-norm by $\|a\|_\infty = \max_k |a_k|$. For a subset I the vector a_I denotes a vector such that $(a_I)_i = a_i$ for $i \in I$ and $(a_I)_i = 0$ otherwise. Moreover, $|I|$ denotes the number of elements of I.

Let π be the stationary distribution of the Markov jump process (MJP), which is defined by an intensity matrix Q. The initial distribution of this process is denoted by ν and we define $\|\nu\|_2^2 = \sum\limits_{\mathbf{s} \in \mathcal{X}} \nu^2(\mathbf{s})/\pi(\mathbf{s})$. Moreover, ρ_1 denotes the smallest positive eigenvalue of the matrix $-1/2(Q + Q^*)$, where Q^* is an adjoint matrix of Q.

5.2 Main results

In this subsection, we state two key results on the structure learning for CTBNs for complete data. In the first one (Theorem 5.5) we show that the estimation error of the minimizer $\hat{\beta}$ given by (5.5) can be controlled with probability close to 1. In the second main result (Corollary 5.6) we state that the thresholded version of (5.5) is able to recognize the structure of the graph with high probability.

First, we introduce the cone invertibility factor (CIF), which plays an important role in the theoretical analysis of the properties of LASSO estimators. Our goal is to show that the estimator $\hat{\beta}$ is close to the true vector β. To accomplish this goal we show in Lemma 5.9 that the gradient of the likelihood (5.4) evaluated at β is close to 0. However, this is not sufficient since the likelihood function cannot be too "flat". Namely, its curvature around the local optimum needs to be relatively high, because we want to avoid the situation when the loss difference can be small whereas the error is large. In the high-dimensional scenario this is often provided by imposing the restricted strong convexity condition (RSC) on (5.4), as in Negahban et al. (2009). CIF defined below in (5.6) plays a similar role to RSC, but gives sharper consistency results (Ye and Zhang, 2010). Therefore, it is used here. CIF is defined analogously to Ye and Zhang (2010); Huang and Zhang (2012); Huang et al. (2013) and is closely related to the compatibility factor (van de Geer, 2008) or the restricted eigenvalue condition (Bickel et al., 2009). Recall that in the previous chapter we also used a version of CIF for Bayesian networks. Thus, for any $\xi > 1$ we define the cone $\mathcal{C}(\xi, S) = \{\theta : \|\theta_{S^c}\|_1 \leq \xi \|\theta_S\|_1\}$, where the set S denotes the support of β as mentioned above. Then CIF is defined as

$$\bar{F}(\xi) = \inf_{0 \neq \theta \in \mathcal{C}(\xi, S)} \frac{\theta^\top \nabla^2 \ell(\beta) \theta}{\|\theta_S\|_1 \|\theta\|_\infty}. \tag{5.6}$$

Notice that only the value of the Hessian $\nabla^2 \ell(\theta)$ at the true parameter β is taken into consideration in (5.6). The main difficulty with CIF in our case is that it is a minimum of the sum of random terms, which number grows exponentially in d. To be able to control this quantity, we bound CIF from below by its deterministic counterpart with much fewer summands. Namely, in Lemma 5.11 we prove that $\bar{F}(\xi)$ is bounded from below by the product of ζ_0 given in Theorem 5.5 and

$$F(\xi) = \inf_{0 \neq \theta \in C(\xi, S)} \sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{c_{S_w} \in \mathcal{X}_{S_w}} \frac{\exp\left(\beta^{w\top}_{s,s'} Z_w(c_{S_w}, 0)\right) \left[\theta^{w\top}_{s,s'} Z_w(c_{S_w}, 0)\right]^2}{\|\theta_S\|_1 \|\theta\|_\infty} \tag{5.7}$$

with probability close to 1. Here we divided each parent configuration $c = (c_{S_w}, c_{-S_w})$ into two parts: the first one corresponds to the true parents nodes and the second part corresponds to the remaining nodes. Below we will also use a similar notation for any state of the network $\mathbf{s} \in \mathcal{X}$ defining it as a triple $\mathbf{s} = (c_{S_w}, c_{-S_w}; s)$ of the state of true parents of the node w, the configuration for the nodes from S_{-w} and the state in the node w. Note, that we restricted the summation in (5.7) only to $c_{S_w} \in \mathcal{X}_{S_w}$ by taking $c_{-S_w} = 0$. This allows us to derive the lower bound on $\bar{F}(\xi)$ without considering exponentially many

random summands. Our argumentation will be also valid in the case, when we choose some nonzero values in c_{-S_w}, unless this values depend on w and c_{S_w}. Next, we state two main results of this chapter.

Theorem 5.5. *Fix arbitrary $\varepsilon \in (0,1)$ and $\xi > 1$. Suppose that*

$$T > \frac{36\left[\left(\max_{w\in\mathcal{V}}|S_w|+1\right)\log 2 + \log\left(d\|\nu\|_2/\varepsilon\right)\right]}{\rho_1 \min\limits_{\substack{w\in\mathcal{V},\, s\in\mathcal{X}_w \\ c_{S_w}\in\mathcal{X}_{S_w}}} \pi^2(c_{S_w},0;s)}. \tag{5.8}$$

We also assume that $T\Delta \geq 2$ and we choose λ such that

$$2\frac{\xi+1}{\xi-1}\log(K/\varepsilon)\sqrt{\frac{\Delta}{T}} \leq \lambda \leq \frac{2\zeta_0 F(\xi)}{e(\xi+1)|S|}, \tag{5.9}$$

where $K = 2(2+e^2)d(d-1)$ and

$$\zeta_0 = \min_{\substack{w\in\mathcal{V},\, s\in\mathcal{X}_w \\ c_{S_w}\in\mathcal{X}_{S_w}}} \pi(c_{S_w},0;s)/2.$$

Then with probability at least $1-2\varepsilon$ we have

$$\|\hat{\beta}-\beta\|_\infty \leq \frac{2e\xi\lambda}{(\xi+1)\zeta_0 F(\xi)}. \tag{5.10}$$

Now consider the Thresholded LASSO estimator with the set of nonzero coordinates $\hat{S}$. The set $\hat{S}$ contains only those coefficients of the LASSO estimator (5.5), which are larger in the absolute value than a pre-specified threshold δ.

Corollary 5.6. *Suppose that assumptions of Theorem 5.5 are satisfied and let R denote the right-hand side of the inequality (5.10). If $R < \beta_{\min}/2$, then for $\delta \in [R, \beta_{\min}/2)$ we have $P\left(\hat{S} = S\right) \geq 1-2\varepsilon$.*

These two results will be proven in the next section and here we give some comments on their meaning and significance. The above two results describe the properties of the proposed estimator (5.5) in recognizing the structure of the graph. Theorem 5.5 gives conditions under which the estimation error of (5.5) can be controlled. Namely, let us for a moment ignore constants, Δ and parameters of MJP such as $\nu, \pi, \rho_1, \zeta_0$, etc. in the assumptions. By condition (5.9), if

$$T \geq \frac{\log^2(d/\varepsilon)|S|^2}{F^2(\xi)}, \tag{5.11}$$

then the estimation error is small. This forms some restrictions on the number of vertices in the graph, sparsity of the graph (i.e. the number of edges has to be small enough) and the expression (5.7), which is discussed in Lemma 5.7 (below). The condition (5.11) is similar to standard results for LASSO estimators in Ye and Zhang (2010); Bühlmann and van de Geer (2011); Huang and Zhang (2012); Huang et al. (2013). The only difference is

that the right-hand side of (5.11) usually depends linearly on $\log(d/\varepsilon)$, but here we have $\log^2(d/\varepsilon)$. The square in the logarithm could be omitted, if we imposed some additional restrictions on observation time T in the crucial auxiliary result (Lemma 5.9), where we use the Bernstein-type inequality for the Poisson random variable. Obviously, it would reduce the applicability of the main result. In our opinion, the gain (having $\log(d/\varepsilon)$ instead of $\log^2(d/\varepsilon)$) is "smaller" than the price (additional assumptions), so we do not focus on it.

The next assumption in Theorem 5.5 that $T\Delta \geq 2$ is quite natural since observation time has to increase when the maximal intensity of transitions decreases. Moreover, the conditions (5.8) and (5.9) depend also on parameters of MJP. More precisely, they depend on the stationary distribution π and the spectral gap ρ_1, which in general decrease exponentially with d. However, in some specific cases, it can be proved that they decrease polynomially.

Corollary 5.6 states that the LASSO estimator after thresholding is able to recognize the structure of a graph with probability close to 1, if the nonzero coefficients of β are not too close to zero and the threshold δ is appropriately chosen. However, Corollary 5.6 does not give a way of choosing the threshold δ, because both endpoints of the interval $[R, \beta_{\min}/2]$ are unknown. It is not a surprising fact and has been already observed, for instance, in linear models (Ye and Zhang, 2010, Theorem 8). In the experimental subsection of this chapter we propose a method of choosing a threshold that relies on information criteria. A similar procedure can be found in Pokarowski and Mielniczuk (2015); Miasojedow and Rejchel (2018).

Now we state a lower bound for (5.7) which has an intuitive interpretation.

Lemma 5.7. *Define* $A_\beta = \sum_{w\in\mathcal{V}} \sum_{s'\neq s} \sum_{u:\beta^w_{s,s'}(u)\neq 0} \exp\left(-\beta^w_{s,s'}(u)\right)$. *Then for every* $\xi > 1$ *we have* $F(\xi) \geq (\xi A_\beta)^{-1}$.

Notice that the term A_β is larger, and in turn $F(\xi)$ is smaller, when negative coefficients of β „dominate" in the absolute value the positive ones. Note that, the more these negative coefficients dominate the more our process „gets stuck", i.e. tends to stay in the same state because intensities in this case tend to be close to zero (recall (5.2)). Such behaviour in the context of MJPs is natural, because multiplying the intensity matrix Q by a constant κ is equivalent to considering time T/κ instead of T. While $F(\xi)$ appears in the lower bound (5.11) on T, such dependence on β is expected.

5.3 Proofs of the main results

This subsection contains the proofs of all the statements made in the previous subsection. The proofs of the theorem and the corollary are based on a number of auxiliary results. Some of these results are well-known facts for LASSO estimators and some of them are new (Lemmas 5.9 and 5.11). The main novelty and difficulty of the considered model is the continuous time nature of the observed phenomena which we investigate.

In Lemma 5.9 we derive the new concentration inequality for MJPs based on the martingale theory. In Lemma 5.11 we give new upper bounds on the occupation time for MJPs.

In the proofs of subsequent results we use the first and second derivatives of ℓ given by (5.4), which can be also expressed in the following form

$$\ell(\theta) = \frac{1}{T} \sum_{w \in \mathcal{V}} \sum_{s \neq s'} \ell^w_{s,s'}(\theta^w_{s,s'}), \tag{5.12}$$

where

$$\ell^w_{s,s'}(\theta^w_{s,s'}) = \sum_{c \in \mathcal{X}_{-w}} \left[-n_w(c; s, s') {\theta^w_{s,s'}}^\top Z_w(c) + t_w(c; s) \exp({\theta^w_{s,s'}}^\top Z_w(c)) \right].$$

Therefore, we can calculate partial derivatives

$$\nabla \ell^w_{s,s'}(\theta^w_{s,s'}) = \sum_{c \in \mathcal{X}_{-w}} [-n_w(c; s, s') + t_w(c; s) \exp({\theta^w_{s,s'}}^\top Z_w(c))] Z_w(c). \tag{5.13}$$

By Remark 5.3 the matrix θ of all parameters has $2d$ rows and $(d-1)$ columns. It can be also considered as a $2d(d-1)$-dimensional vector

$$\theta = \left({\theta^{w_1}_{0,1}}^\top, {\theta^{w_1}_{1,0}}^\top, {\theta^{w_2}_{0,1}}^\top, {\theta^{w_2}_{1,0}}^\top, \ldots, {\theta^{w_d}_{0,1}}^\top, {\theta^{w_d}_{1,0}}^\top \right)^\top,$$

where $(w_1, w_2, \ldots, w_d)$ is a fixed order of the nodes of the graph. Using this order we obtain the following representation of the gradient of ℓ.

$$\nabla \ell(\theta) = \frac{1}{T} \left[\nabla \ell^w_{s,s'}(\theta^w_{s,s'}) \right]_{w \in \mathcal{V}, s \neq s'}. \tag{5.14}$$

Analogously we calculate second derivatives

$$\nabla^2 \ell^w_{s,s'}(\theta^w_{s,s'}) = \sum_{c \in \mathcal{X}_{-w}} t_w(c; s) \exp({\theta^w_{s,s'}}^\top Z_w(c)) Z_w(c) Z_w(c)^\top.$$

The second derivative of $\ell(\theta)$ consists of matrices $\frac{1}{T} \nabla^2 \ell^w_{s,s'}(\theta^w_{s,s'})$ along its diagonal and zeroes elsewhere. Moreover, for any vector $\theta \in \mathbb{R}^{2d(d-1)}$ and the true parameter vector β we have

$$\theta^\top \nabla^2 \ell(\beta) \theta = \frac{1}{T} \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{X}_{-w}} \sum_{s' \neq s} t_w(c; s) \left({\theta^w_{s,s'}}^\top Z_w(c) \right)^2 \exp({\beta^w_{s,s'}}^\top Z_w(c)). \tag{5.15}$$

Next we provide an auxiliary proposition needed to prove an important concentration inequality for MJPs in Lemma 5.9.

Proposition 5.8. *Let $X(\tau)$ be a Markov jump process with a bounded intensity matrix Q. Let*

$$n^\tau_{s,s'} = \sum_{c \in \mathcal{X}_{-w} \colon Z_w(c)[k] = 1} n^\tau_w(c; s, s')$$

be a number of jumps from s to s' on the interval $[0,\tau]$ and t_s^τ be an occupation time at state s on the interval $[0,\tau]$. Then

$$M_\nu(\tau) = n_{s,s'}^\tau - t_s^\tau Q(s,s')$$

is a martingale with respect to the natural filtration $\mathcal{F}_\tau$. The notation $M_\nu(\tau)$ means that the distribution at time 0 is ν.

Proof. For any $u < \tau$ we have

$$\begin{aligned}\mathbb{E}(M_\nu(\tau) \mid \mathcal{F}_u) &= M_\nu(u) + \mathbb{E}(M_\nu(\tau) - M_\nu(u) \mid \mathcal{F}_u) = \\ &= M_\nu(u) + \mathbb{E}(M_\nu(\tau) - M_\nu(u) \mid X(u)) \\ &= M_\nu(u) + \mathbb{E}(M_{X(u)}(\tau - u) \mid X(u)),\end{aligned}$$

where the last equality is the consequence of Proposition 20.3 from Bass (2011). Now it is enough to show that for all $\tau > 0$ and all initial measures ν we have $\mathbb{E}M_\nu(\tau) = 0$, since it implies that $\mathbb{E}\left(\mathbb{E}(M_\nu(\tau) \mid \mathcal{F}_u)\right) = 0$ for $u < \tau$.

For any $n \in \mathbb{N}$ define the sequence $k_i = k_i(n) = \frac{\tau i}{n}$ for all $i = 0, \ldots, n$. Since the trajectory of the process is *càdlàg*, we have

$$\mathbb{E}M_\nu(\tau) = \mathbb{E} \lim_{n\to\infty} \sum_{i=1}^n \left[\mathbb{I}\left(X(k_{i-1}) = s, X(k_i) = s'\right) - \frac{\tau}{n} Q(s,s') \mathbb{I}\left(X(k_{i-1}) = s\right)\right].$$

We observe that for all $n \in \mathbb{N}$

$$\left|\sum_{i=1}^n \left[\mathbb{I}\left(X(k_{i-1}) = s, X(k_i) = s'\right) - \frac{\tau}{n} Q(s,s') \mathbb{I}\left(X(k_{i-1}) = s\right)\right]\right| \le N(\tau) + \tau, \tag{5.16}$$

where $N(\tau)$ is the total number of jumps. Since $N(\tau)$ is a Poisson process with a bounded intensity, the right-hand side of (5.16) is integrable and by the dominated convergence theorem and the definition of Q we get

$$\begin{aligned}\mathbb{E}M_\nu(\tau) &= \lim_{n\to\infty} \mathbb{E} \sum_{i=1}^n \left[\mathbb{I}\left(X(k_{i-1}) = s, X(k_i) = s'\right) - \frac{\tau}{n} Q(s,s') \mathbb{I}\left(X(k_{i-1}) = s\right)\right] \\ &= \lim_{n\to\infty} \mathbb{E} \sum_{i=1}^n \left[\mathbb{E}\left(\mathbb{I}\left(X(k_{i-1}) = s, X(k_i) = s'\right) \mid X(k_{i-1})\right) - \frac{\tau}{n} Q(s,s') \mathbb{I}\left(X(k_{i-1}) = s\right)\right].\end{aligned}$$

Next for $s \neq s'$ we have

$$\begin{aligned}&\mathbb{P}\left(X(k_{i-1}) = s, X(k_i) = s' \mid X(k_{i-1}) = s\right) = \\ &\quad = \mathbb{P}\left(X(k_i) = s' \mid X(k_{i-1}) = s\right) = \frac{Q(s,s')}{n} + o\left(1/n\right)\end{aligned}$$

and for $\sigma \neq s$

$$\mathbb{P}\left(X(k_{i-1}) = s, X(k_i) = s' \mid X(k_{i-1}) = \sigma\right) = 0.$$

Hence,

$$\mathbb{E}\left[\mathbb{I}\left(X(k_{i-1}) = s, X(k_i) = s'\right) \mid X(k_{i-1})\right] = \left(\frac{\tau}{n} Q(s,s') + o(1/n)\right) \mathbb{I}\left(X(k_{i-1}) = s\right).$$

Therefore, we further obtain

$$\begin{aligned}\mathbb{E}M_\nu(\tau) &= \lim_{n\to\infty}\mathbb{E}\sum_{i=1}^{n}\left[\left(\frac{\tau}{n}Q(s,s')+o(1/n)\right)\mathbb{I}\left(X(k_{i-1})=s\right)-\frac{\tau}{n}Q(s,s')\mathbb{I}(X(k_{i-1})=s)\right]\\ &= \lim_{n\to\infty}\mathbb{E}\sum_{i=1}^{n}o(1/n)\mathbb{I}(X(k_{i-1})=s)=0,\end{aligned}$$

where $o(1/n)$ does not depend on i. □

Lemma 5.9. *Let $\varepsilon\in(0,1)$ and $\xi>1$ be arbitrary. Assume that $T\Delta\geq 2$ and*

$$\lambda\geq 2\frac{\xi+1}{\xi-1}\log(K/\varepsilon)\sqrt{\frac{\Delta}{T}},$$

where $K=2(2+e^2)d(d-1)$. Then we have

$$\mathbb{P}\left(\|\nabla\ell(\beta)\|_\infty\leq\frac{\xi-1}{\xi+1}\lambda\right)\geq 1-\varepsilon.$$

Proof. Note that by (5.2), (5.13) and (5.14) we have the following inequality

$$\|\nabla\ell(\beta)\|_\infty\leq\frac{1}{T}\max_{w\in\mathcal{V},s\neq s',1\leq k\leq d-1}\left|\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}\left[n_w(c;s,s')-t_w(c;s)Q_w(c;s,s')\right]\right|,$$

where $Z_w(c)[k]$ is the k-th coordinate of $Z_w(c)$ for each $w\in\mathcal{V}$, $c\in\mathcal{X}_{-w}$. The core step of the proof is to show that for fixed $w\in\mathcal{V}$, $s\neq s'$, $1\leq k\leq d-1$ and $\eta=2\log(K/\varepsilon)$

$$\mathbb{P}\left(\left|\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}\left[n_w(c;s,s')-t_w(c;s)Q_w(c;s,s')\right]\right|>\eta\sqrt{T\Delta}\right)\leq(2+e^2)\exp\left(-\frac{\eta}{2}\right). \tag{5.17}$$

Having (5.17) we finish the proof of Lemma 5.9 using union bounds. More precisely,

$$\begin{aligned}&\mathbb{P}\left(\|\nabla\ell(\beta)\|_\infty>\frac{\xi-1}{\xi+1}\lambda\right)\leq\\ &\leq\mathbb{P}\left(\frac{1}{T}\max_{w\in\mathcal{V},s\neq s',1\leq k\leq d-1}\left|\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}\left[n_w(c;s,s')-t_w(c;s)Q_w(c;s,s')\right]\right|>\eta\sqrt{\frac{\Delta}{T}}\right)\leq\\ &\leq 2d(d-1)\mathbb{P}\left(\left|\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}\left[n_w(c;s,s')-t_w(c;s)Q_w(c;s,s')\right]\right|>\eta\sqrt{T\Delta}\right)\leq\\ &\leq 2d(d-1)(2+e^2)\exp(-\log(K/\varepsilon))=\varepsilon.\end{aligned}$$

Therefore, we focus on proving (5.17). The proof of this inequality is based on the martingale arguments, so we make the dependence on the time explicit in (5.17), that is $n_w(c;s,s')$ and $t_w(c;s)$ become $n_w^T(c;s,s')$ and $t_w^T(c;s)$, respectively.

For $\tau\in[0,T]$ we define a process

$$M(\tau)=\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}\left[n_w^\tau(c;s,s')-t_w^\tau(c;s)Q_w(c;s,s')\right]. \tag{5.18}$$

We use the upper index τ in $n_w^\tau(c;s,s')$ and $t_w^\tau(c;s)$ to indicate that these quantities correspond to the time interval $[0,\tau]$. Using Proposition 5.8 to each summand in (5.18) we obtain that the process $\{M(\tau):\tau\in[0,T]\}$ is a martingale. Let us define its jumps by

$$\Delta M(\tau)=M(\tau)-M(\tau_-)=\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}\mathbb{I}\left[X(\tau_-)=(s,c),X(\tau)=(s',c)\right],$$

where $M(\tau_-)$ is the left limit at τ. By Theorem II.37 of Protter (2005) and Theorem I.4.61 of Jacod and Shiryaev (2003) for any $x>-1$ the process

$$\begin{aligned}\mathcal{E}_x(\tau) &= \exp\left(xM(\tau)\right)\prod_{u\le\tau}(1+x\Delta M(u))\exp(-x\Delta M(u))\\ &= \exp\left\{xM(\tau)-(x-\log(1+x))n_{s,s'}^\tau\right\}\end{aligned}$$

is a local martingale, where $n_{s,s'}^\tau=\sum_{c\in\mathcal{X}_{-w}\colon Z_w(c)[k]=1}n_w^\tau(c;s,s')$ is computed for a trajectory at the time interval $[0,\tau]$. Therefore, by Markov inequality together with the triangle inequality we get for any $x\in(0,1]$

$$\begin{aligned}\mathbb{P}(|M(T)|>L) &\le \mathbb{P}(|xM(T)-(x-\log(1+x))n_{s,s'}^T|>xL/2)+\\ &+ \mathbb{P}((x-\log(1+x))n_{s,s'}^T>xL/2)\le\\ &\le 2\exp\left(\frac{-xL}{2}\right)+\mathbb{P}((x-\log(1+x))n_{s,s'}^T>xL/2)\,.\end{aligned}$$

We observe that $n_{s,s'}^T$ is bounded from above by the total number of jumps up to time T, which in turn is bounded by a Poisson random variable $N(T)$ with the intensity $T\Delta$. Hence, again by Markov inequality we have

$$\mathbb{P}((x-\log(1+x))n_{s,s'}^T>xL/2)\le\exp\left[\frac{-xL}{2}+T\Delta\left(\frac{e^x}{1+x}-1\right)\right].$$

Applying an inequality $e^x\le 1/(1-x)$ for $x<1$ and setting $x=1/\sqrt{T\Delta}$ we get

$$\mathbb{P}((x-\log(1+x))n_{s,s'}^T>xL/2)\le\exp\left(\frac{-L}{2\sqrt{T\Delta}}+\frac{T\Delta}{T\Delta-1}\right).$$

We use $T\Delta\ge 2$ and we plug in $L=\eta\sqrt{T\Delta}$ to conclude the proof. □

The next lemma is a direct application of Theorem 3.4 of Lezaud (1998) and it will be used in the second crucial auxiliary Lemma 5.11.

Lemma 5.10. *For any $w\in\mathcal{V}$, $s\in\mathcal{X}_w$, $c_{S_w}\in\mathcal{X}_{S_w}$ we have*

$$\mathbb{P}\left(\frac{1}{T}t_w(c_{S_w},0;s)\le\pi(c_{S_w},0;s)/2\right)\le\|\nu\|_2\exp\left(-\frac{\pi^2(c_{S_w},0;s)\rho_1 T}{16+20\pi(c_{S_w},0;s)}\right).$$

Proof. Fix $w\in\mathcal{V}$, $s\in\mathcal{X}_w$, $c_{S_w}\in\mathcal{X}_{S_w}$. By the definition we have

$$t_w(c_{S_w},0;s)=\int_0^T\mathbb{I}\left[X(t)=(c_{S_w},0;s)\right]dt.$$

Let us define $f(X(t))=\pi(c_{S_w},0;s)-\mathbb{I}\left(X(t)=(c_{S_w},0,s)\right)$. Taking $\gamma=\pi(c_{S_w},0;s)/2$ in Theorem 3.4 of Lezaud (1998), we conclude the proof. □

Lemma 5.11. *Let $\varepsilon \in (0,1)$, $\xi > 1$ be arbitrary. Suppose that $F(\xi)$ defined in (5.7) is positive and*

$$T > \frac{36\left[\left(\max\limits_{w\in\mathcal{V}}|S_w|+1\right)\log 2 + \log\left(d\|\nu\|_2/\varepsilon\right)\right]}{\rho_1 \min\limits_{\substack{w\in\mathcal{V},\, s\in\mathcal{X}_w \\ c_{S_w}\in\mathcal{X}_{S_w}}} \pi^2(c_{S_w},0;s)}, \tag{5.19}$$

then

$$\mathbb{P}\left(\bar{F}(\xi) \geq \zeta_0 F(\xi)\right) \geq 1-\varepsilon,$$

where $\zeta_0 = \min\limits_{\substack{w\in\mathcal{V},\, s\in\mathcal{X}_w \\ c_{S_w}\in\mathcal{X}_{S_w}}} \pi(c_{S_w},0;s)/2$.

Proof. By the definition of $\bar{F}(\xi)$, the equation (5.7) and the formula for Hessian of ℓ (see (5.15)) we have

$$\frac{\bar{F}(\xi)}{F(\xi)} \geq \frac{1}{T} \min_{w\in\mathcal{V}, s, c_{S_w}\in\mathcal{X}_{S_w}} t_w(c_{S_w},0;s). \tag{5.20}$$

We complete the proof by bounding the right-hand side of (5.20) from below. First, we can calculate that

$$\begin{aligned}
&\mathbb{P}\left(\min_{w\in\mathcal{V}, s\in\mathcal{X}_w, c_{S_w}\in\mathcal{X}_{S_w}} \frac{1}{T} t_w(c_{S_w},0;s) \geq \zeta_0\right) \geq \\
&\geq \mathbb{P}\left(\forall_{w\in\mathcal{V}, s\in\mathcal{X}_w, c_{S_w}\in\mathcal{X}_{S_w}} \quad \frac{1}{T} t_w(c_{S_w},0;s) \geq \pi(c_{S_w},0;s)/2\right) \geq \\
&\geq 1 - 2d \max_{w\in\mathcal{V}, s\in\mathcal{X}_w, c_{S_w}\in\mathcal{X}_{S_w}} 2^{|S_w|}\mathbb{P}\left(\frac{1}{T} t_w(c_{S_w},0;s) < \pi(c_{S_w},0;s)/2\right).
\end{aligned} \tag{5.21}$$

Using Lemma 5.10 we bound (5.21) from below by

$$1 - 2d \max_{w\in\mathcal{V}, s\in\mathcal{X}_w, c_{S_w}\in\mathcal{X}_{S_w}} 2^{|S_w|}\|\nu\|_2 \exp\left(-\frac{\pi^2(c_{S_w},0;s)\rho_1 T}{16+20\pi(c_{S_w},0;s)}\right).$$

Applying (5.19) we conclude the proof. □

Next we state and prove three lemmas, where Lemmas 5.12 and 5.14 will be used in the proof of the main result Theorem 5.5 and Lemma 5.13 is needed to prove Lemma 5.14.

Lemma 5.12. *Let $\tilde{\beta} = \hat{\beta} - \beta$, $z^* = \|\nabla\ell(\beta)\|_\infty$. Then*

$$(\lambda - z^*)\|\tilde{\beta}_{S^c}\|_1 \leq \tilde{\beta}^\top\left[\nabla\ell(\hat{\beta}) - \nabla\ell(\beta)\right] + (\lambda - z^*)\|\tilde{\beta}_{S^c}\|_1 \leq (\lambda + z^*)\|\tilde{\beta}_S\|_1. \tag{5.22}$$

Besides, for arbitrary $\xi > 1$ on the event

$$\Omega_1 = \left\{\|\nabla\ell(\beta)\|_\infty \leq \frac{\xi-1}{\xi+1}\lambda\right\}$$

the random vector $\tilde{\beta}$ belongs to the cone $\mathcal{C}(\xi, S)$.

The proof of Lemma 5.12 is similar to the proof of Lemma 3.1 of Huang et al. (2013) and is based on convexity of $\ell(\theta)$ and properties of the LASSO penalty. For convenience of the reader we will provide it here using our notation.

Proof. Since $\ell(\theta)$ is a convex function, then

$$\tilde{\beta}^\top \Big[\nabla \ell(\hat{\beta}) - \nabla \ell(\beta)\Big] = \tilde{\beta}^\top \Big[\nabla \ell(\beta + \tilde{\beta}) - \nabla \ell(\beta)\Big] \geq 0,$$

which instantly proves the left-hand side inequality in (5.22). As we have already mentioned, the minimized target function, given in (5.5), is also convex because of the convexity of the negative log-likelihood $\ell(\theta)$ and ℓ_1-penalty functions. Hence $\hat{\beta}$ will be a minimizer in (5.5) if and only if the following conditions are met

$$\begin{cases} \dfrac{\partial \ell(\hat{\beta})}{\partial \beta_j} = -\lambda \operatorname{sgn}(\beta_j), & \text{if } \hat{\beta}_j \neq 0, \\ \left| \dfrac{\partial \ell(\hat{\beta})}{\partial \beta_j} \right| \leq \lambda, & \text{if } \hat{\beta}_j = 0, \end{cases} \tag{5.23}$$

where $j \in \{1, \ldots, 2d(d-1)\}$. First, we can write

$$\tilde{\beta}^\top \Big[\nabla \ell(\beta + \tilde{\beta}) - \nabla \ell(\beta)\Big] = \sum_{j \in S^c} \tilde{\beta}_j \frac{\partial \ell(\beta + \tilde{\beta})}{\partial \beta_j} + \sum_{j \in S} \tilde{\beta}_j \frac{\partial \ell(\beta + \tilde{\beta})}{\partial \beta_j} + \tilde{\beta}^\top (-\nabla \ell(\beta)).$$

Since $\tilde{\beta}_j = \hat{\beta}_j$ for $j \in S^c$, then applying the conditions (5.23) we can bound the last expression from above by

$$\sum_{j \in S^c} \hat{\beta}_j (-\lambda \operatorname{sgn}(\hat{\beta}_j)) + \sum_{j \in S} |\tilde{\beta}_j| \lambda + \|\tilde{\beta}\|_1 z^* = \sum_{j \in S^c} -\lambda |\tilde{\beta}_j| + \|\tilde{\beta}_S\|_1 \lambda + z^* \|\tilde{\beta}_S\|_1 + z^* \|\tilde{\beta}_{S^c}\|_1.$$

This in turn equals to $(z^* - \lambda)\|\tilde{\beta}_{S^c}\|_1 + (\lambda + z^*)\|\tilde{\beta}_S\|_1$ meaning that the right-hand side inequality holds as well. Note that we used the fact that $\dfrac{\partial \ell(\hat{\beta})}{\partial \beta_j} = -\lambda \operatorname{sgn}(\beta_j)$ only on the set $S^c \cap \{j : \hat{\beta}_j \neq 0\}$, because $\tilde{\beta}_j = \hat{\beta}_j - \beta_j = 0$, when $j \in S^c$ and $\hat{\beta}_j = 0$. Finally, by (5.22) and the definition of Ω_1 we obtain

$$\|\tilde{\beta}_{S^c}\|_1 \leq \frac{\lambda + z^*}{\lambda - z^*} \|\tilde{\beta}_S\|_1 \leq \|\tilde{\beta}_S\|_1$$

proving the last claim of the lemma. □

Lemma 5.13. *For any $b \in \mathbb{R}^{2d(d-1)}$ we define $c_b = \max\limits_{w \in \mathcal{V},\, s \neq s',\, c \in \mathcal{X}_{-w}} \exp\left(\left|{b^w_{s,s'}}^\top Z_w(c)\right|\right)$. Then we have*

$$c_b^{-1} b^\top \nabla^2 \ell(\beta) b \leq b^\top [\nabla \ell(\beta + b) - \nabla \ell(\beta)] \leq c_b b^\top \nabla^2 \ell(\beta) b \tag{5.24}$$

and

$$c_b^{-1} \nabla^2 \ell(\beta) \leq \nabla^2 \ell(\beta + b) \leq c_b \nabla^2 \ell(\beta), \tag{5.25}$$

where for two symmetric matrices A, B the expression $A \leq B$ means that $B - A$ is a non-negative definite matrix.

Proof. First we prove the inequality (5.24). By (5.14) we have

$$b^\top[\nabla\ell(\beta+b)-\nabla\ell(\beta)] = \\ = \frac{1}{T}\sum_{w\in\mathcal{V}}\sum_{c\in\mathcal{X}_{-w}}\sum_{s'\neq s} t_w(c;s)\left({b^w_{s,s'}}^\top Z_w(c)\right)\exp({\beta^w_{s,s'}}^\top Z_w(c))\left[\exp({b^w_{s,s'}}^\top Z_w(c))-1\right]. \quad (5.26)$$

Moreover, as in (5.15), we have

$$b^\top\nabla^2\ell(\beta)b = \frac{1}{T}\sum_{w\in\mathcal{V}}\sum_{c\in\mathcal{X}_{-w}}\sum_{s'\neq s} t_w(c;s)\left({b^w_{s,s'}}^\top Z_w(c)\right)^2\exp({\beta^w_{s,s'}}^\top Z_w(c)). \quad (5.27)$$

Let us consider an arbitrary summand in (5.26) and the corresponding one in (5.27). We can focus only on cases where $t_w(c;s)>0$ and ${b^w_{s,s'}}^\top Z_w(c)\neq 0$. From the mean value theorem we obtain for all non-zero $x\in\mathbb{R}$

$$\mathrm{e}^{-|x|}\leq\frac{\mathrm{e}^x-1}{x}\leq\mathrm{e}^{|x|}. \quad (5.28)$$

So using (5.28) we can write

$$\exp(-|{b^w_{s,s'}}^\top Z_w(c)|)\leq\frac{\exp({b^w_{s,s'}}^\top Z_w(c))-1}{{b^w_{s,s'}}^\top Z_w(c)}\leq\exp(|{b^w_{s,s'}}^\top Z_w(c)|).$$

Finally, we multiply each side by the expression $t_w(c;s)({b^w_{s,s'}}^\top Z_w(c))^2\exp({\beta^w_{s,s'}}^\top Z_w(c))$ to conclude the proof of (5.24). Similarly we can prove (5.25). Finally, for any vector $x\in\mathbb{R}^{2d(d-1)}$ we have

$$x^\top\nabla^2\ell(\beta)x = \frac{1}{T}\sum_{w\in\mathcal{V}}\sum_{c\in\mathcal{X}_{-w}}\sum_{s'\neq s} t_w(c;s)\left({x^w_{s,s'}}^\top Z_w(c)\right)^2\exp({\beta^w_{s,s'}}^\top Z_w(c))$$

and

$$x^\top\nabla^2\ell(\beta+b)x = \frac{1}{T}\sum_{w\in\mathcal{V}}\sum_{c\in\mathcal{X}_{-w}}\sum_{s'\neq s} t_w(c;s)\left({x^w_{s,s'}}^\top Z_w(c)\right)^2\exp({\beta^w_{s,s'}}^\top Z_w(c))\exp({b^w_{s,s'}}^\top Z_w(c)).$$

Comparing each summand separately and taking into account the definition of c_b and the inequality (5.28) we finish the proof. □

Lemma 5.14. *Let $\xi>1$ be arbitrary and assume that $\bar{F}(\xi)>0$. Moreover, let us denote $\tau=\dfrac{(\xi+1)|S|\lambda}{2\bar{F}(\xi)}$ and the event $\Omega_2=\{\tau<e^{-1}\}$. Then $\Omega_1\cap\Omega_2\subset A$, where*

$$A=\left\{\|\hat{\beta}-\beta\|_\infty\leq\frac{2\xi e^\eta\lambda}{(\xi+1)\bar{F}(\xi)}\right\}$$

and $\eta<1$ is the smaller solution of the equation $\eta e^{-\eta}=\tau$.

Proof. The proof is similar to Theorem 3.1 of Huang et al. (2013) or Lemma 6 of Miasojedow and Rejchel (2018). Suppose we are on the event $\Omega_1\cap\Omega_2$. Denote again $\tilde{\beta}=\hat{\beta}-\beta$, so by the previous lemma we have $\theta=\dfrac{\tilde{\beta}}{\|\tilde{\beta}\|_1}\in\mathcal{C}(\xi,S)$. Let us consider the function

$$g(t)=\theta^\top\nabla\ell(\beta+t\theta)-\theta^\top\nabla\ell(\beta),\quad t\geq 0.$$

This function is non-decreasing, because the negative log-likelihood function is convex. Hence, for every $t \in (0, \|\tilde{\beta}\|_1)$ we have $g(t) \le g(\|\tilde{\beta}\|_1)$. By Lemma 5.12 on the event Ω_1 we obtain

$$\theta^\top [\nabla \ell(\beta + t\theta) - \nabla \ell(\beta)] + \frac{2\lambda}{\xi + 1} \|\theta_{S^c}\|_1 \le \frac{2\lambda\xi}{\xi + 1} \|\theta_S\|_1. \tag{5.29}$$

Using Lemma 5.13 with $b = t\theta$ and (5.24) we obtain

$$t\theta^\top [\nabla \ell(\beta + t\theta) - \nabla \ell(\beta)] \ge t^2 \exp(-t) \theta^\top \nabla^2 \ell(\beta) \theta, \tag{5.30}$$

in this case $c_b = c_{t\theta} \le \exp(t)$. Now using the definition of CIF $\bar{F}(\xi)$, the fact that θ belongs to the cone $\mathcal{C}(\xi, S)$ and applying the bounds (5.29), (5.30) we get

$$\begin{aligned} t \exp(-t) \frac{\bar{F}(\xi) \|\theta_S\|_1^2}{|S|} &\le t \exp(-t) \theta^\top \nabla^2 \ell(\beta) \theta \le \theta^\top [\nabla \ell(\beta + t\theta) - \nabla \ell(\beta)] \le \\ &\le \frac{2\lambda\xi}{\xi + 1} \|\theta_S\|_1 - \frac{2\lambda}{\xi + 1} \|\theta_{S^c}\|_1 = \\ &= 2\lambda \|\theta_S\|_1 - \frac{2\lambda}{\xi + 1} \le \lambda(\xi + 1) \|\theta_S\|_1^2 / 2. \end{aligned}$$

This means that for any t satisfying (5.29) we have

$$t \exp(-t) \le \frac{(\xi + 1)|S|\lambda}{2\bar{F}(\xi)} = \tau. \tag{5.31}$$

Since, as we mentioned, the function $g(t)$ is non-decreasing, the set of all non-negative t satisfying (5.29) is a closed interval $[0, \tilde{t}]$ for some $\tilde{t} > 0$. Hence, (5.31) implies $\tilde{t} \le \eta$, where η is the smallest solution of the equation $\eta e^\eta = \tau$. Now from (5.29) and (5.30) we obtain

$$\begin{aligned} \|\tilde{\beta}\|_1 e^{-\eta} \le \tilde{t} e^{-\tilde{t}} \le \frac{\tilde{t} \exp(-\tilde{t}) \theta^\top \nabla^2 \ell(\beta) \theta}{\bar{F}(\xi) \|\theta_T\|_1 \|\theta\|_\infty} \le \frac{\theta^\top [\nabla \ell(\beta + \tilde{t}\theta) - \nabla \ell(\beta)]}{\bar{F}(\xi) \|\theta_T\|_1 \|\theta\|_\infty} \le \\ \le \frac{2\lambda\xi}{(\xi + 1)\bar{F}(\xi) \|\theta\|_\infty} = \frac{2\lambda\xi \|\tilde{\beta}\|_1}{(\xi + 1)\bar{F}(\xi) \|\tilde{\beta}\|_\infty}, \end{aligned}$$

which finishes the proof. □

Proof of Theorem 5.5. Fix arbitrary $\varepsilon > 0$ and $\xi > 1$. Then $F(\xi)$ is positive by Lemma 5.7. Thus from Lemma 5.11 we know that $\mathbb{P}\left(\bar{F}(\xi) \ge \zeta_0 F(\xi)\right) \ge 1 - \varepsilon$. Using it with the right-hand side of (5.9) we obtain that $\mathbb{P}(\Omega_2) \ge 1 - \varepsilon$. Moreover, from Lemma 5.9 we have that $\mathbb{P}(\Omega_1) \ge 1 - \varepsilon$. Therefore, Lemmas 5.12 and 5.14 (with $\eta = 1$ for simplicity) imply the inequality

$$\mathbb{P}\left(\|\hat{\beta} - \beta\|_\infty \le \frac{2\xi e \lambda}{(\xi + 1)\bar{F}(\xi)} \right) \ge 1 - 2\varepsilon.$$

Finally, we bound $\bar{F}(\xi)$ from below by $\zeta_0 F(\xi)$. □

Proof of Corollary 5.6. The proof is a simple consequence of the uniform bound (5.10) obtained in Theorem 5.5. Indeed, for arbitrary $w \in \mathcal{V}$, $s \ne s'$ and j-th coordinate of the vector $\beta^w_{s,s'}$ such that $\beta^w_{s,s'}(j) = 0$ we obtain

$$|\hat{\beta}^w_{s,s'}(j)| = |\hat{\beta}^w_{s,s'}(j) - \beta^w_{s,s'}(j)| \le \|\hat{\beta} - \beta\|_\infty \le \delta.$$

Analogously, for each $w \in \mathcal{V}$, $s \neq s'$ and j-th coordinate such that $\beta^w_{s,s'}(j) \neq 0$ we have

$$|\hat{\beta}^w_{s,s'}(j)| \geq |\beta^w_{s,s'}(j)| - |\hat{\beta}^w_{s,s'}(j) - \beta^w_{s,s'}(j)| \geq \beta_{\min} - \|\hat{\beta} - \beta\|_\infty > 2\delta - R \geq \delta,$$

which concludes the proof. □

Proof of Lemma 5.7. Fix $\xi > 1$. For each w and c_{S_w} we have $Z_w(c_{S_w}, 0) = (c_{S_w}, 0)$, so

$$F(\xi) = \inf_{0 \neq \theta \in C(\xi, S)} \sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{c_{S_w} \in \mathcal{X}_{S_w}} \frac{\exp\left((\beta^w_{s,s'})^\top_{S_w} c_{S_w}\right) \left[(\theta^w_{s,s'})^\top_{S_w} c_{S_w}\right]^2}{\|\theta_S\|_1 \|\theta\|_\infty},$$

where $\left(\beta^w_{s,s'}\right)_{S_w}$ and $\left(\theta^w_{s,s'}\right)_{S_w}$ are restrictions of $\beta^w_{s,s'}$ and $\theta^w_{s,s'}$ to coordinates from S_w, respectively. Therefore, we need to bound from below the expression

$$\frac{\sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{c_{S_w} \in \mathcal{X}_{S_w}} \exp\left((\beta^w_{s,s'})^\top_{S_w} c_{S_w}\right) \left[(\theta^w_{s,s'})^\top_{S_w} c_{S_w}\right]^2}{\|\theta_S\|_1 \|\theta\|_\infty} \tag{5.32}$$

for each $\theta \in C(\xi, S)$ and $\theta \neq 0$. First, we restrict the third sum in the numerator of (5.32) to the summands corresponding only to vectors $e_u \in \mathcal{X}_{S_w}$ having 1 on the coordinate corresponding to the node $u \in S_w$ and 0 elsewhere. Then we reduce the numerator of (5.32) to the following form

$$\sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{u \in S_w} \exp\left(\beta^w_{s,s'}(u)\right) \left[\theta^w_{s,s'}(u)\right]^2. \tag{5.33}$$

Recall that $S_w = \left\{u \in -w : \beta^w_{0,1}(u) \neq 0 \text{ or } \beta^w_{1,0}(u) \neq 0\right\}$. Therefore, if $\beta^w_{s,s'}(u) \neq 0$, then $u \in S_w$, so the sum (5.33) can be bounded from below by

$$\sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{u : \beta^w_{s,s'}(u) \neq 0} \exp\left(\beta^w_{s,s'}(u)\right) \left[\theta^w_{s,s'}(u)\right]^2, \tag{5.34}$$

because (5.33) has more summands and the summands are nonnegative. Using reverse Hölder's inequality we replace (5.34) by

$$A_\beta^{-1} \left[\sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{u : \beta^w_{s,s'}(u) \neq 0} |\theta^w_{s,s'}(u)|\right]^2, \tag{5.35}$$

where A_β is

$$A_\beta = \sum_{w \in \mathcal{V}} \sum_{s' \neq s} \sum_{u : \beta^w_{s,s'}(u) \neq 0} \exp\left(-\beta^w_{s,s'}(u)\right).$$

Next, recall that S is the set of nonzero coordinates of β, so (5.35) is just $\|\theta_S\|_1^2 / A_\beta$. Summarizing, the sum (5.32) is bounded from below by

$$\frac{\|\theta_S\|_1}{A_\beta \|\theta\|_\infty} \tag{5.36}$$

for each $\theta \in C(\xi, S)$ and $\theta \neq 0$. The vector θ belongs to the cone $C(\xi, S)$, which implies that $\|\theta_{S^c}\|_\infty \leq \|\theta_{S^c}\|_1 \leq \xi \|\theta_S\|_1$ and

$$\|\theta\|_\infty = \max(\|\theta_S\|_\infty, \|\theta_{S^c}\|_\infty) \leq \max(\|\theta_S\|_1, \xi\|\theta_S\|_1),$$

which gives us $\|\theta\|_\infty \leq \xi \|\theta_S\|_1$. Applying it in (5.36), we finish the proof. □

5.4 Numerical examples

In this section we describe the details of algorithm implementation as well as the results of experimental studies.

5.4.1 Details of implementation

We provide in details practical implementation of the proposed algorithm. The solution of (5.5) depends on the choice of λ. Finding the „optimal" parameter λ and the threshold δ is difficult in practice. Here we solve it using the information criteria (Xue et al., 2012; Pokarowski and Mielniczuk, 2015; Miasojedow and Rejchel, 2018).

First, using (5.12) we write the minimized function in (5.5) as the sum

$$\ell(\theta) - \lambda\|\theta\|_1 = \sum_{w\in\mathcal{V}}\sum_{s\neq s'}\left(\frac{1}{T}\ell^w_{s,s'}(\theta^w_{s,s'}) - \lambda\sum_{u\in -w}|\theta^w_{s,s'}|\right),$$

where $s, s' \in \{0,1\}$. Therefore, for fixed $w \in \mathcal{V}$ and $s, s' \in \{0,1\}$ with $s \neq s'$, the corresponding summand is a function which depends on the vector θ restricted only to its coordinate vector $\theta^w_{s,s'}$ (see notation (5.1)). So, for each triple w and $s \neq s'$ we can solve the problem separately. In our implementation we use the following scheme. We start with computing a sequence of minimizers on the grid, i.e. for any triple $w \in \mathcal{V}$, $s \neq s'$ we create a finite sequence $\{\lambda_i\}_{i=1}^N$ uniformly spaced on the log scale, starting from the largest λ_i, which corresponds to the empty model. Next, for each value λ_i we compute the estimator $\hat{\beta}^w_{s,s'}[i]$ of the vector $\beta^w_{s,s'}$

$$\hat{\beta}^w_{s,s'}[i] = \underset{\theta^w_{s,s'}}{\operatorname{argmin}}\left\{\ell^w_{s,s'}(\theta^w_{s,s'}) + \lambda_i\|\theta^w_{s,s'}\|_1\right\}, \tag{5.37}$$

where as in (5.12)

$$\ell^w_{s,s'}(\theta^w_{s,s'}) = \frac{1}{T}\sum_{c\in\mathcal{X}_{-w}}\left[-n_w(c;s,s'){\theta^w_{s,s'}}^\top Z_w(c) + t_w(c;s)\exp\left({\theta^w_{s,s'}}^\top Z_w(c)\right)\right].$$

The notation $\hat{\beta}^w_{s,s'}[i]$ should not be confused with $\beta^w_{s,s'}(u)$ introduced before. Namely, $\hat{\beta}^w_{s,s'}[i]$ is the i-th approximation of $\beta^w_{s,s'}$, while $\beta^w_{s,s'}(u)$ is the coordinate of $\beta^w_{s,s'}$ corresponding to the node u. To solve (5.37) numerically for a given λ_i we use the FISTA algorithm with backtracking from Beck and Teboulle (2009). The final LASSO estimator $\hat{\beta}^w_{s,s'} := \hat{\beta}^w_{s,s'}[i^*]$ is chosen using the Bayesian Information Criterion (BIC), which is a popular method of choosing the value of λ in the literature (Xue et al., 2012; Miasojedow and Rejchel, 2018), i.e.

$$i^* = \underset{1\leq i\leq 100}{\operatorname{argmin}}\left\{n\ell^w_{s,s'}(\hat{\beta}^w_{s,s'}[i]) + \log(n)\|\hat{\beta}^w_{s,s'}[i]\|_0\right\}.$$

Here $\|\hat{\beta}^w_{s,s'}[i]\|_0$ denotes the number of non-zero elements of $\hat{\beta}^w_{s,s'}[i]$ and n is the number of observed jumps of the process. In our simulations we use $N = 100$.

Finally, the threshold δ is obtained using the Generalized Information Criterion (GIC). The similar way of choosing a threshold was used previously in Pokarowski and Mielniczuk (2015); Miasojedow and Rejchel (2018). For a prespecified sequence of thresholds $\mathscr{D}$ we calculate

$$\delta^* = \underset{\delta \in \mathscr{D}}{\operatorname{argmin}} \left\{ n\ell^w_{s,s'}(\hat{\beta}^{w,\delta}_{s,s'}) + \log(2d(d-1))\|\hat{\beta}^{w,\delta}_{s,s'}\|_0 \right\},$$

where $\hat{\beta}^{w,\delta}_{s,s'}$ is the LASSO estimator $\hat{\beta}^w_{s,s'}$ after thresholding with the level δ.

5.4.2 Simulated data

We consider three models defined as follows. For shortness we denote these models later on as $M1$, $M2$ and $M3$, respectively.

Model 1. All vertices have the "chain structure", i.e. for any node, except for the first one, its set of parents contains only a previous node. Namely, we put $\mathcal{V} = \{1, \ldots, d\}$ and $\mathrm{pa}(k) = \{k-1\}$, if $k > 1$ and $\mathrm{pa}(1) = \emptyset$. We construct CIM in the following way. For the first node the intensities of leaving both states are equal to 5. For the rest of the nodes $k = 2, \ldots, d$, we choose randomly $a \in \{0, 1\}$ and we define $Q_k(c, s, s') = 9$, if $s \neq |c - a|$ and 1 otherwise. In other words, we choose randomly whether the node prefers to be at the same state as its parent ($a = 0$) or not ($a = 1$). Say, the node k prefers to be at the same state as the node $k-1$. Then if these two states coincide, the intensity of leaving the current state is 1, otherwise it is 9. The intensity is defined analogously, when the node k does not prefer to be at the same state as the node $k-1$.

Model 2. The first 5 vertices are correlated, while the remaining vertices are independent. We sample 10 arrows between first 5 nodes by choosing randomly 2 parents for each node. In order to define the intensity matrix, consider the node $w \in \mathcal{V}$ with $\mathrm{pa}(w) \neq \emptyset$ and a configuration $c \in \mathcal{X}_{\mathrm{pa}(w)}$ of the parents states. We denote $|c| = 1$ if all the parents of w are in the state 1, and $|c| = 0$ otherwise. Next we define intensities as follows

$$Q_w(c, s, s') = \begin{cases} 5 & \text{if } \mathrm{pa}(w) = \emptyset, \\ 9 & \text{if } \mathrm{pa}(w) \neq \emptyset,\ s \text{ is preferred state and } |c| = 1, \\ 1 & \text{if } \mathrm{pa}(w) \neq \emptyset,\ s \text{ is preferred state and } |c| = 0, \\ 9 & \text{if } \mathrm{pa}(w) \neq \emptyset,\ s \text{ is not preferred state and } |c| = 0, \\ 1 & \text{if } \mathrm{pa}(w) \neq \emptyset,\ s \text{ is not preferred state and } |c| = 1. \end{cases}$$

As in the previous model, the preferred state is chosen randomly from $\{0, 1\}$. In words, for every node w with $\mathrm{pa}(w) \neq \emptyset$ we choose randomly one state, say 0. In this case, if all parents are 1 the process prefers to be in 1 and if some of the parents are 0 the process prefers to be in 0.

Model 3. All vertices have a „binary tree" structure with arrows from leaves to the root. So, leaves have no parents, while the inner nodes have two parents, with the exception that one node has only a single parent, if d is even. If the node has no parents or its

parents have different states, then the intensity of leaving a state is 5. Otherwise, if a node has only one parent or both parents are in the same state, then the intensity of leaving a state are computed as in Model 1.

The model $M1$ has a simple structure, which involves all vertices and satisfies our assumption (5.2). The model $M2$ corresponds to a dense structure on a small subset of vertices and does not satisfy assumption (5.2). Another potential difficulty is related to possible feedback loops, which are usually hard to recognize. Therefore, we also consider model $M2+$, which looks like $M2$, but contains the interaction terms and fulfills (5.2). The model $M3$ has slightly more complex structure than $M1$, but it also satisfies our assumption (5.2).

We consider two cases: $d = 20$ and $d = 50$ for all four models. So, the considered number of possible parameters of the model (the size of β) is $2d^2 = 800$ or 5000, respectively. For model with interactions, number of possible parameters is $d^2(d+1) = 8400$ or 127500. We use $T = 10$ and 50 for all models and we replicate simulations 100 times for each scenario. In Table 5.1 we present averaged results of the simulations in terms of three quality measures

- **power**, which is a proportion of correctly selected edges;
- **false discovery rate (FDR)**, which is a fraction of incorrectly selected edges among all selected edges;
- **model dimension (MD)**, which is a number of selected edges.

We observe that in the models $M1$ and $M3$ the results of experiments confirm that the proposed method works in a satisfactory way. For $T = 10$ the algorithm has high power and its FDR is not large. The final model selected by our procedure is slightly too small (it does not discover a few existing edges). When we increase observation time ($T = 50$), then our estimator behaves almost perfectly.

The model $M2$ is much more difficult and this fact has a direct impact on simulation results. Namely, for $T = 10$ the power of the algorithm is relatively low with FDR also being rather small. The procedure performs slightly better when we take $T = 50$. However, for both observation times the estimator cannot find the true edges in the graph. One of the reasons of such behaviour is that in $M2$ the dependence structure in CIM is not additive in parents. This fact combined with possible feedback loops leads to recovering existing edges, but having the opposite to the true ones directions. Looking deeper into the results for a few examples chosen from our experiments we confirm this claim, i.e. the edges between nodes are correctly selected, but their directions are wrong. Therefore, we can conclude that in the complex model $M2$ our estimator seems at least to be able to recognize interactions between nodes, which is important in many practical problems on its own. The results for $M2+$ confirm that the performance of our method increases, when we consider more complex parametrization with interaction terms.

Table 5.1: Results for simulated data. In the model $M1$ and $M3$ the true dimension is 19 for $d = 20$ and 49 for $d = 50$. In the model $M2$ the true model dimension is 10.

Model	d	T	Power	FDR	MD
$M1$	20	10	0.86	0.03	16.8
		50	1	0.02	19.3
	50	10	0.61	0.01	30.3
		50	1	0.01	49.3
$M2$	20	10	0.16	0.2	2
		50	0.78	0.04	8.1
	50	10	0.10	0.15	1.28
		50	0.62	0.02	6.4
$M2+$	20	10	0.35	0.1	3.9
		50	0.9	0.02	9.2
	50	10	0.17	0.08	2
		50	0.68	0.01	6.9
$M3$	20	10	0.17	0.1	3.7
		50	0.97	0.01	18.7
	50	10	0.6	0.09	3.2
		50	0.88	0.003	43

5.5 Extension of the results

In this chapter we proposed the method for structure learning of CTBNs. We confirmed the good performance of our method both theoretically and experimentally. To simplify the notation and help the reader to follow our reasoning we restricted ourselves to graphs with only two possible states for each node. However, our results can be generalized in a straightforward way to any finite graphs by extending β to other possible jumps and possible values of parents. In terms of the explanatory variable, it is equivalent to the standard encoding of qualitative variables in linear or generalized linear models. To demonstrate the generalization more clearly we present an example similar to Example 5.1 presented in the very beginning of this chapter.

Example 5.15. We consider CTBN with three nodes A, B and C. Let their state spaces be $\mathcal{X}_A = \{0, 1, 2\}$, $\mathcal{X}_B = \{0, 1, 2, 3\}$ and $\mathcal{X}_C = \{0, 1, 2\}$, respectively. Then for the node A we define the function Z_A as

$$Z_A(b, c) = [1, \mathbb{I}(b = 1), \mathbb{I}(b = 2), \mathbb{I}(b = 3), \mathbb{I}(c = 1), \mathbb{I}(c = 2)]^\top$$

for each $b \in \{0, 1, 2, 3\}$ and $c \in \{0, 1, 2\}$. Analogously, we can define representations for the remaining nodes:

$$Z_B(a, c) = [1, \mathbb{I}(a = 1), \mathbb{I}(a = 2), \mathbb{I}(c = 1), \mathbb{I}(c = 2)]^\top$$

for each $a \in \{0,1,2\}$ and $c \in \{0,1,2\}$ and

$$Z_C(a,b) = [1, \mathbb{I}(a=1), \mathbb{I}(a=2), \mathbb{I}(b=1), \mathbb{I}(b=2), \mathbb{I}(b=3)]^\top$$

for each $a \in \{0,1,2\}$ and $b \in \{0,1,2,3\}$.

Therefore, for each node w and for each configuration of parents' states (e.g. for the node A and values in the nodes B and C) the value of the function $Z_w(\cdot,\cdot)$ is still a binary vector with the dimension equal to the sum of the numbers of states in all parents nodes with subtracted number of nodes and added 2. In this example the parameter vector (5.1) is defined as

$$\beta = \left(\beta^A_{0,1}, \beta^A_{1,0}, \beta^A_{0,2}, \beta^A_{2,0}, \beta^A_{1,2}, \beta^A_{2,1}, \beta^B_{0,1}, \beta^B_{1,0}, \beta^B_{0,2}, \ldots, \beta^B_{3,1}, \beta^B_{2,3}, \beta^B_{3,2}, \beta^C_{0,1}, \ldots, \beta^C_{2,1}\right)^\top.$$

With a slight abuse of notation, the vector $\beta^A_{0,1}$ is given as

$$\beta^A_{0,1} = \left[\beta^A_{0,1}(1), \beta^A_{0,1}(B=1), \beta^A_{0,1}(B=2), \beta^A_{0,1}(B=3), \beta^A_{0,1}(C=1), \beta^A_{0,1}(C=2)\right]^\top,$$

and we interpret it as follows: if all $\beta^A_{0,1}(B=1)$, $\beta^A_{0,1}(B=2)$ and $\beta^A_{0,1}(B=3)$ are equal to 0, then the intensity of the change from the state 0 to 1 at the node A does not depend on the state at the node B. Similarly, the coordinates $\beta^A_{0,1}(C=1)$ and $\beta^A_{0,1}(C=2)$ describe the dependence between the above intensity and the state at the node C, and $\beta^A_{0,1}(1)$ corresponds to the intercept. For the node B the coordinates of the vector $\beta^B_{1,3} = \left[\beta^B_{1,3}(1), \beta^B_{1,3}(A=1), \beta^B_{1,3}(A=2), \beta^B_{1,3}(C=1), \beta^B_{1,3}(C=2)\right]$ describe the relation between the intensity of the jump from the state 1 to 3 at the node B to the intercept, states at the nodes A and C, respectively.

Our results can be also easily generalized to the case, where we consider not only additive effect in (5.2), but also interactions between parents. Let us again use an example.

Example 5.16. As previously we consider CTBN with three nodes A, B and C with corresponding state spaces $\mathcal{X}_A = \{0,1\}$, $\mathcal{X}_B = \{0,1,2\}$ and $\mathcal{X}_C = \{0,1\}$. In a linear model we have the following Z_w functions:

$$\begin{aligned}
Z_A(b,c) &= [1, \mathbb{I}(b=1), \mathbb{I}(b=2), \mathbb{I}(c=1)]^\top,\\
Z_B(a,c) &= [1, \mathbb{I}(a=1), \mathbb{I}(c=1)]^\top,\\
Z_C(a,b) &= [1, \mathbb{I}(a=1), \mathbb{I}(b=1)]^\top
\end{aligned}$$

for each $a, c \in \{0,1\}$ and $b \in \{0,1,2\}$. Then after we add pairwise interactions to the linear model the functions above take the form

$$\begin{aligned}
Z_A(b,c) &= [1, \mathbb{I}(b=1), \mathbb{I}(b=2), \mathbb{I}(c=1), \mathbb{I}(b=1,c=1), \mathbb{I}(b=2,c=1)]^\top,\\
Z_B(a,c) &= [1, \mathbb{I}(a=1), \mathbb{I}(c=1), \mathbb{I}(a=1,c=1)]^\top,\\
Z_C(a,b) &= [1, \mathbb{I}(a=1), \mathbb{I}(b=1), \mathbb{I}(a=1,b=1)]^\top.
\end{aligned}$$

For models with more nodes we can also take into account more complex interactions.

Chapter 6

Structure learning for CTBNs for incomplete data

In the previous chapter we considered the case when we observe CTBN at each moment of time. Under this assumption we introduced a novel method of structure learning. In this chapter we show that our method can be adapted to partially observed and noisy data. In the case of partial observations we need to introduce the observation and the likelihood of the observed data given a hidden trajectory of a process. We can again parametrize CIM by (5.2). However, in this case the problem (5.5) becomes more challenging and leads to the following two problems. First, the theoretical analysis becomes more challenging because the loss function is not convex. Second, the likelihood function can not be calculated explicitly, hence, it is difficult to obtain from the computational perspective.

In our solution we formulate the EM algorithm for this case, where the expectation step is standard and concerns the calculation of the expected log-likelihood. The maximization step is performed in the same way as for the complete data. Since the density belongs to the exponential family, the E-step requires to compute the expected values of sufficient statistics, which is done with the MCMC algorithm developed in Rao and Teh (2013). In addition, the results from Majewski et al. (2018) combined with Miasojedow and Niemiro (2017) are used in the analysis of the Monte Carlo scheme.

6.1 Introduction and notation

Let $\boldsymbol{t} = (t_0, t_1, ..., t_n)$ with $0 = t_0 < t_1 < ... < t_n$ and $\boldsymbol{S} = (\boldsymbol{S}_0, \boldsymbol{S}_1, ..., \boldsymbol{S}_n)$ describe the full trajectory X of the process on the interval $[0, T]$ ($\boldsymbol{t}$ denotes times of jumps, $\boldsymbol{S}$ is a skeleton, where $\boldsymbol{S}_i \in \mathcal{X}$ is a state at the moment t_i). Let $\mathbb{X}$ denote the set of all possible trajectories of the process. Let us consider the case when instead of observing the full trajectory X we have access only to the partial and noisy data Y with the conditional density $p(Y \mid X)$. More precisely, we assume that Y is represented by the observation of X at times $t_1^{obs}, \ldots, t_k^{obs}$ with the likelihoods $g_j(\boldsymbol{S}_{j^{\boldsymbol{t}}})$ for $j = 1, \ldots, k$, where

$$j^{\boldsymbol{t}} = \max\{i : t_i \leq t_j^{obs}\}.$$

We assume that $0 < C < g_j < \tilde{C}$ for $1 \leq j \leq k$. In this case the full density is given by

$$p_\beta(X, Y) = p(Y \mid X) p_\beta(X),$$

where $p_\beta(X)$ is given by (2.8). Observe that the dependence of $p_\beta(X)$ on β is mediated through our assumption (5.2), which can be inserted into (2.9). For the clarity of presentation we assume that $p(Y \mid X)$ is known, however, the adaptation of our method to the case where $p(Y \mid X)$ depends also on some unknown parameters is straightforward. The negative of the log-likelihood function in this case is given by

$$\ell(\beta) = -\log\left(\int_{\mathbb{X}} p_\beta(X, Y) dX\right),$$

where symbol dX means the summation first over all possible numbers of jumps of the trajectory X, then over all possible jumps, and the integration with respect to times of jumps. More precisely,

$$\int f(X) dX = \sum_{n=0}^{\infty} \sum_{\boldsymbol{S}_1 \in \mathcal{X}} \cdots \sum_{\boldsymbol{S}_n \in \mathcal{X}} \int_0^{t_2} \int_{t_1}^{t_3} \cdots \int_{t_{n-1}}^{T} f(n, t_1, \ldots, t_n, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n) dt_1 \ldots dt_n.$$

Again we can define the estimator of the parameter vector β, as previously,

$$\hat{\beta} = \underset{\theta \in \mathbb{R}^{2d(d-1)}}{\operatorname{argmin}} \left\{\ell(\theta) + \lambda \|\theta\|_1\right\}. \tag{6.1}$$

Since we are not able to compute $\ell(\theta)$ analytically, we need to propose an efficient algorithm for finding $\hat{\beta}$. One of the efficient algorithms of solving complex optimization problems of the form (6.1) is the projected Proximal Gradient Descent (p-PGD) algorithm (see for example Beck and Teboulle (2009) and Majewski et al. (2018)). For a closed compact convex set $\mathcal{K}$ by $\prod_{\mathcal{K}}(a)$ we denote the projection of a onto $\mathcal{K}$. Then p-PGD is defined iteratively by

$$\beta_{k+1} = \prod\nolimits_{\mathcal{K}} \left(\operatorname{prox}_{\gamma_k, \lambda\|\cdot\|_1}(\beta_k - \gamma_k \nabla \ell(\beta_k))\right), \tag{6.2}$$

where $\{\gamma_k\}_{k \geq 0}$ is a sequence of step-sizes, and "prox" denotes the proximal operator defined for any convex function g by

$$\operatorname{prox}_{\gamma, g}(x) = \underset{y}{\operatorname{argmin}} \left(g(y) + \frac{1}{2\gamma}\|y - x\|^2\right).$$

In the case of ℓ_1 penalty, i.e. $g = \lambda\|\cdot\|_1$, the proximal operator is just a soft-thresholding operator. Element-wise soft-thresholding operator $S_\lambda : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$S_\lambda(x_i) = [|x_i| - \lambda]_+ \operatorname{sgn}(x_i),$$

where $[\cdot]_+$ denotes the positive part.

In our case we are not able to evaluate the gradient $\nabla \ell$ explicitly and we will use stochastic version of the projected proximal gradient algorithm (p-SPGD), where $\nabla \ell$

is replaced by its Monte Carlo approximation. Under the regularity conditions given for example in Assumption $AD.1$ in Douc et al. (2014) we derive that the gradient of the negative log-likelihood is given by

$$\begin{aligned}\nabla\ell(\beta) &= -\nabla \log\left(\int p_\beta(X,Y)dX\right) = -\frac{\nabla\int p_\beta(X,Y)dX}{\int p_\beta(X,Y)dX} = \\ &= -\frac{\int p_\beta(X,Y)\nabla\log(p_\beta(X,Y))dX}{\int p_\beta(X,Y)dX} = \\ &= -\int \frac{\nabla\log(p_\beta(X,Y))p_\beta(X,Y)}{\int p_\beta(X,Y)dX}dX = \\ &= -\int \nabla\log(p_\beta(X,Y))p_\beta(X\mid Y)dX = \\ &= -\mathbb{E}(\nabla\log(p_\beta(X,Y))\mid Y).\end{aligned} \tag{6.3}$$

This equation is sometimes referred as *Fisher's identity*. Now based on (6.3) we can approximate $\nabla\ell$ by

$$\Phi(\beta, X^1,\ldots,X^m) = -\frac{1}{m}\sum_{i=1}^{m}\nabla\log(p_\beta(X^i,Y)), \tag{6.4}$$

where $X^1,\ldots,X^m$ is a set of subsequent states of the Markov chain with the stationary distribution $\pi_\beta = p_\beta(X\mid Y) \propto p_\beta(X,Y)$, where in particular each of $X^1,\ldots,X^m$ is a trajectory of the process X. To generate this Markov chain we will use the procedure described below.

6.2 Sampling the Markov chain with Rao and Teh's algorithm

Consider the set $\mathcal{M}$ of all possible intensity matrices of Markov jump processes with the state space $\mathcal{X}$ equipped with some matrix distance. Therefore, for any $Q\in\mathcal{M}$ and $\mathbf{s}\in\mathcal{X}$ each element $Q(\mathbf{s},\mathbf{s})$ on the diagonal of Q is nonpositive, and otherwise it is non-negative. Let $\mathcal{L}\subset\mathcal{M}$ be a compact set and choose $\eta > \sup_{Q\in\mathcal{L}}\max_{\mathbf{s}\in\mathcal{X}} Q(\mathbf{s})$, where we used the notation $Q(\mathbf{s}) = -Q(\mathbf{s},\mathbf{s})$ introduced in Section 2.5. To generate the Markov chain parametrized by $Q\in\mathcal{L}$ we will use Rao and Teh's algorithm, which uses the idea of *uniformization* and the notion of *virtual jumps* (Rao and Teh (2013)). A virtual jump in a trajectory described by a pair of time points $\boldsymbol{t}$ and a corresponding skeleton $\boldsymbol{S}$ means that for two subsequent time points t_i and t_{i+1} we have $\boldsymbol{S}_i = \boldsymbol{S}_{i+1}$. Simply put, it means that the process can jump from a certain state back to the same state. Here we provide a comprehensive description of a single iteration of the algorithm. Given an arbitrary trajectory $(\boldsymbol{t},\boldsymbol{S})$ of X, such that $Y(t_i^{obs}) = X(t_i^{obs})$ for $1\le i\le k$, we generate another trajectory $(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})$ with this property using the following procedure:

1. We generate times of virtual jumps $\boldsymbol{v}$ from piecewise homogeneous Poisson process with the intensity $\eta - Q(X(t))$, which means that for every interval $[t_i, t_{i+1})$ we

sample a number k_i of virtual jumps from the Poisson distribution with the parameter equal to $(\eta - Q(\boldsymbol{S}_i))(t_{i+1} - t_i)$; then the times of virtual jumps are uniformly distributed on $[t_{i-1}, t_i)$.

2. We add virtual jumps to the trajectory in a correct order and we obtain new times of jumps $\boldsymbol{t'} = \boldsymbol{t} \cup \boldsymbol{v}$ and a new corresponding skeleton $\boldsymbol{S'} = (\boldsymbol{S}'_0, ..., \boldsymbol{S}'_{n'})$, where n' is the number of elements of $\boldsymbol{t'}$. Therefore $\boldsymbol{S'} = (\boldsymbol{S}_0, \ldots, \boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n, \ldots, \boldsymbol{S}_n)$ with k_0 instances of $\boldsymbol{S}_0$, k_1 instances of $\boldsymbol{S}_1$, etc. In other words, this skeleton contains the same jumps at the same time points as $\boldsymbol{S}$ and for every interval $[t_i, t_{i+1})$ the states in $\boldsymbol{S'}$ are equal to $\boldsymbol{S}_{i-1}$.

3. We sample a new skeleton $\bar{\boldsymbol{S}}$ of the size n' using the standard Forward-Filtering Backward-Sampling algorithm (FFBS), which for completeness is provided at the end of this chapter in Section 6.5. Thus, the resulting distribution of the skeleton will be

$$\frac{\nu(\boldsymbol{S}_0) \prod_{i=1}^{n'} P(\boldsymbol{S}'_{i-1}, \boldsymbol{S}'_i) \prod_{j=1}^{k} g_j(\boldsymbol{S}'_{j t'})}{\sum_{\boldsymbol{S}} \nu(\boldsymbol{S}_0) \prod_{i=1}^{n'} P(\boldsymbol{S}_{i-1}, \boldsymbol{S}_i) \prod_{j=1}^{k} g_j(\boldsymbol{S}_{j t'})},$$

where ν is the initial distribution (which does not depend on Q), and

$$P(\mathbf{s}, \mathbf{s}') = \begin{cases} \dfrac{Q(\mathbf{s}, \mathbf{s}')}{\eta}, & \text{if } \mathbf{s} \neq \mathbf{s}', \\ 1 - \dfrac{Q(\mathbf{s})}{\eta}, & \text{if } \mathbf{s} = \mathbf{s}', \end{cases} \tag{6.5}$$

where $\mathbf{s}, \mathbf{s}' \in \mathcal{X}$. The summation in the denominator is taken with respect to all possible skeletons of size n' containing virtual jumps.

4. From the trajectory $(\boldsymbol{t'}, \bar{\boldsymbol{S}})$ we discard newly acquired virtual jumps (i.e. we remove $\bar{\boldsymbol{S}}_i$ such that $\bar{\boldsymbol{S}}_i = \bar{\boldsymbol{S}}_{i-1}$) and we obtain a new set $\bar{\boldsymbol{t}}$ of times of jumps and the resulting trajectory $(\bar{\boldsymbol{t}}, \bar{\boldsymbol{S}})$, which describes the desired Markov chain.

The procedure above describes one step of the algorithm and Rao and Teh (2013) showed that both trajectories $(\boldsymbol{t}, \boldsymbol{S})$ and $(\bar{\boldsymbol{t}}, \bar{\boldsymbol{S}})$ describe the same MJP. Simply put, the Poisson rate η dominates the leaving rates of all states of the MJP and the new skeleton will contain more events than there are jumps in the MJP path. The corresponding trajectory is regarded as a redundant representation of a pure-jump process that always jumps to a new state. Note, that our new stochastic matrix P defined in (6.5) allows self-transitions (we refer to them as virtual jumps), and as η increases their number grows as well. These self-transitions will be discarded in the final step of the algorithm, which compensates for an increased number of events.

The step of the algorithm is the composition of two Markov kernels. First we add

virtual jumps according to the kernel M_Q^J defined by

$$M_Q^J((\boldsymbol{t},\boldsymbol{S}),(\tilde{\boldsymbol{t}},\tilde{\boldsymbol{S}})) = \mathbb{I}(\bar{\boldsymbol{t}} = \boldsymbol{t} \cup \boldsymbol{v})$$
$$\prod_{i=0}^{n-1} \left\{ [(\eta - Q(s_i))(t_{i+1} - t_i)]^{k_i} \, e^{-(\eta - Q(s_i))(t_{i+1}-t_i)} \mathbb{I}(t_i < v_{i,1} < \cdots < v_{i,k_i} < t_{i+1}) \prod_{l=j_i}^{j_{i+1}} \mathbb{I}(\tilde{\boldsymbol{S}}_l = \boldsymbol{S}_i) \right\}. \tag{6.6}$$

Next we draw a skeleton according to the kernel M_Q^S given by

$$M_Q^S((\tilde{\boldsymbol{t}},\tilde{\boldsymbol{S}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})) = \mathbb{I}(\bar{\boldsymbol{t}} = \tilde{\boldsymbol{t}}) \frac{\nu(\bar{\boldsymbol{S}}_0) \prod_{i=1}^n P(\bar{\boldsymbol{S}}_{i-1}, \bar{\boldsymbol{S}}_i) \prod_{j=1}^k g_j(\bar{\boldsymbol{S}}_{j^{\bar{t}}})}{\sum_{\tilde{\boldsymbol{S}}} \left\{ \nu(\tilde{\boldsymbol{S}}_0) \prod_{i=1}^n P(\tilde{\boldsymbol{S}}_{i-1}, \tilde{\boldsymbol{S}}_i) \prod_{j=1}^k g_j(\tilde{\boldsymbol{S}}_{j^{\tilde{t}}}) \right\}}, \tag{6.7}$$

where n denotes the length of the vector $\tilde{\boldsymbol{t}}$. The dependence of M_Q^S on Q is hidden in the definition of P.

Note that adding virtual jump times $\boldsymbol{v}$ defines the new skeleton uniquely and we denote it by $\boldsymbol{S}^{\boldsymbol{v}}$. Therefore, since sampling a new skeleton does not change times of jumps, the full kernel $M_Q = M_Q^S M_Q^J$ is given by

$$M_Q((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})) = M_Q^J((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})) M_Q^S((\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})). \tag{6.8}$$

The kernel M_Q acts on a function $f(\boldsymbol{t},\boldsymbol{S})$ as follows

$$M_Q f(\boldsymbol{t},\boldsymbol{S}) = \mathbb{E}\left[f(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}}) \mid (\boldsymbol{t},\boldsymbol{S}) \right] =$$
$$= \sum_{k_0=0}^{\infty} \cdots \sum_{k_{n-1}=0}^{\infty} \int \prod_{i=0}^{n-1} \mathbb{I}\{t_i < v_{i,1} < \cdots < v_{i,k_i} < t_{i+1}\} \sum_{\bar{\boldsymbol{S}}} f(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}}) M_Q((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})) d\boldsymbol{v}.$$

where the inside sum is taken with respect to all possible skeletons of the same length as $\bar{\boldsymbol{t}}$. Further, when it does not cause any confusion, for convenience we often denote any single trajectory $(\boldsymbol{t},\boldsymbol{S})$ as X, and as X_i – the trajectory obtained after i-th iteration of the algorithm with the starting trajectory X_0. Then for any two adjacent trajectories X_i and X_{i+1} the function $M_Q f(X_i)$ stands for $\mathbb{E}\left[f(X_{i+1}) \mid X_i \right]$. Moreover, for any trajectory X let $V(X) = V(\boldsymbol{t},\boldsymbol{S}) = n+1$ (recall that n denotes the number of jumps on the trajectory X described by $\boldsymbol{t}$ and $\boldsymbol{S}$).

6.3 Structure learning via penalized maximum likelihood function

Recall the assumption (5.2) introduced in the previous chapter

$$\log(Q_w(c,s,s')) = {\beta^w_{s,s'}}^{\top} Z_w(c) \, .$$

For a given parameter vector $\beta \in \mathbb{R}^{2d(d-1)}$ this defines the intensity matrix Q and this mapping can be regarded as an isometry. So, if β belongs to a compact set $\mathcal{K} \in \mathbb{R}^{2d(d-1)}$,

then Q belongs to some compact set $\mathcal{L} \in \mathcal{M}$ and the construction above is still valid. In this case, we will frequently write M_β instead of M_Q. Now we introduce the main theoretical result regarding the convergence of our algorithm.

Theorem 6.1. *Let $\mathcal{K} \in \mathbb{R}^{2d(d-1)}$ be some compact convex set. Denote as $N_\mathcal{K}(\beta)$ the normal cone to the set $\mathcal{K}$ at the point β*

$$N_\mathcal{K}(\beta) = \{a \in \mathbb{R}^{2d(d-1)} : \langle a, z - \beta\rangle \text{ for all } z \in \mathcal{K}\}.$$

Moreover, denote

$$\mathcal{S} = \{\beta \in \mathcal{K} \colon 0 \in \nabla\ell(\beta) + \lambda\partial\|\beta\|_1 - N_\mathcal{K}\},$$

where $\partial\|\beta\|_1$ denotes the subgradient. Suppose that $(\ell+\lambda\|\cdot\|_1)(\mathcal{S})$ has non-empty interior. Assume also $\mathbb{E}V^2(X_0) < \infty$. Let the sequence $\{\gamma_k, k \in \mathbb{N}\}$ satisfy $\gamma_k > 0$, $\lim\limits_{k\to\infty} \gamma_k = 0$, and

$$\sum_{k=1}^{\infty} \gamma_k = \infty, \quad \sum_{k=1}^{\infty} \gamma_k^2 < \infty, \quad \sum_{k=1}^{\infty} |\gamma_k - \gamma_{k+1}| < \infty.$$

Let $\{\beta_k\}$ be a sequence generated by the projected stochastic proximal gradient descent as in (6.2). Then

$$\operatorname{dist}(\beta_k, \mathcal{S} \cap \mathcal{K}) \overset{k\to\infty}{\to} 0 \quad a.s.$$

Remark 6.2.

(1) Obviously, $\lim\limits_{k\to\infty} \gamma_k = 0$ follows from the convergence $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$.

(2) The symbol $(\ell + \lambda\|\cdot\|_1)(\mathcal{S})$ should be understood as the image of the set $\mathcal{S}$ under the function $h(\theta) = \ell(\theta) + \lambda\|\theta\|_1$.

The theorem is a consequence of Theorem 5.4 of Majewski et al. (2018). It states that the sequence of parameter vectors β_k generated by the projected SPGD algorithm converges almost surely to a stationary point of the function being minimized, where instead of the gradient of the negative log-likelihood we used its Markov chain approximation.

Before proving the theorem we need a few auxiliary results and some additional notation. For any function f of the trajectory X and any signed-measure μ, we define its V-variation by

$$\|\mu\|_V = \sup_{|f|\le V} |\mu(f)|, \tag{6.9}$$

where

$$\mu(f) = \int_{\mathbb{X}} f d\mu,$$

and the integration is over all possible trajectories of the process. Also we denote

$$|f|_V = \sup_{X\in\mathbb{X}} \frac{|f(X)|}{V(X)}, \tag{6.10}$$

where supremum is taken with respect to all possible trajectories.

First we prove three auxiliary lemmas concerning the kernels M_Q defined by (6.8).

Lemma 6.3. *Fix a compact set $\mathcal{L} \in \mathcal{M}$. Then there exist constants $\rho_1, \rho_2 \in (0,1)$ and $b_1, b_2 < \infty$ such that for any trajectory X we have*

$$\sup_{Q \in \mathcal{L}} M_Q V(X) \leq \rho_1 V(X) + b_1$$

and

$$\sup_{Q \in \mathcal{L}} M_Q V^2(X) \leq \rho_2 V^2(X) + b_2.$$

Proof. The proof is a simple extension of proofs of Lemma 5 and Proposition 6 in Miasojedow and Niemiro (2017). First we note that in the first step of Rao and Teh's algorithm we do not add any new jumps. This implies that in order to obtain the desired bounds on $M_\beta V(X)$ we simply need to bound the expectation $\mathbb{E}V(X')$ of the jumps for the trajectory X' obtained on the Step 3. Analogously, instead of bounding $M_\beta V^2(X)$ we bound $\mathbb{E}V^2(X')$. Indeed we have

$$M_\beta V^2(X) = \mathbb{E}[V^2(X') \mid X] = \mathbb{E}[\mathbb{E}(V^2(X') \,\big|\, |X'| = n' + 1) \mid X]$$

and $\mathbb{E}(|X'| = n'+1 \mid X) \leq n+1+\eta T$ with $V(X) = n+1$, where $|X'|$ denotes the number of states in the trajectory X'. For the trajectory $X' = (\bar{\boldsymbol{t}}, \boldsymbol{S}')$ we have

$$V(X') = 1 + \sum_{i=0}^{n'-1} \mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1}).$$

Therefore, we get

$$\mathbb{E}V^2(X') = 1 + 2\mathbb{E}\left[\sum_{i=0}^{n'-1} \mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1})\right] + \mathbb{E}\left[\sum_{i \neq j} \mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1})\mathbb{I}(\boldsymbol{S}'_j \neq \boldsymbol{S}'_{j+1})\right]. \tag{6.11}$$

Applying Lemma 2 of Miasojedow and Niemiro (2017) together with the definition (6.5), the definition of η and assumptions on likelihoods $g_j(\boldsymbol{S}_{jt})$, for each $i = 0, \ldots, n'-1$ we obtain

$$\mathbb{P}\left(\boldsymbol{S}'_i = \mathrm{s} \mid \boldsymbol{S}'_{i+1} = \mathrm{s}\right) \geq \delta_i > 0.$$

This is a lower bound for the backward transition probability used by the FFBS algorithm. An analogous inequality for the forward transition probability is also true. Hence,

$$\begin{aligned} \mathbb{E}\left(\mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1})\right) &= \mathbb{E}\left[\mathbb{E}(\mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1}) \mid \boldsymbol{S}'_{i+1})\right] = \\ &= \mathbb{P}\left(\boldsymbol{S}'_i \neq \mathrm{s} \mid \boldsymbol{S}'_{i+1} = \mathrm{s}\right) \leq 1 - \delta_i, \end{aligned} \tag{6.12}$$

which means that we can bound the second term on the RHS of (6.11) from above by $2\sum_{i=0}^{n'-1}(1-\delta_i)$. Also, for each $i \neq j$ we have

$$\mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1})\mathbb{I}(\boldsymbol{S}'_j \neq \boldsymbol{S}'_{j+1}) \leq \mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1}). \tag{6.13}$$

Thus, using (6.12) and (6.13), the third term on the RHS of (6.11) can be bounded by

$$\mathbb{E}\left[\sum_{j=0}^{n'-1}\sum_{\substack{i=0\\ i\neq j}}^{n'-1}\mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1})\right] = \sum_{j=0}^{n'-1}\sum_{\substack{i=0,\\ i\neq j}}^{n'-1}\mathbb{E}\left[\mathbb{E}(\mathbb{I}(\boldsymbol{S}'_i \neq \boldsymbol{S}'_{i+1}) \mid \boldsymbol{S}'_{i+1})\right] =$$

$$= \sum_{j=0}^{n'-1}\sum_{\substack{i=0,\\ i\neq j}}^{n'-1}\mathbb{P}(\boldsymbol{S}'_i \neq \mathbf{s} \mid \boldsymbol{S}'_{i+1} = \mathbf{s}) \leq$$

$$\leq \sum_{j=0}^{n'-1}\sum_{\substack{i=0,\\ i\neq j}}^{n'-1}(1-\delta_i).$$

Combining both bounds we obtain

$$\mathbb{E}V^2(X') \leq 1 + 2n' - 2\sum_{i=0}^{n'-1}\delta_i + n'(n'-1) - \sum_{j=0}^{n'-1}\sum_{\substack{i=0,\\ i\neq j}}^{n'-1}\delta_i \leq$$

$$\leq (1-\delta)(n'+1)^2 + 1 \leq (1-\delta)(n+1) + b,$$

where $\delta = \min_i\{\delta_i\} \in (0,1)$ and b is some finite constant. This finishes the proof of the second part of the lemma.

The first inequality can be shown either analogously to the second one or by applying Jensen's inequality. Indeed,

$$(M_Q V(X))^2 \leq M_Q V^2(X) \leq \rho_2 V^2(X) + b_2,$$

which in turn implies that

$$M_Q V(X) \leq \sqrt{\rho_2 V^2(X) + b_2} \leq \sqrt{\rho_2} V(X) + \sqrt{b_2},$$

which concludes the proof. □

Lemma 6.4. *If* $\mathbb{E}[V(X_0)] < \infty$, *then* $\sup_{n\geq 1} M_Q V(X_n) < \infty$. *If in addition* $\mathbb{E}[V^2(X_0)] < \infty$, *then* $\sup_{n\geq 1} M_Q V^2(X_n) < \infty$.

Proof. Recall that X_n is the trajectory obtained after n-th iteration of Rao and Teh's algorithm starting from the trajectory X_0. As previously we can consider $\mathbb{E}V(X_{n+1})$ instead of $M_Q V(X_n)$. Hence, by the previous lemma we have

$$\mathbb{E}[V(X_{n+1})] = \mathbb{E}[\mathbb{E}[V(X_{n+1}) \mid X_n]] = \mathbb{E}[M_Q V(X_n)] \leq \rho \mathbb{E}V(X_n) + b,$$

where $\rho \in (0,1)$ and $b < \infty$. Then, by iterating this majorization procedure recursively we get

$$\mathbb{E}[V(X_{n+1})] \leq \rho^{n+1}\mathbb{E}V(X_0) + b\sum_{i=1}^{n+1}\rho^i \leq \rho \mathbb{E}V(X_0) + \frac{b}{1-\rho}\,.$$

Since the RHS of this inequality does not depend on n, then $\mathbb{E}[V(X_{n+1})]$ is bounded by a finite constant. This concludes the proof for the first bound. The second inequality can be shown analogously using the bound for $\sup_{Q\in\mathcal{L}} M_Q V^2(X)$ in Lemma 6.3. □

Lemma 6.5. *For any compact set $\mathcal{L} \subset \mathcal{M}$ there exists $C \in (0,\infty)$ such that for any $Q, \tilde{Q} \in \mathcal{L}$ and all trajectories X we have*

$$\|M_Q(X,\cdot) - M_{\tilde{Q}}(X,\cdot)\|_V \le CV(X)\|Q - \tilde{Q}\|.$$

Proof. For any $Q, \tilde{Q} \in \mathcal{L}$ the expression of interest (see the definitions (6.6)-(6.9)) can be bounded by the sum of two terms as follows

$$\begin{aligned}
&\sup_{|f|\le V} \left|M_Q f(\boldsymbol{t},\boldsymbol{S}) - M_{\tilde{Q}} f(\boldsymbol{t},\boldsymbol{S})\right| = \sup_{|f|\le V} \left|M^S_Q M^J_Q f(\boldsymbol{t},\boldsymbol{S}) - M^S_{\tilde{Q}} M^J_{\tilde{Q}} f(\boldsymbol{t},\boldsymbol{S})\right| \\
&\le \sup_{|f|\le V} \left|M^S_Q M^J_Q f(\boldsymbol{t},\boldsymbol{S}) - M^S_Q M^J_{\tilde{Q}} f(\boldsymbol{t},\boldsymbol{S})\right| + \sup_{|f|\le V} \left|M^S_Q M^J_{\tilde{Q}} f(\boldsymbol{t},\boldsymbol{S}) - M^S_{\tilde{Q}} M^J_{\tilde{Q}} f(\boldsymbol{t},\boldsymbol{S})\right| \\
&:= \mathbf{I}_1 + \mathbf{I}_2.
\end{aligned}$$

We can bound $\mathbf{I}_1$ by

$$\begin{aligned}
&\sum_{k_0=0}^{\infty} \cdots \sum_{k_{n-1}=0}^{\infty} \int \prod_{i=0}^{n-1} \mathbb{I}\{t_i < v_{i,1} < \cdots < v_{i,k_i} < t_{i+1}\} \sum_{\bar{\boldsymbol{S}}} |f(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})| \times \\
&\times \left|M^J_Q((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}))M^S_Q((\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})) - M^J_{\tilde{Q}}((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}))M^S_{\tilde{Q}}((\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}}))\right| d\boldsymbol{v}.
\end{aligned}$$

Recall that k_i denotes the number of virtual jumps on the interval $[t_i, t_{i+1})$. Since $|f| \le V$ and for any possible $\bar{\boldsymbol{S}}$ we have $V(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}}) \le 1 + n + \sum_{i=0}^{n-1} k_i$, and $\sum\limits_{\bar{\boldsymbol{S}}} M^S_Q\left((\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})\right) = 1$ (see (6.7)), then we can further bound $\mathbf{I}_1$ by

$$\begin{aligned}
&\sum_{k_0=0}^{\infty} \cdots \sum_{k_{n-1}=0}^{\infty} \int \prod_{i=0}^{n-1} \mathbb{I}\{t_i < v_{i,1} < \cdots < v_{i,k_i} < t_{i+1}\} \times \\
&\qquad \times \left(1 + n + \sum_{i=0}^{n-1} k_i\right) \left|M^J_Q\left((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})\right) - M^J_{\tilde{Q}}\left((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})\right)\right| d\boldsymbol{v},
\end{aligned} \tag{6.14}$$

Next, let U denote the set of indices u such that only u-th rows of Q and $\tilde{Q}$ differ. Let $Q_1 = Q$, $Q_{|U|} = \tilde{Q}$ and for $u \in U$ we define the matrix Q_{u+1} as Q_u with the u-th row replaced by the corresponding row of $\tilde{Q}$. In particular, for $u \in U$ we have $Q_u(\tilde{\mathbf{s}}) \ne Q_{u+1}(\tilde{\mathbf{s}})$ for a certain state $\tilde{\mathbf{s}}$ and for all states $\mathbf{s} \ne \tilde{\mathbf{s}}$ we have $Q_u(\mathbf{s}) = Q_{u+1}(\mathbf{s})$. Then the expression under the integral can be bounded by

$$\begin{aligned}
&\left|M^J_Q\left((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})\right) - M^J_{\tilde{Q}}\left((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})\right)\right| \le \\
&\qquad \le \sum_u \left|M^J_{Q_u}\left((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})\right) - M^J_{Q_{u+1}}\left((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})\right)\right|.
\end{aligned}$$

Each term in the last sum can be expressed in the form

$$
\begin{aligned}
&\left|M^J_{Q_u}((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})) - M^J_{Q_{u+1}}((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}))\right| = \\
&= \left| \prod_{i=0}^{n-1} \left[(\eta - Q_u(\boldsymbol{S}_i))\,(t_{i+1}-t_i)\right]^{k_i} e^{-(\eta - Q_u(\boldsymbol{S}_i))(t_{i+1}-t_i)} - \right. \\
&\quad \left. - \prod_{i=0}^{n-1} \left[(\eta - Q_{u+1}(\boldsymbol{S}_i))\,(t_{i+1}-t_i)\right]^{k_i} e^{-(\eta - Q_{u+1}(\boldsymbol{S}_i))(t_{i+1}-t_i)} \right| = \\
&= \prod_{i=0}^{n-1} (t_{i+1}-t_i)^{k_i} \prod_{\substack{i=0 \\ \boldsymbol{S}_i \neq \tilde{\mathbf{s}}}}^{n-1} (\eta - Q_u(\boldsymbol{S}_i))^{k_i} e^{-(\eta - Q_u(\boldsymbol{S}_i))(t_{i+1}-t_i)} \times \\
&\times \left| (\eta - Q_u(\tilde{\mathbf{s}}))^{\sum_{\boldsymbol{S}_i=\tilde{\mathbf{s}}} k_i} e^{-(\eta - Q_u(\tilde{\mathbf{s}})) \sum_{\boldsymbol{S}_i=\tilde{\mathbf{s}}} (t_{i+1}-t_i)} - (\eta - Q_{u+1}(\tilde{\mathbf{s}}))^{\sum_{\boldsymbol{S}_i=\tilde{\mathbf{s}}} k_i} e^{-(\eta - Q_{u+1}(\tilde{\mathbf{s}})) \sum_{\boldsymbol{S}_i=\tilde{\mathbf{s}}} (t_{i+1}-t_i)} \right|.
\end{aligned}
$$

Now let $x = \eta - Q_u(\tilde{\mathbf{s}})$, $y = \eta - Q_{u+1}(\tilde{\mathbf{s}})$, $a = \sum_{\boldsymbol{S}_i=\tilde{\mathbf{s}}} k_i$ and $b = \sum_{\boldsymbol{S}_i=\tilde{\mathbf{s}}} (t_{i+1}-t_i)$. Let also $r = \max(x,y)$. Hence, we need to bound the expression $|x^a e^{-bx} - y^a e^{-by}|$ for some $x, y, a, b > 0$. From Lagrange's mean value theorem we have

$$
\left|x^a e^{-bx} - y^a e^{-by}\right| \leq \sup_{z\in(x,y)} \left|\frac{d}{dz}(z^a e^{-bz})\right| |x-y| = \sup_{z\in(x,y)} \left|az^{a-1}e^{-bz} - bz^a e^{-bz}\right| |x-y|.
$$

Next we can bound the above supremum by

$$
\sup_{z\in(x,y)} \left|az^{a-1}e^{-bz} - bz^a e^{-bz}\right| \leq \sup_{z\in(x,y)} \max(az^{a-1}e^{-bz}, bz^a e^{-bz}). \tag{6.15}
$$

Let us assume first that the first expression of (6.15) is the maximum, then we obtain

$$
az^{a-1}e^{-bz} \leq a\tilde{\eta}^{a-1}e^{-b\tilde{\eta}}, \tag{6.16}
$$

where $\tilde{\eta} = \min(r, \frac{a-1}{b})$. In the case where the second expression is the maximum we obtain analogously $bz^a e^{-bz} \leq b\tilde{\eta}^a e^{-b\tilde{\eta}}$, where $\tilde{\eta} = \min(r, \frac{a}{b})$. Using the first assumption with the corresponding inequality (6.16), the fact that $a \leq \sum_{i=0}^{n-1} k_i$ and the fact that the set of indices U is finite, we can bound $\mathbf{I}_1$ by

$$
\begin{aligned}
&\sum_u \sum_{k_0=0}^{\infty} \cdots \sum_{k_{n-1}=0}^{\infty} \int \prod_{i=0}^{n-1} \mathbb{I}\{t_i < v_{i,1} < \cdots < v_{i,k_i} < t_{i+1}\} \prod_{\substack{i=0 \\ \boldsymbol{S}_i \neq \tilde{\mathbf{s}}}}^{n-1} (\eta - Q_u(\boldsymbol{S}_i))^{k_i} e^{-(\eta - Q_u(\boldsymbol{S}_i))(t_{i+1}-t_i)} \\
&\qquad \times \prod_{\substack{i=0 \\ \boldsymbol{S}_i = \tilde{\mathbf{s}}}}^{n-1} \tilde{\eta}^{k_i} e^{-\tilde{\eta}(t_{i+1}-t_i)} \cdot \sum_{i=0}^{n-1} k_i \left(1 + n + \sum_{i=0}^{n-1} k_i\right) |Q_{u+1}(\tilde{\mathbf{s}}) - Q_u(\tilde{\mathbf{s}})|\, d\boldsymbol{v} \leq \\
&\qquad \leq \sum_u |Q_{u+1}(\tilde{\mathbf{s}}) - Q_u(\tilde{\mathbf{s}})| \cdot \mathbb{E}\left[\sum_{i=0}^{n-1} k_i \left(1 + n + \sum_{i=0}^{n-1} k_i\right)\right] \leq \\
&\qquad \leq |U| \|Q - \tilde{Q}\| (\eta T + n\eta T + \eta T + (\eta T)^2) = \\
&\qquad = |U| \|Q - \tilde{Q}\| (\eta T(n+2) + (\eta T)^2) = C_1(n) \|Q - \tilde{Q}\|,
\end{aligned}
$$

where $C_1(n)$ is a certain linear function of n.

Using the same technique and the fact that $b \le T$ we can bound $\mathbf{I}_1$ by

$$\sum_u |Q_{u+1}(\tilde{\mathbf{s}}) - Q_u(\tilde{\mathbf{s}})| \cdot \mathbb{E}\left[T\left(1+n+\sum_{i=0}^{n-1} k_i\right)\right] \le T|U|\|Q-\tilde{Q}\| \cdot \mathbb{E}\left(1+n+\sum_{i=0}^{n-1} k_i\right) =$$
$$= T|U|\|Q-\tilde{Q}\|(1+n+\eta T) = C_2(n)\|Q-\tilde{Q}\|,$$

where $C_2(n)$ is a certain linear function of n.

Now we can bound $\mathbf{I}_2$ in a similar way as we did for $\mathbf{I}_1$ by an expression similar to (6.14), namely by

$$\sum_{k_0=0}^{\infty} \cdots \sum_{k_{n-1}=0}^{\infty} \int \prod_{i=0}^{n-1} \mathbb{I}\{t_i < v_{i,1} < \cdots < v_{i,k_i} < t_{i+1}\} \left(1+n+\sum_{i=0}^{n-1} k_i\right) \times$$
$$\times M^J_{\tilde{Q}}((\boldsymbol{t},\boldsymbol{S}),(\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}})) \sum_{\bar{\boldsymbol{S}}} \left| M^S_Q((\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})) - M^S_{\tilde{Q}}((\bar{\boldsymbol{t}},\boldsymbol{S}^{\boldsymbol{v}}),(\bar{\boldsymbol{t}},\bar{\boldsymbol{S}})) \right| d\boldsymbol{v}.$$

Before we continue, for any possible skeleton $\boldsymbol{S}$ let us denote a few auxiliary functions

$$L_Q(\boldsymbol{S}) = \nu(\boldsymbol{S}_0) \prod_{i=1}^{n} P(\boldsymbol{S}_{i-1},\boldsymbol{S}_i) \prod_{j=1}^{k} g_j(\boldsymbol{S}_{k_j}),$$
$$R_Q = \sum_{\boldsymbol{S}} L_Q(\boldsymbol{S}), \quad H_Q(\boldsymbol{S}) = \frac{L_Q(\boldsymbol{S})}{R_Q}.$$

Therefore, we need to obtain the bound for

$$\sum_{\boldsymbol{S}} \left| H_Q(\boldsymbol{S}) - H_{\tilde{Q}}(\boldsymbol{S}) \right| = \sum_{\boldsymbol{S}} \left| \frac{L_Q(\boldsymbol{S})}{R_Q} - \frac{L_{\tilde{Q}}(\boldsymbol{S})}{R_{\tilde{Q}}} \right| \le$$
$$\le \sum_{\boldsymbol{S}} \frac{\left| L_Q(\boldsymbol{S}) - L_{\tilde{Q}}(\boldsymbol{S}) \right|}{R_Q} + \sum_{\boldsymbol{S}} L_{\tilde{Q}}(\boldsymbol{S}) \left| \frac{1}{R_Q} - \frac{1}{R_{\tilde{Q}}} \right|. \tag{6.17}$$

The initial distribution ν and likelihoods g_j for all j are the same for different intensity matrices Q. Using this fact and "triangle" inequality for two products of positive numbers

$$\left| \prod_{j=1}^{n} x_j - \prod_{j=1}^{n} y_j \right| \le \sum_{j=1}^{n} |x_j - y_j| \prod_{i=1}^{j-1} x_i \prod_{i=j+1}^{n} y_i,$$

with $x_i = P(\boldsymbol{S}_{i-1},\boldsymbol{S}_i)$ and $y_i = \tilde{P}(\boldsymbol{S}_{i-1},\boldsymbol{S}_i)$, where $\tilde{P}$ is defined by $\tilde{Q}$ in the same way as P defined by Q (see (6.5)), we obtain the inequality

$$\sum_{\boldsymbol{S}} \left| L_Q(\boldsymbol{S}) - L_{\tilde{Q}}(\boldsymbol{S}) \right| \le \sum_{\boldsymbol{S}} \frac{\tilde{C}^k}{\eta} \|Q-\tilde{Q}\| \sum_{j=1}^{n} \prod_{i=1}^{j-1} P(\boldsymbol{S}_{i-1},\boldsymbol{S}_i) \prod_{i=j+1}^{n} \tilde{P}(\boldsymbol{S}_{i-1},\boldsymbol{S}_i) \le$$
$$\le \frac{\tilde{C}^k}{\eta} \|Q-\tilde{Q}\| \sum_{j=1}^{n} \sum_{\boldsymbol{S}_j} \sum_{\boldsymbol{S}_{<j}} \sum_{\boldsymbol{S}_{>j}} \prod_{i=1}^{j-1} P(\boldsymbol{S}_{i-1},\boldsymbol{S}_i) \prod_{i=j+1}^{n} \tilde{P}(\boldsymbol{S}_{i-1},\boldsymbol{S}_i) \le \frac{\tilde{C}^k}{\eta} \|Q-\tilde{Q}\| \cdot n \cdot |S|.$$

Here we used the assumption that all likelihoods are bounded $C \leq g_j \leq \tilde{C}$ for $1 \leq j \leq k$, and we locally denoted the number of all possible states of the process as $|S|$. Note, that the sum over all possible skeletons $\boldsymbol{S}$ was divided into three sums: the first one is over all possible states of $\boldsymbol{S}_j$, the second - over all possible states of $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_{j-1}$ and the last one is over all possible states $\boldsymbol{S}_{j+1}, \ldots, \boldsymbol{S}_n$. Applying again bounds on the likelihoods we easily obtain that for any Q

$$\frac{1}{\tilde{C}^k} \leq \frac{1}{R_Q} \leq \frac{1}{C^k},$$

which leads to the fact that the first expression in (6.17) is bounded from above by $C_3(n)\|Q - \tilde{Q}\|$, where $C_3(n)$ is some linear function of n. The second expression in (6.17) is bounded by

$$\tilde{C}^k \left| \frac{1}{R_Q} - \frac{1}{R_{\tilde{Q}}} \right| \leq \tilde{C}^k \frac{\sum_{\boldsymbol{S}} \left| L_Q(\boldsymbol{S}) - L_{\tilde{Q}}(\boldsymbol{S}) \right|}{R_Q R_{\tilde{Q}}}.$$

Applying two previously obtained inequalities we derive the bound $C_4(n)\|Q - \tilde{Q}\|$, where $C_4(n)$ is some linear function of n. Now combining all obtained bounds for $\mathbf{I}_1$ and $\mathbf{I}_2$ we conclude the proof. □

Lemma 6.6. *For the measurable function $V : \mathbb{X} \to [1, +\infty)$ let us denote by*

$$D_V(\beta, \beta') = \sup_X \frac{\|M_\beta(X, \cdot) - M_{\beta'}(X, \cdot)\|_V}{V(X)}$$

the V-variation of the kernels M_β and $M_{\beta'}$ and let $F_\beta : \mathbb{X} \to \mathbb{R}^+$ be the function such that $\sup_{\beta \in \mathcal{K}} |F_\beta|_V < \infty$. Moreover, define

$$\hat{F}_\beta = \sum_{n \geq 0} M_\beta^n (F_\beta - \pi_\beta(F_\beta)).$$

Then

$$|M_\beta \hat{F}_\beta - M_{\beta'} \hat{F}_{\beta'}|_V \leq C\{D_V(\beta, \beta') + |F_\beta - F_{\beta'}|_V\}.$$

Proof. The proof follows the same arguments as the proof of the Lemma 4.2 in Fort et al. (2011) in the supplement materials to the paper. In addition, some references to the first papers using similar argumentation can be found there.

First, we use the following decomposition of $M_\beta^k f - M_{\beta'}^k f$ for any $k \geq 1$

$$M_\beta^k f - M_{\beta'}^k f = \sum_{j=0}^{k-1} M_\beta^j (M_\beta - M_{\beta'}) \left(M_{\beta'}^{k-j-1} f - \pi_{\beta'}(f) \right).$$

By the Proposition 7 in Miasojedow and Niemiro (2017) the sets $\{X : |V(X)| < h\}$ are the small sets for any $h \in \mathbb{R}$. Therefore, combining it with Lemma (6.3) we have by Theorem 9 in Roberts and Rosenthal (2004) that there exist constants C_β and $\rho_\beta \in (0, 1)$ such that

$$\|M_\beta^k(X, \cdot) - \pi_\beta\|_V \leq C_\beta \rho_\beta^k V(X).$$

This property is called *geometric ergodicity* of the kernel M_β with invariant distribution π_β. Hence, for any $k \geq 1$ and any trajectory $X_\star$

$$\begin{aligned}
&\|\pi_\beta - \pi_{\beta'}\|_V \\
&\leq \|\pi_\beta - M_\beta^k(X_\star, \cdot)\|_V + \|M_\beta^k(X_\star, \cdot) - M_{\beta'}^k(X_\star, \cdot)\|_V + \|M_{\beta'}^k(X_\star, \cdot) - \pi_{\beta'}\|_V \\
&\leq \left(C_\beta \rho_\beta^k + C_{\beta'} \rho_{\beta'}^k\right)\ V(X_\star) \\
&\quad + \sup_{|f|_V \leq 1} \left| \sum_{j=0}^{k-1} M_\beta^j \left(M_\beta - M_{\beta'}\right) \left(M_{\beta'}^{k-j-1} f - \pi_{\beta'}(f)\right)(X_\star) \right|. \qquad (6.18)
\end{aligned}$$

We can bound each summand from the sum on the RHS by

$$\sup_{|f|_V \leq 1} M_\beta^j \left| (M_\beta - M_{\beta'})(M_{\beta'}^{k-j-1} f - \pi_{\beta'}(f))(X_\star) \right|.$$

Now let us denote $H = [M_{\beta'}^{k-j-1} f - \pi_{\beta'}(f)]$. Then the expression within the absolute value operator is bounded by

$$\begin{aligned}
&\sup_{|f|_V \leq 1} \sup_{|g| \leq |H|} |(M_\beta - M_{\beta'}) g(X_\star)| \leq \sup_{|f|_V \leq 1} |H|_V \sup_{|g| \leq V} |(M_\beta - M_{\beta'}) g(X_\star)| = \\
&\quad = \sup_{|f|_V \leq 1} \sup_X \frac{|M_{\beta'}^{k-j-1} f(X) - \pi_{\beta'}(f)(X)|}{V(X)} \cdot \|M_\beta(X_\star, \cdot) - M_{\beta'}(X_\star, \cdot)\|_V \leq \\
&\quad \leq C_{\beta'} \rho_{\beta'}^{k-j-1} \cdot D_V(\beta, \beta') V(X_\star).
\end{aligned}$$

Thus the last term in the (6.18) is bounded by

$$\begin{aligned}
&C_{\beta'}\ D_V(\beta, \beta')\ \sum_{j=0}^{k-1} \rho_{\beta'}^{k-j-1}\ M_\beta^j V(X_\star) \leq \\
&\quad \leq C_{\beta'}\ D_V(\beta, \beta')\ \sum_{j=0}^{k-1} \rho_{\beta'}^{k-j-1}\ \left\{\pi_\beta(V) + C_\beta \rho_\beta^j V(X_\star)\right\} \leq \\
&\quad \leq \frac{C_{\beta'}}{1 - \rho_{\beta'}}\ D_V(\beta, \beta') \left(\pi_\beta(V) + C_\beta V(X_\star)\right).
\end{aligned}$$

Taking the limit as $k \to +\infty$ in the first term in (6.18) we obtain

$$\|\pi_\beta - \pi_{\beta'}\|_V \leq \frac{C_{\beta'}}{1 - \rho_{\beta'}}\ D_V(\beta, \beta') \left(\pi_\beta(V) + C_\beta V(X_\star)\right). \qquad (6.19)$$

Now from (7) in Fort et al. (2011) we have

$$\begin{aligned}
M_\beta \hat{F}_\beta - M_{\beta'} \hat{F}_{\beta'} = &\sum_{n \geq 1} \sum_{j=0}^{n-1} \left(M_\beta^j - \pi_\beta\right) \left(M_\beta - M_{\beta'}\right) \left(M_{\beta'}^{n-j-1} F_\beta - \pi_{\beta'}(F_\beta)\right) \\
&- \sum_{n \geq 1} \{M_{\beta'}^n (F_{\beta'} - F_\beta) - \pi_{\beta'}(F_{\beta'} - F_\beta)\} - \sum_{n \geq 1} \pi_\beta \{M_{\beta'}^n F_\beta - \pi_{\beta'}(F_\beta)\}. \qquad (6.20)
\end{aligned}$$

Let us consider the first term. Similarly to the previous step by G we denote the operator $G = [M_{\beta'}^{n-j-1} F_\beta - \pi_{\beta'}(F_\beta)]$. Then we can bound

$$\begin{aligned}
\left|\left(M_\beta^j - \pi_\beta\right)\ (M_\beta - M_{\beta'})\left(M_{\beta'}^{n-j-1} F_\beta - \pi_{\beta'}(F_\beta)\right)(X)\right| \leq \\
\leq \sup_{|g|\leq |G|} \left|\left(M_\beta^j - \pi_\beta\right)(M_\beta - M_{\beta'})\, g(X)\right| \leq \\
\leq |G|_V \sup_{|g|\leq V} \left|\left(M_\beta^j - \pi_\beta\right)(M_\beta - M_{\beta'})\, g(X)\right| \leq \\
\leq |G|_V \sup_{|h|\leq \|M_\beta - M_{\beta'}\|_V} \left|\left(M_\beta^j - \pi_\beta\right) h(X)\right| \leq \\
\leq |G|_V D_V(\beta,\beta') \sup_{|h|\leq V} \left|\left(M_\beta^j - \pi_\beta\right) h(X)\right| \leq \\
\leq C_{\beta'} \rho_{\beta'}^{n-j-1} |F_\beta|_V D_V(\beta,\beta') \cdot C_\beta \rho_\beta^j V(X).
\end{aligned}$$

For the second and third terms in (6.20) we obtain the bounds

$$\left|M_{\beta'}^n (F_{\beta'} - F_\beta)(X) - \pi_{\beta'}(F_{\beta'} - F_\beta)\right| \leq C_{\beta'} \rho_{\beta'}^n V(X) |F_{\beta'} - F_\beta|_V$$

and

$$\begin{aligned}
|\pi_\beta\{M_{\beta'}^n F_\beta - \pi_{\beta'}(F_\beta)\}(X)| &= |(\pi_\beta - \pi_{\beta'})\{M_{\beta'}^n F_\beta - \pi_{\beta'}(F_\beta)\}(X)| \leq \\
&\leq \|\pi_\beta - \pi_{\beta'}\|_V |M_{\beta'}^n F_\beta(X) - \pi_{\beta'}(F_\beta)|_V \leq \\
&\leq \|\pi_\beta - \pi_{\beta'}\|_V C_{\beta'} \rho_{\beta'}^n |F_\beta|_V .
\end{aligned} \tag{6.21}$$

Therefore, combining the inequalities (6.19) – (6.21) we get

$$\begin{aligned}
|M_\beta \hat{F}_\beta(X) - M_{\beta'} \hat{F}_{\beta'}(X)| \leq \frac{C_{\beta'} C_\beta}{(1-\rho_{\beta'})(1-\rho_\beta)} |F_\beta|_V D_V(\beta,\beta') V(X) + \\
+ \frac{C_{\beta'}}{1-\rho_{\beta'}} V(X) |F_{\beta'} - F_\beta|_V + \\
+ \frac{C_{\beta'}}{(1-\rho_{\beta'})} |F_\beta|_V D_V(\beta,\beta') \left(\pi_\beta(V) + C_\beta V(X)\right).
\end{aligned}$$

Thus, since $\sup_{\beta\in K} |F_\beta|_V < \infty$, there exists a positive constant $L_{\beta,\beta'}$ for which we have

$$|M_\beta \hat{F}_\beta(X) - M_{\beta'} \hat{F}_{\beta'}(X)| \leq L_{\beta,\beta'} V(X) (D_V(\beta,\beta') + |F_{\beta'} - F_\beta|_V).$$

This concludes the proof. □

The proof of the main Theorem 6.1 is based on Theorem 6.7 which is obtained by combining Theorem 5.4 and Proposition 5.5 of Majewski et al. (2018) with a slight adjustment of the notation to our context. For any compact convex set $\mathcal{K}$ by

$$N_{\mathcal{K}}(x) = \{a \in \mathbb{R}^d : \langle a, z - x\rangle \text{ for all } z \in \mathcal{K}\}$$

we denote the normal cone to $\mathcal{K}$ at the point x. We consider an open set $\mathcal{B} \in \mathbb{R}^d$ and functions $f, g : \mathcal{B} \to \mathbb{R}$. We assume that f is a continuously differentiable function and also for all $\beta \in \mathcal{K}$ its gradient satisfies

$$\nabla f(\beta) = \int_{\mathbb{X}} \Phi(\beta, X) \pi_\beta(dX)$$

for some probability measure π_β and an integrable function $\Phi(\beta, X)$.

By $\{\beta_k, k \in \mathbb{N}\}$ we denote the sequence generated by the projected SPGD:

$$\beta_k \in \prod\nolimits_{\mathcal{K}} \left(\mathrm{prox}_{\gamma_k, g}(\beta_{k-1} - \gamma_k \Phi(\beta_{k-1}, \xi_k))\right), \tag{6.22}$$

where ξ_k is a random variable with $\pi_{\beta_{k-1}}$ distribution. Moreover, by $\{\delta_k, n \in \mathbb{N}\}$ we denote the gradient perturbation sequence defined by

$$\delta_k = \Phi(\beta_{k-1}, \xi_k) - \nabla \ell(\beta_{k-1}).$$

Moreover, for any measurable function $W : \mathbb{X} \to [1, +\infty)$ recall the definitions of $\|\mu\|_W$ and $|f|_W$ given in (6.9) and (6.10). Then we define W-variation of the kernels M_β and $M_{\beta'}$ by

$$D_W(\beta, \beta') = \sup_X \frac{\|M_\beta(X, \cdot) - M_{\beta'}(X, \cdot)\|_W}{W(X)}.$$

Theorem 6.7. *Denote*

$$\mathcal{S} = \{\beta \in \mathcal{K} : 0 \in \nabla f(\beta) + \partial g(\beta) - N_{\mathcal{K}}(\beta)\},$$

where ∂g is a subgradient of $g : \mathbb{B} \to \mathbb{R}$ (see e.g. Rockafellar (1970)). Suppose that the set $(f+g)(\mathcal{S})$ has empty interior and $\sup_{k \in \mathbb{N}} \|\delta_k\| \leq \infty$. We also make the following assumptions.

(1) The function g is convex, Lipschitz and bounded from below.

(2) The sequence of step sizes $\{\gamma_k\}$ satisfies $\gamma_k > 0$ and $\lim_{k \to \infty} \gamma_k = 0$ and

$$\sum_{k=1}^{\infty} \gamma_k = \infty, \quad \sum_{k=1}^{\infty} |\gamma_k - \gamma_{k-1}| < \infty, \quad \sum_{k=1}^{\infty} \gamma_k^2 < \infty.$$

(3) There exist constants $\rho \in [0, 1)$ and $b < \infty$ and a measurable function $W : \mathbb{X} \to [1, +\infty)$ such that

$$\sup_{\beta \in \mathcal{K}} |\Phi(\beta, \cdot)|_{W^{1/2}} < \infty, \quad \sup_{\beta \in \mathcal{K}} M_\beta W \leq \lambda W + b.$$

In addition, for any $l \in (0, 1]$ there exists $C < \infty$ and $\rho \in (0, 1)$ such that for any $X \in \mathbb{X}$

$$\sup_{\beta \in \mathcal{K}} \|M_\beta^n(X, \cdot) - \pi_\beta\|_{W^l} \leq C \rho^n W^l(X).$$

(4) The kernels M_β and the stationary distributions π_β are locally Lipschitz with respect to β, i.e. for any compact set $\mathcal{K}$ and any $\beta, \beta' \in \mathcal{K}$ there exists $C < \infty$ such that

$$\sup_{\beta \in \mathcal{K}} \|\Phi(\beta, \cdot) - \Phi(\beta', \cdot)\|_{W^{1/2}} + D_{W^{1/2}}(\beta, \beta') \leq C \|\beta - \beta'\|.$$

(5) $\mathbb{E}[W(\xi_1)] < \infty$.

Then the sequence $\{x_k, k \in \mathbb{N}\}$ generated by iterations (6.22) converges to $\mathcal{S}$.

Proof of Theorem 6.1. For better transparency of the proof we will use $m = 1$ in (6.4), the generalization of the reasoning to the case of $m > 1$ is straightforward. In our case the role of the function f plays the negative log-likelihood $\ell(\beta)$ and the function g is the ℓ_1-penalty. Both functions satisfy the assumptions of Theorem 6.7.

Then, by the formula (5.14) for the gradient of the negative log-likelihood function and the Fisher identity in (6.3) the function $\Phi(\beta_k, X)$ in our case takes the form

$$\Phi(\beta_k, X) = \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{X}_{-w}} \sum_{s \neq s'} \left[-n_w(c;\ s, s') + t_w(c;\ s) \exp({\beta^w_{k,s,s'}}^\top Z_w(c)) \right] Z_w(c), \quad (6.23)$$

where we take β_k from the k-th iteration of the p-SPGD algorithm as the parameter vector. Other components such as $n_w(c;\ s, s')$, $t_w(c;\ s)$ and $Z(c)$ correspond to a single trajectory X of the Markov jump process. Integrating this function over all possible trajectories with respect to $\pi_\beta = p_\beta(Y \mid X)$ gives us the desired gradient $\nabla \ell(\beta)$ of the negative log-likelihood.

In place of the function W in the assumptions of Theorem 6.7 we take the function V^2. Note that the original Theorem 5.4 of Majewski et al. (2018) on the convergence of the algorithm does not use the function W, instead it has an additional assumption on the gradient perturbation sequence

$$\delta_k = \Phi(\beta_{k-1}, X_k) - \nabla \ell(\beta_{k-1}), \quad k \in \mathbb{N}.$$

That assumption states that the sequence $\{\delta_k, k \in \mathbb{N}\}$ can be decomposed as $\delta_k = e^\delta_k + r^\delta_k$, where $\{e^\delta_k, k \in \mathbb{N}\}$ and $\{r^\delta_k, k \in \mathbb{N}\}$ are two sequences satisfying $\lim_{k\to\infty} \|r^\delta_k\| = 0$ and the series $\sum_{k=1}^\infty \gamma_k e^\delta_k$ converges. However, Proposition 5.5 in Majewski et al. (2018) implies that by introducing Assumptions (3)–(5) we obtain the required decomposition of δ_k. Therefore let us check the rest of the assumptions of Theorem 6.7.

Assumption (2) on step-sizes is automatically satisfied. First we review Assumption (3) with $W = V^2$, which consists of three conditions. The first condition that $\sup_{\beta_k \in \mathcal{K}} |\Phi(\beta_k, \cdot)|_V$ is bounded is easy to check because for any trajectory X the sum of the terms $n_w(c;\ s, s')$ is bounded by the total number of jumps $V(X)$, the sum of the terms $t_w(c;\ s)$ is bounded by the total observation time T and vectors β_k come from the compact set $\mathcal{K}$, which means that exponent is bounded by some constant. The second condition follows directly from Lemma 6.3. The last condition representing geometric ergodicity was shown in Lemma 6.6.

In our setting Assumption (4) takes the form

$$D_V(\beta, \beta') + |\hat{\Phi}(\beta, \cdot) - \hat{\Phi}(\beta', \cdot)|_V \leq C\|\beta - \beta'\|$$

for some constant C. We obtain it by combining Lemma 6.5 and the trivial fact that $|\Phi(\beta, \cdot) - \Phi(\beta', \cdot)|_V \leq C\|\beta - \beta'\|$ for some constant C.

In the course of the proof of the mentioned above decomposition Majewski et al. (2018) used the following property of the function W, which needs to be checked as well. For any trajectory ξ_k under the assumption $\mathbb{E}W(\xi_0) < \infty$ there holds $\sup_{k\geq 1} \mathbb{E}[W(\xi_k)] < \infty$. In our

case we can obtain the same property. Assuming $\mathbb{E}V^2(X_0) < \infty$ by Lemma 6.4 we have that $\sup_{k\geq 1} \mathbb{E}[V^2(X_k)] < \infty$. This concludes the proof of the theorem. □

6.4 Numerical results

In this section we describe the details of implementation of the proposed algorithm as well as the results of experimental studies.

6.4.1 Details of implementation

We provide in details implementation of the proposed algorithm in practice. Recall that the optimization problem (6.1) is solved by the iterative algorithm called projected stochastic proximal gradient descent given in (6.2). Instead of the gradient of the negative log-likelihood $\nabla\ell(\beta)$ we use its MCMC approximation $\Phi(\beta, X^1, \ldots, X^m)$, where $X^1, \ldots, X^m$ is a set of trajectories generated by Rao and Teh's scheme given in Section 6.2. The solution of (6.1) depends on the choice of λ. As we mentioned in previous chapters, finding the „optimal" parameter λ and the threshold δ is difficult in practice. In this case we also solve it using the same information criteria as in Chapter 5, where again instead of the gradient of the negative log-likelihood we use its MCMC approximation.

The function $\Phi(\theta, X^1, \ldots, X^m)$ is an average of the functions $\Phi(\theta, X^i)$ introduced in (6.23) (recall that we use the symbol β only for the true parameter vector and θ otherwise). Now, in the analogous way as we divided the optimization problem (5.5) in Subsection 5.4.1 we can divide the current one. Namely, for fixed $w \in \mathcal{V}$ and $s, s' \in \{0, 1\}$ with $s \neq s'$, the corresponding summand in $\Phi(\theta, X^1, \ldots, X^m)$ is a function which depends on the vector θ restricted only to its coordinate vector $\theta^w_{s,s'}$ (see notation (5.1)). So, for each triple w and $s \neq s'$ we can solve the problem separately. Let us denote these summands of $\Phi(\theta, X^1, \ldots, X^m)$ as $\Phi^w_{s,s'}(\theta^w_{s,s'})$.

Hence, in the current implementation we can use the scheme from Subsection 5.4.1. Namely, we start with computing a sequence of minimizers on the grid, i.e. for any triple $w \in \mathcal{V}$, $s \neq s'$ we create a finite sequence $\{\lambda_i\}_{i=1}^N$ uniformly spaced on the log scale, starting from the largest λ_i, which corresponds to the empty model. Next, for each value λ_i we compute the estimator $\hat{\beta}^w_{s,s'}[i]$ of the vector $\beta^w_{s,s'}$

$$\hat{\beta}^w_{s,s'}[i] = \underset{\theta^w_{s,s'}}{\operatorname{argmin}} \left\{ \Phi^w_{s,s'}(\theta^w_{s,s'}) + \lambda_i \|\theta^w_{s,s'}\|_1 \right\}. \tag{6.24}$$

The notation $\hat{\beta}^w_{s,s'}[i]$ means the i-th approximation of $\beta^w_{s,s'}$. To solve (6.24) numerically for a given λ_i we use the SPGD algorithm without the projection onto the compact set. In practice, the algorithm still converges well so we did not use the projection. The final LASSO estimator $\hat{\beta}^w_{s,s'} := \hat{\beta}^w_{s,s'}[i^*]$ is chosen using the Bayesian Information Criterion (BIC) applied to the MCMC approximation of the gradient of the negative log-likelihood, i.e.

$$i^* = \underset{1\leq i\leq N}{\operatorname{argmin}} \left\{ n\Phi^w_{s,s'}(\theta^w_{s,s'})(\hat{\beta}^w_{s,s'}[i]) + \log(n)\|\hat{\beta}^w_{s,s'}[i]\|_0 \right\}.$$

Here $\|\hat{\beta}^w_{s,s'}[i]\|_0$ denotes the number of non-zero elements of $\hat{\beta}^w_{s,s'}[i]$ and n is the number of jumps in the trajectory generated by Rao and Teh's algorithm. In our simulations we use $N = 100$.

Finally, the threshold δ is obtained using the Generalized Information Criterion (GIC) as in Subsection 5.4.1, also applied to the MCMC approximation of the gradient of the negative log-likelihood. For a prespecified sequence of thresholds $\mathscr{D}$ we calculate

$$\delta^* = \underset{\delta \in \mathscr{D}}{\operatorname{argmin}} \left\{ n\Phi^w_{s,s'}(\hat{\beta}^{w,\delta}_{s,s'}) + \log(2d(d-1))\|\hat{\beta}^{w,\delta}_{s,s'}\|_0 \right\} ,$$

where $\hat{\beta}^{w,\delta}_{s,s'}$ is the LASSO estimator $\hat{\beta}^w_{s,s'}$ after thresholding with the level δ.

6.4.2 Simulated data

We consider the chain model analogous to the model $M1$ in Subsection 5.4.2. All vertices have the "chain structure", i.e. for any node, except for the first one, its set of parents contains only a previous node. Namely, we put $\mathcal{V} = \{1, \ldots, d\}$ and $\mathrm{pa}(k) = \{k-1\}$, if $k > 1$ and $\mathrm{pa}(1) = \emptyset$. We construct CIM in the same way as in Subsection 5.4.2. Namely, for the first node the intensities of leaving both states are equal to 5. For the rest of the nodes $k = 2, \ldots, d$, we choose randomly $a \in \{0, 1\}$ and we define $Q_k(c, s, s') = 9$, if $s \neq |c - a|$ and 1 otherwise. In other words, we choose randomly whether the node prefers to be at the same state as its parent ($a = 0$) or not ($a = 1$).

We consider two cases with the number of nodes equal to $d = 5$ and $d = 10$. So, the considered number of possible parameters of the model (the size of β) is $2d^2 = 50$ or 200, respectively. We use $T = 10$ for 5 nodes and $T = 20$ for 10 nodes. We replicate simulations 100 times for each scenario. As the partial observation we take 100, 200 and 400 equally spaced points for 5 nodes and 200, 400 and 800 for 10 nodes. In Figure 6.1 we present averaged results of the simulations in terms of three quality measures

- **power**, which is a proportion of correctly selected edges;
- **false discovery rate (FDR)**, which is a fraction of incorrectly selected edges among all selected edges;
- **true model (TM)**, which is an indicator whether the algorithm selected the true model without any errors.

In Figure 6.2 we provide the results of simulations for the same models in case of complete trajectories. We observe that the results of experiments confirm that the proposed method works in a satisfactory way. We observe that with increasing number of observation points results are close to the ones in case of complete data. The larger the number of points the higher the power of the algorithm and tends to 1. The FDR is quite low in all cases. For the half simulations in case of 10 nodes and the time $T = 20$ the algorithm discovers the true model when we choose a big enough number of observation points.

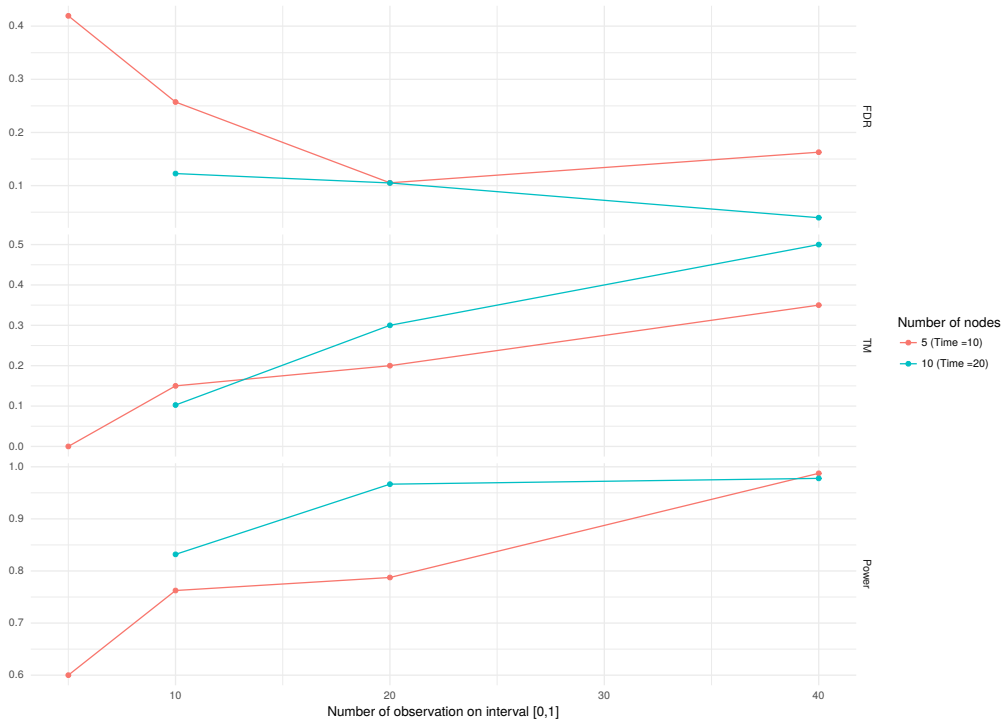


Figure 6.1: Results of simulations for partially observed data.

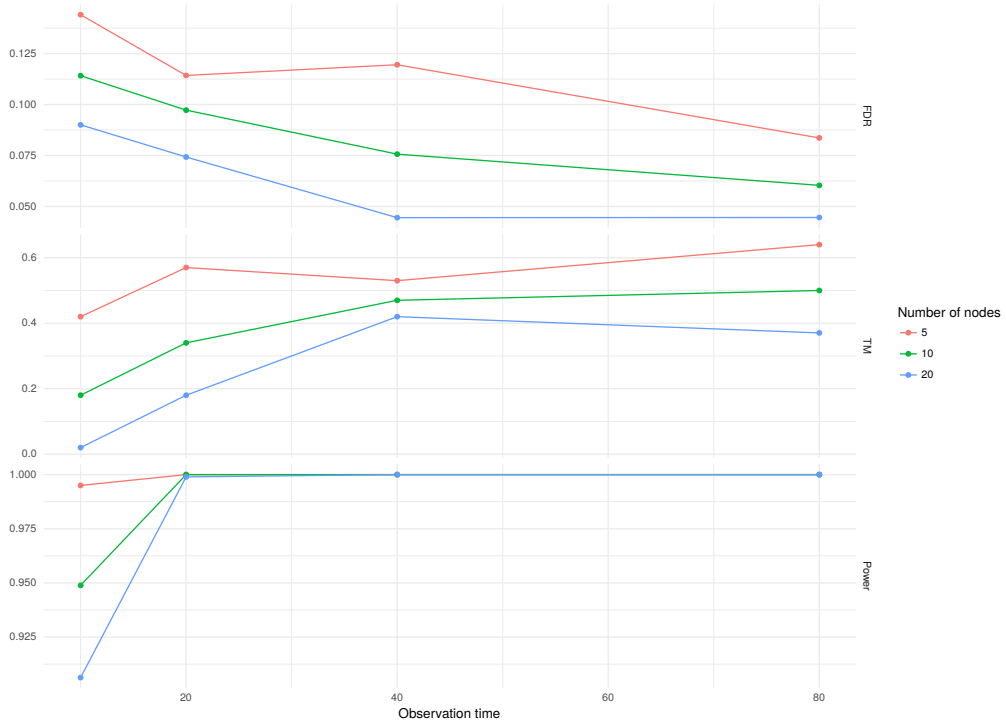


Figure 6.2: Results of simulations for fully observed data.

6.5 FFBS Algorithm

For completeness of the proposed scheme we provide the description of the forward-filtering backward-sampling algorithm for discrete-time Markov chains taken from Rao and Teh (2013) with a slightly changed notation. Earlier references for the FFBS algorithm can be found there as well.

Let $(\boldsymbol{S}_0, \ldots, \boldsymbol{S}_n)$ be a discrete-time Markov chain with a discrete state space $\mathcal{X} = \{1, \ldots, N\}$. Let P be a transition matrix $P(\mathbf{s}, \mathbf{s}') = p(\boldsymbol{S}_{j+1} = \mathbf{s}' \mid \boldsymbol{S}_j = \mathbf{s})$. Let ν be an initial distribution over states at time point 0 and let $Y = (Y_0, \ldots, Y_n)$ be a sequence of noisy observations with likelihoods $g_j(\mathbf{s}) = p(Y_j \mid \boldsymbol{S}_{j^t} = \mathbf{s})$. Given a set of observations $Y = (Y_0, \ldots, Y_n)$, FFBS returns an independent posterior sample of the state vector.

Define $a_j(\mathbf{s}) = p(Y_0, \ldots, Y_{j-1}, \boldsymbol{S}_j = \mathbf{s})$. From the Markov property, we have the following recursion:

$$a_{j+1}(\mathbf{s}') = \sum_{\mathbf{s}} a_j(\mathbf{s}) g_j(\mathbf{s}) P(\mathbf{s}, \mathbf{s}').$$

We calculate this for all possible states $\mathbf{s}' \in \mathcal{X}$ performing a forward pass. At the end of the forward pass we obtain the distribution

$$b_n(\mathbf{s}) = g_n(\mathbf{s}) a_n(\mathbf{s}) = p(Y, \boldsymbol{S}_n = \mathbf{s}) \propto p(\boldsymbol{S}_n = \mathbf{s} \mid Y)$$

and sample $\boldsymbol{S}_n$ from it. Next, note that

$$\begin{aligned} p(\boldsymbol{S}_j = \mathbf{s} \mid \boldsymbol{S}_{j+1} = \mathbf{s}', Y) &\propto p(\boldsymbol{S}_j = \mathbf{s}, \boldsymbol{S}_{j+1} = \mathbf{s}', Y) = \\ &= a_j(\mathbf{s}) g_j(\mathbf{s}) P(\mathbf{s}, \mathbf{s}') p(Y_{j+1}, \ldots, Y_n \mid \boldsymbol{S}_{j+1} = \mathbf{s}') \propto \\ &\propto a_j(\mathbf{s}) g_j(\mathbf{s}) P(\mathbf{s}, \mathbf{s}'), \end{aligned}$$

where the second equality follows from the Markov property. This is also an easy distribution to sample from, and the backward pass of FFBS successively samples new elements of Markov chain from $\boldsymbol{S}_{n-1}$ to $\boldsymbol{S}_0$. The pseudocode for the algorithm is given below.

Algorithm 1 The forward-filtering backward-sampling algorithm

Input: An initial distribution over states ν, a transition matrix P, a sequence of noisy observations $Y = (Y_0, \ldots, Y_n)$ with likelihoods $g_j(\mathbf{s}) = p(Y_j \mid \boldsymbol{S}_{j^t} = \mathbf{s})$.

Output: A realization of the Markov chain $(\boldsymbol{S}_0, \ldots, \boldsymbol{S}_n)$

Initialize $a_0(\mathbf{s}) = \nu(\mathbf{s})$.

for $j = 0$ **to** $n - 1$

  $a_{j+1}(\mathbf{s}') = \sum_{\mathbf{s}} a_j(\mathbf{s}) g_j(\mathbf{s}) P(\mathbf{s}, \mathbf{s}')$ for $\mathbf{s}' \in \mathcal{X}$.

Sample $\boldsymbol{S}_n \sim b_n(\cdot)$, where $b_n = g_n(\mathbf{s}) a_n(\mathbf{s})$.

for $j = n - 1$ **to** 0

  Define $b_j(\mathbf{s}) = a_j(\mathbf{s}) g_j(\mathbf{s}) P(\mathbf{s}, \boldsymbol{S}_{j+1})$;

  Sample $\boldsymbol{S}_j \sim b_j(\cdot)$.

return $(\boldsymbol{S}_0, \ldots, \boldsymbol{S}_n)$

Chapter 7

Conclusions and discussion

In this thesis we explored two types of probabilistic graphical models (PGM): Bayesian networks (BN) and continuous time Bayesian networks (CTBN). First, we explained the concept of PGMs and the motivation to study them with a few examples of successful applications. Then, we discussed more thoroughly PGMs of interest describing the problems within both frameworks and provided necessary preliminaries. In terms of contributions we were focused on structure learning, which is one of the most challenging tasks in the process of exploring PGMs and is interesting in itself. We also discussed other types of problems and reviewed some previously known results concerning these problems to provide some context.

The problem of structure learning for BNs is difficult due to the superexponential growth of the space of directed acyclic graphs (DAG) with the number of variables and also because the underlying graph needs to be acyclic. We solve this problem by dividing it into two tasks. First, we use a known method called partition MCMC to slice the set of variables into layers where any variable in any layer can have parents only from the previous layers and has at least one parent from the previous adjacent layer. Second, we find the arrows using the knowledge about the layers. In the case of continuous data we use the assumption that our network is a Gaussian Bayesian network and hence each variable is a linear combination of its parents. Thus, we solve the problem of finding arrows by finding the non-zero coefficients in the linear combination of all the variables from previous layers using Thresholded LASSO estimator. In the case of discrete and binary data we use the assumption that probability of each variable being equal to 1 is a sigmoid function of a linear combination of its parents. Hence, again we solve the problem of finding arrows by finding the non-zero coefficients in the linear combination of all the variables from previous layers using Thresholded LASSO estimator for logistic regression. Finally, for the discrete data where each variable has a finite state space we use a softmax function instead of the sigmoid function. We demonstrated theoretical consistency of LASSO and Thresholded LASSO estimators for the continuous model and showed their effectiveness on the benchmark Bayesian networks of different sizes and structure comparing the proposed method to several existing methods for structure learning.

The problem of structure learning for CTBNs in the case of complete data is also

reduced to solving the optimizational problem for the penalized with ℓ_1-penalty maximum likelihood function. We assumed that a conditional intensity of a variable is a linear function of the states of its parents, which can be easily extended to a polynomial dependence. Starting from the full graph we remove irrelevant edges and estimate parameters for existing ones simultaneously in case of LASSO estimator. In case of thresholded version of this LASSO estimator we only learn the structure. We proved the consistency of the proposed estimators and demonstrated coherence of theoretical results with numerical results from simulated data.

The last problem considered in the thesis was structure learning for CTBNs in the case of incomplete data. The optimizational problem takes the same form as for complete data but we cannot write the likelihood function explicitly anymore. Instead of the negative log-likelihood function we used its Markov chain Monte Carlo approximation, where Markov chain was generated using Rao and Teh's algorithm. The optimizational problem itself was solved by projected stochastic proximal gradient descent algorithm. We proved the convergence of this algorithm to the set of stationary points of the minimized function. We used the same assumption on conditional intensities as in the case of complete data. In practice to discover the arrows we used the thresholded version of the obtained estimator. We showed on a small simulated example that the quality of the proposed method is similar to the case of complete data and increases with the number of observed points per interval.

As for the future research we want to obtain similar theoretical results for Bayesian networks in the case of discrete data as we have obtained for the continuous data. In future we intend to perform more experiments and comparisons with existing approaches for all proposed methods. For some methods there are no open implementations or there are implementations in different programming languages, which makes it difficult to perform the comparison. The main goal was to show theoretical value of the proposed methods and show that the results of experiments are consistent with theory, which in our opinion was achieved.

Bibliography

Andersen, P. K. and Gill, R. D. (1982). Cox's regression model for counting processes: A large sample study. *Ann. Statist.*, 10:1100–1120.

Baraniuk, R., Duarte, M., and Hegde, C. (2011). Introduction to compressive sensing. *Connexions e-textbook*.

Bass, R. F. (2011). *Stochastic Processes*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Bathla Taneja, S., Douglas, G., Cooper, G., Michaels, M., Druzdzel, M., and Visweswaran, S. (2021). Bayesian network models with decision tree analysis for management of childhood malaria in Malawi. *BMC Medical Informatics and Decision Making*, 21.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202.

Bickel, P. J., Ritov, Y., and Tsybakov, A. B. (2009). Simultaneous analysis of Lasso and Dantzig selector. *The Annals of Statistics*, 37:1705–1732.

Boudali, H. and Dugan, J. B. (2006). A continuous-time Bayesian network reliability modeling, and analysis framework. *IEEE transactions on reliability*, 55(1):86–97.

Bühlmann, P. and van de Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Springer Series in Statistics, New York: Springer.

Casella, G. and George, E. I. (1992). Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174.

Chen, X. and Xuan, J. (2020). Bayesian inference of gene regulatory network. In Tang, N., editor, *Bayesian Inference on Complicated Data*, chapter 5. IntechOpen, Rijeka.

Chung, K. and Walsh, J. (2005). *Markov processes, Brownian motion, and time symmetry. 2nd ed.*

Colombo, D. and Maathuis, M. H. (2014). Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2):393–405.

Daly, R., Shen, Q., and Aitken, S. (2011). Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26(2):99–157.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Douc, R., Moulines, E., and Stoffer, D. (2014). *Nonlinear time series. Theory, methods and applications with R examples.*

Eaton, D. and Murphy, K. (2007). Bayesian structure learning using dynamic programming and MCMC. *UAI.*

Fan, Y. and Shelton, C. R. (2012). Learning continuous-time social network dynamics. *arXiv:1205.2648.*

Fort, G., Moulines, E., and Priouret, P. (2011). Convergence of adaptive and interacting Markov chain Monte Carlo algorithms. *Ann. Statist.*, 39:3262–3289.

Frey, B. and Jojic, N. (2005). A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1392–1416.

Friedman, N. and Koller, D. (2001). Being Bayesian about network structure: A bayesian approach to structure discovery in bayesian networks. *Mach Learn*, 50.

Gasse, M., Aussem, A., and Elghazel, H. (2014). A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Syst. Appl.*, 41(15):6755–6772.

Gatti, E., Luciani, D., and Stella, F. (2012). A continuous time Bayesian network model for cardiogenic heart failure. *Flexible Services and Manufacturing Journal*, 24(4):496–515.

Gelman, A. and Shirley, K. (2012). Inference from simulations and monitoring convergence. *Handbook of Markov Chain Monte Carlo.*

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Geyer, C. (2011). *Introduction to Markov Chain Monte Carlo*, pages 3–48. CRC Press.

Giudici, P. and Castelo, R. (2003). Improving Markov Chain Monte Carlo model search for data mining. *Machine Learning*, 50:127–158.

Grzegorczyk, M. and Husmeier, D. (2008). Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71(2-3):265–305.

Gupta, A., Slater, J., Boyne, D., Mitsakakis, N., Beliveau, A., Druzdzel, M., Brenner, D., Hussain, S., and Arora, P. (2019). Probabilistic graphical modeling for estimating risk of coronary artery disease: Applications of a flexible machine-learning method. *Medical Decision Making*, 39:1032–1044.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC.

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

Heckerman, D. (2021). A tutorial on learning with Bayesian networks. *arXiv 2002.00269*.

Huang, J., Sun, T., Ying, Z., Yu, Y., and Zhang, C.-H. (2013). Oracle inequalities for the lasso in the Cox model. *Annals of statistics*, 41(3):1142–1165.

Huang, J. and Zhang, C.-H. (2012). Estimation and selection via absolute penalized convex minimization and its multistage adaptive applications. *Journal of Machine Learning Research*, 13:1839–1864.

Jacod, J. and Shiryaev, A. N. (2003). *Limit Theorems for Stochastic Processes*. Springer Berlin Heidelberg.

Jorge, P., Abrantes, A., Lemos, J., and Marques, J. (2007). Long term tracking of pedestrians with groups and occlusions. *Bayesian Network Technologies: Applications and Graphical Models*, pages 151–175.

Jorge, P., Abrantes, A., and Marques, J. (2004). On-line object tracking with Bayesian Networks. *https://www.researchgate.net/publication/251372022*.

Koivisto, M. and Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *J. Mach. Learn. Res.*, 5:549–573.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press.

Komodakis, N., Paragios, N., and Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

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

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224.

Lezaud, P. (1998). Chernoff-type bound for finite Markov chains. *The Annals of Applied Probability*, 8(3):849–867.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press.

Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International Statistical Review / Revue Internationale de Statistique*, 63(2):215–232.

Majewski, S., Miasojedow, B., and Moulines, E. (2018). Analysis of nonsmooth stochastic approximation: the differential inclusion approach. *arXiv: 1805.01916v1*.

Meng, X.-L. and Rubin, D. B. (1991). Using EM to obtain asymptotic variance-covariance matrices: The SEM algorithm. *Journal of the American Statistical Association*, 86(416):899–909.

Miasojedow, B. and Niemiro, W. (2017). Geometric ergodicity of Rao and Teh's algorithm for Markov jump processes and CTBNs. *Electronic Journal of Statistics*, 11(2):4629–4648.

Miasojedow, B. and Rejchel, W. (2018). Sparse estimation in Ising model via penalized Monte Carlo methods. *Journal of Machine Learning Research*, 19(75):1–26.

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Negahban, S., Yu, B., Wainwright, M. J., and Ravikumar, P. K. (2009). A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, pages 1348–1356.

Nodelman, U. (2007). *Continuous Time Bayesian Networks*. PhD thesis, Department of Computer Science, Stanford University.

Nodelman, U. and Horvitz, E. (2004). Continuous time Bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation. *https://www.researchgate.net/publication/228686433*.

Nodelman, U., Koller, D., and Shelton, C. (2005). Expectation propagation for continuous time Bayesian networks. In *Proceedings of the Twenty-first Conference on Uncertainty in AI (UAI)*, pages 431–440, Edinburgh, Scottland, UK.

Nodelman, U., Shelton, C., and Koller, D. (2002). Continuous time Bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 378–387.

Nodelman, U., Shelton, C. R., and Koller, D. (2012). Expectation Maximization and complex duration distributions for continuous time Bayesian networks. *arXiv 1207.1402*.

Opgen-Rhein, R. and Strimmer, K. (2007). From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology*, 1(37).

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proc. of Cognitive Science Society (CSS-7)*.

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Pokarowski, P. and Mielniczuk, J. (2015). Combined l_1 and greedy l_0 penalized least squares for linear model selection. *J. Mach. Learn. Res.*, 16:961–992.

Protter, P. E. (2005). *Stochastic Integration and Differential Equations*. Springer Berlin Heidelberg.

Rabiner, L. and Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16.

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological Methodology*, 25:111–163.

Rao, V. and Teh, Y. W. (2013). Fast MCMC sampling for Markov jump processes and extensions. *Journal of Machine Learning Research*, 14:3207–3232.

Roberts, G. O. and Rosenthal, J. S. (2004). General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Schäfer, J. and Strimmer, K. (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1).

Scutari, M. (2010). Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22.

Scutari, M., Graafland, C. E., and Gutiérrez, J. M. (2018). Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *arXiv:1205.2648*.

Sontag, D. and Jaakkola, T. (2007). New outer bounds on the marginal polytope. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

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

Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review - SOC SCI COMPUT REV*, 9:62–72.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search, 2nd Edition*.

Stella, F., Acerbi, E., Vigano, E., Poidinger, M., Mortellaro, A., and Zelante, T. (2016). Continuous time Bayesian networks identify Prdm1 as a negative regulator of TH17 cell differentiation in humans. *Scientific Reports*, 6.

Stella, F., Acerbi, E., Zelante, T., and Narang, V. (2014). Gene network inference using continuous time Bayesian networks: A comparative study and application to Th17 cell differentiation. *BMC Bioinformatics*, 15.

Stella, F. and Amer, Y. (2012). Continuous time Bayesian network classifiers. *Journal of Biomedical Informatics*, 45(6):1108–1119.

Thiesson, B. (1995). Accelerated quantification of Bayesian networks with incomplete data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, KDD'95, pages 306–311. AAAI Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

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

van de Geer, S. (2008). High-dimensional generalized linear models and the LASSO. *The Annals of Statistics*, 36:614–645.

van de Geer, S. (2016). *Estimation and Testing Under Sparsity: Cole d't de Probabilits de Saint-Flour XLV - 2015*. Springer Publishing Company, Incorporated, 1st edition.

Villa, S. and Stella, F. (2018). Learning continuous time Bayesian networks in non-stationary domains (extended abstract). In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 5656–5660. AAAI Press.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2005). MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51:3697–3717.

Wasyluk, H., Onisko, A., and Druzdzel, M. (2001). Support of diagnosis of liver disorders based on a causal Bayesian network model. *Medical science monitor : international medical journal of experimental and clinical research*, 7 Suppl 1:327–32.

Xu, J. and Shelton, C. R. (2008). Continuous time Bayesian networks for host level network intrusion detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 613–627. Springer.

Xue, L., Zou, H., and Cai, T. (2012). Nonconcave penalized composite conditional likelihood estimation of sparse Ising models. *The Annals of Statistics*, 40:1403–1429.

Yang, S., Khot, T., Kersting, K., and Natarajan, S. (2016). Learning continuous time Bayesian networks in relational domains: A non-parametric approach. In *AAAI*.

Ye, F. and Zhang, C.-H. (2010). Rate Minimaxity of the Lasso and Dantzig Selector for the l_q loss in l_r Balls. *Journal of Machine Learning Research*, 11:3519–3540.

Yedidia, J., Freeman, W., and Weiss, Y. (2001). Generalized belief propagation. *Advances in Neural Information Processing Systems 13*, 13.