Abstract

Bayesian networks are used to model complex uncertain systems with interrelated components. A reason for using Bayesian networks to model these systems is the correspondence to the abstract human notion of causal reasoning. Since many real-world relations are assumed to be deterministic, we also allow the models in this work to contain deterministic relations. This thesis concerns discrete Bayesian networks and corresponding inference methods.

Exact inference in Bayesian networks is often too computationally intensive. On the other hand, most approximate inference methods have serious drawbacks in handling Bayesian networks with deterministic relations. In the case of importance sampling, we have a rejection problem: many of the samples may have high probability to get rejected due to the fact that the samples are incompatible with the evidence. In the case of Markov chain Monte Carlo methods, the deterministic relations can prevent the Markov chain from visiting certain regions in the state space.

A sound Markov chain Monte Carlo algorithm is presented: prune sampling. This algorithm guarantees convergence under certain circumstances and it achieves better results than the commonly used methods, especially in the presence of determinism.

Title: Inference methods in discrete Bayesian networks
Author: Ron Weikamp, ron.weikamp@student.uva.nl, 6060919
Supervisor UvA: prof. dr. R. Núñez Queija
Internship supervisor TNO: dr. F. Phillipson
Second examiner: prof. dr. M.J. Sjerps
Final date: 30-10-2015
Acknowledgements

First and foremost I would like to thank my supervisors Sindo Núñez Queija (UvA) and Frank Phillipson (TNO) for being my main supervisors during this project. I explicitly want to thank them for reviewing my work and the useful suggestions they have made. Secondly, I am grateful to TNO for providing an internship position and all necessary facilities to perform this research. It was a pleasure to be part of the department Performance of Networks and Systems. Lastly, I am indebted to Marjan Sjerps (UvA) for her time to be the second reviewer.
Contents

1 Introduction ................. 6
  1.1 Bayesian networks and inference .................. 6
  1.2 Goals and approach of the research project .......... 7
  1.3 Overview of the thesis .................. 7

2 Bayesian networks .............. 9
  2.1 Example Bayesian network: BloodPressure ............ 9
    2.1.1 Belief propagation and independencies ........... 10
    2.1.2 Conditional probability table representation .......... 11
    2.1.3 Reasoning .................................. 12
    2.1.4 Bayesian probability .......................... 12
  2.2 Mathematical representation of a Bayesian network .......... 13
    2.2.1 Chain rule and conditional independence .......... 13
    2.2.2 Factorization of the joint distribution .......... 14
    2.2.3 D-separation ................................ 17
    2.2.4 Entering evidence and the posterior distribution .... 18

3 Inference in Bayesian networks ............ 20
  3.1 Complexity .......................... 20
  3.2 Exact inference methods .......................... 21
    3.2.1 An example of exact marginalization ............... 22
    3.2.2 Variable elimination .................................. 23
    3.2.3 Clique trees .................................. 24
    3.2.4 Recursive conditioning .................................. 24
  3.3 Sampling inference methods .......................... 25
    3.3.1 Introduction to approximate inference methods ...... 25
    3.3.2 Notation and definitions .................................. 26
    3.3.3 Forward sampling .................................. 26
    3.3.4 Likelihood sampling .................................. 28
    3.3.5 Importance sampling .................................. 29
    3.3.6 Gibbs sampling .................................. 29
    3.3.7 Markov chain Monte Carlo methods ............... 31
  3.4 Drawbacks in common sampling inference methods .......... 33
Chapter 1

Introduction

1.1 Bayesian networks and inference

A Bayesian network (BN) is a graphical model that represents a set of random variables and their conditional dependencies. This graphical representation can be achieved by considering a graph where the set of nodes is induced by the set of random variables and where the set of edges is given by the conditional dependencies between the random variables.

The ingenious feature of a BN is that it endeavors to model the knowledge domain: things that have been perceived, discovered or learned. It is a useful tool for inferring a hypothesis from the available information. This information, or data, can be inserted in the BN and various hypotheses can be adjusted. More formal, BN’s are used to determine or infer the posterior probability distributions for the variables of interest, given the observed information.

Several systems have been modeled using BN’s, inter alia: target recognition [16], business continuity [22] and the effects of chemical, biological, radiological and nuclear (CBRN) defense systems [21].

There are multiple methods to derive the posterior probability distributions for the variables of interest, given the (newly) observed information. This activity is also known as belief propagation, belief updating or inference. In this project we will first introduce the concept of a BN and consider and compare the various methods to do inference in these networks. Inference methods can roughly be divided into two classes: the exact methods and the approximate methods. Common problems that arise in exact methods are time consumption and memory intensity. Common problems that arise in approximate methods are rejection of samples due to inconsistency with the evidence or lack of convergence.
1.2 Goals and approach of the research project

This project is part of an internship at TNO and a master thesis in Mathematics at the University of Amsterdam.

Within a specific TNO project a BN is used to model the effects of a CBRN defense system [21]. Due to the size and high connectivity of this network, exact inference methods are computationally too intensive to use. Fortunately, in most reasoning cases an approximation of the distribution of interest will suffice, provided that the approximation is close to the exact posterior distribution. A plethora of different inference strategies [4, 14, 15, 20] and inference software [7, 9, 13, 17, 19] have been developed.

The main question of this project is: Can we find a reliable (approximate) inference method? Besides a theoretically reliable algorithm, it also needs to work in practice.

To approach this problem we first investigate the commonly used inference methods in Bayesian networks and the extent to which these commonly used algorithms give reliable results.

Secondly, we will investigate the Bayes linear (BL) methodology, which is an approach suggested by TNO. In this methodology a Bayesian network is represented and updated by the use of moments of variables, instead of probability densities. We will investigate whether the BL method is a useful method for doing inference in BN’s.

To summarize, our main question and the two sub-questions are:

Can we find a reliable (approximate) inference method that works theoretically as well as practically?

1. To what extent do the commonly used algorithms give reliable results?

2. To what extent is the BL method useful in doing inference in Bayesian networks?

1.3 Overview of the thesis

We start with explaining the framework of Bayesian networks in Chapter 2. In this chapter the representation of BN’s, the notations used, the rules and reasoning patterns are discussed. Note that we only deal with discrete (and finite) BN’s, where all the nodes are represented by discrete random variables and may attain finitely many states, as opposed to continuous BN’s, where the nodes represent continuous random variables.

In Chapter 3 inference in BN’s is discussed. The complexity of doing inference, exact inference methods, approximate inference methods and their benefits and drawbacks are addressed.

In Chapter 4 an algorithm to overcome several difficulties is proposed: prune sampling. We show that prune sampling is beneficial in relation to the
commonly used methods and we show how an implementation practically can be realized.

In Chapter 5 the Bayes linear methodology is discussed. Theoretically this method is able to do inference in BN’s. In practice it turns out that the Bayes linear representation cannot be calculated exactly. Moreover, an approximation method of this representation comprises similar drawbacks as discussed in Chapter 3, making the Bayes Linear method no fruitful direction to evade problems arising in common Bayesian inference methods.

The algorithm Prune sampling is implemented in the programming language Python and a manual can be found in Appendix A.
Chapter 2

Bayesian networks

In modeling complex uncertain systems with interrelated components we often make use of multiple variables that may attain certain values with certain probability and which may be dependent of each other. The joint distribution of such a collection of variables becomes quickly unmanageable due to the exponential growth of the total number of possible states of the system. To be able to construct and utilize models from complex systems we make use of probabilistic graphical models [15]. In this chapter we discuss a particular probabilistic graphical framework, namely a Bayesian network.

The reason for using Bayesian networks is their graphical representation corresponding to our abstract notion of causal reasoning. Without directly bothering about the formal mathematical definition, we will start in Section 2.1 with an example corresponding to our real world intuitive way of reasoning.

Subsequently we will discuss the general mathematical framework of a Bayesian network in Section 2.2.

2.1 Example Bayesian network: BloodPressure

In this section we discuss the main idea of a Bayesian network by an educational Bayesian network BloodPressure, see Figure 2.1. Suppose we want to model a situation where we want take a measurement of someones blood pressure. Assume that the blood pressure is dependent of the functioning of the persons kidney and the persons lifestyle (think of food and sports habits). Furthermore, a particular lifestyle can induce that someone is doing sports. See Figure 2.1 for a graphical representation of these relations.

For now suppose that the dependency relations are as described here and as depicted in the figure, and that there are no additional links or additional variables involved in this process. Note that we do not claim that this is how it works in reality! Of course, the functioning of the kidney in real life probably is directly dependent of the lifestyle, however in case of younger
For simplicity all variables are binary. We deal with five random variables: the persons Kidney functioning (bad $k_b$ or good $k_g$), the persons Lifestyle (unhealthy $l_u$ or healthy $l_h$), the persons BloodPres (normal $b_n$ or exceptional $b_e$), the person Sports (no $s_n$ or yes $s_y$) and the Measurement (normal $m_n$ or exceptional $m_e$) of the blood pressure.

### 2.1.1 Belief propagation and independencies

Suppose we receive the knowledge that a person has been doing sports ($S = s_y$), then this knowledge would increase our belief that the persons lifestyle is healthy and then this could increase our belief that the person has an normal blood pressure. Here we see an intuitive version of propagation of evidence.

Now suppose that we also know, in addition to the knowledge about the person was doing sports, the state of the persons lifestyle. The the fact that we know the person was doing sports, does not influence our belief about the persons blood pressure anymore. We say that the persons blood pressure is independent of doing sports given the fact that we know his lifestyle, written as

$$(BP \perp S \mid L).$$

Similarly, intuitively, we might argue that the outcome of the measurement
is independent of the persons lifestyle given the knowledge about its actual blood pressure (which we somehow happen to know by using an other measurement), which results in the independence relation \((M \perp L \mid BP)\). More general we see that a node is independent of its non descendents given the state of its parents. These independencies are called the local independencies defined by the graph structure and we will see them again later.

### 2.1.2 Conditional probability table representation

Notice that in Figure 2.1 we have attached the conditional probability density of each node, given the state of its parents. In the discrete finite case we call these Conditional Probability Tables (CPT) because of the table structure. We could ask whether this representation, which is a directed acyclic graph (DAG) and a collection of CPT’s is sufficient to represent a joint distribution of the collection of variables.

It turns out that this representation is indeed sufficient. Using the independence relations we saw before the following product of conditional probabilities defines a joint distribution over the variables:

\[
P(K, L, BP, S, M) = P(K)P(L)P(BP \mid K, L)P(S \mid L)P(M \mid BP).
\]

The probability of a certain state, say the boldfaced state in Figure 2.1 \((K = k_g, L = l_b, BP = b_e, S = s_y, M = m_e)\), which is an entry in the joint probability distribution, can be calculated by collecting all relevant information from the CPT’s

\[
P(k_g, l_b, b_e, s_y, m_e) = P(k_g)P(l_b)P(b_e \mid k_g, l_b)P(s_y \mid l_b)P(m_e \mid b_e) = 0.5 \cdot 0.5 \cdot 0.8 \cdot 0.2 \cdot 0.9 = 0.144
\]

This factorization of the joint probability distribution in CPT’s has two major benefits relative to an explicit joint distribution (where each individual state of the network is assigned a certain probability).

The first benefit is that it is more compact. Note that the BloodPressure network deals with 5 binary random variables, hence this network has \(2^5 = 32\) possible configurations. The CPT representation specifies only 20 probabilities (see Figure 2.1), whereas an explicit joint distribution would ask for a probability specification of the whole state space (which is of size 32). In general the explicit formulation of a joint probability distribution of a BN is evaded due to the exponential growth in size of the state space.

The second benefit of the CPT representation is the fact that experts, for example a doctor, can more easily specify conditional probabilities between a few variables, rather than specifying joint probabilities over all variables involved.
2.1.3 Reasoning

We want to model real world systems and subsequently we want to answer queries of the form:

What is the probability of the event $X = x$ given observations $E = e$?

In the BloodPressure model possible queries of this form are

- What is the probability that the measurement of a person’s blood pressure results in exceptional, given the fact that his kidney is functioning bad?
- What is the probability that the student has an exceptional blood pressure, given the fact that we know he has been doing sports.
- What is the probability that his kidney is bad functioning given the fact that the blood pressure of the student is exceptional but we saw him do sports?
- What is the probability that we measure a high blood pressure given the fact that we know nothing else about this person?

If the model corresponds reasonable with the ‘real world’ situation, and if we are able to calculate the above probabilities, then this would give the user the ability to reason and predict in a quantitative way about real world situations.

The two conditions

1. The model corresponds reasonable with the real world situation, and

2. We are able to calculate so-called posterior distribution $P(X = x \mid E = e)$.

are both nontrivial premises. We will only deal with the second one. The calculation of such probabilities is called Bayesian inference.

2.1.4 Bayesian probability

The reason why these networks are called Bayesian networks is because of the way probability is interpreted in this case. Bayesian probability is one interpretation of the concept of probability. A Bayesian assigns a probability/belief to a certain hypothesis. Within a Bayesian framework this belief can be updated if data/evidence becomes available. This is a different view from the frequentist (classical) view of probability, where the frequency or propensity underlies a phenomenon. The difference here is that a Bayesian evades a ontological specification of the object of interest, whereas a frequentist assumes that the object of interest is distributed with fixed parameters.
by nature. According to a frequentist view the more data becomes available, the more we know the underlying process of our object of interest. On the other hand a Bayesian updates the belief in a certain object of interest using available data.

2.2 Mathematical representation of a Bayesian network

In the previous section some of the ingenious features of Bayesian networks where shown. In this section the general mathematical framework of Bayesian networks is discussed.

Firstly, essential rules in probability theory are stated in 2.1. Subsequently the factorized joint distribution is discussed in 2.2.2. It turns out that the defined distribution inherits independence relations, which will be discussed in 2.2.3. Finally the mathematical interpretation of evidence and the posterior distribution is given.

2.2.1 Chain rule and conditional independence

Let $A, B$ be two events in an outcome space with a probability function $P$. The conditional probability is defined as

$$P(A \mid B) := \frac{P(A, B)}{P(B)}.$$ 

Intuitively we can interpret this as the restriction of $A$ to the sample space of $B$. For random variables we have the similar expression

$$P(X = x \mid Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}.$$ 

Often we use the shorthand notation $P(X)$ to denote a (marginal) probability distribution over the values that can be attained by $X$ and we write $P(X \mid Y)$ to denote a set of conditional probability distributions $P(X \mid Y = y)$. A fundamental rule in probability theory is the chain rule

$$P(X, Y) = P(X \mid Y)P(Y).$$ 

Applying this rule multiple times to a collection $\mathcal{X} = \{X_1, \ldots, X_n\}$ we obtain a general form of the chain rule

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, \ldots, X_{i-1}) \quad (2.1)$$
Definition 2.2.1 (Conditionally independent). Two events $A$ and $B$ are conditionally independent given a third event $C$, if the outcome of $A$ and the outcome of $B$ are independent events in their conditional probability distribution given $C$. Formally, we say that $A$ and $B$ are conditionally independent given $C$ if

$$P(A \mid B, C) = P(A \mid C),$$

or equivalently

$$P(A, B \mid C) = P(A \mid C)P(B \mid C)$$

and we write $(A \perp B \mid C)$. For probability distributions similar expressions hold, but instead of the events $A, B$ and $C$, the random variables $X, Y$ and $Z$ are involved.

2.2.2 Factorization of the joint distribution

This section serves as a basis to the framework of Bayesian networks. A considerable amount of literature has been published on the framework of BN’s [4,14,15,20], we follow the exposition used in [15, Chapter 3].

Recall Section 2.1.1 where independence relations where derived from the graph structure of the BloodPressure example. From this a factorized form of the joint probability distribution was defined. But does it also work the other way around: from a factorized form to conditional independencies? In this section we show it does.

This equivalence guarantees that if we talk about the joint distribution over a DAG and its local independence relations, or the joint distribution over the DAG and CPT’s, we deal with the same uniquely defined joint probability distribution (with the same independence relations).

Definition 2.2.2 (Bayesian network structure graph). A Bayesian network structure graph is a directed acyclic graph (DAG) $G$ whose nodes represent random variables $X_1, \ldots, X_n$ and whose edges represent parent-child relations between the nodes, e.g. Figure 2.1.

Let $Pa(X_i)$ denote the parents of $X_i$ in $G$ and $ND(X_i)$ denote the nodes that are not descendants of $X_i$. Then $G$ ‘encodes’ the following set of conditional independence relations, called the local independencies, denoted by $I_L(G)$

For each $X_i$: $(X_i \perp ND(X_i) \mid Pa(X_i))$.

Definition 2.2.3 (Independencies in $P$). Let $P$ be a distribution of $X$. We define $I(P)$ to be the set of independencies of the form $(X \perp Y \mid Z)$ that hold in $P$.

Definition 2.2.4 (I-map). Let $G$ be a graph object with a set of independencies $I(G)$. We say that $G$ is an I-map for a set of independencies $I$ if $I(G) \subseteq I$. For simplicity we now say that $G$ is an I-map for $P$ if $G$ is an I-map for $I(P)$. 

14
Definition 2.2.5 (Factorization). Let $\mathcal{G}$ be a Bayesian network structure graph over the variables $\mathcal{X} = X_1, \ldots, X_n$. We call a distribution $P$ over the space $\mathcal{X}$ a distribution that factorizes according to $\mathcal{G}$ if $P$ can be expressed as the product

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

(2.2)

This equation is also called the chain rule for Bayesian networks and the factors $P(X_i \mid \text{Pa}(X_i))$ are called the conditional probability distributions (CPDs) or (in the discrete finite case) conditional probability tables (CPTs).

The following defines the representation of a Bayesian network

Definition 2.2.6 (Bayesian network). A Bayesian network $\mathcal{B}$ over a collection of random variables $\mathcal{X}$ is represented by the pair $(\mathcal{G}, P)$ where $P$ is a probability distribution that factorizes over the DAG $\mathcal{G}$ and where $P$ is specified as a set of CPDs associated with the nodes in $\mathcal{G}$.

Definition 2.2.7 (Topological ordering). Let $\mathcal{G}$ be a Bayesian network structure over the nodes $\mathcal{X} = \{X_1, \ldots, X_n\}$. An ordering of the nodes $X_1, \ldots, X_n$ such that if $X_i \rightarrow X_j$ (there is a directed edge from $X_i$ to $X_j$), then $i < j$, is called a topological ordering of $\mathcal{X}$ given $\mathcal{G}$.

The following theorem proves the one-on-one correspondence between the conditional independencies in the Bayesian network structure graph $\mathcal{G}$ and the factorization of the joint distribution in CPDs, guaranteeing that a Bayesian network as in Definition 2.2.6 is well-defined.

Theorem 2.2.1. Let $\mathcal{G}$ be a Bayesian network structure graph over $\mathcal{X} = \{X_1, \ldots, X_n\}$ and let $P$ be a joint distribution over the same space $\mathcal{X}$. The graph $\mathcal{G}$ is an I-map for $P$ if and only if $P$ factorizes according to $\mathcal{G}$.

Proof. $\Rightarrow$ Assume without loss of generality that $X_1, \ldots, X_n$ is a topological ordering of the variables in $\mathcal{X}$ according to $\mathcal{G}$. Using the chain rule (see (2.1)) we have

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \{X_1, \ldots, X_{i-1}\}).$$

Suppose $\mathcal{G}$ is an I-map for $P$, then $(X_i \perp \text{ND}(X_i) \mid \text{Pa}(X_i)) \in \mathcal{I}(P)$. The assumption of the topological ordering guarantees that all parents of $X_i$ are in the set $\{X_1, \ldots, X_{i-1}\}$ and none of the descendents of $X_i$ is in $\{X_1, \ldots, X_{i-1}\}$. Therefore $\{X_1, \ldots, X_{i-1}\} = \text{Pa}(X_i) \cup Z$ for some subset $Z \subseteq \text{ND}(X_i)$. i.e. the collection $\{X_1, \ldots, X_{i-1}\}$ partitions in a set of parents of $X_i$ and a subset of non descendents of $X_i$. Using the independencies from $\mathcal{G}$ we get

$$(X_i \perp Z \mid \text{Pa}(X_i))$$

and hence

$$P(X_i \mid X_1, \ldots, X_{i-1}) = P(X_i \mid \text{Pa}(X_i)).$$
Since $i$ was arbitrary the above equation holds for all factors and hence the results follows [15, p. 62-63].

$\Leftarrow$ We proof this by induction on the number of nodes of the network. For the base case (one node) the result holds trivially. Now the induction hypothesis is: for BN’s with $n - 1$ nodes we have that if $P$ factorizes over $G$, then $G$ encodes the independence relations $(X_i \perp \text{ND}(X_i) \mid \text{Pa}(X_i))$.

Let $B$ be a network over $n$ nodes $\mathcal{X} = \{X_1, \ldots, X_n\}$ (in topological ordering) and suppose that $P$ factorizes according to $G$ (the graph structure of $B$). By definition of conditional independence we can equivalently show for all $i$ we have

$$P(X_i \mid \mathcal{X} \setminus X_i) = P(X_i \mid \text{Pa}(X_i)).$$

It holds that $X_n$ is a leaf node by the topological ordering. Let $B'$ be the network $B$ with $X_n$ removed and let $G'$ be its graph. We have that the distribution of $B'$ is given by

$$\sum_{X_n} P(\mathcal{X}) = \sum_{X_n} \prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i))$$

$$= \prod_{i=1}^{n-1} P(X_i \mid \text{Pa}(X_i)) \sum_{X_n} P(X_n \mid \text{Pa}(X_n))$$

$$= \prod_{i=1}^{n-1} P(X_i \mid \text{Pa}(X_i)) \cdot 1,$$

which is a factorized distribution over the graph $G'$. Using the induction hypothesis we get the independence relations

$$(X_i \perp \text{ND}(X_i) \mid \text{Pa}(X_i)),$$

for $i = 1, \ldots, n - 1$.

Using the definition of conditional probability in the first equality and the factorization property in the second we obtain

$$P(X_n \mid \mathcal{X} \setminus X_n) = \frac{P(\mathcal{X})}{P(\mathcal{X} \setminus X_n)}$$

$$= \frac{\prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i))}{\sum_{X_n} \prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i))},$$

$$= \frac{\prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i))}{\prod_{i=1}^{n-1} P(X_i \mid \text{Pa}(X_i)) \sum_{X_n} P(X_n \mid \text{Pa}(X_n))}$$

$$= \frac{\prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i))}{\prod_{i=1}^{n-1} P(X_i \mid \text{Pa}(X_i)) \cdot 1}$$

$$= P(X_n \mid \text{Pa}(X_n)),$$

yielding the independence relation $(X_n \perp \text{ND}(X_i) \mid \text{Pa}(X_n))$. □
2.2.3 D-separation

In the previous section we showed how the correspondence between the local independencies in the Bayesian network graph structure $G$ correspond to a specific factorization of the joint distribution over $X_1, \ldots, X_n$. It turns out that the Bayesian graph structure inherits even more independence relations. For almost all (except for a measure zero set) probability distributions $P$ that factorize over $G$, the set of independence relations of $P$ corresponds to the set of independence relations of $G$. As a consequence, we can derive independence relations of $P$ by examining the connectivity of $G$.

Again, the exposition used in [15, Chapter 3] is followed.

**Definition 2.2.8** (Trail). If there is a path between two nodes $A$ and $B$ in the undirected graph induced by the directed graph (if there is an edge between two nodes in the DAG, this edge will also be present in the induced undirected graph), we call this a trail.

**Definition 2.2.9** (V-structure). A (part of) a Bayesian network graph structure where $X \rightarrow Z \leftarrow Y$ is called a v-structure.

**Definition 2.2.10** (Active trail). Let $G$ be a Bayesian network graph structure, $Z$ be a subset of the evidence nodes and $\langle X_1, X_2 \rangle \ldots \langle X_{n-1}, X_n \rangle$ be a trail between $X_1$ and $X_n$. This trail is called an active trail given $Z$ if

- Whenever we have a v-structure, $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, then $X_i$ or one of its descendants is in $Z$;

- no other nodes along the trail is in $Z$.

**Definition 2.2.11** (d-separation). Let $X, Y, Z$ be sets of nodes in $G$. We say that $X$ and $Y$ are d-separated given $Z$ if there is no active trail between any node $X \in X$ and $Y \in Y$ given $Z$.

The set of independencies that correspond to d-separation is denoted by $I(G)$.

The ‘d’ in d-separation reflects that the separation is a consequence of the directed graph structure.

So far we only used the directed graph structure to characterize these independence properties. The question is whether these dependency properties are also present in the joint distribution that is specified over the graph. The answer is yes:

**Theorem 2.2.2** (Thm 3.5 [15]). For almost all distributions $P$ that factorize over $G$ (except for a set of measure zero in the space of CPD parametrizations), we have $I(G) = I(P)$.
2.2.4 Entering evidence and the posterior distribution

As noted (recall Definition 2.2.5) the distribution of a Bayesian network over the nodes \( \mathcal{X} = \{X_1, \ldots, X_n\} \) can be written in the factorized form

\[
P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i)).
\]

Inference means the calculation of the posterior distribution \( P(X \mid \mathcal{E} = e) \) for some subsets \( X, \mathcal{E} \subset \mathcal{X} \) and some assignment \( e \) to the nodes in \( \mathcal{E} \). The posterior \( P(X = x \mid \mathcal{E} = e) \) can be calculated by

\[
P(X = x \mid \mathcal{E} = e) = \frac{P(X = x, \mathcal{E} = e)}{P(\mathcal{E} = e)} = \frac{\sum_{X' \in \mathcal{X} \setminus \{X, \mathcal{E}\}} P(X', x, e)}{\sum_{X' \in \mathcal{X} \setminus \mathcal{E}} P(X', e)}.
\] (2.3)

The probability of the evidence \( P(\mathcal{E} = e) \) acts as a normalizing constant which will be denoted by \( Z \).

The conditional probabilities \( P(X_i \mid \text{Pa}(X_i)) \) can be written abbreviated as factors

\[
\phi_i : X_i \cup \text{Pa}(X_i) \rightarrow \mathbb{R}.
\]

The domain of a factor is written as \( D(\phi_i) \). The collection of possible states of a variable \( X \) is written as \( \text{Val}(X) \).

The evidence \( \mathcal{E} = e \) can also be interpreted as a collection of constraints \( \mathcal{C} = \{C_1, \ldots, C_k\} \), where each \( C_i \) is a 0/1 function with domain a subset of \( \mathcal{X} \). If a state of the network \( x \) satisfies the evidence \( E_i = e_i \), then \( C_i(x) = 1 \), otherwise \( C_i(x) = 0 \). The probability distribution over the distribution with the constraints is written as

\[
P_M(x) := \begin{cases} 
P(x) & \text{if } C_i(x) = 1 \quad \forall i \\ 
0 & \text{otherwise}
\end{cases}.
\]

This Bayesian network with additional constraints is also known as a mixed network [11].

Using the above abbreviations we can write (2.3) as

\[
P_M(x) = \frac{1}{Z} \prod_i \phi_i(x_i) \prod_i C_i(x),
\] (2.4)

where \( x_i \) contains the appropriate assignment of \( x \) for the factor \( \phi_i \).

Mostly only states \( x \) of the BN which are compatible with the evidence are considered, meaning \( C_i(x) = 1 \) for all \( i \). Often we write \( P_M(x) \) or simply \( P(x) \) as the posterior distribution of interest if we mean (2.4), without explicitly writing down the evidence. A state \( x \) with \( P(x) > 0 \) is called a feasible state.

In the following example we illustrate how this notation is used in a concrete BN:
Example 2.2.1. Recall the BloodPressure network from Section 2.1. A state $x$ of this network is given by

$$x = (k_g, l_b, b_e, s_g, m_e).$$

If no evidence is given then $x$ is a feasible state and

$$P(x) = 0.5 \cdot 0.5 \cdot 0.8 \cdot 0.2 \cdot 0.9 = 0.144.$$

If evidence $M = m_n$ is given, then state $x$ is not compatible with the evidence since it contains $M = m_e$. Equivalently, $P(x \mid M = m_n) = 0$, or we simply write $P(x) = 0$. 

19
Chapter 3

Inference in Bayesian networks

One of the main questions that the framework of Bayesian networks has to deal with is:

Given a Bayesian network $\mathcal{B}$ over $\mathcal{X}$, a (collection of) variable(s) $\mathbf{X} \subset \mathcal{X}$, a realization $\mathbf{e}$ of a (collection of) variable(s) $\mathbf{E} \subset \mathcal{X}$. What is the probability distribution $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$?

The realization of the collection $\mathbf{E}$ is usually called evidence. The activity of answering such questions in the Bayesian network framework is called inference. The distribution $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$ is called the posterior distribution.

In Section 3.1 we will briefly explain the complexity of doing inference. In Section 3.2 an overview of the exact inference methods in BN’s is given. Exact methods do not work in practice if networks become large. Therefore we elaborate the commonly used sampling inference methods in Section 3.3. The sampling methods inherit serious drawbacks, which are revealed in Section 3.4.

3.1 Complexity

We have seen that we can access each entry in the joint distribution $P(\mathcal{X})$ by collecting the corresponding CPT-values from the corresponding CPT’s. So why could we, to do inference, not just marginalize the joint distribution as follows

$$P(\mathbf{X} \mid \mathbf{E} = \mathbf{e}) = \frac{P(\mathbf{X}, \mathbf{e})}{P(\mathbf{e})} = \frac{\sum_{\mathbf{x} \in \mathcal{X} \setminus \mathbf{X}, \mathbf{E}} P(\mathbf{x}, \mathbf{e})}{\sum_{\mathbf{x} \in \mathcal{X} \setminus \mathbf{E}} P(\mathbf{x}, \mathbf{e})}?$$

Recall the exponential blowup of the joint distribution discussed in Section 2.1.2. It is because of the size of the joint distribution, the exponential
growth in number of possible cases, that this naive way of summing out variables yields an exponential blow up in number of calculations.

Unfortunately the inference process in Bayesian networks is \( \mathcal{NP} \)-hard:

**Theorem 3.1.1** (Thm. 9.4 [15]). The following problem is \( \mathcal{NP} \)-hard for any \( \epsilon \in (0, \frac{1}{2}) \):

Given a Bayesian network \( \mathcal{B} \) over \( \mathcal{X} \), a variable \( X \in \mathcal{X} \), a value \( x \), and an observation \( E = e \) for \( E \subseteq \mathcal{X} \), find a number \( \rho \) such that

\[
|P_{\mathcal{B}}(X = x) - \rho| < \epsilon.
\]

Theorem 3.1.1 implies that, given evidence, doing exact or approximate inference in a BN is \( \mathcal{NP} \)-hard. This means in practice that we do not know a polynomial-time algorithm to solve these problems. We note furthermore that exact marginalization, the calculation of \( P(X = x) \) for \( X \in \mathcal{X} \), is \( \mathcal{NP} \)-hard [15]. In the case of approximating the marginal probability \( P(X = x) \), there exists a polynomial time algorithm which approximates the true value with high probability [15, p. 490-491].

It also has been proven that, given evidence \( E = e \), finding a state \( x \) with strictly positive probability is \( \mathcal{NP} \)-hard:

**Theorem 3.1.2** (The main theorem in [26]). The following problem is \( \mathcal{NP} \)-hard for any positive number \( \rho \):

Given a Bayesian network \( \mathcal{B} \) over \( \mathcal{X} \), an observation \( E = e \) for \( E \subseteq \mathcal{X} \) and \( e \in \text{Val}(E) \), find a find a state \( x \) of the network with \( P(x) > \rho \).

We claimed that a naive ‘summing out’ approach yields an exponential running time. Fortunately, in practice, we do not always have to naively pass all configurations and we can appeal to smarter methods. Roughly speaking, there are two classes of inference methods that we can address: exact inference methods and approximate inference methods. We will discuss both classes briefly to get a basic understanding of the methods and their implications. For a more detailed discussion and overview of the research in inference methods in Bayesian networks we refer to [4,14,15,20].

In the following two subsections we describe the classes of exact and approximate inference.

### 3.2 Exact inference methods

We start with a simple example in which the main ideas of exact inference methods become clear: there is not always need for an explicit joint distribution and a specific elimination ordering might lead to a significant decrease of computations. Subsequently an overview of some of the popular exact inference algorithms is given.
3.2.1 An example of exact marginalization

Consider a Bayesian network with the three binary variables $A, B$ and $C$. Suppose the graph structure is given by

$$A \rightarrow B \rightarrow C.$$ 

The marginal probability $P(B = 0)$ is calculated by marginalizing the joint distribution (recall the factorized form)

$$P(B = 0) = \sum_{A \in \{0,1\}} P(B = 0 \mid A = a)P(A = a)$$

$$= P(B = 0 \mid A = 0)P(A = 0) + P(B = 0 \mid A = 1)P(A = 1).$$

Now observe the calculation of the marginal probability $P(C = 0)$

$$P(C = 0) = \sum_{A,B} P(C = 0 \mid B)P(B \mid A)P(A)$$

$$= P(C = 0 \mid B = 0)P(B = 0 \mid A = 0)P(A = 0)$$
$$+ P(C = 0 \mid B = 0)P(B = 0 \mid A = 1)P(A = 1)$$
$$+ P(C = 0 \mid B = 1)P(B = 1 \mid A = 0)P(A = 0)$$
$$+ P(C = 0 \mid B = 1)P(B = 1 \mid A = 1)P(A = 1)$$

where $P(B = 0)$ and $P(B = 1)$ denote the marginal probabilities of $B$. Here we see that we can save computations if we first calculate $P(B)$ and reuse these values. In this example this would not save us that much computations, but similarly for a BN $A \rightarrow B \rightarrow C \rightarrow D$, we can calculate $P(D = 0)$ by

$$P(D = 0) = P(D = 0 \mid C = 0)P(C = 0) + P(D = 0 \mid C = 1)P(C = 1),$$

where $P(C = 0)$ and $P(C = 1)$ are marginal probabilities of $C$. In this case we can save significantly in the number of calculations when using the a specific order of marginalization and saving intermediate values.

This simple example shows us two main ideas that are used in exact inference algorithms in the marginalization process:

1. There is not always need for an explicit joint distribution.

2. Specific elimination ordering might lead to a significant decrease of computations.
Although in the above example we saved a significant amount of computations, in more complex structures we are often still haunted by exponentially many operations.

We will discuss the available exact methods in more detail in next sections. The goal is to get an idea of the way they work, the benefits and the possible drawbacks.

### 3.2.2 Variable elimination

Recall the factorized form of the joint distribution of a Bayesian network. Each $P(X_i \mid Pa(X_i))$ can be written as a factor $\phi_i : X_i \cup Pa(X_i) \rightarrow \mathbb{R}$, where $X_i \cup Pa(X_i)$ is the scope of the factor.

Using the following definition (also used in [15, p. 297]) the elimination process gets more structured.

**Definition 3.2.1 (Factor marginalization).** Let $X$ be a set of variables, and $Y \notin X$ a variable. Let $\phi(X,Y)$ be a factor. We define the factor marginalization of $Y$ in $\phi$, denoted $\sum_Y \phi$, to be a factor $\psi$ over $X$ such that:

$$\psi(X) = \sum_Y \phi(X,Y)$$

Factor multiplication is commutative, i.e. $\phi_1 \phi_2 = \phi_2 \phi_1$ and also factor marginalization is commutative $\sum_X \sum_Y \phi = \sum_Y \sum_X \phi$. Furthermore, we have associativity: $(\phi_1 \phi_2) \phi_3 = \phi_1 (\phi_2 \phi_3)$, and also distributivity: if $X \notin D(\phi_1)$ then $\sum_X \phi_1 \phi_2 = \phi_1 \sum_X \phi_2$.

**Example 3.2.1.** Using the above interpretation of factors we derive another interpretation of equation (3.1) in Example 3.2.1

$$P(D) = \sum_C \sum_B \sum_A P(A,B,C,D)$$

$$= \sum_C \sum_B \sum_A \phi_A \phi_B \phi_C \phi_D$$

$$= \sum_C \phi_D \left( \sum_B \phi_C \left( \sum_A \phi_A \phi_B \right) \right) .$$

As showed before, in this example we can reduce costs using these principles. Also note that this evades constructing the complete joint distribution $P(A,B,C,D)$, which is unmanageable in most cases.

By choosing an elimination sequence, applying factor marginalization and finally normalizing we can obtain marginal probabilities like $P(X)$ where $X$ is a variable of interest in the Bayesian network. This method is also known as variable elimination (VE). A drawback of this method is that if there are many connections in the network, we will create new factors
\( \psi \), which are products of multiple factors \( \psi = \prod_{\phi \in \Phi} \phi \), leading to large domains, and hence this will lead to exponential growth of the CPT of \( \psi \). Therefore this method is known to be memory intensive, which especially can be problematic in case of large and dense networks.

The order in which we sum-out variables, the elimination sequence, is important. As noted it is desirable to minimize the domains of the factors \( \psi \) that arise during the process of elimination. For a non-trivial example of selecting an optimal elimination sequence see [14, p.110].

### 3.2.3 Clique trees

There exists a data structure, named a clique tree or junction tree, which uses a message passing system to guarantee the usage of an optimal elimination sequence. A graphical example is found in [14, p. 124].

The clique tree method is basically variable elimination, but it has the benefit to eliminate variables in the most efficient way.

The methods comprises the following drawbacks:

1. Space needed to store products of factors (still) grows exponentially.

2. Due to the initialization of the data structure, the computation is fixed and predetermined, and we cannot take advantage of efficiencies that arise because of specific features of the evidence and the query.

Note that the search for cliques in graphs is also known to be a hard problem. However, it is not the graph structure of the BN that is intractable, but rather the size of the scope of the intermediate factors that arise in variable elimination.

### 3.2.4 Recursive conditioning

Without giving the details of this method, we note that the computations and summations in the variable elimination process can be done recursively by conditioning on the states of the variables and do a recursive call. This method is called recursive conditioning. For a comprehensive discussion of this method see [14, p. 140].

Basically, it does the same as VE, but roughly speaking this method fills in the factors and multiplies with a recursive call. This prevents multiplying full factors \( \phi_i \) with each other and thus the the exponential blow-up in memory consumption. Unfortunately this process comes at a price: we create an exponential blow-up in terms of recursive calls.

Besides a hard trade-off between memory and computational intensity, a more smooth tradeoff has been proposed to get a ratio between the proportion of memory used and the computational effort, therefore this exact method is known for its time-space tradeoff [1].
3.3 Sampling inference methods

Before elaborating on the sampling inference methods, we start with an introduction to approximate inference methods. Secondly, the notation that is used and the definitions that are needed are introduced in Section 3.3.2. Subsequently, the commonly used sampling inference methods are explained.

3.3.1 Introduction to approximate inference methods

The class of approximate inference methods can roughly be divided into two subclasses: optimization and sampling methods.

The optimization subclass is based on optimization techniques that find a solution \( Q \) in a space of distributions \( Q \) that best approximates the real distribution \( P \) according to some objective function. A major benefit is that this opens the door to an area of optimizations techniques not necessarily related to Bayesian networks. The user has the freedom to choose the space \( Q \), the objective function that must be minimized and the algorithm used to perform the optimization. The algorithm \textit{loopy belief propagation} is included in this class. Major drawbacks are lack of convergence, especially in case of strong dependencies between nodes, and qualification of the approximation [15, p. 474]. We have decided to not go into further detail of this specific type of inference.

In sampling methods events are simulated from a distribution, ideally from a distribution which is close to the real posterior distribution. The simulated events, or samples, are used to estimate probabilities or other parameters of the distribution.

For example, suppose \( Y \) is a variable of interest and \( Y_1, \ldots, Y_N \) is an independent identically distributed sample of \( Y \), the probability of \( Y = y \) can be approximated by

\[
P(Y = y) = E[1_{Y=y}] \approx \frac{1}{N} \sum_{i=1}^{N} 1_{Y_i = y},
\]

where \( 1_{Y = y} \) is the indicator function of the event \( Y = y \).

As noted, the problems that arise are

1. If the distribution \( P(Y \mid E = e) \) is not known, then how to obtain samples?
2. How many samples are needed to obtain a reliable estimation?

The general frameworks that tackle the first point are \textit{importance sampling} methods or \textit{Markov chain Monte Carlo} (MCMC) methods. Sadly both approaches yield no out-of-the-box algorithms and the second point is not easily answered. A lot of things have to be specified and pitfalls will arise as we will see in this chapter.
3.3.2 Notation and definitions

We introduce some simplified notation. Recall that the conditional probabilities \( P(X_i \mid \text{Pa}(X_i)) \) are often represented as so-called conditional probability tables and written as factors \( \phi_i \).

An assignment of a state to every node in the network, a state of the BN, can be written as

\[
(X_1 = x_1, \ldots, X_n = x_n),
\]

such that for all \( i, x_i \) lies in the domain of \( X_i \). For convenience we can also write \( (x_1, \ldots, x_n) \) as a state of the network, or simply use the boldface notation \( \mathbf{x} \) to denote a state of the BN.

With slight abuse of notation we write \( P(\mathbf{x}) \) for the probability of the state \( \mathbf{x} \) and if we write \( X_i = \mathbf{x} \) we assign the \( i \)-th component of \( \mathbf{x} \) to \( X_i \). Similarly, \( \phi(\mathbf{x}) = P(X_i = \mathbf{x} \mid \text{Pa}(X_i) = \mathbf{x}) \) if we mean the unique CPT-value of the CPT corresponding to node \( X_i \) and corresponding to state \( \mathbf{x} \).

If a CPT \( \phi_i \) contains zeros at certain positions we call this determinism, since such zeros represent hard logical clauses: \( \text{Pa}(X_i) = \mathbf{x} \implies \neg(X_i = x) \) for some state \( x \) of \( X_i \) and some configuration \( \mathbf{x} \) of the parents of \( X_i \).

If determinism is present in a BN, then there are infeasible states: \( \mathbf{x} \) such that \( P(\mathbf{x}) = 0 \). Recall that a feasible state \( \mathbf{x} \) is a state such that \( P(\mathbf{x}) > 0 \).

3.3.3 Forward sampling

One of the most basic sampling strategies for BN’s is called forward sampling. Before forward sampling is defined, we discuss how to sample from a finite discrete distribution.

**Definition 3.3.1 (Sampling from a categorical distribution).** For each node \( X_i \) in a BN we have that the distribution of \( X_i \) given the state of its parents

\[
P(X_i \mid \text{Pa}(X_i) = \mathbf{x})
\]

is a so-called categorical distribution, meaning that each of the \( k \) possible states of \( X_i \) is attained with probability \( p_1, \ldots, p_k \), where \( \sum p_i = 1 \). In practice the probabilities \( p_i \) can be interpreted as the column of a CPT, see Figure 2.1. Sampling from such distributions can be done by partitioning the interval \([0, 1)\) in \( k \) smaller intervals (corresponding to the states) of the form

\[
[0, p_1), [p_1, p_1 + p_2), [p_1 + p_2, p_1 + p_2 + p_3), \ldots, [1 - p_k, 1)
\]

and subsequently use a psuedo-random number generator (which is available in most programming languages) to generate a random number \( r_0 \) in the interval \([0, 1)\). The unique smaller interval in which \( r_0 \) is found yields the sampled state.
Definition 3.3.2 (Forward sampling). Given a topological ordering (recall Definition 2.2.7) of the network we can generate samples of the network by sampling nodes in the order of the topological ordering.

Due to sampling of nodes in the topological ordering we start at root nodes of the network, for every non-root node that is sampled we have that its parents are already sampled (hence we need to sample from a categorical distribution). Continuing this process we obtain a state of the whole BN and this method is called Forward sampling.

Note that the probability of sampling a certain state $x$ with forward sampling equals $P(x)$, where $P$ is the factorized distribution over the CPTs. Therefore we sample from the desired distribution.

To deal with evidence we could simply ‘trow away’ samples which are not compatible with the evidence; this is called rejection sampling. A major drawback is that evidence with low probability is unlikely to occur in the sample and therefore a large part of the samples we create get rejected. This will be explained in more detail in Section 3.4.

In fact we use the following estimation for events $A$ and $B$ given a collection of samples $S$

$$P(A \mid B) = \frac{P(A, B)}{P(B)} = \frac{E[1_{A,B}]}{E[1_B]} \approx \frac{\sum_{x \in S} 1_{A,B}(x)}{\sum_{x \in S} 1_B(x)}.$$

Example 3.3.1. Recall the BN BloodPressure in Figure 2.1. A topological ordering is given by $(K, L, BP, S, M)$ and forward sampling according to this order (from left to right) could yield the samples

$$(k_g, l_b, b_e, s_y, m_e)$$
$$(k_g, l_g, b_g, s_y, m_n)$$
$$(k_b, l_b, b_e, s_n, m_e)$$
$$(k_g, l_b, b_n, s_n, m_n)$$
$$(k_b, l_g, b_e, s_y, m_e)$$
$$(k_g, l_g, b_n, s_y, m_n).$$

From the above samples we estimate $P(M = m_n) \approx \frac{3}{6}$.

Suppose the evidence is given by $M = m_e$, then all samples with $M = m_n$ get rejected:

$$(k_g, l_b, b_e, s_y, m_e)$$
$$(k_g, l_g, b_g, s_y, m_n)$$
$$(k_b, l_b, b_e, s_n, m_e)$$
$$(k_g, l_b, b_n, s_n, m_n)$$
$$(k_b, l_g, b_e, s_y, m_e)$$
$$(k_g, l_g, b_n, s_y, m_n).$$
From the above sample we estimate \( P(K = k_g \mid M = m_e) \approx \frac{1}{3} \).

### 3.3.4 Likelihood sampling

We can set evidence nodes to have a fixed value (the state corresponding to the evidence) and sample the other nodes of the network using forward sampling, but then we would not take into account the upward propagation (to the parents) of the evidence at non-root nodes. This can be compensated by ‘attaching’ a weight factor to each sample, which represents the likelihood that the concerning sample generated the evidence. The name of this method is chosen appropriately: likelihood-weighted sampling or simply likelihood sampling. In the following example likelihood sampling is applied to a concrete BN.

**Example 3.3.2.** Again, recall the BN BloodPress in Figure 2.1. With no evidence fixed, likelihood sampling is equal to forward sampling. Suppose evidence is given by \( K = k_b, S = s_n \) and \( M = m_e \).

First the evidence is fixed \((K = k_b, L = *, BP = *, S = s_n, M = m_e)\) and the non-evidence nodes are left open (denoted by *). Subsequently we sample the non-evidence nodes according to the (topological) ordering \((L, BP)\). Suppose this generates \( x = (k_b, l_g, b_e, s_n, m_e) \), then the weight \( w_x \) that is attached is the likelihood that the sample would arrive at the evidence given \( L = l_g \) and \( BP = b_e \), which equals the product of the CPT-entries

\[
w_x = P(S = S_n \mid L = l_g) \cdot P(M = m_e \mid BP = b_e) = 0.8 \cdot 0.9.
\]

Similarly we could generate a collection of samples \( S \):

\[
\begin{align*}
(k_b, l_b, b_e, s_n, m_e) &: w_1 = 0.8 \cdot 0.9 \\
(k_b, l_b, b_e, s_n, m_e) &: w_2 = 0.8 \cdot 0.9 \\
(k_b, l_g, b_e, s_n, m_e) &: w_3 = 0.2 \cdot 0.1 \\
(k_b, l_b, b_e, s_n, m_e) &: w_4 = 0.8 \cdot 0.9 \\
(k_b, l_g, b_e, s_n, m_e) &: w_5 = 0.2 \cdot 0.9
\end{align*}
\]

The posterior of event \( A \) given \( B \) can be estimated using a collection of samples \( S \) by

\[
P(A \mid B) \approx \frac{\sum_{x \in S} w_x \cdot 1_{A,B}(x)}{\sum_{x \in S} w_x \cdot 1_B(x)} = \frac{\sum_{x \in S} w_x \cdot 1_A(x)}{\sum_{x \in S} w_x},
\]

where the latter equality follows from the fact that event \( B \), the evidence, is fixed.
Soon we will see that likelihood sampling is a special case of the more general sampling strategy *importance sampling*, see Section 3.3.5.

It is an improvement of forward sampling, since it does not necessarily have to reject samples (unless the weight is zero). However, if the probability of evidence is small, we attach small weight to the samples or even reject samples, which would imply that we need a lot of samples to estimate properly. If the CPT of the evidence node contains zeros, then it is possible that the likelihood of a generated sample is zero. We will discuss this problem in more detail in Section 3.4.

### 3.3.5 Importance sampling

If we look carefully at likelihood sampling, we see that we do not sample from the actual posterior distribution \( P(X | e) \) (we do not have access to this distribution in general, that is why we moved to sampling methods in the first place), but from a distribution \( Q \), which at most nodes equals \( P \) except at the evidence nodes. The evidence nodes are fixed (attain a certain state with probability 1) irrespectively of the states of the parents. See Example 3.3.3

In a more general setting the posterior \( P \) is called the *target distribution* and \( Q \) is called the *proposal distribution* or *sampling distribution* (likelihood sampling has a particular choice of \( Q \)). We should compensate for the difference between \( P \) and \( Q \) using weights or an acceptance probability.

**Example 3.3.3.** Observe Example 3.3.2. The actual distribution from which we sample in likelihood sampling is displayed in Figure 3.1. This distribution is changed at the evidence nodes (colored red). The upward influence is removed at non-root evidence nodes. The downward influence of evidence implies that certain columns can be neglected.

There are more choices of \( Q \) possible, e.g. evidence pre-propagation importance sampling [30]. Often there is a tradeoff between how close \( Q \) is to \( P \) and how much weight we attach to samples.

The difficulty is to find a proposal distribution close to the real posterior. A major drawback is a high probability of generating low weight samples, making the sampling process inefficient and hard to derive reliable estimations. More discussion and examples of this drawback are given in Section 3.4.

### 3.3.6 Gibbs sampling

Suppose we have a feasible initial state of our network \( x^{(0)} \) (finding one is nontrivial, recall Theorem 3.1.2, but we could use for example forward sampling to search for a state compatible with the evidence). From this
Figure 3.1: The proposal distribution $Q$ for the original BloodPressure network from Figure 2.1 with (red colored) evidence nodes $K = k_B, S = s_n$ and $M = m_e$.

initial state we can, one by one, sample the unobserved variables given the current state of all the other variables, e.g. a transition

$$(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}) \rightarrow (x_1^{(1)}, x_2^{(0)}, \ldots, x_n^{(0)})$$

by sampling $x_1^{(1)}$ from $P(X_1 | x_2^{(0)}, \ldots, x_n^{(0)})$. Subsequently $x_2$ is sampled from $P(X_2 | x_1^{(1)}, x_3^{(0)}, \ldots, x_n^{(0)})$, or simply $P(X_2 | x_{-2})$ where $x_{-i}$ means the current state minus the $i$-th component. Doing this for all unobserved variables

$$x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$$

$$\rightarrow (x_1^{(1)}, x_2^{(0)}, \ldots, x_n^{(0)})$$

$$\rightarrow (x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(0)})$$

$$\vdots$$

$$\rightarrow (x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}) = x^{(1)}$$

a new state $x^{(1)}$ of the BN is obtained. Repeating this process generates a Markov chain of states of the BN:

$$x^{(0)} \rightarrow x^{(1)} \rightarrow x^{(2)} \rightarrow \ldots$$

This method is called Gibbs sampling and is part of the broader class of Markov chain Monte Carlo (MCMC) sampling algorithms.
Example 3.3.4. Recall the BloodPressure example. Suppose the initial state is
\[ x^{(0)} = (k_g, l_g, b_n, s_y, m_n). \]
A possible Gibbs transition is given by:
\[
\begin{align*}
&x^{(0)} = (k_g, l_g, b_n, s_y, m_n) \\
&\quad\quad\rightarrow (k_b, l_g, b_n, s_y, m_n) \\
&\quad\quad\rightarrow (k_b, l_g, b_n, s_y, m_n) \\
&\quad\quad\rightarrow (k_b, l_g, b_e, s_y, m_n) \\
&\quad\quad\rightarrow (k_b, l_g, b_e, s_n, m_n) \\
&\quad\quad\rightarrow (k_b, l_g, b_n, s_y, m_e) \\
&\quad\quad= x^{(1)}.
\end{align*}
\]

In this example we show how to calculate the transition probability of
\((k_g, l_b, b_n, s_y, m_n) \rightarrow (k_g, l', b_n, s_y, m_n)\), which is given by \(P(L \mid k_g, b_n, s_y, m_n)\).

It turns out that the calculation of \(P(X_i \mid x_{-i})\) for a node \(X_i\) in a BN, reduces to an expression where only the Markov blanket of \(X_i\) is important, meaning the parents, children and the parents of the children of \(X_i\). This is due to the fact that factors independent of \(X_i\) will cancel out each other, as in the following equations
\[
P(L \mid k_g, b_n, s_y, m_n) = \frac{P(L, k_g, b_n, s_y, m_n)}{\sum_{l \in \text{Val}(L)} P(l, k_g, b_n, s_y, m_n)}
= \frac{1}{Z} \prod_i \phi_i(L, k_g, b_n, s_y, m_n)
= \prod_{l \in \text{MB}(L)} \phi(L, k_g, b_n, s_y, m_n)
\]
where \(\text{MB}(X_i)\) denotes the collection of factor indices corresponding to the Markov blanket of \(X_i\).

3.3.7 Markov chain Monte Carlo methods

Gibbs sampling can be placed in the broader class of sampling methods named Markov chain Monte Carlo methods. In this section we discuss this class. We only consider Markov chains on finite state spaces.

Definition 3.3.3 (Regular Markov chain). A Markov chain is called regular if there exists a \(k\) such that for every two states \(x\) and \(x'\), the probability of transitioning from \(x\) to \(x'\) in precisely \(k\) steps is strictly positive.
**Definition 3.3.4** (Reversible Markov chain). A finite-state Markov chain is called reversible if there exists a unique distribution $\pi$ such that for all states $x$ and $x'$:

$$
\pi(x)T(x \to x') = \pi(x')T(x' \to x),
$$

where $T$ denotes the transition probability.

**Definition 3.3.5** (Stationary distribution). A distribution $\pi$ is a stationary distribution for a Markov chain if

$$
\pi(x') = \sum_x \pi(x)T(x \to x').
$$

In words this means that the probability of being in a state is equal to the probability of transitioning into it from a randomly sampled predecessor.

From the theory of Markov chains we know that if a Markov chain is regular and satisfies reversibility with respect to a distribution $\pi$ then $\pi$ is the unique stationary distribution [15, p. 516].

**Example 3.3.5.** Gibbs sampling can also be viewed as a Markov chain which cycles over multiple transition models $T_i$, where

$$
T_i((x_{-i}, x_i) \to (x_{-i}, x'_i)) = P(x_i | x_{-i}),
$$

and where $x_{-i}$ denotes the state of the variables $X \setminus X_i$. From this we see that for each $x$ and $x'$ which differ at the $i$-th component we have

$$
P(x)T_i(x \to x') = P(x_i, x_{-i})P(x'_i | x_{-i})
= P(x_i | x_{-i})P(x_{-i})P(x'_i | x_{-i})
= P(x_i | x_{-i})P(x')
= T_i(x' \to x)P(x'),
$$

where we used the chain rule two times. Thus Gibbs sampling generates a Markov chain that satisfies reversibility.

Unfortunately, Gibbs sampling does not guarantee the uniqueness of the stationary distribution since it does not necessarily satisfy regularity. In Gibbs sampling for Bayesian networks the regularity property often fails to hold in case of deterministic dependencies between nodes, as we will see in Section 3.4. As a consequence, the chain generated by Gibbs sampling does not visit the whole state space and we cannot derive accurate knowledge about the real posterior distribution.

Fortunately, there exist other transition models which generate Markov chains and we will introduce one in Chapter 4.
3.4 Drawbacks in common sampling inference methods

The sampling methods discussed in the previous section contain some serious drawbacks. In this section we discuss these drawbacks and give concrete examples that reveal them.

3.4.1 Deterministic relations

One of the main causes of the drawbacks in sampling methods is caused by deterministic relations between variables. The difficulties with sampling methods regarding BN’s with deterministic information has been observed before [11,23,27,30].

We already mentioned the notions of determinism, incompatibleness and feasibility and for convenience we repeat them here.

**Definition 3.4.1** (Deterministic relations and incompatibleness). If a CPT corresponding to a node $X_i$ contains a 0, this means that for a certain state $x$ of $X_i$ and a certain state $\mathbf{x}$ of the parents of $X_i$ we have

$$P((X_i = x \mid Pa(X_i)) = \mathbf{x}) = 0.$$ 

We call this zero entry of a CPT a deterministic relation.

In that case, if $X_i = x$ is given as evidence, then the parent state $Pa(X_i) = \mathbf{x}$ has zero probability, i.e. $P(Pa(X_i) = \mathbf{x} \mid X_i = x) = 0$. We will often say in such cases that the state $\mathbf{x}$ is incompatible with the evidence or infeasible.

**Example 3.4.1.** Consider the BN in Figure 3.2. We see that the CPT corresponding to $B$ contains a deterministic relation:

$$P(B = 1 \mid A = 0) = 0.$$ 

3.4.2 Forward and rejection sampling (drawbacks)

Recall the method rejection sampling discussed in Section 3.3.3.
Example 3.4.2. \textit{Lets apply the reject sampling method on the BN in Figure 3.3.} The following set of 10 samples is generated using Forward sampling \[ S = \{ (0, 0), (0, 1), (0, 1), (0, 0), (0, 1), (0, 0), (0, 1), (0, 0), (0, 0), (0, 0) \}, \]
which is a realistic draw. Suppose evidence is given by \( A = 1 \). Now, according to reject sampling, we need to reject the samples which are not compatible with the evidence, but this implies the rejection of all samples in \( S \)!

A large part of the generated samples get rejected due to the fact that with high probability samples are generated that are incompatible with the evidence.

In the previous example on average 1 out of 100 samples that is generated using forward sampling is compatible with the evidence. In general, when the probability of the evidence is small, this process is inefficient.

Moreover, finding one feasible state compatible with the evidence is \( \mathcal{NP} \)-hard, as was mentioned in Section 3.1.

3.4.3 Likelihood and importance sampling (drawbacks)

In the following example likelihood sampling is applied and we show what happens in the case that with high probability samples are generated that are incompatible with the evidence.

Example 3.4.3. \textit{Consider the BN in Figure 3.4.}

Figure 3.4: An example BN which causes problems with likelihood sampling

\[ \text{Given the evidence } B = 1, \text{ it holds that } A = 1 \text{ or } A = 2 \text{ with equal probability.} \]
Now, apply likelihood sampling to determine the posterior $P(A \mid B = 1)$. Due to the distribution of $A$ we generate with high probability samples with $A = 0$, which leads to a zero weight sample, i.e. rejection. On average only 2 out of the 10000 samples that are generated will be compatible with the evidence, making this process inefficient and time consuming to derive an accurate estimation of the real posterior.

In this example one of the major drawbacks of importance sampling is shown. If the proposal distribution $Q$ is not ‘close’ to the real posterior $P$, the sampling process will be inefficient. As already mentioned, smarter proposal distributions have been developed, e.g. EPIS [30]. Although the authors present promising empirical results, they also admit that the role of the heuristics they use are not well understood. They cannot specify how ‘good’ their proposal is and furthermore the way they cope with determinism is somewhat loose (using an $\epsilon$-cutoff heuristic, where CPT values smaller than $\epsilon$ are replaced by $\epsilon$, the choice of this $\epsilon$ is based on experimental results and seems to be somewhat arbitrary).

The above discussion asks for methods that are guaranteed to improve (get closer to the posterior) during the sampling process and Markov chain Monte Carlo methods are capable of this.

### 3.4.4 Gibbs sampling (drawbacks)

One reproach that is made regarding importance sampling techniques is that is does not get ‘closer’ to the real distribution over time. Over and over samples are generated by sampling from a proposal distribution and this proposal distribution might be far away from the real posterior. MCMC methods can, under certain circumstances, guarantee that the sampling process gets closer to the real distribution as the number of iterations increases.

However, MCMC is not an out-of-the-box solution for inference problems. Several specifications have to be made. Firstly, the MCMC algorithm must start with an initial solution and how to find one? Secondly, a transition model needs to be specified for transitioning from state to state.

Recall Gibbs sampling as discussed in Section 3.3.6. To find a feasible initial state forward sampling can be used (but might be ineffective). The transition model was defined as updating each of the unobserved variables $X_i$ according to the distribution $P(X_i \mid x_{-i})$.

We already mentioned that the Markov chain generated by the Gibbs method is not necessarily regular, as will appear from the following example.

**Example 3.4.4.** Consider the BN in Figure 3.5. Suppose $(A = 0, B = 0)$ is the initial state and suppose no evidence is available, hence $A$ and $B$ are unobserved variables and are sampled according to Gibbs.

We have that the transition probability for $A$ and $B$ equal $P(B = 0 \mid A = 0) = P(A = 0 \mid B = 0) = 1$ (using Bayes rule in the second equality).
As a consequence the Markov chain behaves like

\[(0, 0) \to (0, 0) \to (0, 0) \to \ldots,\]

yielding all samples to equal \((0, 0)\). The real distribution for \(A\) on the other hand must equal \(P(A = 0) = P(A = 1) = 0.5\).

Starting from the initial state \((1, 1)\) would lead to

\[(1, 1) \to (1, 1) \to (1, 1) \to \ldots.\]

We see that the chains get trapped in a small part of the so-called state space and that it is unable to move freely around in the whole state space. Here the main problem of Gibbs sampling in BN’s with deterministic relations becomes visible: the chain gets trapped and cannot visit the whole state space and therefore the Markov chain generated by this process does not has the desired unique stationary distribution.

If \(P(B \mid A)\) would be given by

\[
P(B \mid A) = \begin{array}{c|cc}
A & A = 0 & A = 1 \\
\hline
B = 0 & 0.9999 & 0.0001 \\
B = 1 & 0.0001 & 0.9999 \\
\end{array}
\]

which contains a near-deterministic relation, then the chain behaves similar: being trapped in state \((0, 0)\) or \((1, 1)\) for most of the time, making it hard to derive knowledge about the real distribution.

The problem just discussed is also known as bad mixing, where the term mixing is commonly used in MCMC methods to point out the extent to which the chain is able to visit the whole state space.

Methods to improve mixing of MCMC methods have been proposed \([2, 5, 10]\). For example one could update (with Gibbs sampling) two or more variables at once from their joint distribution conditioned on all other variables. This is called blocked Gibbs. The pairwise deterministic relations between nodes would be evaded, but this will not capture all deterministic dependencies as we will see in the next example.
Example 3.4.5. Suppose we have the following chain shaped BN (not to be confused with a Markov chain)

\[ X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_N, \]

where each \( X_i \) can attain the values \( \{0, 1, 2, 3\} \). Let \( X_1 \) be uniformly distributed and let each \( X_i \) for \( i = 2, \ldots, n \) be conditionally distributed according to

\[
\begin{array}{c|cccc}
  & X_{i-1} = 0 & X_{i-1} = 1 & X_{i-1} = 2 & X_{i-1} = 3 \\
  \hline
  X_i = 0 & 0.5 & 0.5 & 0 & 0 \\
  X_i = 1 & 0.5 & 0.5 & 0 & 0 \\
  X_i = 2 & 0 & 0 & 0.5 & 0.5 \\
  X_i = 3 & 0 & 0 & 0.5 & 0.5 \\
\end{array}
\]

Applying Gibbs sampling will not reveal the correct posterior distribution since if \( X_1 \in \{0, 1\} \), then all \( X_i \in \{0, 1\} \) and the sample is trapped in \( \{0, 1\} \) for all nodes. We see that the state space contains two disconnected regions for Gibbs sampling, namely all \( X_i \in \{0, 1\} \) or all \( X_i \in \{2, 3\} \).

Also, we see that, due to the dependence in this BN, blocked Gibbs will not solve this problem unless we update all \( n \) nodes jointly (which would mean we could sample directly from the posterior distribution).

Observing the behavior of multiple Gibbs chains from different starting points will give useful insight in this example, but in general it is hard to determine the disconnected regions.
Chapter 4

Prune sampling

As we saw with the MCMC method Gibbs sampling, the deterministic dependencies can prevent the Markov chain from visiting certain regions of the state space. The Markov chain generated by Gibbs sampling is not necessarily regular.

In this section we present the MCMC method: prune sampling. It turns out that prune sampling generates a regular and reversible Markov chain with respect to the desired distribution.

The mathematical definition of prune sampling is addressed in Section 4.1. We have implemented prune sampling in the programming language Python, see Appendix A. In Section 4.2 it is explained how several non-trivial steps in the algorithm can be realized in practice. The prune sampling algorithm is inspired by the MC-SAT algorithm [23] and comparisons to MC-SAT and to other methods are made in Section 4.3.

4.1 Pruning and the prune sampling algorithm

We will start with defining prune sampling. Secondly, we will identify the transition probability of prune sampling. Finally, we proof that it generates a reversible Markov chain.

4.1.1 Definition of prune sampling

Let $\mathcal{B}$ be a Bayesian network and let

$$\mathcal{C} := \{k(i) : c_{k(i)} \text{ is a CPT-value in the } i\text{-th CPT } P(X_i \mid Pa(X_i)), \text{ indexed by } k(i), i = 1, \ldots, n\}$$

be the collection of all CPT-indices of a BN.

Example 4.1.1. Consider the following BN
The collection of CPT-indices $C$ for this BN contains 6 indices: 2 for the CPT corresponding to node $A$ and 4 for the CPT corresponding to node $B$.

Suppose we prune CPT indices around a certain state $x$ of the network. To make this more formal, consider the following definitions:

**Definition 4.1.1 (CPT-indices corresponding to a state $x$).** A state $x = (X_1 = x_1, \ldots, X_n = x_n)$ of the BN corresponds to a unique collection of CPT-values $c_{k(i)}$ for $i = 1, \ldots, n$. The collection $C_x$ of CPT-indices $k(i)$, corresponding to these values, are called the CPT-indices corresponding to $x$. Note that $P(x) = \prod_{k(i) \in C_x} c_{k(i)}$.

**Definition 4.1.2 (States corresponding to a set of CPT-indices).** We can have a collection of CPT-indices $C$. The set $S_C$ of states $x$ that use only the CPT-indices in the collection $C$ are called states corresponding to the CPT-indices in $C$.

**Definition 4.1.3 (Pruning given/around $x$).** Let $C_{x,p}$ be the subset of $C$ that is constructed by adding each CPT-index $k(i) \in C \setminus C_x$ with probability $1 - c_{k(i)}$ to the set $C_{x,p}$ and with probability $c_{k(i)}$ not. We say that the collection $C_{x,p}$ contains the pruned CPT-indices. Note that $C_{x,p}$ is a random set.

The collection of CPT-indices that do not get pruned is given by $C_{x,n} := C \setminus C_{x,p}$.

Note that in particular $C_x \subset C_{x,n}$ and therefore $x \in S_{C_{x,n}}$.

The probability of generating $C_{x,p}$ and $C_{x,n}$ is given by

$$\prod_{k(i) \in C_{x,p}} (1 - c_{k(i)}) \cdot \prod_{k(i) \in C_{x,n} \setminus C_x} c_{k(i)}.$$ 

This process is called pruning given/around $x$.

**Definition 4.1.4 (Uniform sampling over a set of states).** Let $S_{C_{x,n}}$ be the set of feasible states corresponding to the CPT-indices which are not pruned. We define $\mathcal{U}(S_{C_{x,n}})$.
as the uniform distribution over the states in $S_{C_x,n}$ and we write

$$U(S_{C_x,n})(y) = \frac{1}{|S_{C_x,n}|}$$

for the probability of sampling state $y$ with respect to this uniform distribution.

**Definition 4.1.5** (Prune sampling algorithm). Start with an initial state $x^{(0)}$. For $i = 1, 2, \ldots$ prune around $x^{(i-1)}$ to obtain $C_{x^{(i-1)},n}$ and sample $x^{(i)}$ from $U(S_{C_x^{(i-1)},n})$. See Algorithm 1.

**Algorithm 1** Prune sampling algorithm

```plaintext
function PruneSampling(BN, initial, numsamples)
    $x^{(0)} \leftarrow$ initial
    $S \leftarrow \{x^{(0)}\}$
    for $i \leftarrow 1$ to numsamples do
        $C_{x^{(i-1)},p} \leftarrow$ Prune around $x^{(i-1)}$  \(\triangleright\) See Definition 4.1.3
        $C_{x^{(i-1)},n} \leftarrow C \setminus C_{x,p}$
        $x^{(i)} \sim U(S_{C_x,n})$
        $S \leftarrow S \cup x^{(i)}$
    end for
    return $S$
end function
```

Note that with strictly positive probability we have that $C_{x^{(i-1)},n}$ contains all the non-zero indices in $C$, implying that $S_{C_x^{(i-1)},n}$ contains all feasible states of the BN. This means the prune sampling algorithm generates a regular Markov chain: with positive probability a state $x$ can transition to any other feasible state $y$ in one step. The reversibility conditions takes more effort to show and is addressed in the next two sections.

We finish this section with an example:

**Example 4.1.2.** Consider the BN in Figure 4.1.

Note that the lower the value of the CPT-entry, the higher the probability that the index gets pruned. We see that $S_{C_x,n}$ contains two feasible states, namely $(k_y, l_b, b_e, s_y, m_e)$ (which is boldfaced) and $(k_y, l_b, b_e, s_n, m_e)$.

### 4.1.2 Transition probability

To transition from a state $x$ to a state $y$ we need to prune around $x$ such that non of the indices corresponding to $y$ is pruned. This leads to the following definition:
Figure 4.1: A pruned version of the BloodPressure network around the boldfaced state $x = (k_g, l_b, b_e, s_y, m_e)$.

**Definition 4.1.6** (Pruning around $x$ and $y$). Let $C_{\{x,y\},p}$ be the subset of $C$ that is constructed by pruning around $x$ or pruning around $y$ such that none of the indices corresponding to $x$ and non of the indices corresponding to $y$ is contained in $C_{\{x,y\},p}$.

The collection of CPT-indices that do not get pruned is given by

$$C_{\{x,y\},n} := C \setminus C_{\{x,y\},p}.$$  

For each two states $x$ and $y$ there are finitely many ways ($K$) to create a pruned collection $C_{\{x,y\},p,j}$ and a non-pruned collection $C_{\{x,y\},n,j}$ ($j = 1, \ldots, K$), such that $x$ can transition to $y$ by sampling from $\mathcal{U}(S_{C_{\{x,y\},n,j}})$. For $j = 1, \ldots, K$ we define the transition probabilities from $x$ to $y$ by pruning a certain collection by:

$$Q_j(x \rightarrow y) := \left( \prod_{k(i) \in C_{\{x,y\},p,j}} (1 - c_{k(i)}) \right) \cdot \left( \prod_{k(i) \in C_{\{x,y\},n,j}\setminus C_x} c_{k(i)} \right) \cdot \mathcal{U}(S_{C_{\{x,y\},n,j}})(y)$$

$$= \left( \prod_{k(i) \in C_{\{x,y\},p,j}} (1 - c_{k(i)}) \right) \cdot \left( \prod_{k(i) \in C_{\{x,y\},n,j}\setminus C_x} c_{k(i)} \right) \cdot \frac{1}{|S_{C_{\{x,y\},n,j}}|}.$$  

(4.1)

In words Equation (4.1) means the probability of pruning certain CPT-indices around $x$, such that non of the CPT-indices corresponding to $y$ is
pruned, and subsequently sampling \( y \) uniformly from the states corresponding to the CPT-indices that where not pruned.

The total probability of transitioning from \( x \) to \( y \) is therefore given by

\[
Q(x \rightarrow y) = \sum_{j=1}^{K} Q_j(x \rightarrow y).
\]

### 4.1.3 Reversibility

To show reversibility we need to show that the transition probability satisfies the detailed balance equation:

\[
P(x)Q(x \rightarrow y) = P(y)Q(y \rightarrow x)
\]

which is the same as

\[
P(x) \left( \sum_{j=1}^{K} Q_j(x \rightarrow y) \right) = P(y) \left( \sum_{j=1}^{K} Q_j(y \rightarrow x) \right),
\]

but then it is sufficient to show

\[
P(x)Q_j(x \rightarrow y) = P(y)Q_j(y \rightarrow x), \quad (4.3)
\]

for all \( j = 1, \ldots, K \).

The following equation shows that (4.3) holds:

\[
P(x)Q_j(x \rightarrow y) = P(x) \cdot \left( \prod_{k(i) \in C_{(x,y), p,j}} (1 - c_{k(i)}) \right) \cdot \left( \prod_{k(i) \in C_{(x,y), n,j} \setminus C_x} c_{k(i)} \right) \cdot \frac{1}{|S_{(x,y), n,j}|}
\]

\[
= \prod_{k(i) \in C_x} c_{k(i)} \cdot \left( \prod_{k(i) \in C_{(x,y), p,j}} (1 - c_{k(i)}) \right) \cdot \left( \prod_{k(i) \in C_{(x,y), n,j} \setminus C_x} c_{k(i)} \right) \cdot \frac{1}{|S_{(x,y), n,j}|}
\]

\[
= \left( \prod_{k(i) \in C_{(x,y), p,j}} (1 - c_{k(i)}) \right) \cdot \left( \prod_{k(i) \in C_{(x,y), n,j} \setminus C_x} c_{k(i)} \right) \cdot \frac{1}{|S_{(x,y), n,j}|}
\]

\[
= \left( \prod_{k(i) \in C_{(x,y), p,j}} (1 - c_{k(i)}) \right) \cdot \left( \prod_{k(i) \in C_y} c_{k(i)} \right) \cdot \left( \prod_{k(i) \in C_{(x,y), n,j} \setminus C_y} c_{k(i)} \right) \cdot \frac{1}{|S_{(x,y), n,j}|}
\]

\[
= \left( \prod_{k(i) \in C_{(x,y), p,j}} (1 - c_{k(i)}) \right) \cdot P(y) \cdot \left( \prod_{k(i) \in C_{(x,y), n,j} \setminus C_y} c_{k(i)} \right) \cdot \frac{1}{|S_{(x,y), n,j}|}
\]

\[
= P(y)Q_j(y \rightarrow x).
\]

We conclude that prune sampling generates a regular and a reversible Markov chain with respect to the desired distribution \( P \). As discussed in Section 3.3.7 we know that this implies that \( P \) is the unique stationary distribution of the Markov chain generated by prune sampling.
4.2 Practical implementation of prune sampling.

The algorithm requires the following two non-trivial steps

1. Generating an initial state.

2. Sampling uniformly over the pruned BN, i.e. sampling from the distribution \( \mathcal{U}(S_{c_x,n}) \).

In the next two subsections we will in each subsection propose methods to meet these requirements.

4.2.1 Generation of an initial states.

Finding a feasible state of the BN is \( \mathcal{NP} \)-hard. In this section we suggest a heuristic method to search for initial states of the BN. We start with a commonly used method named Forward sampling, see Definition 3.3.2. Subsequently we present two variations.

A problem with forward sampling is that in presence of evidence of low probability many of the samples we generate may be infeasible (zero probability due to incompatibility with the evidence). How many forward sampling walks will it take to generate one feasible sample?

Furthermore, the states we obtain from forward sampling are guided by the CPT probabilities, hence heavily biased. To obtain more diversity in the samples that are generated, we propose a custom forward sampling strategy. In this strategy the state of a node \( X_i \) is sampled uniformly from the set \( \{ x : P(X_i = x \mid Pa = x) > 0 \} \) (the states with non-zero probability).

Definition 4.2.1 (Random forward sampling). Suppose we apply Forward sampling, but instead of sampling node \( X_i \) from \( P(X_i \mid Pa(X_i) = x) \), we sample uniformly from \( \{ x : P(X_i = x \mid Pa = x) > 0 \} \). This method is called random forward sampling.

Example 4.2.1. In random forward sampling it is only relevant to know whether a CPT-value is zero or non-zero. The CPT in Example 3.4.1 reduces therefore to

\[
P(B \mid A) = \begin{bmatrix}
A=0 & A=1 \\
B=0 & * & * \\
B=1 & 0 & *
\end{bmatrix},
\]

where \( * \) means that the state corresponding to that entry is non-zero and thus available.

Definition 4.2.2 (Hybrid forward sampling). Consider a hybrid approach in which in the basis we apply forward sampling, but at each node \( X_i \) either (say with probability \( p \)) the sampling distribution \( P(X_i \mid Pa(X_i) = x) \) is chosen or (with probability \( 1-p \)) the uniform distribution over \( \{ x : P(X_i = x) > 0 \} \).
\( x \mid Pa = x \) > 0} is chosen. The parameter \( p \) can be chosen as wished by the user.

Using hybrid forward sampling we can try to generate a broad collection of initial states of the BN. This search strategy also provides a heuristic method for estimation of the MPE/MAP (most probable explanation/maximum a posteriori estimate, which is a state \( x \) such that \( P(x) \) is maximized) state, [18]. Starting a Markov chain from a highly (or the most) probable state feels intuitively smart, since this suggests we are already close to the posterior distribution.

For the BN of interest, hybrid forward sampling was able to generate multiple feasible initial solutions within acceptable time. Local search techniques to address the generation of feasible states have been suggested in [18] and we would advise to investigate this area if one wants to develop more intelligent ways to generate initial states.

### 4.2.2 Sampling from \( U(S_{\mathcal{C},n}) \).

In this section we explain how sampling from \( U(S_{\mathcal{C},n}) \) can be realized.

Note that due to the pruning of CPT-entries in the original BN, the number of feasible states in the pruned BN is much smaller in comparison to the number of feasible states in the original BN. This can be seen from Example 4.1.2, where the initial network had \( 2^5 \) feasible states, the pruned BN had only 2!

Assuming we have sufficient memory, a breath first search approach can be used to list all feasible states of the pruned BN. From this collection we can easily draw uniformly a state. Where exhaustive listing of all feasible states of the original BN was impossible (due to too much memory and time consumption), the exhaustive listing of all solutions of the pruned BN went surprisingly well within acceptable time with our BN of interest. This was a major breakthrough in this research project. However, in comparison to Gibbs sampling the uniform sampling step is relatively expensive.

If one still runs into memory problems or one wants to reduce the computational effort heuristic methods can be developed. We propose to use random forward sampling to construct a set \( S \) (of predetermined fixed size) of feasible states of the pruned BN. Subsequently a state from \( S \) can be sampled uniformly. This can be interpreted as a trade-off between uniformity and computational effort. A more intelligent method, based on simulated annealing is suggested by [28]. We advise to investigate this area if one wants to develop an intelligent heuristic for the uniform sampling step.
4.3 Comparison to other sampling techniques.

In comparison to importance sampling techniques, MCMC techniques have less trouble with low evidence and achieve better results when the posterior distribution is very different from the prior [15]. As opposed to importance sampling, MCMC methods have the potential to move closer to the desired posterior distribution over time.

4.3.1 Comparison to MC-SAT

The algorithm MC-SAT [23] is a special case of the more general MCMC strategy *slice sampling* or the strategy of using *auxiliary variables* [2, 8, 10, 23]. The authors of [23] present the more general framework called Markov Logic Networks (MLN’s) which can be used to represent a BN [24]. Furthermore, in the MLN setting, they have designed a novel algorithm named MC-SAT, which in theory is a sound MCMC algorithm, meaning it generates a Markov chain that is regular and which is reversible, even in the presence of deterministic relations.

To apply MC-SAT on BN’s a translation to a weighted SAT problem is necessary. This has two drawbacks. Firstly, an explicit translation of a BN to a weighted SAT problem is memory intensive (every CPT entry needs to have an associated clause representing the state of the nodes and the state of the parent nodes). Secondly, the graphical dependencies are lost when translating to a collection of weighted clauses.

To address the generation of an initial state MC-SAT uses local search SAT solvers. To address the uniform sampling step, MC-SAT uses a heuristic. More specific, a hybrid local search strategy (which can be found in [28]) to generate states nearly uniform (instead of completely uniform). This is a major difference between MC-SAT and prune sampling. Prune sampling gives a way to guarantee the uniform sampling step.

4.3.2 Comparison to Gibbs

The usual MCMC method in case of discrete BN’s is Gibbs sampling. Gibbs sampling has the major drawback that it generally does not generate a regular Markov chain. Especially in the case of deterministic relations in the BN, the chains constructed by Gibbs sampling often fails to visit certain areas of the state space. Prune sampling does not suffer from this drawback, even in case of deterministic relations, see Example 4.3.1. Within the class of MCMC methods, the guarantee of convergence of prune sampling makes it an attractive MCMC algorithm since the user does not need to take into account disconnected regions in the state space.

Using the following analysis tools we show the power of prune sampling.
Definition 4.3.1 (Mean trace plot). Let \( S_N := \{s_0, s_1, \ldots, s_N\} \) be a collection of samples of a node \( X_i \) of interest. If in the long-term the samples we draw are from the desired distribution, then for large samples we expect convergence in the mean \( S_N \) of the sample to the real mean \( E[X_i] \). The exact mean \( E[X_i] \) is not known, but we can check whether the plot of \( S_i \), where \( S_i := \{s_j \in S_N : 0 \leq j \leq i\} \), over \( i \), shows asymptotic behavior, see Figure 4.2. The plot of the mean \( S_i \) for \( i = 1, \ldots, N \), is called the mean trace plot.

If the prune sampling algorithm is started from several different initial solutions and all mean trace plots of the chains show asymptotic behavior and furthermore they converge to the same value, then this is a strong indication of converging behavior, see Figure 4.2 for an example of a mean trace plot of three chains.

Example 4.3.1. Consider the BN in Figure 3.5. Suppose no evidence is given. As was shown before, applying Gibbs on this example generates either the Markov chain

\[
(0, 0) \to (0, 0) \to \ldots, \tag{4.4}
\]

if the initial state is \((0, 0)\) or

\[
(1, 1) \to (1, 1) \to \ldots, \tag{4.5}
\]

if the initial state is \((1, 1)\). Again we see that the Gibbs chains are trapped in the state space and therefore do not converge to the desired distribution (which would generate samples \((0, 0)\) and \((1, 1)\) with equal probability). To graphically display this we can make a so-called trace plot (see Figure 4.2) of the mean of a variable of the chain. We see that the mean \( \bar{A}_1 \) of \( A \) in chain (4.4) equals 0, whereas the mean \( \bar{A}_2 \) of \( A \) in the chain (4.5) equals 1. The mean (or expectation) \( E[A] \) of the real distribution should equal 0.5.

If we apply Prune sampling (we have implemented the algorithm in Python) on this example we obtain much better results, see Figure 4.3. From both initial states the prune sampling algorithm converges to the correct mean.
Figure 4.2: An example of three mean trace plots of a query node of interest, starting from three different initial solutions and thus three different Markov chains. Asymptotic behavior is shown for all three chains and also to the same mean. As noted in Definition 4.3.1, this is a strong indication of convergence.

Figure 4.3: A mean trace plot of Prune vs Gibbs sampling algorithm from two initial solutions on the BN from Example 4.3.1. The horizontal lines at 0 and 1 correspond to the mean trace plot of Gibbs sampling. The moving lines correspond to the mean trace plot of prune sampling. The horizontal line at 0.5 represents the exact mean of A.
Chapter 5

Bayes Linear Methodology

In this chapter we describe the difference between the common Bayesian approach and the Bayes linear (BL) approach. These two areas both include probabilistic graphical models, e.g. Bayesian networks. The methods differ in the specification of the models uncertainty and as a consequence differ in the way inference is done.

The goal is to investigate whether we can evade the inference methods in the common Bayesian approach by using the Bayes linear approach. To do so we first explain the difference between both methods. Secondly the Bayes linear method is explained. We interpret the Bayes linear method as an approximation to the common Bayes method and show that under certain conditions the BL method equals the common Bayes method. Finally we discuss to what extent the Bayes linear method can evade inference problems in common Bayes.

5.1 Bayes linear expectation

Up to now we have specified uncertainties (or Bayesian belief) in a probabilistic form, see for example the representation of the Bayesian network in Section 2.2.2 where we needed to specify CPTs. The Bayesian networks and inference methods described in the previous chapters will be called full Bayes or common Bayes methods.

Goldstein and Wooff argue that the full Bayes approach might be too ambitious for certain problems, since the specification requires knowledge to an extreme level of detail which is in practice beyond our reach [12, p.xvii]. To respect the limitations on our abilities to specify meaningful beliefs, Goldstein and Wooff propose an alternative method, the Bayes linear approach, which requires limited aspects of our beliefs and which we are able to specify. The Bayes linear approach uses expectation rather than probability as a primitive quantity for the quantification of uncertainty. By doing so, we move from a position that requires full probability specification (hence the
name full Bayes) to a less demanding position in which a collection of expectation statements appropriate to the situation is specified.

We want to have uncertainty in hypothesis, possibilities to adjust beliefs using data, but do not want to specify ourselves in terms of probability (which suffers from the disadvantage of difficulty (or even inability) to specify adequately). To cite Wisse: “So there is a need for a methodology that possesses the benefits of the full probabilistic Bayesian approach, but suffers less from the disadvantages just described” [29, p. 2]. And: “[...] we will consider the Bayes linear adjustment rules as approximation to full probabilistic updating and evaluate the accuracy of this approximation. We are not aware of any research that has been performed up to date on this topic” [29, p.60].

5.2 Bayes linear representation and adjusted expectation

Definition 5.2.1 (Bayes linear representation). A collection of random variables $\mathcal{X}$ for which prior expectation, variance and covariances are specified is called a Bayes linear representation.

If the values of some of the variables become known (i.e. data $D = d$ for $D \subset \mathcal{X}$ is available) then the expected values of the variables can be adjusted by linear fitting, see Definition 5.2.2. The data collection $D$ can be viewed as the Bayes linear analogue of the evidence collection $E$ in full Bayes.

Definition 5.2.2 (Adjusted expectation). The adjusted expectation $E_D[X]$ of a random variable $X$, given (a collection of) random variables $D = \{D_1, \ldots, D_k\}$ which have been observed ($D = d$), is the linear combination.

$$E_D[X] = \sum_{i=0}^{k} h_i D_i,$$

which minimizes

$$E \left( \left[ X - \sum_{i=0}^{k} h_i D_i \right]^2 \right)$$

over all vectors $h = (h_0, h_1, \ldots, h_k) \in \mathbb{R}^{k+1}$, where $D_0 := 1$. The adjusted expectation is also known as the Bayes linear rule for $X$ given $D$.

It holds that

$$E_D[X] = E[X] + \text{Cov}(X, D) \text{Var}(D)^{-1}(D - E[D])$$

49
is the unique solution to the minimization problem defined in Definition 5.2.2. Here \( \text{Cov}(X, D) \) denotes the vector with entries \( \text{Cov}(X, D_i) \), \( \text{Var}(D)^{-1} \) denotes the (generalized) inverse of the variance-covariance matrix of the elements in \( D \).

The adjusted expectation can be viewed as the analogue of the posterior distribution in full Bayes. Therefore the calculation of \( E_D[X] \) can be viewed as doing inference in the Bayes linear method.

Let \( \Omega \) be the space of possible outcomes of the observed variables \( D \). If \( \Omega \) is a finite set, then the adjusted expectation can be viewed as a function from space of possible outcomes of the observed variables, to a real number

\[
E_D[X] : \Omega \rightarrow \mathbb{R}.
\]

Since \( E_D[X] \) is a linear function on the domain of \( k \) random variables (\( D = \{D_0, \ldots, D_k\} \)), this function is uniquely defined by the \( k + 1 \) parameters, which are \( h_0, \ldots, h_k \).

In the following section we give more interpretations of the adjusted expectation.

### 5.2.1 Interpretations of adjusted expectation

In this section we show that the adjusted expectation \( E_D[X] \) can be interpreted as an approximation to the conditional expectation \( E[X \mid D] \): it is the same minimization problem, but over a more restricted space: the variables written as linear functions. To understand the relation between the adjusted expectation and the conditional expectation we give definitions of both.

In the general setting of prediction theory the adjusted expectation \( E_D[X] \), see Definition 5.2.2, is the best linear predictor of \( X \) given \( D \) in the squared error [25]:

\[
\arg\min_{f, f \text{ is linear}} E[(X - f(D))^2]. \tag{5.1}
\]

If we do not restrict our predictor function \( f \) then the conditional expectation equals

\[
E[X \mid D] = \arg\min_f E[(X - f(D))^2],
\]

or as the orthogonal projection of \( X \) onto \( V_D \) where \( V_D \) is the subspace of \( L^2(\mathbb{P}) \) containing the random variables \( Z \in L^2(\mathbb{P}) \) which can be written as \( f(D) \), for some \( f : \Omega \rightarrow \mathbb{R} \) [12, p. 82]. For a more formal and measure theoretic definition of the conditional expectation we refer to [3, 6].

Note that if \( \Omega \) is finite, then the conditional expectation is a function which is uniquely defined by \( |\Omega| \) parameters (a mapping for each outcome in \( \Omega \), including an empty event).
Another way to define (and often used to calculate) the conditional expectation $E[X \mid D]$ is by summing over the $X$ values with respect to the conditional distribution $P(X \mid D)$ [6, 25]:

$$E[X \mid D] = \sum_{x \in \text{val}(X)} x \cdot P(X = x \mid D).$$

5.2.2 Adjusted polynomial expectation.

As mentioned, we can interpret the adjusted expectation as an approximation to the conditional expectation. As also mentioned, in the discrete finite case, the conditional expectation is a function defined by finitely many parameters.

The number of possible states of the observed variables $D = \{D_1, \ldots, D_k\}$ equals $|\Omega| := \prod_{i=1}^{k} |\Omega_i| + 1$, where $\Omega_i$ is the set of values that can be attained by $D_i \in D$, and the +1 counts for no outcome (no available data).

Note that an arbitrary (single variable) linear function is uniquely defined by two points, and similarly a multivariate linear function defined on $k$ variables is uniquely defined on $k + 1$ points. Therefore the adjusted expectation equals the conditional expectation if the size of the space of outcomes $\Omega$ of $D$ equals $k + 1$, which in general is not the case (if all $D_i$ are binary, then $|\Omega| = 2^k + 1$).

We can define a projection on a larger subspace, that equals the conditional expectation:

**Definition 5.2.3 (Adjusted polynomial expectation).** Let

$$D^* := \left\{ \prod_{i=1}^{k} D_i^{p_i} \mid 0 \leq p_i \leq |\Omega_i| - 1 \text{ and not all powers } p_i \text{ equal zero} \right\}$$

and define the adjusted expectation of $X$ given $D^*$ as

$$E_{D^*} = E[X] + \text{Cov}(X, D^*) \ Var(D^*)^{-1}(D^* - E[D^*]).$$ (5.2)

Due to the polynomial form of elements in $D$ we call this the adjusted polynomial expectation of $X$ given $D$.

We can view $E_{D^*}$ as a polynomial in function in $D$ that minimizes the mean squared distance to $X$. This polynomial function is defined on the outcome space of $\Omega$ and is precisely defined by $|\Omega|$ parameters. Therefore it has to equal the conditional expectation.

Note that the matrix $\text{Var}(D^*)$ is of size $|\Omega|^2$, and $|\Omega| = \prod_i |\Omega_i|$, hence $\text{Var}(D^*)$ grows exponential in the number of data variables and corresponding outcomes.
5.2.3 The moment representation

In the previous section we saw that for a collection of discrete finite variables the adjusted polynomial expectation equals the conditional expectation. In this section we show how this can be used for inference.

**Proposition 5.2.1.** If $X$ is a finite discrete random variable with $N$ possible outcomes (say $X \in \{0, 1, \ldots, N-1\}$), then knowing the first $N-1$ moments of $X$ is sufficient for finding the probability distribution $P(X)$ of $X$.

**Proof.** The distribution $P(X)$ has $N$ unknowns. If we know the first $N-1$ moments, we can solve the linear system with the (linear independent) equations $E[X^i] = \sum_{x \in \text{val}(X)} x^i P(X = x)$ for $i = 1, \ldots, N-1$ and $\sum_{x \in \text{val}(X)} P(X = x) = 1$.

**Definition 5.2.4** (Moment representation.). Given a Bayesian network over $\mathcal{X} = \{X_1, \ldots, X_n\}$. Given $D \subset \mathcal{X}$ a specification of the following values

$$E[X_i^{p_i} X_j^{p_j}], \quad \forall X_j \in D, \quad 0 \leq p_i \leq |\Omega_i| - 1, \quad 0 \leq p_j \leq |\Omega_j| - 1,$$

is called a moment representation. These values can be used to calculate the adjusted polynomial representation (5.2). The moment representation can be viewed as a different prior specification of the BN, using moments instead of conditional probabilities.

Using independence relations in the BN we might save some calculations and storage, see also [12, ch. 10] which discusses graphical models in combination with adjusted expectation.

Suppose we have access to the moment representation. Given evidence or data $D$, the adjusted polynomial expectation can be used for the calculation of the moments $E[X^i | D]$ of $X$. Using Proposition 5.2.1 subsequently the posterior probability distribution of a variable of interest can be found. In this way full Bayes inference via the moment representation can be realized.

5.2.4 Discussion of inference via BL

One problem is: how to determine the moment representation of a Bayesian network?

We discuss two ways to address this problem:

1. instead of specifying a full Bayes BN, we could directly specify a moment representation of a collection of variables.

2. Given a BN defined by a DAG with CPTs, calculate all values in the moment representation.
The first solution makes sense in the case moments are of order 1 (Goldstein and Wooff and Wisse discuss how to obtain such knowledge from experts [12,29]), but experts may find it more difficult to specify higher order moments for certain variables.

The second option implies the calculation of moments, which implies the calculation of marginal probabilities of (products of) variables. But recall that the calculation of marginal probabilities is infeasible for general BN’s, see Theorem 3.1.1.

Can we use approximate methods to calculate the moment representation? To approximate the prior moment distribution we need to generate samples from the prior distribution (with no evidence available). Forward sampling is the most suitable method for this task. But this means that evidence (or data) of low probability will hardly be sampled. As a consequence, the approximated moment representation may not be accurate enough to make an reliable estimation of the exact adjusted polynomial expectation, see Example 5.2.1.

Example 5.2.1. In this example we show that the approximation of the moment representation using sampling methods results in an inaccurate approximation of the posterior distribution.

Consider the BN in Example 3.4.3 and suppose we generated 10000 samples \( S \) using forward sampling. Suppose the sample \( S \) consists of 9998 times \((A = 0, B = 0)\) and 2 times \((A = 1, B = 1)\) (which is a realistic draw). Note that the feasible state \((A = 2, B = 2)\) is not drawn.

Suppose the data is \( B = 1 \). Using this sample we obtain an approximation of the moment representation and subsequently an estimation of the adjusted (polynomial) expectation:

\[
E_B[A] \approx \hat{E}[A] + \hat{\text{Cov}}(A, B) \hat{\text{Var}}(B)^{-1}(B - \hat{E}[B])
\]
\[
= \frac{2}{10000} + 1 \cdot \left(1 - \frac{2}{10000}\right)
\]
\[
= 1,
\]

and similarly \( E_B[A^2] \approx 1 \). Solving the system of linear equations (see Proposition 5.2.1)

\[
\begin{bmatrix}
0 & 1 & 2 \\
0 & 1 & 4 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
P(A = 0) \\
P(A = 1) \\
P(A = 2)
\end{bmatrix}
= \begin{bmatrix}1 \end{bmatrix}
\]

results in the posterior \( P(A = 1) = 1 \), whereas the real posterior distribution should be \( P(A = 1) = P(A = 2) = 0.5 \).

In other words, this method also contains the drawbacks that occur in the rejection sampling strategy in common Bayes: it does not take into account the influence of evidence.
Another problem is exponential growth of the size of the matrix \( \text{Var}(\mathbf{D}^*) \), which has to be inverted to calculate the adjusted polynomial expectation. We conclude that inference via the BL method was no fruitful direction to evade the problems which arise in common Bayes.
Chapter 6

Conclusion

The main goal of this research project was to find a reliable (approximate) inference method that works both theoretically and practically. To this aim, we discussed inference methods in discrete Bayesian networks. Besides the mathematical framework, an overview of the common inference methods and their implications was given.

First we discussed to what extent commonly used inference methods gave reliable results. Exact inference in BN’s is often too computationally intensive. On the other hand, most approximate inference methods have serious drawbacks in handling BN’s with deterministic relations, as was revealed in this thesis. Importance sampling methods were found to be inefficient, since there was a high probability of generating samples incompatible with the evidence. In addition, Gibbs sampling, an MCMC method, was not able to visit certain regions in the state space. This was especially the case when deterministic relations in the BN were present. As a consequence, a reliable estimation of the posterior distribution could not be derived. Careful analysis of the obtained samples is thus advised when using the previously discussed common inference methods.

Secondly, we addressed to what extent the BL method is useful in doing inference in BN’s. This was the original approach suggested by TNO. As was shown in Chapter 5, we may conclude that inference via the BL method is no fruitful direction to evade the problems that arose in common Bayes.

We are not left empty-handed. Within the class of sampling inference methods, we proposed a Markov chain Monte Carlo algorithm: prune sampling. A major breakthrough in this research project was that prune sampling worked well on our BN of interest, making it practically interesting. Theoretically, prune sampling guarantees convergence and it outperforms the described sampling methods, especially in the presence of determinism. These factors make prune sampling also of scientific interest. However, prune sampling comes at a price, as it comprises two non-trivial and expensive steps. Optimization of these steps is therefore suggested for future research.
Appendices
Appendix A

Prune Sampling Toolbox for Python
Prune Sampling Toolbox for Python

A toolbox for approximate inference in Bayesian networks. The inference engine is Prune Sampling, an MCMC method.

Files
- parse_xdsl.py
- prune.py

Requirements
- Python 3 + IPython console (both are included in the Anaconda Python)

Functions
- Help functions:
  - parse_xdsl
  - print_nodes
  - print_states
- Main sampling functions:
  - prune_sampling
  - hybrid_fw
- Analysis functions:
  - probability_node
  - mean_trace_plot

Example

Open a IPython console from the Anaconda launcher and type the following commands (without 'In [number]'). The path to the file can be obtained by drag and drop into terminal. Note that in the first two lines no quotation marks are present. The command %matplotlib inline can be used for inline plots.

```python
In [1]: run //tsn.tno.nl/Data/Users/weikampr/Home/Desktops/parse_xdsl.py
In [2]: run //tsn.tno.nl/Data/Users/weikampr/Home/Desktops/prune.py
In [3]: monty_network = parse_xdsl('/Users/ronweikamp/Dropbox/TNO/python/MontyHall.xdsl')
In [4]: print_nodes(monty_network) PrizeDoor GuestDoor MontyDoor
In [5]: print_states(monty_network, 'GuestDoor')
In [6]: print_states(monty_network, 'GuestDoor')
In [7]: evm = {'GuestDoor':'A', 'MontyDoor':'B'}
In [8]: S1 = prune_sampling(monty_network, ['PrizeDoor'], evidence = evm, num_samples = 10000, num_prints=5000)
In [9]: S2 = prune_sampling(monty_network, ['PrizeDoor'], evidence = evm, num_samples = 10000, num_prints=5000)
In [10]: S3 = prune_sampling(monty_network, ['PrizeDoor'], evidence = evm, num_samples = 10000, num_prints=5000)
In [11]: %matplotlib inline #for inline plot, instead of new windows
```

58
In [12]: probability_node(monty_network,'PrizeDoor', S1)

In [13]: mean_trace_plot('PrizeDoor', samples=[S1,S2,S3])

In [14]: states = hybrid_fw(monty_network, evidence=evm, num_walks=100, num_sols=2)
In [15]: initial = states[0][0]
In [16]: S4 = prune_sampling(network, ['PrizeDoor'], initial=initial)

Function description

Help functions

• `parse_xdsl('path_to_xdsl_file')`. Reads GeNIe networks and parses to a structure in Python. **Input:** a GeNIe .xdsl file. **Output:** network structure in Python.

• `print_nodes(network)`. Reads Python network structure created by `parse_xdsl` and prints all nodes of the network.

• `print_states(network, node)`. Prints the states of the given node in the given network

Main sampling functions

• `prune_sampling(network, query_nodes)`. Generates samples using prune sampling. **Input:** First two mandatory arguments network and query_nodes (list of strings of node_names). **Output:** a dictionary with keys from query_nodes and values a list of sampled state indices.
Optional arguments (written as ‘option=value’, without quotes):

- **evidence** (dict with node keys and state values) Evidence of the network. Default=None.
- **num_samples** (integer) Desired number of samples. Default=10000.
- **burn** (integer) Number of samples to burn. Default = 0.
- **thin** (integer) Thinning value. Default = 1.
- **initial** (state, a dict with node keys and state values) An initial state from which prune sampling should start. Default = hybrid_fw.
- **num_walks** (integer) Number of walks performed by hybrid_fw. Default = 10000.
- **prob** (float, decimal number in [0,1]) The probability of selecting CPT values in hybrid_fw, with 1-prob it samples uniform over the positive values. Default = 0.5.
- **num_prints** (integer) At which rounds sampling info should be printed. Default = 1000.
- **heuristic** (0 or 1) If 1 is given, then at each step the random forward sampling heuristic is used to sample uniform in the prune sampling algorithm. Default = 0.
- **heuristic_values** ([num_walks, num_sols]) The values used in the random forward sampling heuristic. Default = [100,10].

**hybrid_fw(network)**. Generates states using hybrid forwad sampling. Input: a network (mandatory). Output: states (x) and corresponding weight (+log(P(x))), dependent on num_sols, it returns the lowest weight (highest probability states).

- Optional arguments (written as ‘option=value’, without quotes):
  - **evidence** (dict with node keys and state values) Evidence of the network. Default=None.
  - **num_walks** Number of walks performed. Default = 10000.
  - **prob** (float, decimal number in [0,1]) The probability of selecting CPT values in hybrid_fw, with 1-prob it samples uniform over the positive values. Default = 0.5.
  - **num_sols** (integer) Maximal number of solutions returned. Default=1.

**Analysis functions**

- **probability_node**(network, node, sample). Output: prints an estimation of the probability distribution of the given node from the given sample.

  - Optional arguments (written as ‘option=value’, without quotes):
    - **burn** (integer) Number of samples to burn. Default = 0.
    - **thin** (integer) Thinning value. Default = 1.
- **pie** (0 or 1) Plot a pie of the distribution
  Yes (1) or No (0). Default = 1.

- **mean_trace_plot**(node, samples=[S1, S2,...]) Function that
  create a mean trace plot for a node and given samples S1, S2, ...
  o Optional arguments (written as ‘option=value’, without quotes):
    - **burn** (integer) Number of samples to burn.
      Default = 0.
    - **thin** (integer) Thinning value. Default = 1.
    - **axis** ([x_1, x_2, y_1, y_2]). Custom axis of
      the plot.
Appendix B

Populaire samenvatting

Een Bayesiaans netwerk is een grafisch model dat een verzameling variabelen en hun afhankelijkheden representeert. Zie bijvoorbeeld Figuur B.1. In dit netwerk modelleren we de situatie waarin we een meting doen van iemands bloeddruk. We nemen aan dat de variabele bloeddruk afhankelijk is van het functioneren van iemands nier en iemands levensstijl (denk aan sport en eetgewoontes). Daarnaast kan een zekere levensstijl het al dan niet beoefenen van sport beïnvloeden. Voor het gemak nemen we aan dat iedere variabele twee toestanden kan aannemen. Zo kan de nier slecht of goed zijn, de levensstijl kan slecht of goed zijn, iemand doet niet of wel aan sport, de bloeddruk is normaal of exceptioneel en de meting is normaal of exceptioneel. Omdat dit vijf variabelen betreft die ieder in twee verschillende toestanden kunnen verkeren, heeft dit netwerk in totaal $2^5 = 32$ mogelijke toestanden.

Gegeven dit netwerk willen we de volgende vragen kunnen beantwoorden:

- Wat is de kans dat we een exceptionele bloeddruk meten, gegeven de observatie dat iemands nier slecht functioneert?
- Wat is de kans iemand een exceptionele bloeddruk heeft, gegeven de observatie dat de persoon sport?
- Wat is de kans dat iemands nier slecht functioneert, gegeven het feit dat we een exceptionele bloeddruk meten en daarnaast weten dat iemand een goede levensstijl heeft?

In dit ‘kleine’ Bayesiaanse netwerk is het uitrekenen van deze kansen mogelijk, maar zodra de netwerken groter worden, wordt dit uitrekenen zelfs voor een computer problematisch. Daartoe moeten we slimme methodes gebruiken om deze zogenaamde gevolgtrekkingen uit te kunnen rekenen. In dit project hebben we verschillende van deze methodes bekeken en vergeleken.

We hebben gezien dat de vergeleken methodes allen hun voor- en nadelen hebben. Daarnaast blijkt een specifieke methode, genaamd prune sampling,
Figure B.1: Het Bloeddruk netwerk. De variabelen en hun conditionele afhankelijkheden worden hier weergegeven. De getallen representeren zogenaamde (conditionele) kansen. De dikgedrukte getallen corresponderen met een zekere staat van het netwerk \( (K_g = k_g, L = l_g, BP = b_e, S = s_y, M = m_e) \)

goed te werken voor het netwerk dat gebruikt werd in dit project. Het algoritme kenmerkt zich door het snoeiende karakter en blijkt goede schattingen te geven van de kansen in Bayesiaanse netwerken.


