Stockholms universitet

Causal Inference through Structure Learning in Bayesian Networks

Simon Berggren

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2015:12
Matematisk statistik
Juni 2015

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Bachelor Thesis **2015:10**
http://www.math.su.se

Causal Inference through Structure Learning in Bayesian Networks

Simon Berggren*

June 2015

Abstract

This thesis is about structure learning in Bayesian Networks, and how this may be used for causal inference. A Bayesian Network is a graphical representation of a probability distribution, that provides a clear representation of conditional independences among the random variables. It consists of a graph and a set of probability tables. In the graph, each node represent a random variable, while edges and their orientations represent association and its nature. Causal inference is an analysis concerned with queries about cause and effect relationships. Specifically, in the phenomenon we wish to analyze, we use random variables to model the factors that describe this phenomenon, and infer causal associations among those variables.

Structure learning is a method for obtaining a Bayesian Network from data, i.e. given data consisting of a number of observations, where each observation consists of a realization of all the random variables in our model, the task is to infer a graph structure that represent the distribution of the variables. Hence, the main focus of this thesis will be to obtain a graph structure from data, such that the cause and effect relationships it exhibits, represents the corresponding causal associations in the underlying distribution. There are different approaches to structure learning in Bayesian Networks. In this thesis, focus is on the constraint-based approach, in which we use conditional independences, inferred from data, to build the graph structure. Since the number of required independence tests increases fast with the number of variables in our model, algorithms are necessary to handle the learning process.

One part of this thesis consists of a theoretical treatment of structure learning in Bayesian Networks, and in another part, I have composed and implemented my own version of a structure learning algorithm, SLBN, which is available for download at `https://github.com/SiboBerggren/SLBN`. Furthermore, simulation studies were performed to test the capabilities of structure learning, and of the SLBN algorithm. Altogether, we will see that structure learning is a powerful tool that may be used for causal inference, and that the SLBN algorithm successfully handles the learning process.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: simon.berggren@gmail.com. Supervisor: Michael Höhle.

Contents

Chapter 1

Introduction

I would rather discover one causal law, than be king of Persia. This statement by Democritus 400 BC demonstrates that answers to queries about cause and effect relationships has always appealed to humanity. In this text, structure learning in Bayesian Networks is presented, as a tool to infer causal associations. Such analysis is referred to as *Causal inference*.

Causal inference, based on data from experimental settings, is desired in many fields such as medicine and economics. The following made up example will be used throughout the text to exemplify the different concepts.

1.1 Drug test example

Consider a new drug attempting to provide remedy for Ulcerative colitis (UC), an inflammatory bowel decease with symptoms appearing in breakouts. Suppose that the drug has a number of side effects, namely that some people may experience a decreased appetite, and that some may have an increased level of a specific intestinal bacteria (IB). In a possible setting for testing the drug, we have a random sample of a number of individuals, supposed to represent the population of people suffering from UC. Half of them are given the drug, and the other half is given placebo (i.e. a sugar pill), without being aware of what substance they actually got. After some time, we measure each individuals C-reactive protein level (CRP), an indicator of inflammation. We also observe whether each individual has had a breakout or not during the time of the experiment, if they have experienced decreased appetite, and if their level of IB has increased above a specific level. From this information, we attempt to determine how the factors are related. If this investigation indicates an association between taking the drug and absence of a breakout, it does not necessarily imply that the drug is a direct cause. The absence of a breakout may be caused by increased IB level or reduced food in-

take, resulting from decreased appetite. Such knowledge about the relationships between the various factors, would be crucial in the future treatment of UC.

The relationships between the observed factors in this setting may be represented by a *Bayesian Network* (BN). The factors are represented by random variables (RV) and pictured by nodes in a graph, where an edge and its orientation represents association and its nature. In the example, we have binary categorical RVs "drug" (D), "increased IB" (IB), "decreased appetite" (DA), "hi CRP level" (CRP), and "breakout" (B). The BN in figure 1.1a represents that D affects DA, IB and CRP, and that CRP affects B. The second BN in figure 1.1b represents that D affects IB and DA, that those two affects CRP, which in its turn affects B. It also states that D has no direct impact on B, but affects it only through other factors. If the latter was the case, we may consider another treatment, also increasing the IB level, and suggest reduced food intake. BNs and their properties are explained in chapter 2.

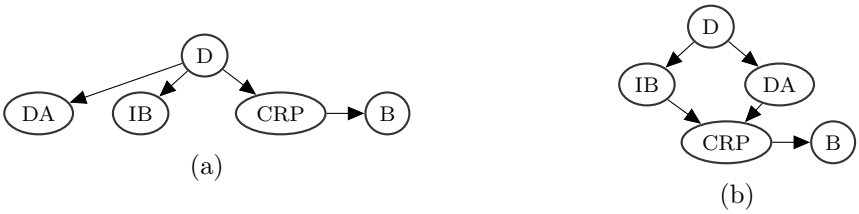


Figure 1.1: Two Bayesian Networks representing different relationships among the factors in a drug test setting.

Structure learning in Bayesian Networks is concerned with obtaining the structure of the graph, given data, i.e. to find out how the RVs relate to each other. In the example we could use it in an attempt to obtain the graphs in figure 1.1 from data. There are different approaches to this, commonly divided into two families, the *score-based* and the *constraint-based*. In the Score-based approach, how good different structures fits data are evaluated, and the best scored structure is chosen. In the constraint-based, conditional independence tests successively builds up the desired structure. Since the number of tests increases fast with the number of RVs in the model, this cannot (apart from small settings) be done by hand, and algorithms that automatically handle the learning process are necessary. In this text, focus is on the constraint-based approach, which is explained in chapter 3.

As a part of the thesis, I have composed and implemented my own version of a structure learning algorithm, SLBN, which takes advantage of ideas from the constraint-based approach. SLBN is presented in chapter 4, and is available for download at github [13]. Results from experiments, where SLBN is used to learn the structure from a number of data-sets, are presented in chapter 5.

Before going into structure learning in Bayesian Networks, some basic theory has to be covered. In the following sections, the idea of causal inference, the most important concepts of probability theory, and a description of what data we are concerned with, are presented.

1.2 Causal Inference

A statement about causality specifies the direction of an association, and is therefore stronger than one about a dependence. For RVs A and B, "A causes B" is interpreted as "A is a cause of B and B is an effect of A, but not the other way around", i.e. the outcome of A will affect B's probability distribution, but the outcome of B does not affect A's.

Causal inference is an analysis concerned with queries about causal associations among RVs in a model, i.e. based on data we attempt to determine how a RV is affected by changes in another. In e.g. a multiple linear regression-model, a prediction is based on how the "regression-plane" corresponds to changes in all the explanatory variables, and if those covariates, the actual cause of a change in the response variable remains unknown[1]. Causal inference attempts to distinguish covariation from causation. In the drug test example of section 1.1, it may give the answer to if absence of a breakout is actually caused by taking the drug, in addition to if those are associated.

1.3 Probability theory

Conditional independences among RVs forms the basis in constraint-based structure learning, this section will introduce some important concepts of independence.

Let A and B be two RVs. In general, the correspondence between the joint and the conditional probabilities are

$$P(A, B) = P(A|B)P(B). \tag{1.1}$$

A and B is said to be independent if they do not affect each other, and we write $(A \perp\!\!\!\perp B)$. If they are dependent, we write $(A \not\perp\!\!\!\perp B)$. Formally, A and B are independent if and only if

$$P(A, B) = P(A)P(B) \quad \text{or equivalent} \quad P(A|B) = P(A). \tag{1.2}$$

Conditional independence of A and B given a set of RVs $\mathbf{C}$, disjoint of $\{A, B\}$, states that A does not provide any information about B, given information about

[1] A "causal" alternative to prediction in a multiple linear regression-model is presented in section 2.4.

$\mathbf{C}$, and we write $(A \perp\!\!\!\perp B|\mathbf{C})$. This holds if and only if

$$P(A|B,\mathbf{C}) = P(A|\mathbf{C}) \quad \text{or equivalent} \quad P(A,B|\mathbf{C}) = P(A|\mathbf{C})P(B|\mathbf{C}). \tag{1.3}$$

We will refer to a specific outcome $\mathbf{c}$, of the RVs in $\mathbf{C}$ as a *configuration* of $\mathbf{C}$.

The law of the total probability specifies that $\sum_{a\in\mathcal{A}} P(A=a) = 1$, where $\mathcal{A}$ is A's sample space, and we have

$$P(A) = \sum_{\mathbf{c}\in\mathcal{C}} P(A, \mathbf{C}=\mathbf{c}), \tag{1.4}$$

where $\mathcal{C}$ denotes the set of configurations of $\mathbf{C}$.

Consider a joint probability distribution P, over a set of RVs $\mathbf{V} = (V_1, V_2, \ldots, V_n)$. Repeated use of (1.1) gives

$$\begin{aligned} P(V_1, V_2, \ldots, V_n) &= P(V_n|V_1, \ldots, V_{n-1})P(V_1, \ldots, V_{n-1}) = \ldots \\ &= P(V_n|V_1, \ldots, V_{n-1})P(V_{n-1}|P(V_1, \ldots, V_{n-2}) \ldots P(V_1). \end{aligned} \tag{1.5}$$

1.4 Data and Estimation

When learning the structure of a Bayesian Network from data, we are interested in how the factors affect each other, or specifically how the RVs in our models are related. Data will therefore consists of a number of observations, where each observation consist of a realization of all the fators we want to model. Hence, in a setup containing a set of k RVs, $\mathbf{V} = (V_1, V_2, \ldots, V_k)$, we have, in a sample of size n, data

$$\mathbf{D} = \{\mathbf{d}^1, \mathbf{d}^2, \ldots, \mathbf{d}^n\}, \text{ where } \mathbf{d}^j = (d_1^j, d_2^j, \ldots, d_k^j),\ j = 1, 2, \ldots, n,$$

and where d_i^j denotes the outcome of $V_i, i = 1, 2, \ldots, k$, in the jth observation. The sample space of each RV corresponds to the possible outcomes of the factor it represents. As an example, a sample of size n from the drug test setting in section 1.1 will consists of observations on D, IB, DA, CRP and B from n individuals, and the sample space of each RV will consist of two values (typically 1 and 0 for "yes" and "no" respectively). Note that we in this text assume that there are no missing observations in data.

Constraint-based structure learning applies to ordinal data (continuous or discrete values), as well as nominal data (categorical levels without a meaningful ordering). In the case of nominal data, each configuration of the RVs in our model has a specific probability to occur, and we assume that the RVs follows a joint multinomial distribution. We will estimate those probabilities, i.e. the parameters in the multinomial distribution, from the proportions in data. Specifically, given

a model containing a set of k RVs $\mathbf{V} = (V_1, V_2, \ldots, V_k)$, and a sample $\mathbf{D}$ of size n (without missing values of these RVs). Let $r_{\mathbf{V}}$ denote the number of possible configurations of all RVs in $\mathbf{V}$, this is typically given by the Cartesian product of the sizes of each RVs sample space, i.e. $r_{\mathbf{V}} = \prod_{i=1}^{k} r_{V_i}$, where r_{V_i} is the size of V_is sample space. Let furthermore n_i denote the number of observations in $\mathbf{D}$ that match the ith configuration. Then the maximum likelihood (ML) estimator $\hat{\pi}_i$, of the probability of the ith configuration $\pi_i, i = 1, 2, \ldots, r_{\mathbf{V}}$, is given by n_i/n, where

$$\sum_{i=1}^{r_{\mathbf{V}}} n_i = n \quad \text{and} \quad \sum_{i=1}^{r_{\mathbf{V}}} \pi_i = 1.$$

Given data from the drug test example in section 1.1, we will be interested in estimating

$$\pi_{i,j,k,l,m} = P(D = i, IB = j, DA = k, CRP = l, B = m), \; i, j, k, l, m = 0, 1, \quad (1.6)$$

i.e. the probability of observing the RVs at the specified levels in one individual. For each of the $r_{\mathbf{V}} = 2^5 = 32$ possible configurations of the RVs, we will estimate these probabilities with $\hat{\pi}_{i,j,k,l,m} = n_{i,j,k,l,m}/n$, where $n_{i,j,k,l,m}$ denotes the number of observations that match this specific configuration. For the sample of size $n = 500$, presented in the contingency table 1.1, we have e.g. $\hat{\pi}_{0,0,0,1,1} = 127/500 = 0.256$, estimating $P(D = 0, IB = 0, DA = 0, CRP = 1, B = 1)$.

		B	0		1	
		CRP	0	1	0	1
D	IB	DA				
0	0	0	12	7	5	127
		1	15	0	8	13
	1	0	44	0	10	12
		1	6	0	1	2
1	0	0	1	0	0	5
		1	12	1	2	7
	1	0	17	0	13	7
		1	124	0	26	23

Table 1.1: Contingency table showing the outcomes of the five RVs in the drug test setting of section 1.1, for a sample of size 500.
(Data is simulated according to the distribution specified by the Bayesian Network in figure 2.2)

The size of a model refers to the number of parameters required to specify the distribution, i.e. the size is $r_{\mathbf{V}} - 1 = \prod_{i=1}^{k} r_{V_i} - 1$ in a model with k RVs. For each of the r_{V_1} possible values of V_1, V_2 assumes one of r_{V_2} values, V_3 one of r_{V_3}, and so on, and the fact that the probabilities sums to one restricts the number of parameters by one. In the drug test example, with 5 binary RVs, the joint multinomial distribution is specified by $2^5 - 1 = 31$ parameters, i.e. the size of the model is 31. In chapter 2 we will see that a BN representation of a model, based on conditional independences, may reduce its size, and in a setup with a large number

of RVs, this reduction may be crucial, e.g. in order to estimate probabilities such as the one in (1.6).

Conditional independences among the RVs, forms the basis in constraint-based structure learning. The asymptotically χ^2 distributed Pearson's χ^2 test for independence, is used to infer about such independences from data, and is explained in the appendix 7.1.

When the aim with structure learning is a complete causal analysis, we have to assume that the observed RVs describes the phenomenon of interest sufficiently good. Specifically, we assume that no unobserved factors affects two or more of the observed ones. Such "unobserved RVs" are referred to as *latent* variables or *confounders* (Pearl, 2000).

As mentioned, the distributions will be represented by Bayesian Networks, now we are ready to explore those in the next chapter.

Chapter 2

Bayesian Networks

A Bayesian Network is a graphical representation of a probability distribution, which provides a clear representation of how the RVs are related. BNs are suitable for causal inference since the graph structure specifies a factorization of the distribution, as will be explained in section 2.3. Constraint-based structure learning takes advantage of this factorization. Conditional independences inferred from data specifies a factorization of the distribution, and is used to obtain a graph. Relationships read from this graph then tells us something about the distribution via this factorization. In this chapter, BNs and their properties will be explained, and in chapter 3, we will see how we can use independences inferred from data to obtain the graph. The relevant graph terminology is introduced in the following section.

2.1 Graphs

Our graphs consist of *nodes*, representing RVs, and *edges*. A *directed* edge and its orientation represents a causal association between the corresponding RVs, and undirected edges represents unspecified associations. Connected nodes are said to be *adjacent*, and in a *complete* graph, each pair of nodes are adjacent. A *path* is series of edges and the nodes they connect, such that no node is revisited. If all edges in a path are directed, and follows the same orientation, we call it a *directed path*. A *cycle* is a directed path, starting and finishing at the same node. In figure 2.1, A−C−E−D is a undirected path from A to D and B→D→E→F is a directed path from B to F. Changing the orientation of the edge D→E to E→D would introduce the cycle B→C→E→D→B. In BNs, cycles are not allowed, but only *directed acyclic graphs* (DAG), as the one in figure 2.1. Acyclic graphs with both directed and undirected edges is referred to as a *PDAG*s, partially directed acyclic graphs. The *skeleton* of a DAG (or a PDAG) refers to the corresponding

undirected graph. In figure 2.1, A and B are the *parents* of C, E is the only parent of F, $\{A, B, C, D\}$ are the *predecessors* of E, and A has no predecessors. C and D are the *children* of B, and E and F are the *descendants* of D. The *neighborhood* of a node consist of its parents and children, e.g. $\{A, B, E\}$ are the *neighbors* of C.

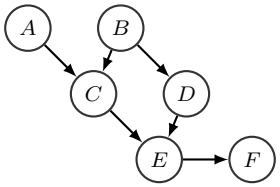


Figure 2.1: Example of a DAG.

2.2 Bayesian Networks

The term "Bayesian Network" was coined by Pearl (Pearl,1985), and the name stems from a parallel to Bayesian statistics, we update our prior knowledge by observed data. Among several definitions of a BN, the following is used in this text.

Definition 1 *(Koller,Friedman,2009)* ***Bayesian Network***
A Bayesian Network is a graphical representation of a probability distribution over a set of RVs. It consists of a DAG in which each node corresponds to a RV, and a set of conditional probability tables, one for each RV.

The intended conditional probability tables are determined by the joint distribution. To see this, consider a model with a set of RVs $\mathbf{V} = (V_1, \ldots, V_k)$. For a specific outcome $V_i = v_i$ and a configuration $\mathbf{u}$, of the RVs in $\mathbf{U} \subseteq \mathbf{V} \backslash V_i$, the identities (1.1) and (1.4) gives,

$$P(v_i|\mathbf{u}) = \frac{P(v_i, \mathbf{u})}{P(\mathbf{u})} = \frac{\sum_{\mathbf{s}\in\mathcal{S}} P(v_i, \mathbf{u}, \mathbf{s})}{\sum_{\mathbf{s}\in\mathcal{S}, v\in\mathcal{V}_i} P(\mathbf{u}, \mathbf{s}, v)} \tag{2.1}$$

where $\mathbf{s}$ is a configuration of $\mathbf{S} = \mathbf{V} \backslash \{\mathbf{U}, V_i\}$, and $\mathcal{S}$ and $\mathcal{V}_i$ are the sample spaces of $\mathbf{S}$ and V_i respectively. Since $V_i \cup \mathbf{U} \cup \mathbf{S} = \mathbf{V}$, the desired conclusion follows.

2.3 Markovian factorization

In this section, the factorization mentioned in the introduction to this chapter, will be formalized. Given an ordering of a set of RVs, the Markovian parents of a RV A is the minimal subset of A's predecessors, according to that ordering, such that A is independent of all its other predecessors given this subset.

Definition 2 *(Koller,Friedman,2009)* ***Markovian parents***
Let $\mathbf{V} = (V_1, \ldots, V_k)$ *be an ordered set of RVs, distributed according to* P*. A set of variables* $PA(V_i) \subseteq \{V_1, \ldots, V_{i-1}\}$ *is said to be the Markovian parents of* V_i *if*

$$P(V_i|PA(V_i)) = P(V_i|V_{i-1}, V_{i-2}, \ldots, V_1) \tag{2.2}$$

and no proper subset of $PA(V_i)$ *satisfies (2.2).*

According to the identity (1.5), the joint probabilities can be written,

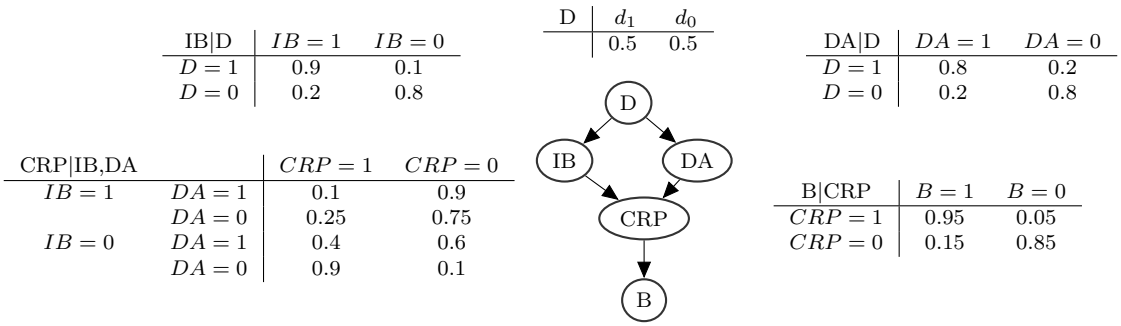
$$P(V_1, V_2, \ldots, V_k) = \prod_{i=1}^{k} P(V_i|\text{PA}(V_i)) \tag{2.3}$$

In a BN, the oriented edges induces a natural ordering of the RVs, and the Markovian parents of a RV is just its parents. Due to this observation and (2.3), the size, i.e. number of parameters required to describe the distribution, may be reduced in a BN. Let G be the graph of a BN with RVs $\mathbf{V} = (V_1, \ldots, V_k)$, distributed according to P_G, where P_G factors according to G. Let furthermore r_i denote the cardinality of V_i's sample space, and $q_i = \prod_{\{i:V_j \in \text{PA}(V_i)\}} r_i$, be the number of configurations of the RVs in $\text{PA}(V_i)$ for $i = 1, 2, \ldots, k$. Then

$$\text{size}(G) = \sum_{i=1}^{k} q_i(r_i - 1) \tag{2.4}$$

is the number of parameters in the joint multinomial distribution, given that it factors according to G. Compared to the expression for the size $\prod_{i=1}^{k} r_i - 1$, as computed in section 1.4, (2.4) is always smaller for a DAG. Consider the drug test example in section 1.1, and the BN in figure 1.1b. Due to the independences implied by the graph,
$P(D, IB, DA, CRP, B) = P(D)P(IB|D)P(DA|D)P(CRP|IB, DA)P(B|CRP)$,
and 11 parameters will specify the distribution, instead of the 31 found in section 1.4. The distribution is therefore determined by the probability tables in figure 2.2, which illustrates how a BN typically is specified (the probabilities are made up).

If A→C→B is the graph of a BN, $P(B|C, A) = P(B|\text{PA}(B)) = P(B|C)$, so $(B \perp\!\!\!\perp A|C)$ holds, i.e. A and B is independent given C. When learning the structure of a BN, we use such conditional independences, inferred from data, to obtain the graph, and this graph to infer about causal associations. How this is done, and under what circumstances the relationships in the obtained graph actually corresponds relationships in the distribution, is explained in chapter 3. In the next section, an important application of BNs is exemplified, and in section 2.5, a description of how we can simulate data from a BN is explained.



IB|D	$IB=1$	$IB=0$
$D=1$	0.9	0.1
$D=0$	0.2	0.8

D	d_1	d_0
	0.5	0.5

DA|D	$DA=1$	$DA=0$
$D=1$	0.8	0.2
$D=0$	0.2	0.8

CRP|IB,DA		$CRP=1$	$CRP=0$
$IB=1$	$DA=1$	0.1	0.9
	$DA=0$	0.25	0.75
$IB=0$	$DA=1$	0.4	0.6
	$DA=0$	0.9	0.1

B|CRP	$B=1$	$B=0$
$CRP=1$	0.95	0.05
$CRP=0$	0.15	0.85

Figure 2.2: Bayesian Network, represented as a DAG and a set of probability tables, specifying the distribution in the drug test example of section 1.1.

2.4 Propagation in Bayesian Networks

We have seen that the BN representation of a distribution provides a clear representation of conditional independences among the RVs, and the probabilities of a specific RV's outcomes may be easily accessed, since those depends only on a subset of all RVs in the distribution, namely the Markovian parents. As an example, in the BN of figure 2.2, the probabilities of B's different outcomes depends only on the outcome of PA(B)=CRP, and is available in the probability table for B|CRP, i.e. we do not have to account for the outcomes of D, IB and DA to specify those probabilities, if we know the outcome of CRP.

As motivated above, the BN representation of a distribution is useful when we wish to predict the outcome of a specific RV, given information about the remaining RVs. Such prediction is sometimes referred to as propagation, and is an important application of BNs. Specifically, let $\mathbf{V}$ be the set of RVs, and suppose that $V_I \in \mathbf{V}$ is of specific interest, i.e. we are interested in probability vector $\mathbf{P}(V_I|\mathbf{V}\backslash V_I = \mathbf{v})$, where $\mathbf{v}$ is a realization of the RVs in $\mathbf{V}\backslash V_I$. This expression may be laborious to derive from the joint probability distribution if the factorization of 2.3 is unknown, but in a BN the graph specifies this factorization, and $\mathbf{P}(V_I|\mathbf{V}\backslash V_I = \mathbf{v}) = \mathbf{P}(V_I|\mathrm{PA}(V_I) = \mathbf{u})$, where $\mathbf{u} = \{v_i \in \mathbf{v} : V_i \in \mathrm{PA}(V_I)\}$ is the outcomes of the variables in $\mathrm{PA}(\mathrm{V}_I)$ that match $\mathbf{v}$. As an example, consider a bank, concerned with whether or not to approve a loan to a new customer, and suppose that the bank collects data from all its customers. Specifically, the bank has data on a number of RVs $\mathbf{V}_E = (V_{E1}, V_{E2}, \ldots, V_{Ek})$, representing different factors such as income and age, that may affect a customers ability to pay back a loan, from each customer in the bank[1]. The bank has also repayment

[1]It is convenient to use ordinal RVs when exemplifying $\mathbf{V}_E$, even though it contravenes this texts restriction to nominal categorical data. However, the theory naturally extends to ordinal data, as will be discussed in chapter 6.

history for each customer, i.e. in a simple setup, the bank has data on the RV V_I, with two possible outcomes, "customer did pay back loan" and "customer did not pay back loan". Suppose furthermore that we can represent the distribution of $(V_{E1}, V_{E2}, \ldots, V_{Ek}, V_I)$ with a BN. We are interested in the probability that a new customer that requests a loan, will be able to pay it back, given information about the factors represented by $\mathbf{V}_E$ for this costumer, i.e. the probability $P(V_I|\mathbf{V}_E = \mathbf{v})$, where $\mathbf{v}$ is a realization of the RVs in $\mathbf{V}_E$. In a BN, the probability table for $\mathrm{V}_I|\mathrm{PA}(\mathrm{V}_I)$ contains this information. Structure learning may be used to infer this BN from data consisting of observations on all the customers in the bank, i.e. we estimate the probability distribution through structure learning, and use this estimation for propagation. This exemplifies an important application of structure learning. How the BN can be inferred from data is explained in chapter 3.

2.5 Simulation

In this section, we will see how data may be generated, according to a distribution specified by a BN. The described technique is used for the simulations in chapter 5. For convenience, the set-builder notation will sometimes be used even though an ordered vector is intended.

Given a BN with k RVs $\mathbf{V} = (V_1, V_2, \ldots, V_k)$, one sample $\mathbf{d} = (d_1, d_2, \ldots, d_k)$ from this BN is generated as follows. Let $\mathbf{V}^{(1)} \subseteq \mathbf{V}$ denote the set all parentless RVs in $\mathbf{V}$, then for each RV $V_i \in \mathbf{V}^{(1)}$, one realization d_i of V_i is generated from a multinomial distribution. The parameters in this distribution are specified by the corresponding probability table in the BN, i.e. for each $V_i \in \mathbf{V}^{(1)}$, we have a vector of probabilities $\mathbf{P}(V_i)$, and generate d_i from a $\mathrm{MN}(1, \mathbf{P}(V_i))$ distribution. After having realized all RVs in $\mathbf{V}^{(1)}$, we will have a vector $\mathbf{r}^{(1)} = \{d_i : V_i \in \mathbf{V}^{(1)}\}$, consisting of the realizations of the RVs in $\mathbf{V}^{(1)}$. Next, we realize each RVs that has all of its parents in $\mathbf{V}^{(1)}$, let $\mathbf{V}^{(2)} = \{V_j : \mathrm{PA}(V_j) \subseteq \mathbf{V}^{(1)}\}$ denote this set, i.e. for each $V_j \in \mathbf{V}^{(2)}$, we generate d_j according to a $\mathrm{MN}\big(1, \mathbf{P}(V_j|\mathbf{V}^{(1)} = \mathbf{r}^{(1)})\big)$ distribution, where the probability vector $\mathbf{P}(V_j|\mathbf{V}^{(1)} = \mathbf{r}^{(1)})$ is specified in the row of the corresponding probability table that match the realizations in $\mathbf{r}^{(1)}$. When all RVs in $\mathbf{V}^{(2)}$ is realized, we will have the vector of realizations, $\mathbf{r}^{(2)} = \{d_j : V_j \in \mathbf{V}^{(2)}\}$. Next, we realize each RV that has all of its parents in $\mathbf{V}^{(1)} \cup \mathbf{V}^{(2)}$, in the same way as described above, and the procedure continues until all the RVs in $\mathbf{V} = \bigcup_l \mathbf{V}^{(l)}$ is realized. Note that, due to the DAG structure, the sets $\mathbf{V}^{(l)}$ are all disjoint.

We are typically interested in simulating a data-set consisting of $n > 1$ samples, i.e. n independent realizations of the RVs in the BN. One could therefore repeat the described procedure n times, but it is more efficient to generate n realizations of the

RVs in $\mathbf{V}^{(i)}$, before proceeding to the RVs in $\mathbf{V}^{(i+1)}$. Suppose that $\bigcup_{l=1}^{M} \mathbf{V}^{(l)} = \mathbf{V}$, then algorithm 1 will produce a data-set of sample size n from a BN. In algorithm 1, d_i^j denotes the jth realization of $V_i \in \mathbf{V}$, $\mathbf{d}^j$ denotes the jth realization of the RVs in $\mathbf{V}$, and $\mathbf{r}^{(m),j} = \{d_i^j : V_i \in \mathbf{V}^{(m)}\}$ denotes the jth realization of the RVs in $\mathbf{V}^{(m)}$ for $m = 1, 2, \ldots, M$. The probability vectors are specified in the corresponding probability table of the BN.

foreach $V_i \in \mathbf{V}^{(1)}$ **do**
  generate $\{d_i^1, d_i^2, \ldots, d_i^n\}$ from $\mathrm{MN}(n, \mathbf{P}(V_i))$
end
$\mathbf{r}^{(1),j} = \{d_i^j : V_i \in \mathbf{V}^{(1)}\}$
for $m = 2, 3, \ldots, M$ **do**
  foreach $V_i \in \mathbf{V}^{(m)}$ **do**
    for $j = 1, 2, \ldots, n$ **do**
      generate d_i^j from $\mathrm{MN}\big(1, \mathbf{P}(V_i | \mathbf{V}^{(m-1)} = \mathbf{r}^{(m-1),j})\big)$
    end
  end
  $\mathbf{r}^{(m),j} = \{d_i^j : V_i \in \mathbf{V}^{(m)}\}$
end
$\mathbf{d}^j = (d_1^j, d_2^j, \ldots, d_m^j)$
$\mathbf{d} = \{\mathbf{d}^1, \mathbf{d}^2, \ldots, \mathbf{d}^n\}$

Algorithm 1: Simulation of a BN.

The simulations in chapter 5 is done in R. When realizing multinomial distributed RVs, the *sample*-function is used.

Chapter 3

Constraint-based structure learning

The idea behind constraint-based structure learning is to obtain a graph from conditional independences inferred from data, and to use this graph to infer about causal associations in the distribution. If relationships in the graph does not correspond to ditto in the distribution, conclusions from reading the graph would not be justified, and the idea of causal inference through structure learning would be useless. To justify such analysis, we need to ensure that the graph only entails relationships that actually corresponds to relationships in the underlying distribution. Before explaining how such a graph is obtained in sections 3.1 through 3.4, a general overview of constraint-based structure learning follows. Figure 3.1 and the following schematic description intends to guide the reader through this chapter.

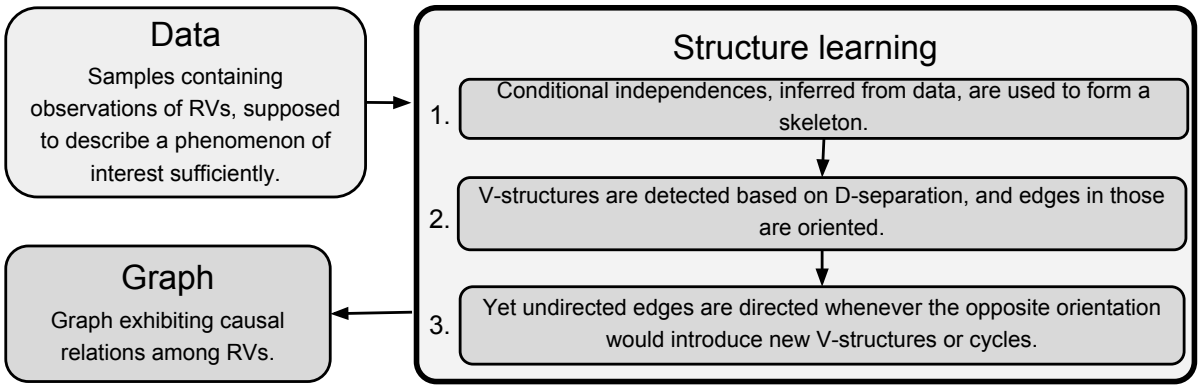


Figure 3.1: Constraint based structure learning.

1. Starting from a graph consisting of nonadjacent nodes, corresponding to the RVs in our model, a skeleton is formed based on conditional independences inferred from data. Two nodes are connected if and only if there are no subset of

the remaining RVs that renders their corresponding RVs independent.
2. Based on the connections in the skeleton, and the rules of D-separation, a specific kind of node triples, called V-structures are detected, and the edges in those are oriented. V-structures, and what makes them important, is explained in section 3.2, and D-separation is explained in section 3.3.
3. Finally, yet undirected edges are oriented, such that the graph remains acyclic and now further V-structures are introduced. Through this, we obtain a PDAG which exhibits causal associations. Under assumptions, specified in sec 4.1, those relationships will be valid for the underlying probability distribution.

3.1 Correspondence in Bayesian Networks

The theory in the following sections serves to ensure the correspondence between relationships in the graph we obtain from data, and relationships in the distribution that data is supposed to follow. D-separation, defined in section 3.3, provides a convenient way to identify such inferred independences that can be used to obtain a graph that satisfies this. To motivate the ideas behind D-separation, we will distinguish independence in a graph from independence in a distribution, i.e. initially, independence between two nodes will not be equivalent to independence between the RVs they represent. Notations-wise it will be distinguished by subscripts P and G, for independences in distributions and in graphs respectively. The following definition refers to the factorization introduced in section 2.3.

Definition 3 *(Pearl,2000)* ***Markov Compatibility***
If a probability distribution P admits the factorization of (2.3), relative to the ordering induced by a DAG G, then G and P are said to Markov-compatible.

Theorem 1 *(Pearl,2000)*
For any three disjoint sets of nodes, $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in a DAG G the following implications holds:

- $(\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_G \Rightarrow (\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_P$ *for every distribution P, Markov-compatible with G.*
- $(\mathbf{A} \not\perp\!\!\!\perp \mathbf{B}|\mathbf{C})_G \Rightarrow (\mathbf{A} \not\perp\!\!\!\perp \mathbf{B}|\mathbf{C})_P$ *for at least one distribution P, Markov-compatible with G.*

The following direct consequence is instructive, in order to formulate the desired correspondence.

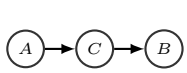
For any three disjoint sets of nodes $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in a DAG G, and for all distributions P the following holds:

- $(\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_G \Rightarrow (\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_P$ whenever G and P are Markov-compatible.
- if $(\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_P$ holds in every P that is Markov-compatible with G, then $(\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_G$.[1]

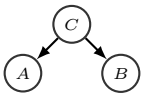
If there were only one DAG being Markov-compatible with P, independences among RVs distributed according to P would certainly hold in this. Unfortunately, this is not the case in general. However, there are certain structures, such that if those are not present in a DAG, this will not be Markov-compatible with a distribution satisfying the corresponding relationships, i.e. such structures has to be present in every DAG being Markov compatible with P. If considering graphs entailing only such structures, independence in graphs and distributions would be equivalent, and due to mutual exclusivity of independence and dependence, this equivalence would also hold for dependences. Towards to identify those structures, we will examine the *fundamental connections*.

3.2 Equivalent structures

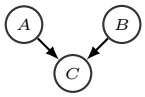
Consider a substructure of a graph, consisting of three nodes, where two of them are adjacent with the third, but not connected to each other. There are three possible orientations of the edges in such a triple if we consider the nodes as interchangeable. Those are the *serial connection*, the *diverging connection* and the *V-structure*, referred to as the *fundamental connections* (Scutari,Nagarajan,Lébre,2013). The serial connection consists of the nodes and edges in a directed path, in the diverging connection one of the nodes has outgoing edges pointing towards the other two, and in the V-structure two of the nodes have edges converging to the third, the *collider* (see figure 3.2).



(a) Serial connection



(b) Diverging connection



(c) V-structure

Figure 3.2: The fundamental connections.

Decomposing the probabilities of the corresponding RVs according to the identity (2.2), and the ordering implied by the graphs gives

$$P(A, B, C) = P(A|\mathrm{PA}(A))P(B|\mathrm{PA}(B))P(C|\mathrm{PA}(C)) = P(A)P(C|A)P(B|C), \tag{3.1}$$

[1]The second statement, that a graph G embodies all independences in a probability distribution P, is with the additional assumption that P is stable (see section 3.4) also referred to as G being an *I-map* of the set of independences in P (Koller,Friedman, 2009).

$$P(A,B,C) = P(A|\text{PA}(A))P(B|\text{PA}(B))P(C|\text{PA}(C)) = P(A|C)P(B|C)P(C), \tag{3.2}$$

$$P(A,B,C) = P(A|\text{PA}(A))P(B|\text{PA}(B))P(C|P(C)) = P(A)P(C|A,B)P(B), \tag{3.3}$$

for the serial connection, the diverging connection, and the V-structure respectively. For (3.2), the definition of conditional independence (1.1) gives

$$P(A|C)P(B|C)P(C) = \frac{P(A,C)}{P(C)}\frac{P(B,C)}{P(C)}P(C) = P(C|A)P(A)P(B|C), \tag{3.4}$$

which equals (3.1). Hence, based on the outcomes of conditional independence tests exclusively, we will not be able to distinguish the diverging connection from the serial connection. However, the expression for the V-structure (3.3) differs from the others, and we can uniquely determine a class of PDAGs, defined by its skeleton and set of V-structures. Such a class satisfies the transitive, symmetric and reflexive properties and is referred to as an equivalence class.

Definition 4 *(Pearl,2000)* ***Equivalent structures***
Two PDAGs are equivalent if and only if they have the same skeletons and the same set of V-structures.

The uniqueness of the skeleton follows from the fact that two nodes are adjacent if and only if **no** subset of all RVs renders their corresponding RVs independent, and that independence when conditioned on only **one** such subset is sufficient to separate them. Therefore the conditional independence tests will always identify whether two nodes are neighbors or not.

Consider the PDAGs in figure 3.3. Changing the orientation of the edge connecting A and B will not change the set of V-structures, therefore the first and the second PDAG belongs to the same equivalence class, defined by the third PDAG. Redirecting the edge connecting D and E, as in the fourth and fifth PDAG, will result in PDAGs from different equivalence classes, the first with V-structures A→C←B and C→E←D, and the second with A→C←B and C→D←E.

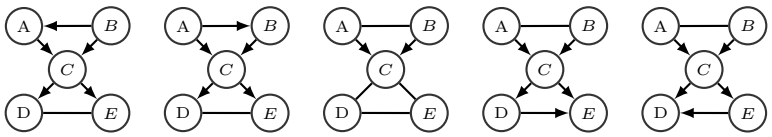


Figure 3.3: PDAGs illustrating equivalence classes. The first, second and third graph belongs to the same equivalence class, but the forth and fifth belongs to different.

3.3 D-separation

D-separation is used to orient edges in constraint-based structure learning, and provides a convenient way to identify such directed substructures of a graph, that is uniquely determined by conditional independences inferred from data.

Consider three nodes A,B and C in a graph G, and the fundamental connections. Intuitively, in the serial connection A→C→B, and in the diverging connection A←C→B, A and B are marginally dependent, but independent when conditioned on C, i.e. $(A \not\perp\!\!\!\perp B)_G$, but $(A \perp\!\!\!\perp B|C)_G$. This extends to larger graphs, in a directed path traversing a number of nodes,

$$A \to C_1 \to \cdots \to C_n \to B,$$

or two directed paths, starting from a node $C_i \in \{C_1, C_2, \ldots, C_n\}$,

$$A \leftarrow C_1 \leftarrow \cdots \leftarrow C_i \to \cdots \to C_n \to B,$$

$(A \not\perp\!\!\!\perp B)_G$, but $(A \perp\!\!\!\perp B|C_j)_G$ holds for $j = 1, 2 \ldots, n$. For the V-structure A→C←B, we will instead introduce a dependence by conditioning, i.e. $(A \perp\!\!\!\perp B)_G$ but $(A \not\perp\!\!\!\perp B|\mathbf{U})_G$ holds for any subset **U** of C and its descendants. D-separation is a formalization of this ideas.

Definition 5 *(Pearl,1988)* ***D-separation***
Let **A**, **B** *and* **C** *be three disjoint sets of nodes in a DAG G. A path p is said to be D-separated, or blocked, by* **C** *if and only if*

- *there is a serial or a diverging connection along p such that the middle node is in* **C**, *or*
- *there is a V-structure along p, centered at a node Z, s.t. neither Z or any of its descendants are in* **C**.

If **C** *blocks every path between nodes in* **A** *and* **B**, *then* **C** *D-separates* **A** *and* **B**.

Note that **C** D-separates **A** and **B** if and only if $(\mathbf{A} \perp\!\!\!\perp \mathbf{B}|\mathbf{C})_G$.

In the DAG in figure 3.4, B D-separates C from D but $\{B, E\}$ does not since the path C→E→D is open. E D-separates A from F, but C does not, since C will open the path A→C→B→D→E→F. A and D are D-separated by $\emptyset$, but not by F, since F opens the paths A→C→B→D and A→C→E→D.

In the following, graph-independence and independence among RVs will be distinguished only if the context motivates it.

To be able to determine the equivalence class uniquely, i.e. to ensure the desired correspondence between relationships in the graph, obtained by constraint-based structure learning, and relationships in the underlying distribution, we also have to assume that the distribution is stable. Stability is explained in the following section.

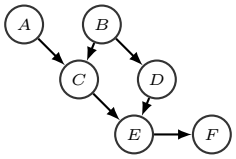


Figure 3.4: DAG used in examples of D-separation and Markov blankets.

3.4 Stability

A distribution is said to be stable[2] if its set of independences are invariant under different parameterizations (Pearl,2000). The following example of an unstable distribution illustrates the concept.

Consider a distribution with three binary categorical RVs A,B and C, where A and B is independent and Bernoulli distributed with

$$P(A=1)=p_A,\ P(A=0)=1-p_A,\ P(B=1)=p_B \text{ and } P(B=0)=1-p_B,$$

and where C equals 1 if A=B and 0 otherwise. By this definition of the probability distribution, $(A \not\perp\!\!\!\perp C)$ and $(B \not\perp\!\!\!\perp C)$ holds in general. However, with $p_A = p_B = 1/2$, $(A \perp\!\!\!\perp C)$ and $(B \perp\!\!\!\perp C)$ holds. To see this, note that given that $A = 0$, then $C = 0$ if and only if $B = 1$, and by similar arguments we get the following two equalities, from which the proclaimed independences follows:

$$\begin{aligned} P(C=0|A=0) = P(B=1) = p_B = 1/2 = 1-p_B = P(B=0) = P(C=0|A=1) \\ P(C=0|B=0) = P(A=1) = p_A = 1/2 = 1-p_A = P(A=0) = P(C=0|B=1). \end{aligned} \tag{3.5}$$

This specific parametrization introduces extraneous independences, and the distribution is not stable. No consider the following inequalities,

$$\begin{aligned} P(C=0,A=0|B=0) = 0 \neq p_A = P(C=0,A=1|B=0), \\ P(C=0,B=0|A=0) = 0 \neq p_B = P(C=0,B=1|A=0), \\ P(A=0,B=0|C=0) = 0 \neq p_B p_A = P(B=1)P(A=1) \\ = P(A=0|C=0)P(B=0|C=0). \end{aligned} \tag{3.6}$$

Evidently, each pair of the RVs are marginally independent, but dependent when conditioned on the third. This illustrates why unstable distributions are problematic in constraint-based structure learning. The V-structure we expect to obtain is A→C←B, which exhibits the marginal independence of A and B, their dependence when conditioned on C, and the unaffected dependences of A and C, and of B and C. However, for $p_A = p_B = 1/2$, each of the three conditional dependencies in

[2]The concept of *faithfulness* is equivalent to stability (Koller,Friedman,2008).

(3.6) together with the pairwise independences in (3.5), satisfies the factorization linked to the V-structures (3.3), and the equivalence class cannot be determined. While Markov compatibility ensures the existence of an equivalence class, stability ensures its uniqueness.

Another concept used in most of the constraint-based learning algorithms (and in SLBN), is that of Markov blankets, which is explained in the following section.

3.5 Markov blankets

Markov blankets (MB) are used when building up the skeleton, and when detecting the V-structures in the constraint based learning algorithms[3]. The MB of a node is the minimal set of nodes that D-separates it from all others.

Definition 6 *(Pearl,2000)* ***Markov blanket***
Let $\mathbf{V}$ *be the set of nodes in a Bayesian Network. The Markov blanket of* $A \in \mathbf{V}$ *is a set* $\mathbf{MB}(A) \subseteq \mathbf{V} \backslash A$ *s.t.*

1. $(A \perp\!\!\!\perp B | \mathbf{MB}(A))_G$ *for all* $B \subseteq \mathbf{V} \backslash \{\mathbf{MB}(A), A\})$
2. *if a set* $\mathbf{H}$ *satisfies 1 (in the place of* $\mathbf{MB}(A)$*), then* $\mathbf{MB}(A) \subseteq \mathbf{H}$

The MB of a node consist of its parents, its children and the parents of its children, e.g. in figure 3.4, B is the parent of A's child C, so B $\in \mathbf{MB}(A)$, and $(A \not\perp\!\!\!\perp C | B)_G$ holds as expected. We have $\mathbf{MB}(A) = \{B, C\}$, $\mathbf{MB}(B) = \{A, C, D\}$, $\mathbf{MB}(C) = \{A, B, D, E\}$, $\mathbf{MB}(D) = \{B, C, E\}$, $\mathbf{MB}(E) = \{C, D, F\}$ and $\mathbf{MB}(F) = \{E\}$.

A goodness of fit measure for model selection is instructive to distinguish different graph structures. One such measure, the BIC-score, is explained in the following section.

3.6 Comparing structures

If there are several graph structures available, a goodness of fit measure indicating how well different structures match data is instructive. The more complex a graph structure is, the less explanatory power it has, and therefore such a metric is desired to also account for the complexity of the graph. The Bayesian information criterion (BIC), known from classical statistics, can be adopted to the BN context. Let G be a DAG representing a set of k RVs distributed according to P_G, where P_G factors according to G as described in section 2.3, and let $\mathbf{D} = \{\mathbf{d}^1, \mathbf{d}^2, \ldots, \mathbf{d}^n\}$ be a data-set, where $\mathbf{d}^i, i = 1, 2, ...n$ denotes the ith sample. Let furthermore $\hat{\pi}$ be

[3]The PC-algorithm does not use MBs, and is an exception (Scutari,Nagarajan,Lébre,2013).

the ML estimator of P_G's parameters based on $\mathbf{D}$ as described in section 1.4, i.e. $\hat{\pi}$ is the probabilities that matches data best, given P_Gs factorization. Then we define the BIC-score (Jensen,Nielsen,2007)

$$\mathrm{BIC}(G|\mathbf{D}) = \sum_{i=1}^{n} \log P(\mathbf{d}^i|\hat{\pi}, G) - \mathrm{size}(G)\frac{\log n}{2}.$$

This expression is such that higher values indicates better fit. The less factorized, according to equation (2.3), a joint distribution is, the higher the log-likelihood term will be, but the less explanatory power the graph structure will posses. Hence, a goodness of fit measure based exclusively on the log-likelihood term favors densely oriented graphs. The size term, defined in section 2.3, compensates for this fact by decreasing the BIC-score proportional to the number of parameters required to describe the distribution. The BIC-score is score equivalent, i.e. it cannot distinguish one graph from another in the same equivalence class.

Interpreting the BIC-score may be summarized as follows (Kass,Raftery,1995). Given a data-set, let BIC_1 and BIC_2 denote the BIC-scores of two different graph structures that we wish to compare, then

$\lvert\mathrm{BIC}_1 - \mathrm{BIC}_2\rvert$	evidence against the lowest scored structure
< 2	no difference worth mentioning
$2 - 6$	positive
$6 - 10$	strong
$10 <$	very strong

The BIC-score is used to compare different structures in chapter 5.

Now, constraint-based structure learning is explained and motivated, but as mentioned in the introduction, algorithms are required to handle the learning process. In the next chapter such an algorithm is presented.

Chapter 4

The SLBN Algorithm

As a part of this thesis, I have composed and implemented my own version of a structure learning algorithm, called SLBN, which is an acronym for Structure Learning in Bayesian Networks. SLBN is implemented in R and consist of 545 lines of code. The source-code is available for download at github [13].

SLBN is a constraint-based learning algorithm, composed of five steps. The first step, obtaining the MBs, and the forth step, removing cycles, are inspired by the Grow/Shrink algorithm (Margaritis,2003). Step two, three and five, obtaining the skeleton and orienting the edges, are inspired by the IC-algorithm (Pearl,2000)[1].

In section 4.2 each step of SLBN will be explained and motivated, and some details about the implementation are found in section 4.3. In the following section, the input and output of SLBN is specified.

4.1 Input and output

input: Data containing observations of RVs, following a stable distribution, that sufficiently describes the modeled phenomenon.
output: A *maximally oriented graph*, i.e. a PDAG from the equivalence class, defined by the inferred skeleton and V-structures, in which undirected edges are oriented whenever the opposite orientation would introduce new V-structures or cycles (Pellet,Elisseff,2008).

The sufficiency assumption ensures that relationships in the obtained graph represents relationships among the observed RVs. Presence of latent variables would violate the results of SLBN since "false" edges, connecting RVs affected by the same latent variable, may occur. As an example, consider the BN in figure

[1]The algorithms in the constraint-based family stems from the IC-algorithm (IC for Inductive causation). IC was proposed by Verma and Pearl in 1990.

4.1 and suppose that A is not observed in data. Since no subset of the observed RVs would render B and C independent, there would be a "false" edge connecting them in the inferred graph structure.

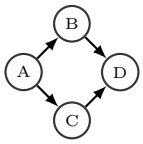


Figure 4.1: Graph used to illustrate a model with a latent variable A.

Unstable distributions may entail conditional independences that are not structural, as explained in section 3.4, and relationships inferred from such independences are not desired in this context.

4.2 Step by step

In this section, each step of SLBN is explained. The following list introduces the five steps. Starting from a graph consisting of one node for each RV in the distribution and no edges.

Step 1 **Determine the Markov blanket** for each node.

Step 2 **Obtain the graph skeleton** by finding the neighbors of each node.

Step 3 **Propagate V-structures**. Orients edges that are part of a V-structure.

Step 4 **Remove cycles** from the graph.

Step 5 **Add compelled edges.** Orients yet undirected edges.

Note that independences are tested, not dependences. However, the notation of dependence sometimes makes the ideas more intuitive, and will therefore be used in the following descriptions.

4.2.1 Step 1. Determine the Markov blankets

Step 1 determines a set of MBs (defined in section 3.5), one for each node, in two phases, the growing- and the shrinking-phase. Consider a BN with a set of RVs $\mathbf{V}$, distributed according to P. Let for A $\in \mathbf{V}$, $\mathbf{S}(A)$ be the set that is to be formed into $\mathbf{MB}(A)$, and let $\mathbf{N}(A)$ denote the neighborhood of A. Note that instances in $\mathbf{V}$ and in the sets defined above represent both nodes and RVs, e.g. A represent both a RV and a node, and $\mathbf{MB}(A)$ consist of instances representing both the

RVs that corresponds to the nodes in A's MB, and the nodes in $\mathbf{MB}(A)$. Step 1 obtains the MBs through the procedure presented in the pseudo-code of algorithm 2.

for $A \in \mathbf{V}$ **do**
  $\mathbf{S}(A) = \emptyset$
  #growing:
  while $\Big(\exists\ B \in \mathbf{V} \backslash A\ s.t.\ (A \not\perp\!\!\!\perp B | \mathbf{S}(A))\Big)$ **do**
    $\mathbf{S}(A) = \mathbf{S}(A) \cup B$
  end
  #shrinking:
  while $\Big(\exists\ B \in \mathbf{S}(A)\ s.t.\ (A \perp\!\!\!\perp B | \mathbf{S}(A) \backslash B)\Big)$ **do**
    $\mathbf{S}(A) = \mathbf{S}(A) \backslash B$
  end
  $\mathbf{MB}(A) = \mathbf{S}(A)$
end

Algorithm 2: Step 1 of SLBN.

For each node in the BN, say A, we initialize a set, $\mathbf{S}(A) = \emptyset$, which is suppose to be completed into A's MB, $\mathbf{MB}(A)$. If the test $(A \perp\!\!\!\perp B | \mathbf{S}(A))$ indicates that A is dependent of another RV B when conditioned on $\mathbf{S}(A)$, which is the best candidate of $\mathbf{MB}(A)$ at this point, B is included in $\mathbf{S}(A)$. This continues until all RVs but those in $\mathbf{S}(A)$ is independent of A given $\mathbf{S}(A)$. In this way the growing-phase provides a set of potential MBs, s.t. $\mathbf{MB}(V) \subseteq \mathbf{S}(V)$ for each $V \in \mathbf{V}$. At this point, there may be nodes in $\mathbf{S}(A)$, that are not members of $\mathbf{MB}(A)$. This will be the case if A and another RV, say B, is dependent when conditioned on $\mathbf{S}(A)$ at the time of the test $(A \perp\!\!\!\perp B | \mathbf{S}(A))$, but another RV C, subsequently added to $\mathbf{S}(A)$, renders A and B independent when conditioned on it. This motivates the shrinking-phase, which excludes false members from the $\mathbf{S}$ sets. If there exist a RV B $\in \mathbf{S}(A)$ that is independent of A when conditioned on $\mathbf{S}(A) \backslash B$, it is a false member of $\mathbf{MB}(A)$ and we exclude it from $\mathbf{S}(A)$. This continues until each RV B in $\mathbf{S}(A)$ is dependent of A when conditioned on $\mathbf{S}(A) \backslash B$. Then we assign $\mathbf{MB}(A) = \mathbf{S}(A)$.

A walk-through of Step 1, when obtaining a MB, is presented in section 7.3 of the appendix.

4.2.2 Step 2. Obtain the graph skeleton

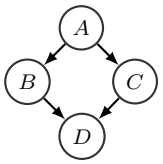
Step 2 obtains the skeleton by connecting each node with its neighbors, through the following procedure (using the notation introduced in the last section):

For each A $\in$ **V** and for each B $\in$ **MB**(A), if there is no set **H** $\subseteq$ **U** s.t. $(A \perp\!\!\!\perp B|\mathbf{H})$, where **U** is the smaller of **MB**$(A)\backslash B$ and **MB**$(B)\backslash A$, then include B in **N**(A).

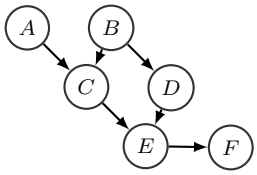
Two nodes, say A and B, are to be adjacent if and only if there is no subset **W** $\subseteq$ **V**$\backslash\{A, B\}$ s.t. $(A \perp\!\!\!\perp B|\mathbf{W})$. However, all potential neighbors of a node belongs to its MB. Furthermore, two nodes in the same MB are to be separated if and only if they share a common child, and this child belongs to both of the nodes MBs. Hence, all information that is needed to determine if A and B are neighbors is contained both in **MB**(A) and in **MB**(B), and we may condition on subsets of the smaller of those two without loss of generality. Note that it would take 2^{k-2} independence tests to determine if two nodes are to be neighbors if we were to consider the entire set of k RVs in a BN (2^{k-2} being the cardinality of **V**$\backslash\{A, B\}$'s power set). In general, the cardinality of a MB is small compared to 2^{k-2}, and the number of independence tests in Step 2 may be substantially reduced.

As an example of Step 2, consider the BN in figure 4.2a and suppose that we determine the neighborhood of B. When testing $(B \perp\!\!\!\perp C|A)$, where $A \in$ **MB**$(B)\backslash C = \{A, D\} =$ **MB**$(C)\backslash B$, this will indicate dependence, and B and C will not be connected. On the other hand, when testing $(B \perp\!\!\!\perp D|\mathbf{H})$, where **H** $\subseteq \{C\} =$ **MB**$(D)\backslash B < \{A, C\} =$ **MB**$(B)\backslash D$, no **H** will separate B and D.

A walk through of Step 2, when obtaining the neighborhood of a node in a larger BN is found in section 7.4 of the appendix.



(a) DAG in example of Step 2 of SLBN.



(b) DAG in example of Step 3 of SLBN.

Figure 4.2

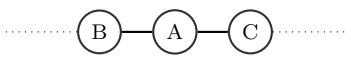
4.2.3 Step 3. Propagate V-structures

Step 3 orients edges that are parts of V-structures through the following procedure (using the notation introduced in section 4.2.1):

For each A $\in$ **V** and for each B $\in$ **N**(A), if there exist C $\in$ **N**$(A)\backslash\{$**N**$(B), B\}$ s.t. $(B \not\perp\!\!\!\perp C|A \cup \mathbf{H})$ for all **H** $\subseteq$ **U**, where **U** is the smallest of **MB**$(B)\backslash\{A, C\}$ and **MB**$(C)\backslash\{A, B\}$, then orient the edges B$\rightarrow$A and C$\rightarrow$A.

The condition identifies the kind of node triples that forms the skeleton of the fundamental connections. In such triples, D-separation provides a way to orient edges in the graph, so that the implied relationships is valid in the underlying distribution, as explained in section 3.3.

In a BN with RVs $\{A, B, C\} \cup \mathbf{V}$, consider the subgraph



and note that B and C are nonadjacent, but that there might exist other paths connecting them in the remaining graph. With A $\in \mathbf{V}$, B $\in \mathbf{N}(A) = \{B, C, \ldots\}$ and C $\in \mathbf{N}(A)\backslash\{\mathbf{N}(B), B\}$, this triple is identified in Step 3. If $(B \not\perp\!\!\!\perp C | A \cup \mathbf{W})$ holds for every subset $\mathbf{W} \subseteq \mathbf{V}\backslash\{A, B, C\}$, there cannot be an edge B←A, because in that case there would be an unblocked path B←A−C, regardless of the orientation of the edge connecting A and C, and $(B \perp\!\!\!\perp C | A \cup \mathbf{W})$ would hold (one $\mathbf{W}$ satisfying this is $\mathbf{W} = \emptyset$). Hence, orienting the edge B→A is the only option, and by a symmetry argument also A←C follows, i.e. we have the V-structure B→A←C. The subsets we condition on serves to block all paths connecting B and C but the one traversing A. If there exist such paths, those will be blocked both by nodes in $\mathbf{MB}(B)\backslash\{A, C\}$ and in $\mathbf{MB}(C)\backslash\{A, B\}$, hence we might condition on subsets of the smaller of those two MBs without a loss of generality.

As an example, picture the skeleton of the BN in figure 4.2b, and suppose that E is examined. With D $\in \mathbf{N}(E)$ and C $\in \mathbf{N}(E)\backslash\{\mathbf{N}(D), D\}$, and since $(D \not\perp\!\!\!\perp C | E \cup \mathbf{H})$ holds for any H $\subseteq \{B\}$, where B is the smaller of B $= \mathbf{MB}(D)\backslash\{E, C\}$ and $\{A, B\} = \mathbf{MB}(C)\backslash\{E, D\}$, the desired V-structure C→E←D would be detected.

4.2.4 Step 4. Remove cycles

In the absence of erroneously oriented edges Step 4 is excessive, since we assume DAGs. However, with a large number of independence tests, the graph resulting from Step 3 may contain such. If those result in cycles, Step 4 removes those cycles through the following two steps:

1. **step one, remove cycles**
   As long as there are cycles in the graph, iterate the following two operations:
   - form $\mathbf{C}$, the collection of all edges that are part of a cycle ($\mathbf{C}$ may contain duplicates),
   - remove the most frequent member of $\mathbf{C}$ from the graph[2], and add it to

[2]Removing the most frequent edge is a heuristic. The problem of removing the edge, s.t. most cycles disappear is NP-complete, i.e. a problem without known efficient solutions (Margaritis,2003).

the set $\mathbf{R}$.

2. **step two, reverse edges**
   Reverse the edges in $\mathbf{R}$, and add them to the graph in the reversed removal order.

Note that Step 4 (and Step 5), is concerned only with the graph structure, i.e. from this point SLBN does not use data.

For Step 4 to be meaningful, we need to show that no new cycles are introduced by the procedure described above. The proof of this, which also clarifies the reason for the reversed order insertion, is found in section 7.2 of the appendix.

A pseudo-code for Step 4 is presented in algorithm 3. In this, $\mathbf{C}$ is a collection and $\mathbf{R}$ is a set, both containing edges (note that $\mathbf{C}$ may contain several instances of the same edge), $\mathbf{G}_e$ is the set of edges in the graph G, $\mathbf{V}$ is the set of nodes, and (A,B) denotes the oriented edge A→B.

$\mathbf{R} = \emptyset$
#step one, remove cycles:
while *G is cyclic* **do**
  $\mathbf{C} = \emptyset$;
  for *edge (A,B)* $\in \mathbf{G}_e$ *s.t.* (A, B) *is part of a cycle in* G **do**
    $\mathbf{C} = \mathbf{C} \cup (A, B)$
  end
  e_c = most frequent edge in $\mathbf{C}$;
  $\mathbf{G}_e = \mathbf{G}_e \backslash e_c$
  $\mathbf{R} = \mathbf{R} \cup e_c$
end
#step two, reverse edges:
reverse $\mathbf{R}$
for *edge* $(A, B) \in \mathbf{R}$ **do**
  $\mathbf{G}_e = \mathbf{G}_e \cup (B, A)$
end

Algorithm 3: Step 4 of SLBN.

As an example of Step 4, consider the graphs in figure 4.3, where a dashed edge represent a directed path traversing zero or more nodes. Initially, i.e. in the first graph, there are three cycles, (B,...,D,...,F,E,...,C,...,A,B), (A,B,...,A) and (C,D,...,F,E,...,C). In the first transit of step one, all edges in the graph will be included in $\mathbf{C}$, and all edges but those in the dashed path from B to A and (C,D) will occur twice. Say (F,E) is removed, resulting in $\mathbf{R} = \{(F, E)\}$ and the second graph, which still contains the cycle (A,B,...,A). Say (A,B) is removed in the next

transit of step one, resulting in $\mathbf{R} = ((A, B), (F, E))$ and the third acyclic graph. Inserting the reversed edges in $\mathbf{R}$ gives the forth DAG.

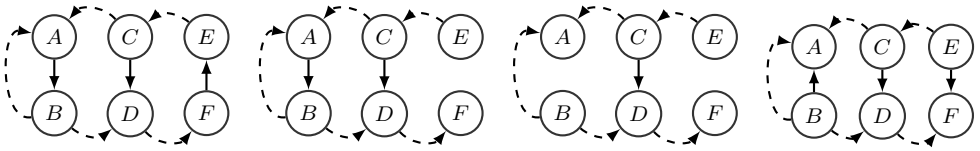


Figure 4.3: Cyclic graph is converted into a DAG by Step 4 of SLBN.

4.2.5 Step 5. Add compelled edges

Step 5 orients yet undirected edges whenever the opposite orientation would result in a new V-sructure, or in a cycle. Edges that are oriented as a result of this procedure are called *compelled* (Scutari,Nagarajan,Lébre,2013).

The following four rules are executed until they no longer applies, and whenever a change is made, the transit restarts.

R1 If there exist two nonadjacent nodes A,B $\in \mathbf{V}$ with a common neighbor C $\in \mathbf{V}$, s.t. A$\rightarrow$C$-$B, then orient the edge C$\rightarrow$B.

R2 If there exist nodes A $\in \mathbf{V}$ and B $\in \mathbf{N}(A)$ with an undirected connecting edge, and a directed path A$\rightarrow \cdots \rightarrow$B, then orient the edge A$\rightarrow$B.

R3 If there exist four nodes A,B,C,D $\in \mathbf{V}$ s.t. A$-$B, C and D are nonadjacent, and forming the paths A$-$C$\rightarrow$B and A$-$D$\rightarrow$B, then orient the edge A$\rightarrow$B.

R4 If there exist four nodes A,B,C,D $\in \mathbf{V}$ s.t. A$-$B, C and B are nonadjacent, A and D are adjacent, and forming the path A$-$C$\rightarrow$D$\rightarrow$B, then orient the edge A$\rightarrow$B.

The PDAG that Step 5 operates on entails only such relationships that can be uniquely determined by independences inferred from data, i.e. as explained in section 3.2, all directed edges in the PDAG are oriented based on their attendance in V-structures[3]. The four rules above are based on the idea that the equivalence class of the PDAG is to be maintained throughout Step 5. A motivation to each rule follows.

[3]Note that this is violated whenever Step 4 is executed. The disputable approach of combining Step 4 and Step 5 is discussed in section 4.4.

R1

R1 identifies node-triples of the form A→B−C. Since A and C are non-adjacent, the edge C→B would introduce a V-structure of the kind that would have been detected by Step 3, hence B→C is the only alternative.

R2

R2 identifies structures such as the one in figure 4.4a (a dashed line denotes a directed path traversing one or more nodes). The edge A→B would make the graph cyclic, hence B→A, as in figure 4.4b, is the only alternative. The proof in section 7.2 ensures that no new cycles will be introduced.

(a) (b)

Figure 4.4: Example where R2 in Step 5 of SLBN orients the edge A→B to avoid a cycle.

R3

R3 identifies substructure as the one in figure 4.5. To see why A→B is the only alternative, suppose that B→A constitutes the true orientation. Then D→A and C→A follows from avoiding cycles, resulting in the V-structure C→A←D. However, since C and D are nonadjacent, this V-structure would have been detected by Step 3, and B→A is ruled out.

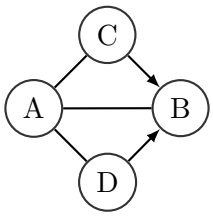


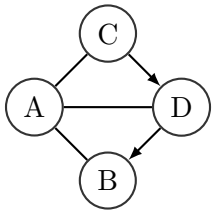
Figure 4.5: Example of graph where R3 in Step 5 of SLBN orients the edge A→B.

R4

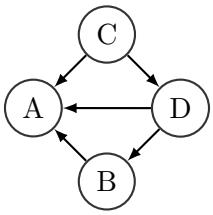
R4 identifies substructures as the one in figure 4.6a. To see why A→B is the only alternative, suppose that B→A constitutes the correct orientation. Then

the structure in figure 4.6b follows from avoiding cycles, and since B and C are nonadjacent, the V-structure centered at A in 4.6b would have been detected by Step 3. Hence B→A is ruled out.

Verma and Pearl showed that those four rules are required to obtain a maximally oriented graph (Pearl,2000), and Meek proved that they are also sufficient (Meek,1995). Meek also showed that R4 is excessive when starting from a PDAG, only containing directed edges, orientated based on their attendance in V-structures.



(a) Example of graph where R4 orients the edge A→B.



(b) Graph in the counterexample of R4.

Figure 4.6: Graphs used to illustrate R4 in Step 5 of SLBN.

4.3 Implementation of SLBN

SLBN is written in R and consists of 545 lines of code, the source-code is available for download at github [13]. The SLBN implementation takes advantage of the *ci.test*-function from the R package bnlearn when testing conditional independences [6] and the *graphviz.plot*-function from the R package Rgraphviz when drawing the graph [10]. When computing the BIC-score of the inferred structure, the *cextend*- and the *score*-function from bnlearn is used [6]. R4 of Step 5 is left out in the implementation.

SLBN takes, except from data, four optional arguments:

- A threshold value α, specifying at what limit two RVs are concluded to be independent (see section 4.4). If no argument is passed for α, the default level $\alpha = 0.05$ is used.
- A boolean "info" argument, specifying if information about the learning process is printed. An example of this information output is found in section 7.5. The default value is FALSE, i.e. no information output will be printed.

- A boolean "viewGraph" argument, specifying if the resulting graph is to be drawn. The default value is TRUE, but the option to prevent the graphical output is useful e.g. when learning the structures of a large number of simulated data-sets in a loop.

- A boolean "sCheck" argument, which allows the user to prevent synchronization of the MBs (see section 4.3.2). The default value is "TRUE".

The SLBN implementation draws the inferred graph, but it also returns a list object, consisting of the three following instances:

1. The adjacency matrix of the inferred graph structure,

2. The BIC-score of the inferred graph structure (evaluated on input data),

3. The number of performed independence tests.

4.3.1 Adjacency matrix

When constructing the graph in SLBN, it is represented by an *adjacency matrix*, i.e. with k nodes, we have a $k \times k$ matrix $\mathbf{M}$, initially consisting of zeros. If the ith node have an edge pointing towards the jth node, then $\mathbf{M}_{i,j} = 1,\ i, j = 1, 2, \ldots, k$. Undirected edges are represented by ones in both directions, i.e. if the ith and jth node are connected by an undirected edge, then $\mathbf{M}_{i,j} = \mathbf{M}_{j,i} = 1$. This representation admits the use of results from graph theory, e.g. when examining the existence of a directed path between two nodes (no details of these actions will be given here).

4.3.2 Symmetry

We expect symmetry in the MBs. If A is in B's MB, then A is either a direct neighbor of B, or they share a common child, so B also has to be in A's MB, and a similar symmetry is expected for the neighborhoods. When building the MBs in Step 1 of SLBN, errors in the independence tests, when conditioned on certain subsets of the RVs, may cause asymmetries in the MBs. If this happens, SLBN includes or excludes edges from the MBs to obtain the desired symmetry. The way this is done is based on experiments, i.e. I have implemented a mechanism that handles asymmetries based on studies the learning process of a large number of data-sets, generated to reveal systematic failures. With this solution SLBN tends to return more dense connected graphs in data-sets with a large number of RVs than e.g. the *gs*-function from the bnlearn package [6], but also to capture the "desired" graph more frequent in data-sets with a smaller number of RVs.

4.4 Discussion

In this section a number of subjects linked to SLBN is discussed. It aims to clarify shortcomings of constraint-based structure learning, improvements that are available among other algorithms, and to motivate my decisions when composing SLBN.

Numerical data

All the theory and all the examples in this text is concerned with nominal data, but SLBN also operates on numerical data. The difference in how such is handled by SLBN, is due to how the independences are tested. In the case of numerical data, the exact t-test for Pearson's correlation coefficient is used (Scutari,Nagarajan,Lébre,2013).

Combining Step 4 and Step 5

The motivation to the rules of Step 5 relies on the idea, that the graph Step 5 operates on entails only relationships that certainly corresponds to relationships in the underlying distribution, i.e. that all orientations in it is a result of Step 3. This is not the case if the graph resulting from Step 3 is cyclic and Step 4 was executed. One might therefore argue that combining Step 4 and Step 5 is a passable approach, (e.g. the IC-algorithm does not have a functionality corresponding to Step 4). However, that the graph resulting from Step 3 contains cycles indicates that data follows a distribution that doubtfully satisfies the assumptions, and the reliability of SLBN's output is in this case already violated. Furthermore, the arguments that motivates the rules in Step 5 also relies on the assumption of acyclicity, and the approach of combining Step 4 and Step 5 seems like a reasonable compromise.

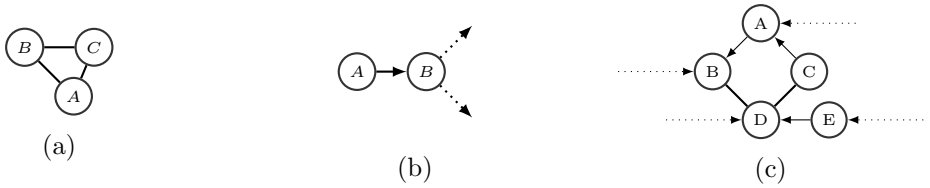
Efficient performance

Efficient performance in terms of the number of performed actions, and the execution time, have been of low priority when implementing SLBN. To gain speed, a number of excessive actions could be avoided, e.g. in Step 3, by not attempting to orient edges in a V-structure that was identified in an earlier transit. Another way to gain speed would be to use compiled code, e.g. written in C.

Undirected edges and order-determined orientations

As we have seen, not all structures can be detected by the constraint-based learning algorithms. As an example, edges in completely connected node triples as the one

in figure 4.7a are left undirected. Regardless of the orientations in this structure, the condition in Step 3 is never satisfied, and Step 5 requires at least one already oriented edge. Also edges connecting a single parent, such as the edge A→B in figure 4.7b, is left undirected.



(a) (b) (c)

Figure 4.7: Graphs used to illustrate shortcomings of constraint based structure learning.

Orientations resulting from Step 5 may depend on the variable ordering, i.e. the output of SLBN may depend on how we organize data. As an example, consider the sub-graph of a BN in figure 4.7c. If E precedes A (and C precedes B) in the variable ordering, then R1 will detect E→D−C and orient the edge D→C, and in the next transit R2 will orient the edge B←D. On the other hand, if A precedes E in the variable ordering, R1 will first orient B→D, and in the next transit R2 will orient D←C, resulting in B→D←C. Hence, for two different variable orderings, but for the same data, SLBN will return two different graphs.

The choice of threshold

Two RVs are concluded to be conditionally independent when the p-value of the independence test exceeds a specified threshold-value α, as explained in section 7.1. The statement "Small values of α, e.g. $\alpha \in [0.01, 0.05]$ work well for networks with up to hundred variables" specifies a reasonable range for the choice of α (Scutari,Nagarajan,Lébre,2013). However, to test different threshold-values and compare the results is instructive. The consequences of different threshold-values are easily concluded from how the first three steps constructs the graph in SLBN, and may be summarized as follows:

- Smaller α may result in a PDAG with fewer connections and orientations. The risk that SLBN fails to detect a true relationship increases while the risk that the output exhibits false relationships decreases.
- Higher α may result in a PDAG with more connections and orientations. The risk that the output exhibits false relationships increases, but the risk that SLBN fails to detect true relationships decreases.

At the extremes, when α is sufficiently close to one, or to zero, the result will be a complete, or a totally separated graph respectively.

Chapter 5

Experimental results

In this chapter the structure of BNs will be learned from simulated and real world data. The aim is to test the SLBN algorithm and its implementation, and to illustrate the capabilities and shortcomings of constraint-based structure learning. In sections 5.1, 5.2, 5.3 and 5.4, data-sets are simulated based on examples in this text, and in section 5.5, a real world data-set on potential risk factors of coronary trombosis is analyzed.

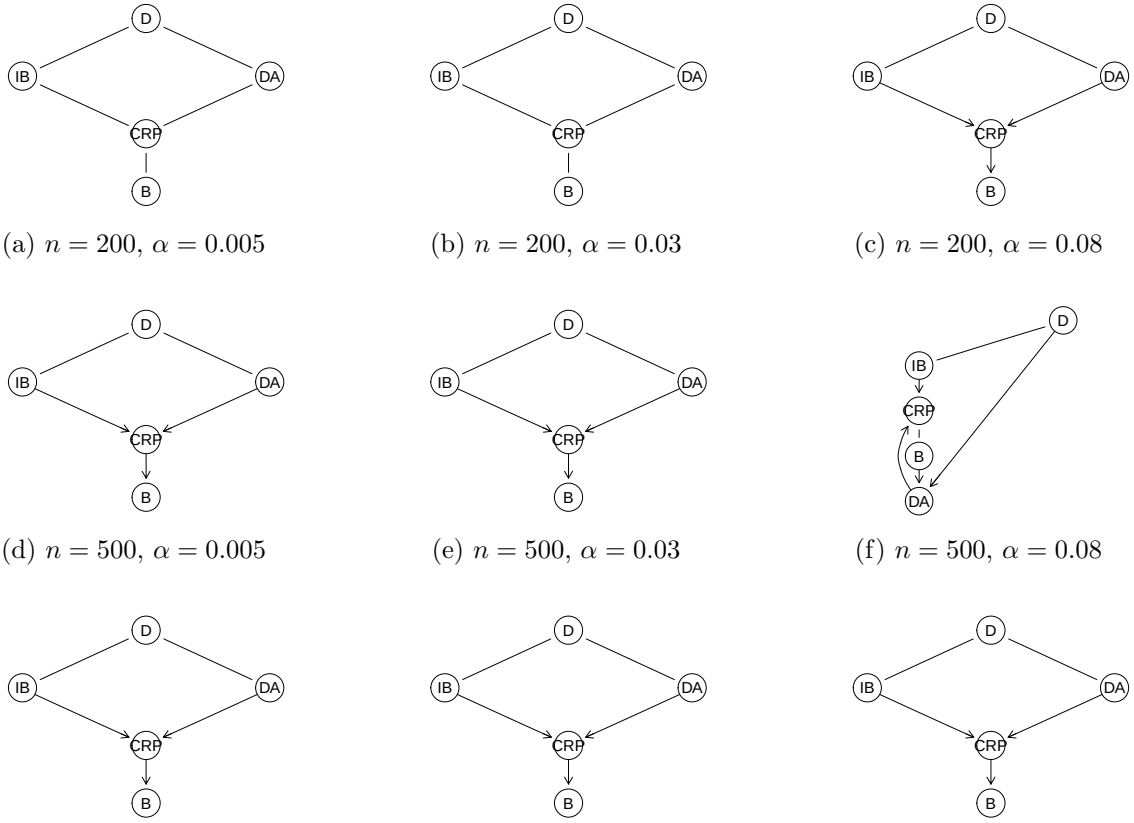
Data is simulated according to the method that was described in section 2.5 of chapter 2.

5.1 Drug test example

The drug test example was introduced in section 1.1. Simulations of it, according to the distribution specified in figure 2.2, was generated in order to illustrate the influence of the threshold-value α, and the sample size n[1]. For each $n = 200, 500, 800$, one data-set was generated, and for each $\alpha = 0.005, 0.03, 0.08$, the structure of each data-set was learned by SLBN. The inferred graphs are presented in figure 5.1.

The skeleton similar in all graphs but 5.1f, which indicates that the grow/shrink approach appears robust. The orientations of D's outgoing edges are of the kind that cannot be detected, as explained in section 3.2 and 4.1, which the graphs also shows. Therefore the graph structure in e.g. 5.1d is the best maximally oriented graph we can expect from SLBN. For $n = 200$, the higher threshold value $\alpha = 0.08$ allows Step 3 of SLBN to determine the V-structure centered at CRP, and with this present, R1 in Step 5 also orients CRP→B. For $n = 500$, the graphs indicate that SLBN perceives more edges and orientations when increasing the threshold to $\alpha = 0.08$, then an additional V-structure centered at DA is inferred. For $n = 800$,

[1]The simulation code is available for download at github [13].



(a) $n = 200$, $\alpha = 0.005$ (b) $n = 200$, $\alpha = 0.03$ (c) $n = 200$, $\alpha = 0.08$

(d) $n = 500$, $\alpha = 0.005$ (e) $n = 500$, $\alpha = 0.03$ (f) $n = 500$, $\alpha = 0.08$

(g) $n = 800$, $\alpha = 0.005$ (h) $n = 800$, $\alpha = 0.03$ (i) $n = 800$, $\alpha = 0.08$

Figure 5.1: Outputs from SLBN, when learning the structure from simulations of the drug test example.

the output is not sensitive to different threshold values, i.e. the p-values of the crucial independence tests are not in the range $\alpha \in [0.005, 0.08]$, which indicates that data, probably due to the larger sample size, is rich on information and unambiguous.

The BIC-scores of the inferred structures are presented in table 5.1 (BIC is undefined for undirected graphs, as will be explained in section 5.6). The BIC-score for the graph in figure 5.1f is much smaller than that of the other $n = 500$ graphs, and due to the BIC-score interpretation in section 3.6, a difference of 78.2 provides very strong evidence against 5.1f. Note that it is meaningless to compare structures with the BIC-score evaluated on different data-sets.

Another simulation of the same distribution, with $n = 500$ and $\alpha = 0.04$, was done to show SLBNs information output. This resulting data is presented in table 1.1 of chapter 1, and the text output in the appendix 7.5.

To illustrate how the number of detections, of the best graph we can expect

BIC	$\alpha = 0.005$	$\alpha = 0.03$	$\alpha = 0.08$
$n = 200$	undefined	undefined	-511.0
$n = 500$	-1213.5	-1213.5	-1291.7
$n = 800$	-1958.9	-1958.9	-1958.9

Table 5.1: BIC-scores for three different structures, learned from simulations of the drug test setting.

from SLBN, i.e. 5.1d, varies with the sample size, another experiment was done. For each $n = 100, 200, \ldots, 2000$, 100 data-sets was generated, again according to the distribution specified in figure 2.2. The number of those 100, for which SLBN learned the desired PDAG 5.1d, when using $\alpha = 0.04$, was counted. The result is presented in figure 5.2, where the number of detections is plotted against the sample size n. The number of detections increases with the sample size, and stabilizes around 79.85 detections of 100 when $n \geq 700$ (79.85 is the mean number of detections for $n = 700, 800, \ldots, 2000$).

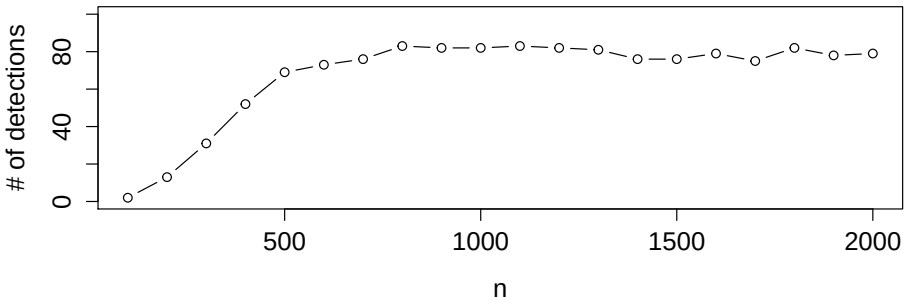


Figure 5.2: Number of detections of the desired PDAG 5.1d, when the structure was learned from 100 simulations of the drug test setting, for different sample sizes n.

5.2 Testing SLBNs capabilities

The following simulation aims to test the capabilities of SLBN, and to illustrate a goodness of fit comparison of different graph structures over the same data-set, using the BIC-score, which was explained in section 3.6.

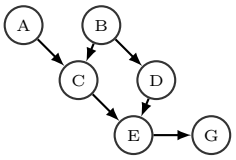
One data-set, from the BN specified in figure 5.3, was generated for each the six different sample sizes, $n = 200, 400, 600, 900, 1300, 2000$, and for each of those six

A	a_0	a_1
	0.5	0.5

B	b_0	b_1	b_2
	1/3	1/3	1/3

D|B	d_0	d_1
b_0	0.2	0.8
b_1	0.5	0.5
b_2	0.8	0.2

C|A,B		c_0	c_1
a_0	b_0	0.1	0.35
	b_1	0.35	0.65
	b_2	0.5	0.5
a_1	b_0	0.5	0.5
	b_1	0.7	0.3
	b_2	0.9	0.1



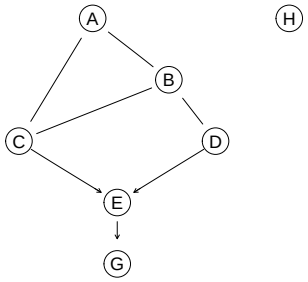
H	h_0	h_1
	0.5	0.5

G|E	g_0	g_1
e_0	0.9	0.1
e_1	0.1	0.9

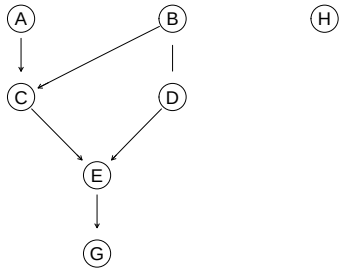
E|C,D		e_0	e_1	e_2
c_0	d_0	0.1	0.35	0.55
	d_1	0.35	0.55	0.1
c_1	d_0	0.55	0.1	0.35
	d_1	0.8	0	0.2

Figure 5.3: BN specifying the distribution used for simulations in section 5.2.

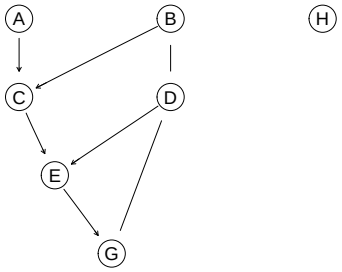
data-sets D_1,D_2,...,D_6, the structure was learned by SLBN, using $\alpha = 0.04$. This resulted in three different graph structures, S_1 from D_1, S_2 from D_2,D_4,D_5 and D_6, and S_3 from D_3, which are presented in figure 5.4. S_2 captures all relationships specified in the distribution but the edge B→D, which is of the kind that cannot be inferred (see section 3.2), i.e. in four of the six data-sets, SLBN captures the best graph we can expect. The V-structure C→E←D and the edge E→G are detected in all graphs.



(a) S_1



(b) S_2



(c) S_3

Figure 5.4: Graph structures learned by SLBN from data simulated according to the BN in figure 5.3.

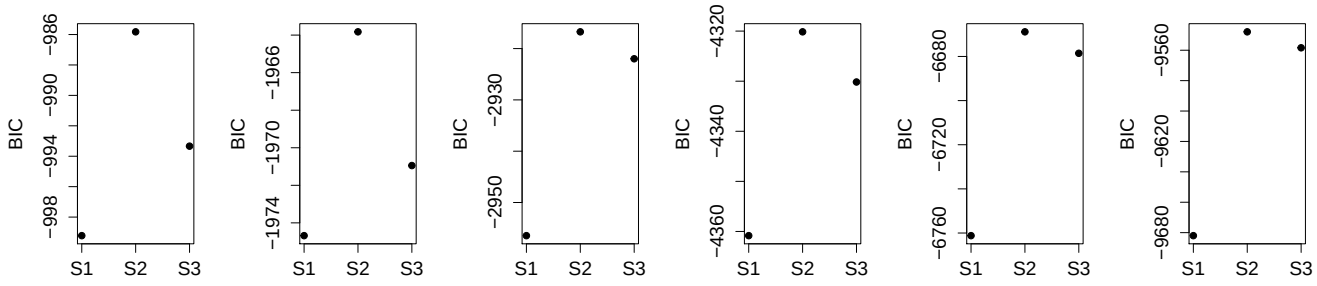
The following experiment illustrates SLBNs capabilities of capturing the best scored structure. For each of the six data-sets $D_1, \ldots, D_6$, the BIC-score was computed for S_1,S_2 and S_3, the results are plotted in figure 5.5. Next, for each of the

six data-sets, say D_1, the BIC of the structure that was learned from D_1, i.e. S_1, was compared with the BIC of the best scored of S_1, S_2 and S_3, when evaluated on D_1. The results of this, together with a similar comparison between the BIC of the highest and the lowest scored of S_1, S_2 and S_3, for each data-set, are presented in table 5.2, where $S_4 = S_5 = S_6 = S_2$.

i :	1	2	3	4	5	6
$\lvert \mathrm{BIC}(S_i, D_i) - \max_j\{\mathrm{BIC}(S_j, D_i)\} \rvert$	13.40	0	5.25	0	0	0
$\lvert \min_k\{\mathrm{BIC}(S_k, D_i)\} - \max_j\{\mathrm{BIC}(S_j, D_i)\} \rvert$	13.40	10.85	39.73	40.69	92.35	134.08

Table 5.2: Comparison of BIC-scores of the structures $S_1, \ldots, S_6$, evaluated on six different data-sets $D_1, \ldots, D_6$.

The result that would favor constraint-based structure learning, would be that the inferred graph always scored best, but as figure 5.5 indicates, this is not the case. S_2 scores best in all of the data-sets, even in those where SLBN inferred another graph. With differences exceeding 10, there is very strong evidence against S_1 in all data-sets, also in D_1 from which S_1 was learned. For D_3, the difference $|\mathrm{BIC}(S_3, D_3) - \mathrm{BIC}(S_2, D_3)| = 5.25$ is smaller, i.e. the evidence against S_3 is positive but not strong. We see that SLBNs output in the smallest data-set D_1, is substandard, but that the outputs for the larger sample-sizes are satisfying. Furthermore we see that the BIC-score favors S_2, which is the graph structure that is closest to the BN that all of the data-sets was generated from.



D_1 D_2 D_3 D_4 D_5 D_6

Figure 5.5: BIC-scores for the structures S_1,S_2 and S_3, evaluated on six different data-sets, D_1,...,D_6.

The number of executed independence tests when learning each structure was, for increasing n: $93, 103, 137, 107, 131, 129$, i.e. in a model with 7, not very densely

connected RVs, in general over 100 tests are performed. Step 4 of SLBN was not executed in any of the learning processes, i.e. Step 3 did not introduce any cycles. It is also worthwhile to note that R1 was the only rule of Step 5 that was applied (such information is provided by the optional information output of SLBN).

Another experiment, based on the distribution specified in figure 5.3 was done, again to test SLBNs capabilities of capturing the optimal structure. For each $n = 100, 200, \ldots, 1300$, 100 data-sets was generated, and for each data-set, the structure was learned by SLBN, using $\alpha = 0.04$. To compare SLBN with another implementation of a constraint-based learning algorithm, for each data-set, the structure was learned also by the *gs*-function from the R package bnlearn [6]. Next, for each sample size n, the number of learned structures that matched the "best" graph we can expect from the constraint-based learning algorithms, i.e. the graph in figure 5.4b which captures all specified edges but B→D, was counted.

The results, presented in figure 5.6, where the number of detections out of the 100, are plotted against the sample size n, are not very convincing. The number of detections increases with n, but the mean number of detections for $n = 500, 600, \ldots, 1300$ is just 50.7 of 100 for SLBN, and even lower for the gs-function. However, to count only the outputs that perfectly match 5.4b is a dubious approach since it does not account for how close the inferred structure is to the desired ditto. Hence, also the number of the six edges in 5.4b that was captured by the two algorithms (with its correct orientation), was counted for each data-set. Specifically, the mean value of the number of detected edges, for the 100 simulations was computed, for each sample size n. Those mean values are plotted against the sample size n in figure 5.7, and the plot indicates that the number of captured edges increases with n. For SLBN, it stabilizes around 5.32 edges of 6 for $n \geq 500$. Also this latter approach is dubious in order to investigate the capabilities of capturing the optimal structure, since it does not account for potential additional edges that might have been inferred. However, the two approaches together indicates that the capability of inferring a satisfying structure for small sample sizes, say $n \leq 400$, is bad. For those n we have less than 40 detections of 5.4b per 100 data-sets. For larger sample sizes, say $n > 500$, this capability is acceptable. The plots of 5.6 and 5.7 also indicates that, in this specific comparison, SLBN is slight better than the gs-function over all. Furthermore, it does not appear that the inferred structure will be more reliable when increasing the sample size beyond say $n = 600$.

5.3 Simulation with latent variables

This simulation illustrates constraint-based structure learning from data, following a distribution that does not satisfy the sufficiency assumption, which was explained in section 4.1. Data-sets of sample size $n = 1000$, was generated according to a

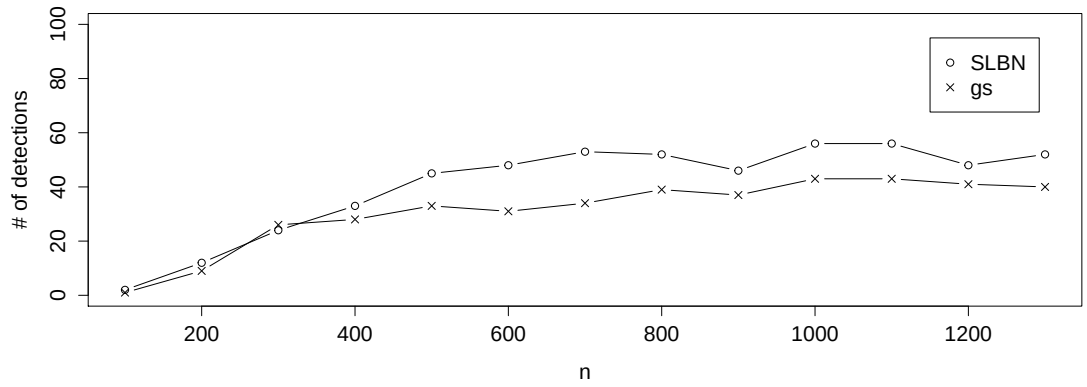


Figure 5.6: Number of detections of desired PDAG in figure 5.4b, per 100 simulations of the BN in 5.3, for different sample sizes n, and for two different algorithms.

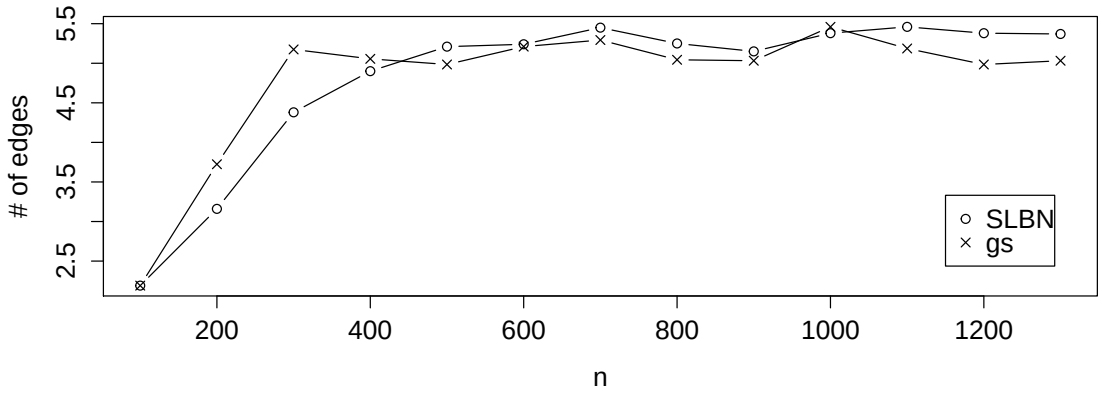
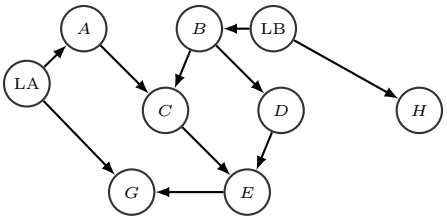


Figure 5.7: Mean number of detected edges in 100 simulations of the BN in figure 5.3, for different sample sizes n, and for two different algorithms.

BN, similar to the one in figure 5.3, but with two additional latent variables LA and LB, this BN is specified in figure 5.8. Next, the structure was learned from generated data, but in a model without LA and LB. One graph resulting from this experiment is presented in figure 5.9. It indicates that the effect of a latent

variable is local, LA's common impact on A and G results in the edge A→G, and LB's common impact on B and H ties those two together. Except for those two edges, the skeleton is identical to the corresponding sub-skeleton in figure 5.8, and the V-structures centered at C and E are present. The graph does not differ very much from the ones presented in section 5.2, and in this specific example, the presence of the latent variables may not lead to completely incorrect conclusions about the analyzed phenomenon.



LA	la_0	la_1
	0.5	0.5

LB	lb_0	lb_1
	0.5	0.5

A|LA	a_0	a_1
la_0	0.2	0.8
la_1	0.8	0.2

B|LB	b_0	b_1	b_2
lb_0	0.2	0.5	0.3
lb_1	0.1	0.2	0.7

C|A,B		c_0	c_1
a_0	b_0	0.1	0.35
	b_1	0.35	0.65
	b_2	0.5	0.5
a_1	b_0	0.5	0.5
	b_1	0.7	0.3
	b_2	0.9	0.1

D|B	d_0	d_1
b_0	0.2	0.8
b_1	0.5	0.5
b_2	0.8	0.2

H|LB	h_0	h_1
lb_0	0.85	0.15
lb_1	0.15	0.85

E|C,D		e_0	e_1	e_2
c_0	d_0	0.1	0.35	0.55
	d_1	0.35	0.55	0.1
c_1	d_0	0.55	0.1	0.35
	d_1	0.8	0	0.2

G|LA,E		g_0	g_1
la_0	e_0	0.1	0.9
	e_1	0.15	0.85
la_1	e_0	0.25	0.75
	e_1	0.05	0.95

Figure 5.8: BN used for the simulations in section 5.3

5.4 Simulation of an unstable distribution

The following experiment illustrates structure learning from data, with an unstable distribution. To this aim, consider the unstable distribution that was described in section 3.4. For eight different values of p_A, namely $p_A = 0.50, 0.52, \ldots, 0.64$, and for $p_B = 1 - p_A$, 100 data-sets of sample size $n = 300$ was generated. For each data-set, the structure was learned by SLBN, using $\alpha = 0.05$, and for each value of p_A, the number of the 100 learned graphs that captured the desired V-structure was counted. The result is presented in figure 5.10, where the number of detections of the desired V-structure is plotted against p_A. When $|p_A - p_B| \leq 0.4$, i.e. for

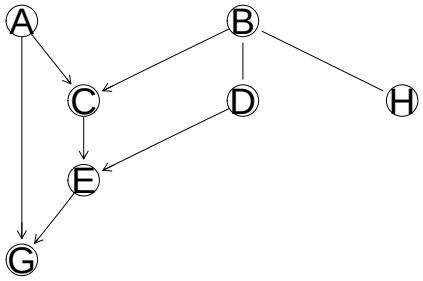


Figure 5.9: Graph inferred by SLBN in the experiment of section 5.3.

$p_A = 0.5$ and $p_A = 0.52$, only 0 and 3 outputs of 100 capture the desired V-structure respectively. The number of detections increases with p_A, and there appears to be a breakpoint between $\alpha = 0.56$ and $\alpha = 0.58$. For $p_A = 0.64$, 96 of 100 outputs capture the desired structure. This experiment illustrates that the constraint-based learning algorithms cannot handle unstable distributions satisfactory.

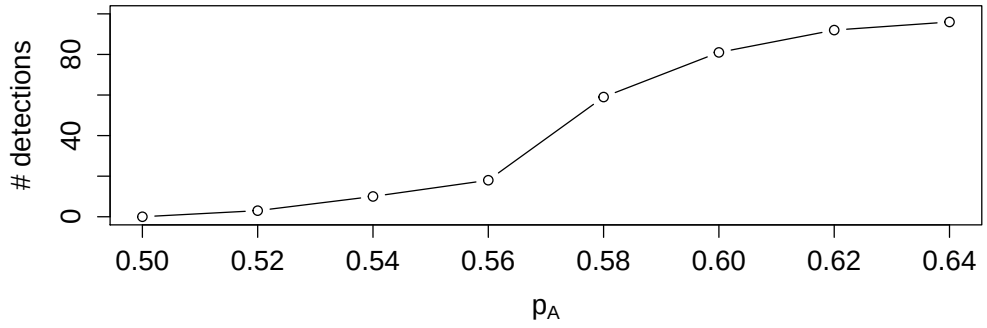


Figure 5.10: The number of learned structures per 100, that capture the desired V-structure, for different values of the probability p_A.

5.5 Coronary data-set

In this section, a real world data-set is analyzed. The coronary data-set contains information about potential risk factors for coronary trombosis, measured in 1841 men employed in a Czechoslovakian car factory. Data was collected in the 1970s,

and is available for download at bnlearns web page [7]. It contains six categorical RVs, which are presented in table 5.3. Note that none of the RVs is of specific interest, i.e. the aim with the study was to survey the relationships between the measured RVs, rather than to determine their effect on coronary trombosis.

Description	Abbreviation	Levels
smoking	S	yes/no
strenuous mental work	MW	yes/no
strenuous physical work	PW	yes/no
systolic blood pressure (unit: mm/HG)	Pse	(Pse $<$ 140) / (140 $<$ Pse)
ratio of beta and alpha lipoproteins	Prt	(Prt $<$ 3) / (3 $<$ Prt)
family anamnesis of coronary heart disease	F	yes/no

Table 5.3: Coronary data-set.

In (Whittaker,1990), the coronary data-set was used to exemplify a graphical log-linear model selection procedure, for multi-way contingency tables. The aim with Whittakers analysis is to infer an *independence graph*, i.e. a graph skeleton without directed edges. This model selection procedure is, in conformity with the introducing steps of SLBN, based on conditional independences, and Whittakers proposal is presented in figure 5.11a. The structure of the data-set was also learned both by SLBN, and by the *gs*-function from the bnlearn package [6], using both $\alpha = 0.005$ and $\alpha = 0.02$. The results of this are presented in figure 5.11.

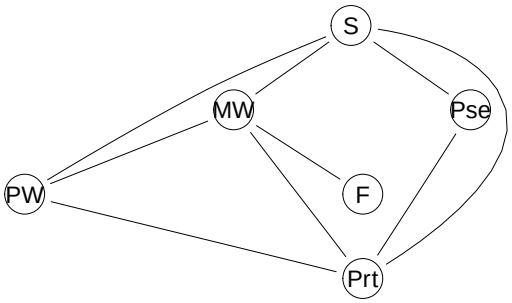
For $\alpha = 0.005$, both SLBN and gs returns structures with the same skeleton as Whittakers proposal 5.11a, but with some additional orientations. For $\alpha = 0.02$, both algorithms infers the additional edge Pse→MW (higher α-values, $0.02 \leq \alpha \leq 0.1$, result in the same graph for both SLBN and gs). The BIC-scores for the extended graphs are:

BIC	$\alpha = 0.005$	$\alpha = 0.02$
SLBN	-6726.42	-6718.54
gs	-6730.99	-6728.76

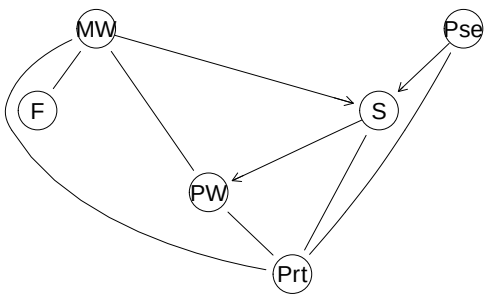
The graph in 5.11d, obtained by SLBN for $\alpha = 0.02$ scores best, and since the difference to the second best scored structure, the SLBN $\alpha = 0.005$ output 5.11b, is 7.88, 5.11d is to prefer, with strong or very strong evidence against the other structures (see the BIC interpretation in section 3.6). Since no BIC-score is available for the proposed skeleton, as explained in section 5.6, this cannot be compared to the others graphs. The fact that we obtain different graph structures, stresses the importance of testing different threshold-values.

128 and 127 independence tests were performed by SLBN when using $\alpha = 0.005$ and $\alpha = 0.02$ respectively.

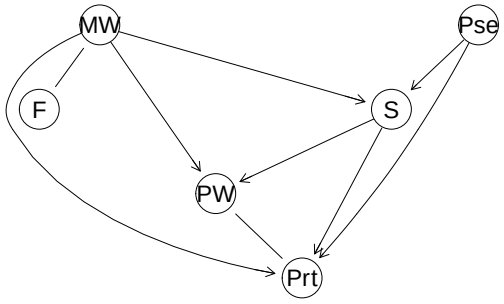
In this analysis, the constraint-based learning approach does not provide much extra information, as opposed to a method without causal concerns, i.e. a method



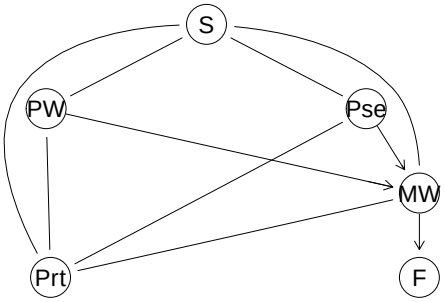
(a) Whittaker's proposal



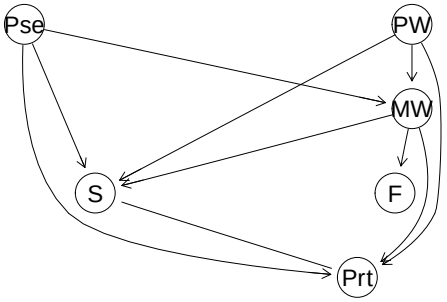
(b) SLBN, $\alpha = 0.005$



(c) gs, $\alpha = 0.005$



(d) SLBN, $\alpha = 0.02$



(e) gs, $\alpha = 0.02$

Figure 5.11: Graphs exhibiting relationships among the RVs in the coronary dataset, learned using five different approaches.

that also reveals associations among the RVs, but that does not specify their nature. Without interpreting the results in detail, it is worthwhile to note the counterintuitive edge MV→F in the graphs resulting from $\alpha = 0.02$. It appears far fetched that strenuous mental work would affect the family anamnesis of coronary heart decease. This compelled edge is a result of R1 in step 5 of SLBN (and of the corresponding action in gs), and is applied whenever the V-structure PW→MW←Pse is inferred in the sub skeleton consisting of PW−MW−Pse and MW−F. The inferred counterintuitive edge MV→F illustrates that, even though

the constrain-based learning approach captures the broad picture of the relationships, it may result in dubious causal implications, and has to be used with care.

5.6 Extending structures

The BIC-score is used to compare different structures and require DAGs, as described in section 3.6. A graph, learned by a constraint-based learning algorithm, may be only partially directed, and in order to compute the BIC-score of a PDAG, it has to be extended to a *consistent extension*, which is any DAG that belongs to the same equivalence class as the PDAG. The undirected edges in the PDAG are oriented based on methods that stems from the score-based learning approach (Daly,Shen,2009), but no details will be given here. Since the BIC-score is score equivalent, it assign the same value to this extension as to the original PDAG, which motivates a BIC-score comparison of PDAGs, through extending them. However, if not all edges in a PDAG can be oriented without introducing new V-structures or cycles, no consistent extension is available. In those cases, the BIC-score is undefined for the PDAG. In the preceding sections, graphs are extended by the *cextend*-function from the R package bnlearn [6].

Chapter 6

Discussion

The focus in this thesis was to understand the concepts of constraint-based structure learning in Bayesian Networks (BN), and how those may be used for causal inference. This was subsequently split into two main tasks. The first was to do a theoretical study of constraint-based structure learning and BNs, and to present and motivate this theory in a clear and concise matter, while also clarifying its limits. Composing this text was challenging since a large part of the theory was not part of any previous course curriculum in the math program at Stockholm University. Hence, the difficulties have been to present the basis in an adequate level of abstraction while staying concise, in order to leave room for a description of the second main task.

The second task was to compose and implement my own version of a constraint-based structure learning algorithm. The results presented in chapter 5 indicates that the resulting algorithm, SLBN (available for download at github [13]), is able to learn the structure from data. The work with this algorithm involved, besides translating the theory into practice, an large number of experiments. Those experiments consisted of using SLBN (and other available implementations of similar algorithms) to learn the structure of simulated BNs, generated to satisfy specific properties, in order to revel SLBN's (and other algorithm's) capability to infer the desired structure.

Future work

A number of desirable features were, because of the time limit of the thesis, deliberately left out in SLBN. However, a large part of the presented theory naturally extends to a broader set of applications. A number of examples of developments follows.

- Information about the strengths of the directed edges, i.e. about the strength

of the inferred causal associations, is already obtained via the p-values of the independence tests that is used in the algorithm. Hence, what remains in order to complement the output of SLBN with values that describes such strengths is to prepare, and present those values. Information about such strengths is valuable. As an example, if we in the drug test example of section 1.1 knew that decreased appetite has a large impact, but that IB has a small impact, on the probability of observing a breakout, this information would lead to different conclusions than if all inferred relationships are considered equal.

• A number of actions that would improve the efficiency of SLBN, in terms of the number of executed independence tests, are available among other existing constraint-based structure learning algorithms, and other improvements of SLBNs efficiency where discussed in section 4.4. When learning the structure of data-sets with a large number of variables, such efficiency is desired.

• SLBN assumes sufficiency, as explained in section 4.1. Pearl suggests the IC*-algorithm to handle "latent structures". The output of IC* is a *marked pattern*, in which orientations are allowed to be determined at different levels of certainty, and the sufficiency assumption is relaxed to "distributions where every unobserved variable is a parent-less common cause of exactly two nonadjacent observed variables" (Pearl,2000). The ideas of IC* could be adapted to SLBN. Note that, even though the assumption is relaxed, it still restricts the applications of structure learning to data where the random variables are carefully chosen.

• One shortcoming of the structure learning methods is the lack of significance measures. The BIC-score, and other similar metrics such as AIC or BDe, indicates which graph structure, among a number of candidates, that fit data best, but non of them provides a meaningful measure of the probability that the graph actually exhibits the "true" relationships. A familywise error rate, taking all independence tests into account, may result in a passable measure, since not all independence tests affect the learned structure, e.g. if an erroneous independence test in Step 3 of SLBN results in a suspended V-structure, this might have been the final result anyway, due to a subsequent "correct" test. As opposed to statistical methods that provides a level of significance, the result from a structure learning method is therefore to be treated as a best guess, rather than a result that is "correct" with a specific probability. However, other methods to provide a significance level are available. We may get an indicator of an inferred structures reliability through simulations, i.e. through generating a large number of data-sets according to the distribution inferred from the data of interest, and compare those simulated data-sets with original data. Furthermore, with large sample sizes, a

possible approach is to learn the structure from a part of data, and to compare a prediction based on the inferred structure with the remaining data. Based on this comparison, an indicator of the inferred structures reliability may be obtained. A significance level, indicating the reliability of a learned structure would make the method more applicable, and to complete SLBN with a functionality that provides such a measure would be a major improvement.

Applications of structure learning

The results presented in chapter 5 shows that structure learning may be used for causal inference, but we have also seen that the method is limited by strict assumptions on data. The sufficiency assumption, explained in section 4.1, is never completely satisfied, there always exist latent variables at a higher level of abstraction, and in general we will not know if the random variables used to model a phenomenon, constitutes a sufficient description of it. However, as the results presented in 5.3 indicates, the effect of a latent variable is local, i.e. its impact is limited to a part of the structure, and a large part of the inferred relationships may be valid even though the sufficiency assumption is not satisfied. Furthermore, a sufficient model is not always crucial, i.e. the aim is not always to get a complete description of a phenomenon, and an inferred causal association may provide valuable information even though it is due to latent variables.

Structure learning may be used for propagation, which was exemplified in section 2.4. The inferred graph structure together with p-values from the independence tests that the algorithm use, estimates the probability distribution. We may use this estimation to predict the outcome of a random variable of interest, for specific values on the remaining variables. Estimating a probability distribution may be done in several ways, but due to a BN's clear representation of conditional independences among the variables, structure learning in BNs is suitable for this kind of propagation.

In this text, structure learning have been exemplified in statistical contexts. However, the methods are also used for data driven prediction and decisions in the machine learning field. Specifically, automatically executed predictions, based on structures learned from a dynamic flow of incoming data, forms the basis for a machines "decisions". Sometimes, such data driven decisions constitutes a robots "intelligent" behavior in Artificial Intelligence.

Altogether, we have seen that structure learning in Bayesian Networks is a powerful tool, that provides the basis for a lucrative analysis, causal inference, but we have also seen that there are important shortcomings of the method, and that it has to be used with care. However, since the structure learning algorithms have been developed during a relatively short period, improvements in the field are probably imminent.

Chapter 7

Appendix

7.1 Independence test

In this section, Pearsson's χ^2 test for contingency tables, due to the English mathematician Karl Pearson, is explained, following (Agresti,2013) and (Scutari,Nagarajan,Lébre,2013).

In a model with a set of categorical RVs $\mathbf{V}$, we typically test $(A \perp\!\!\!\perp B|\mathbf{C})$, for RVs A,B$\in \mathbf{V}$, and a set of RVs $\mathbf{C} \subseteq \mathbf{V}\backslash\{A, B\}$. Let r_A denote the the size of A's sample space, and $r_{\mathbf{C}}$ the number of configurations of the the RVs in $\mathbf{C}$. We will present data, consisting of n independent samples, in contingency tables. The number of observations $n_{i,j}$, $i = 1, 2, \ldots, r_B,\ j = 1, 2, \ldots, r_A$ that match the jth and ith level of A and B respectively, will for each configuration $\mathbf{c}_l, l = 1, 2, \ldots, r_{\mathbf{C}}$ of $\mathbf{C}$ be given in the $r_A \times r_B$ table 7.1. In this table $\sum_{j=1}^{r_A} \sum_{i=1}^{r_B} n_{i,j} = n_{r_{\mathbf{c}}}$, where $n_{\mathbf{c}}$ is the number of samples that match $\mathbf{c}$, and $\sum_{l=1}^{r_{\mathbf{C}}} n_{\mathbf{c}_l} = n$.

$\mathbf{C} = \mathbf{c}$	A			
B	$n_{1,1}$	$n_{1,2}$	$\cdots$	n_{1,r_A}
	$n_{2,1}$	$n_{2,2}$	$\cdots$	n_{2,r_A}
	$\vdots$			
	$n_{r_B,1}$		$\cdots$	n_{r_B,r_A}

Table 7.1: $r_A \times r_B$ contingency table.

Each count in cell (i, j) is a Bernoulli distributed RV with parameter $\pi_{i,j}$, and the $n_{i,j}$s follows a joint multinomial distribution with parameters

$$\boldsymbol{\pi} = \{\pi_{i,j} : i = 1, 2, \ldots, r_B, j = 1, 2, \ldots, r_A\}, \text{ where } \sum \pi_{i,j} = 1.$$

The asymptotically unbiased ML estimator of $\pi_{i,j}$ is given by $\hat{\pi}_{i,j} = \frac{n_{i,j}}{n}$, and since the parameters sum to one, it will have $r_A r_B - 1$ degrees of freedom (d.f.). Under

$H_0 : (A \perp\!\!\!\perp B|\mathbf{C})$, the probability of an outcome in cell (i, j) is given by the product of the probability of an outcome in row i, and the probability of an outcome in column j, i.e. $\pi_{i,j} = \pi_{i\cdot}\pi_{\cdot j}$, where $\pi_{i\cdot} = \sum_{j=1}^{r_A} \pi_{i,j}$ and $\pi_{\cdot j} = \sum_{i=1}^{r_B} \pi_{i,j}$. The ML estimator of those probabilities is given by

$$\tilde{\pi}_{i,j} = \bar{n}_{i\cdot}\bar{n}_{\cdot j} = \frac{1}{n}\sum_{i=1}^{r_B} n_{i,j} \ \frac{1}{n}\sum_{j=1}^{r_A} n_{i,j}.$$

By the restrictions $\sum_{i=1}^{r_B} n_{i\cdot} = \sum_{j=1}^{r_A} n_{\cdot j} = n$, $\tilde{\pi}$ will have $(r_A - 1) + (r_B - 1)$ d.f..

We expect a test statistic that rejects H_0 for large values, to increase as data indicates deviation from the decomposition $\pi_{i,j} = \pi_i\pi_j$.

$$\sum_{i,j} \frac{(n_{i,j} - n\ \bar{n}_{i\cdot}\ \bar{n}_{\cdot j})^2}{n\ \bar{n}_{i\cdot}\ \bar{n}_{\cdot j}} \tag{7.1}$$

satisfies this. It is sometimes written $\sum_k (\mathrm{O}_k - \mathrm{E}_k)^2/\mathrm{E}_k$, where we sum over the cells in the contingency table. O denotes the observed value in each cell, and E refers to what we expect under H_0. The d.f. is given by the difference between the d.f. under the alternative hypothesis "H_0 is false", and the d.f. under H_0, i.e. $r_A r_B - 1 - \big((r_A - 1) + (r_B - 1)\big) = (r_A - 1)(r_B - 1)$. Under H_0, and for large samples, (7.1) is approximately $\chi^2_{(r_A-1)(r_B-1)}$ distributed. Furthermore, if H_0 is true we expect the square sums from different configurations of $\mathbf{C}$ to be independent. The sum of independent χ^2 RVs are again χ^2 distributed, with d.f. given by the sum of the summands ditto. Therefore the desired test statistic is

$$\sum_{\mathbf{c}\in\mathbf{C}} \sum_{i=1}^{r_B} \sum_{j=1}^{r_A} \frac{(n_{i,j} - n\ \bar{n}_{i\cdot}\ \bar{n}_{\cdot j})^2}{n\ \bar{n}_{i\cdot}\ \bar{n}_{\cdot j}},$$

which under H_0 is asymptotically $\chi^2_{r_{\mathbf{C}}(r_A-1)(r_B-1)}$ distributed, and rejects H_0 when exceeding a suitable χ^2 quantile.

7.2 Proof of correctness of Step 4 in SLBN

Step 4 is explained in 4.2.4 of chapter 4. We first prove that adding an edge to an acyclic graph, such that the opposite orientation of the edge would involve it in a cycle, does not introduce another cycle.

Consider an acyclic graph G. Towards a contradiction, suppose that the edge (B,A) is a part of the cycle (A,R_1,...,R_m,B,A), and that inserting (A,B) in the place of (B,A) in G would introduce another cycle, (B,L_1,...,L_n,A,B), see figure 7.1. Then G would also contain the cycle (A,R_1,...,R_m,B,L_1,...,L_n,A), but G was supposed to be acyclic, and the desired contradiction is met.

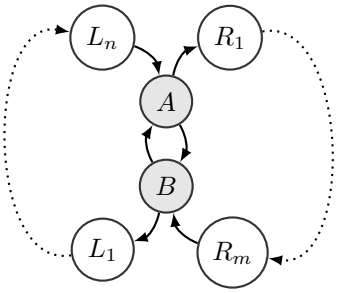


Figure 7.1: Graph G in the proof of correctness of Step 4 in SLBN.

What remains is to prove that no cycles will be introduced when inserting several edges, specifically the edges in **R** (see section 4.2.4 for specifications of **R**, **S** and the two steps, and note that those steps refers to the two sub-steps in Step 4 of SLBN).

Step one will terminate first when the graph G is acyclic, and the first inserted edge in step two, say (V_1,V_2), is the one that was removed last in step one. Since G was cyclic before (V_1,V_2) was removed, inserting (V_2,V_1) in the place of (V_1,V_2) will result in an acyclic graph, by the first part of the proof. Suppose that removing (V_1,V_2) also would have made another cycle, which was handled in an earlier transit of step one, disappear. Then (V_1,V_2) would have been involved in this last handled cycle, and the ones taken care of in the earlier transit. In this case, since the most frequent edges in **S** are removed first, (V_1,V_2) would have been removed in an earlier transit of step one, i.e. we are in the situation considered in the first part of the proof. This completes the proof.

7.3 Walk through of Step 1 in SLBN

A walk through of Step 1 in SLBN, explained in section 4.2.1, obtaining the MB of the node D in the DAG of figure 7.2 is presented in table 7.2. The variable ordering D,A,E,B,F,C is selected to stress the fact that the result does not depend on this. Note that $\mathbf{MB}(D)$ provided by Step 1 in the walk through agrees with the one that was found in section 3.5.

If the MB of F in figure 7.2 were to be determined, but in the variable ordering A,B,C,D,E,F. Then all nodes (but F itself) will be included in $\mathbf{S}(F)$ during the growing-phase, and all nodes but E will have to be removed, which stresses the importance of the shrinking-phase.

Growing phase:		
$\mathbf{S}(D) = \emptyset$,		
$\mathbf{S}(D) = \emptyset$,	$(D \not\perp\!\!\!\perp A \mid \emptyset)$ =true,	A not added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \emptyset$,	$(D \not\perp\!\!\!\perp E \mid \emptyset)$ =false,	E added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \{E\}$	$(D \not\perp\!\!\!\perp A \mid \{E\})$ =false,	A added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \{A, E\}$	$(D \not\perp\!\!\!\perp B \mid \{A, E\})$ =false,	B added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \{B, A, E\}$	$(D \not\perp\!\!\!\perp F \mid \{B, A, E\})$ =true,	F not added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \{B, A, E\}$	$(D \not\perp\!\!\!\perp C \mid \{B, A, E\})$ =false,	C added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, A, E\}$	$(D \not\perp\!\!\!\perp F \mid \{C, B, A, E\})$ =true,	F not added to $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, A, E\}$	Growing phase complete	
Shrinking phase:		
$\mathbf{S}(D) = \{C, B, A, E\}$	$(D \perp\!\!\!\perp C \mid \{B, E, A\})$ =false,	C not removed from $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, A, E\}$	$(D \perp\!\!\!\perp B \mid \{C, E, A\})$ =false,	B not removed from $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, A, E\}$	$(D \perp\!\!\!\perp A \mid \{C, B, E\})$ =true,	A removed from $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, E\}$	$(D \perp\!\!\!\perp C \mid \{B, E\})$ =false,	C not removed from $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, E\}$	$(D \perp\!\!\!\perp B \mid \{C, A\})$ =false,	B not removed from $\mathbf{S}(D)$
$\mathbf{S}(D) = \{C, B, E\}$	$(D \perp\!\!\!\perp E \mid \{C, B\})$ =false,	E not removed from $\mathbf{S}(D)$
Shrinking phase complete		
$\mathbf{MB}(D) = \mathbf{S}(D) = \{C, B, E\}$		

Table 7.2: Step 1 of SLBN determines the MB of D in the DAG of figure 7.2.

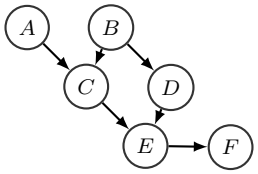


Figure 7.2: DAG in the walk throughs of Step 1 and Step 2 of SLBN.

7.4 Walk through of Step 2 in SLBN

A walk through of Step 2 in SLBN, obtaining the neighborhood of D in the DAG of figure 7.2, in the variable ordering D,A,E,B,F,C, is presented in table 7.3.

D	$\mathbf{MB}(D) = \{E, B, C\}$
	E: $\lvert\mathbf{MB}(D)\backslash\{E\}\rvert = \lvert\{B, C\}\rvert = \lvert\{F, C\}\rvert = \lvert\mathbf{MB}(E)\backslash\{D\}\rvert$ $\quad \{B, C\}$ is chosen.
	$(D \perp\!\!\!\perp E \lvert \emptyset) =$ false
	$(D \perp\!\!\!\perp E \lvert \{B\}) =$ false
	$(D \perp\!\!\!\perp E \lvert \{C\}) =$ false
	$(D \perp\!\!\!\perp E \lvert \{B, C\}) =$ false
	all tests false $\Rightarrow$ undirected edge between D and E added.
	B, similar to case above...
	all tests false $\Rightarrow$ undirected edge between D and B added.
	C: $\lvert\mathbf{MB}(D)\backslash\{C\}\rvert = \lvert\{E, B\}\rvert < \lvert\{A, E, B\}\rvert = \lvert\mathbf{MB}_C\backslash\{D\}\rvert$ $\quad \{E, B\}$ is chosen.
	$(D \perp\!\!\!\perp C \lvert \emptyset) =$ true
	$(D \perp\!\!\!\perp C \lvert \{E\}) =$ false
	$(D \perp\!\!\!\perp C \lvert \{B\}) =$ true
	not all tests false $\Rightarrow$ no edge added.
	$\mathbf{N}(D)$ completed
A...	

Table 7.3: Walk through of Step 2 of SLBN, obtaining the neighborhood of D in the DAG of figure 4.2b.

7.5 Drug test simulation, information output

```
1  SLBN.
2
3  Step 1, learn Markov blankets.
4
5
6  Obtaining Markov blanket of D, growing:
7  ( D ind IB |  ), p= 0 , indicates dep. S={IB}, restarting
8  ( D ind DA | IB ), p= 0 , indicates dep. S={IB, DA}, restarting
9  ( D ind CRP | IB, DA ), p= 0.232 , indicates ind. S unchanged
10 ( D ind B | IB, DA ), p= 0.078 , indicates ind. S unchanged, shrinking:
11 ( D ind IB | DA ), p= 0 , indicates dep. S unchanged.
12 ( D ind DA | IB ), p= 0 , indicates dep. S unchanged.
13 Obtaining Markov blanket of IB, growing:
14 ( IB ind D |  ), p= 0 , indicates dep. S={D}, restarting
15 ( IB ind DA | D ), p= 0.811 , indicates ind. S unchanged
16 ( IB ind CRP | D ), p= 0 , indicates dep. S={D, CRP}, restarting
17 ( IB ind DA | D, CRP ), p= 0.003 , indicates dep. S={D, CRP, DA}, restarting
18 ( IB ind B | D, CRP, DA ), p= 0.185 , indicates ind. S unchanged, shrinking:
19 ( IB ind D | CRP, DA ), p= 0 , indicates dep. S unchanged.
20 ( IB ind CRP | D, DA ), p= 0 , indicates dep. S unchanged.
21 ( IB ind DA | D, CRP ), p= 0.003 , indicates dep. S unchanged.
22 Obtaining Markov blanket of DA, growing:
23 ( DA ind D |  ), p= 0 , indicates dep. S={D}, restarting
24 ( DA ind IB | D ), p= 0.811 , indicates ind. S unchanged
25 ( DA ind CRP | D ), p= 0 , indicates dep. S={D, CRP}, restarting
26 ( DA ind IB | D, CRP ), p= 0.003 , indicates dep. S={D, CRP, IB}, restarting
27 ( DA ind B | D, CRP, IB ), p= 0.326 , indicates ind. S unchanged, shrinking:
28 ( DA ind D | CRP, IB ), p= 0 , indicates dep. S unchanged.
29 ( DA ind CRP | D, IB ), p= 0 , indicates dep. S unchanged.
30 ( DA ind IB | D, CRP ), p= 0.003 , indicates dep. S unchanged.
31 Obtaining Markov blanket of CRP, growing:
32 ( CRP ind D |  ), p= 0 , indicates dep. S={D}, restarting
33 ( CRP ind IB | D ), p= 0 , indicates dep. S={D, IB}, restarting
34 ( CRP ind DA | D, IB ), p= 0 , indicates dep. S={D, IB, DA}, restarting
35 ( CRP ind B | D, IB, DA ), p= 0 , indicates dep. S={D, IB, DA, B}, restarting, shrinking:
36 ( CRP ind D | IB, DA, B ), p= 0.276 , indicates ind. S={ IB, DA, B }, restarting
37 ( CRP ind IB | DA, B ), p= 0 , indicates dep. S unchanged.
38 ( CRP ind DA | IB, B ), p= 0 , indicates dep. S unchanged.
39 ( CRP ind B | IB, DA ), p= 0 , indicates dep. S unchanged.
40 Obtaining Markov blanket of B, growing:
41 ( B ind D |  ), p= 0 , indicates dep. S={D}, restarting
42 ( B ind IB | D ), p= 0 , indicates dep. S={D, IB}, restarting
43 ( B ind DA | D, IB ), p= 0 , indicates dep. S={D, IB, DA}, restarting
44 ( B ind CRP | D, IB, DA ), p= 0 , indicates dep. S={D, IB, DA, CRP}, restarting, shrinking:
45 ( B ind D | IB, DA, CRP ), p= 0.124 , indicates ind. S={ IB, DA, CRP }, restarting
```

```
( B ind IB | DA, CRP ), p= 0.333 , indicates ind. S={ DA, CRP }, restarting
( B ind DA | CRP ), p= 0.773 , indicates ind. S={ CRP }, restarting
( B ind CRP |  ), p= 0 , indicates dep. S unchanged.
Step 1 complete, Markov blankets:
 D : IB, DA
 IB : D, CRP, DA
 DA : D, CRP, IB
 CRP : IB, DA, B
 B : CRP

Step 2, learn neigborhoods, obtain skeleton.

Obtaining neighborhood of D
testing IB , condition-set: { DA }, no separating set, D IB neighbors
testing DA , condition-set: { IB }, no separating set, D DA neighbors
Obtaining neighborhood of IB
testing D , condition-set: { DA }, no separating set, IB D neighbors
testing CRP , condition-set: { D, DA }, no separating set, IB CRP neighbors
testing DA , condition-set: { D, CRP }, ( IB ind DA | D ), not neighbors
Obtaining neighborhood of DA
testing D , condition-set: { IB }, no separating set, DA D neighbors
testing CRP , condition-set: { D, IB }, no separating set, DA CRP neighbors
testing IB , condition-set: { D, CRP }, ( DA ind IB | D ), not neighbors
Obtaining neighborhood of CRP
testing IB , condition-set: { DA, B }, no separating set, CRP IB neighbors
testing DA , condition-set: { IB, B }, no separating set, CRP DA neighbors
testing B , condition-set: {  }, no separating set, CRP B neighbors
Obtaining neighborhood of B
testing CRP , condition-set: {  }, no separating set, B CRP neighbors
Step 2 complete, neighbors:
 D : IB DA
 IB : D CRP
 DA : D CRP
 CRP : IB DA B
 B : CRP

Step 3, propagate V-structures.

Centernode: D ,
Centernode: IB ,
Centernode: DA ,
Centernode: CRP ,        detected V-structure: IB->CRP<-DA
Centernode: B ,
Step 3 complete, detected orientations:
 IB -> CRP
 DA -> CRP

Step 4, Graph is acyclic

Step 5, orient compelled edges
Rule 1 satisfied, orient CRP -> B
Step 5 complete, detected orientations:
 CRP -> B

SLBN complete, Adjacency Matrix:

       D IB DA CRP B
D      0  1  1  0   0
IB     1  0  0  1   0
DA     1  0  0  1   0
CRP    0  0  0  0   1
B      0  0  0  0   0

BIC: -1211.146, number of tests: 81
```

References

[1] Judea Pearl, *Causality and reasoning*, first edition, Cambridge University Press, 2000

[2] R. Nagarajan, M. Scutari and S. Lébre, *Bayesian networks in R*, Springer, 2013

[3] D. Koller and N. Friedman, *Probabalistic graphical models*, The MIT Press, 2009

[4] F. V. Jensen and Thomas D. Nielsen *Bayesian networks and decision graphs*, second edition, Springer, 2007

[5] D. Margaritis, *Learning Bayesian networks model structure from data*, PHD thesis, 2003

[6] Marco Scutari, *bnlearn R-package*, URL:
`http://www.bnlearn.com`

[7] *Coronary data-set*, URL:
`http://www.bnlearn.com/documentation/networks/index.html`

[8] Robert E. Kass and Adrian E. Raftery, *Bayes Factors*,
Journal of the American Statistical Association, Vol. 90, No. 430, pp. 773-795, 1995

[9] Jean Philippe Pellet and André Elisseeff, *Using Markov Blankets for Causal Structure Learning*, Journal of Machine Learning Research 9, 2008

[10] K.D.Hansen J.Gentry L.Long R.Gentleman S.Falcon F.Hahne and D.Sarkar, *R-package Rgraphviz*, URL:
`http://www.bioconductor.org/packages/release/bioc/html/Rgraphviz.html`,
version: Release (3.1)

[11] Rónán Daly and Qiang Shen, *Learning Bayesian Network Equivalence Classes with Ant Colony Optimization*, Journal of Artificial Intelligence Research 35, 2009

[12] Judea Pearl, *Probabilistic reasoning in Intelligent Systems*, Morgan Kaufmann publisher, Inc., 1988

[13] Simon Berggren, *SLBN source-code,* URL: `https://github.com/SiboBerggren/SLBN`, 2015

[14] Alan Agresti, *Categorical Data Analysis*, third edition, Wiley, 2013

[15] Joe Whittaker, *Graphical Models in Applied Multivariate Statistics*, John Wiley & Sons Ltd., 1990